ULPGC Universidad de Las Palmas de Gran Canaria | eite ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

Uso y evaluación de inteligencia artificial para el desarrollo de una herramienta de marcado de vídeos sobre Moodle

Titulación:	Grado en Ingeniería en Tecnologías de la Telecomunicación
Mención:	Telemática
Autor/a:	Francisco José Cazorla Hernández
Tutor/a:	Miguel Ángel Quintana Suárez
Fecha:	Julio 2025

Agradecimientos

Al terminar esta etapa tan importante de mi vida académica, quiero agradecer a quienes me han acompañado en el camino.

En primer lugar, mi agradecimiento más grande a mi familia. Gracias por su cariño, por su paciencia cuando no podía estar y por entender mis momentos de estrés. Gracias por todo lo que han hecho por mí y por creer siempre en mis capacidades. Sus ánimos y su ayuda han sido indispensables para seguir adelante.

A mi pareja, quiero agradecer de corazón por ser mi gran compañera en esta etapa tan exigente. Gracias por entender mis horarios y mi dedicación al estudio, por tu paciencia y por tu apoyo sin condiciones. Tu presencia me ha dado la tranquilidad que necesitaba en los momentos difíciles y la alegría en los logros. Contar contigo ha hecho todo más fácil y la meta más gratificante.

A mi tutor de este trabajo, Miguel Ángel, le doy mi más sincero agradecimiento. Su ayuda y paciencia han sido muy valiosas. Gracias por su dedicación, consejos, por animarme a mejorar y confiar en mis capacidades. Sus conocimientos y experiencia me ayudaron mucho en los momentos más difíciles, convirtiendo los problemas en oportunidades para aprender.

Finalmente, quiero agradecer al grado de ingeniería en tecnologías de la telecomunicación. Estos años en la universidad me han dado conocimientos técnicos y una forma de ver y entender los problemas muy valiosa, dándome las herramientas necesarias para afrontar el futuro. Me ha enseñado la importancia de perseverar, de saber resolver problemas y de pensar de forma paciente, además de aportarme la disciplina y el interés por la innovación que ahora me preparan para lo que venga en este campo.

Gracias a todos, este logro también es suyo.

Resumen

El presente Trabajo de Fin de Grado aborda el diseño y desarrollo de una herramienta interactiva para la anotación de vídeos, integrada en la plataforma educativa Moodle. El objetivo principal es transformar la visualización pasiva de vídeos en una experiencia de aprendizaje activo, permitiendo a los estudiantes interactuar directamente con ellos mediante marcas visuales y anotaciones textuales. Una parte fundamental del proyecto es la aplicación de modelos de Inteligencia Artificial Generativa (IAG) para asistir y optimizar el proceso de desarrollo de software, actuando como un asistente en la generación de código funcional.

Para ello, se realiza un análisis exhaustivo de diferentes modelos de IAG, tanto en la nube como en entornos locales, evaluando su rendimiento en la generación de código, su eficiencia y sus capacidades contextuales. Este estudio lleva a la selección de Gemini 2.5 Flash como el modelo más adecuado para esta labor, destacando por su equilibrio entre alta precisión, velocidad de respuesta y su accesibilidad gratuita. Paralelamente, se investiga la plataforma Moodle a fondo para asegurar su correcta integración, aprovechando sus recursos nativos (como *Página* o *Tarea*) y las tecnologías web estándar (HTML5, JavaScript, CSS).

La herramienta implementa una interfaz web cliente-ligera que permite a los alumnos pausar vídeos, realizar anotaciones y asociarles información detallada. A partir de estas interacciones, el sistema genera informes estructurados en formatos PDF y ZIP (incluyendo imágenes del fotograma anotado y metadatos), los cuales pueden descargarse o subirse automáticamente a las actividades de *Tarea* en la plataforma. Además, un módulo innovador transforma estas anotaciones en vídeos interactivos bajo el estándar H5P, creando recursos educativos dinámicos y reutilizables que pueden ser fácilmente integrados.

A lo largo del proyecto, la IAG demuestra su eficacia en tareas complejas como la manipulación de la interfaz gráfica y la generación de archivos. Si bien acelera la creación del código base, su rol se complementa con la guía y el ajuste crítico por parte del desarrollador para asegurar la calidad y coherencia del resultado final. La solución lograda es accesible y adaptable, lo que facilita su replicabilidad por cualquier institución educativa sin necesidad de grandes inversiones en infraestructura.

Abstract

This Final Degree Project addresses the design and development of an interactive tool for video annotation, integrated into the Moodle educational platform. The main objective is to transform passive video viewing into an active learning experience, allowing students to interact directly with them through visual markers and textual annotations. A fundamental part of the project is the application of Generative Artificial Intelligence (GAI) models to assist and optimize the software development process, acting as an assistant in the generation of functional code.

To this end, an exhaustive analysis of different GAI models is carried out, both in the cloud and in local environments, evaluating their performance in code generation, their efficiency, and their contextual capabilities. This study leads to the selection of Gemini 2.5 Flash as the most suitable model for this task, standing out for its balance between high precision, response speed, and its free accessibility. In parallel, the Moodle platform is thoroughly investigated to ensure its correct integration, leveraging its native resources (such as Page or Assignment) and standard web technologies (HTML5, JavaScript, CSS).

The tool implements a lightweight client-side web interface that allows students to pause videos, make annotations, and associate detailed information with them. Based on these interactions, the system generates structured reports in PDF and ZIP formats (including images of the annotated frame and metadata), which can be downloaded or automatically uploaded to Assignment activities on the platform. Furthermore, an innovative module transforms these annotations into interactive videos under the H5P standard, creating dynamic and reusable educational resources that can be easily integrated.

Throughout the project, GAI demonstrates its effectiveness in complex tasks such as graphical interface manipulation and file generation. While it accelerates the creation of the base code, its role is complemented by the developer's guidance and critical adjustment to ensure the quality and consistency of the result. The solution achieved is accessible and adaptable, which facilitates its replicability by any educational institution without the need for large infrastructure investments.

Índice de contenido

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

En este capítulo se presenta el contexto, los antecedentes y los objetivos del presente Trabajo de Fin de Grado. Se define el problema a resolver en el ámbito de la tecnología educativa, se exploran las soluciones existentes y se establecen las metas específicas del proyecto.

1.1. Contexto

En la última década, la educación superior ha experimentado una profunda transformación digital, donde los **Sistemas de Gestión de Aprendizaje** (*Learning Management System*, *LMS*) se han convertido en una pieza central del ecosistema educativo. Plataformas como Moodle [1] han democratizado el acceso a recursos formativos, permitiendo a las instituciones ofrecer una experiencia de aprendizaje flexible y estructurada a una escala global. Dentro de este entorno, el contenido en formato de vídeo se ha establecido como uno de los recursos didácticos más efectivos y utilizados, gracias a su capacidad para presentar información compleja de manera visual y atractiva [2].

Sin embargo, la integración estándar de los vídeos en estas plataformas se limita, en la mayoría de los casos, a su simple reproducción. Este enfoque puede conducir a una experiencia de aprendizaje pasiva, en la que el estudiante se convierte en un espectador en lugar de un participante activo del proceso. La falta de herramientas que permitan interactuar directamente con el contenido (como la capacidad de realizar anotaciones en momentos clave, resaltar secciones específicas o extraer fragmentos de forma sencilla) constituye una barrera para un análisis más profundo y una retención efectiva del conocimiento.

Paralelamente, el campo de la ingeniería de software está siendo redefinido por los avances en la **Inteligencia Artificial Generativa** (IAG). Estas herramientas son capaces de interpretar descripciones en lenguaje natural para generar, optimizar y depurar código, acelerando significativamente los ciclos de desarrollo [3]. Este paradigma no solo mejora la productividad de los desarrolladores, sino que también facilita la creación de soluciones software más sofisticadas y personalizadas, haciendo viable la ejecución de proyectos ambiciosos dentro de marcos de tiempo y recursos acotados, como el de un Trabajo de Fin de Grado.

Este proyecto se ubica en la influencia de estos dos ámbitos: la necesidad de mejorar la interacción pedagógica con el contenido visual en plataformas de *e-learning* y el potencial de la IAG para agilizar el desarrollo de software. Por tanto, el propósito de este trabajo es diseñar e implementar una herramienta integrada en Moodle que aborde las limitaciones del visionado pasivo de vídeos. La solución propuesta ofrecerá funcionalidades para la anotación visual y textual sobre fotogramas, la generación automática de informes de análisis y la creación de contenido interactivo, transformando así el vídeo de un recurso estático a una experiencia de aprendizaje dinámica.

1.2. Antecedentes

Como se ha mencionado anteriormente, la IAG se ha consolidado como una herramienta fundamental en el desarrollo de software, pues son capaces de generar de código a partir de descripciones en lenguaje natural. Entre las soluciones basadas en la nube destacan GitHub Copilot [4], ChatGPT [5], Claude [6], DeepSeek [7] y Gemini [8], las cuales ofrecen modelos de lenguaje avanzados y proporcionan acceso a actualizaciones constantes. Sin embargo, su dependencia de una conexión a internet plantea desafíos en términos de privacidad de datos y disponibilidad del servicio, pues la interrupción de la conectividad puede ralentizar el desarrollo y exponer información sensible a riesgos potenciales [9].

En contraste, las soluciones que permiten la ejecución de modelos de IAG de manera local, como LMStudio [10], GPT4All [11] y Ollama [12], proporcionan un mayor control sobre la privacidad al no requerir el envío de datos a servidores externos y permiten un rendimiento más estable. No obstante, esta autonomía conlleva un mayor consumo de recursos computacionales y puede requerir una configuración inicial más compleja [13]. Por ello, la elección entre soluciones en la nube y locales debe basarse en un análisis de factores clave como la escalabilidad, el coste y la infraestructura disponible. Mientras que las soluciones en la nube destacan por su flexibilidad y facilidad de acceso, las opciones locales priorizan la seguridad y el control de los datos.

En línea con la transformación digital de la educación superior y el rol de plataformas como Moodle en la gestión de cursos y el acceso a recursos educativos, destaca la relevancia del contenido en vídeo como herramienta didáctica. No obstante, como se ha señalado en el apartado 1.1, la interacción con este formato se limita fundamentalmente a su reproducción pasiva, lo que obstaculiza la capacidad del estudiante para interactuar activamente con el material, realizar anotaciones o extraer fragmentos clave.

Para abordar esta limitación, se propone desarrollar una herramienta integrada en Moodle que permita pausar vídeos, realizar anotaciones visuales sobre los fotogramas y generar informes en formato PDF con la información recopilada. Dichos informes incluirán el fotograma original y el anotado, junto con metadatos relevantes de cada marca, como el usuario, el tiempo de ejecución del vídeo y las coordenadas concretas de la anotación. El proceso se automatizará para que los documentos generados sean comprimidos en formato ZIP y enviados a la tarea correspondiente en la plataforma, optimizando así el flujo de trabajo.

Este hecho ha quedado patente en el proyecto desarrollado por el Grupo de Innovación Educativa - VETFUN de la Universidad de Las Palmas de Gran Canaria. En él, la inclusión y el trabajo con vídeos obtenidos a partir de imágenes tomográficas mejoran

significativamente el proceso de aprendizaje en asignaturas fundamentales de Anatomía Veterinaria.

Para implementar esta funcionalidad se emplearán tecnologías web estándar como HTML5, JavaScript y CSS [14]. Una de las bases del desarrollo será el uso de la etiqueta `<video>` de HTML5 permite reproducir contenido multimedia directamente en el navegador sin necesidad de complementos externos [15]. La superposición de las anotaciones se podrá gestionar mediante la interfaz de programación de aplicaciones (*Application Programming Interfaces, API*) Canvas con la etiqueta `<canvas>`, que ofrece un entorno gráfico para dibujar líneas, formas y texto [16]. Las capturas de las anotaciones se procesarán para su posterior integración en los informes, utilizando bibliotecas como jsPDF [17] o PDFKit [18], que permiten crear documentos personalizados con gráficos y texto enriquecido. Además, se contempla la creación de vídeos interactivos en los que las marcas realizadas sean visibles y dinámicas durante la reproducción. Para ello, se explorará la integración de tecnologías como H5P [19], permitiendo al usuario visualizar la información relevante en tiempo real durante la reproducción.

De esta manera, el desarrollo e implementación de esta herramienta ofrecerá una experiencia interactiva que facilitará el análisis y la revisión de los contenidos visuales en un entorno educativo. Esta solución integrada en Moodle representa un paso significativo hacia la mejora de la interacción con recursos multimedia en entornos de aprendizaje en línea. La combinación de tecnologías web avanzadas con la capacidad de generación de código mediante IAG facilita la creación de una solución eficiente, adaptable y escalable, demostrando cómo la integración de tecnologías emergentes puede transformar la implementación de funcionalidades tecnológicas en la enseñanza.

1.3. Objetivos

Como se ha identificado, el objetivo principal de este proyecto es desarrollar una herramienta integrada sobre la **interfaz de usuario** (*User Interface, UI*) de Moodle que permita la anotación en vídeos mediante marcas y etiquetas con la información solicitada, la generación y entrega automatizada de informes, y la visualización interactiva del contenido multimedia editado. Para ello, se utilizará un modelo de IAG, previamente analizado, comparado y seleccionado, para optimizar la generación de código y mejorar la eficiencia del desarrollo.

Para lograr este propósito, se han definido los siguientes objetivos específicos:

Objetivo 1. Evaluación de herramientas de IAG: Definir criterios de selección de herramientas de IAG en la nube y en entornos locales de código abierto u *open-source*

para determinar cuáles ofrecen mayor eficiencia en la generación de código a partir de descripciones en lenguaje natural, evaluando criterios como inteligencia, velocidad de generación, adaptabilidad al contexto del código y facilidad de integración en el desarrollo del proyecto.

Objetivo 2. Desarrollo de un sistema de anotación en vídeos sobre Moodle: Diseñar e implementar sobre la UI de Moodle una funcionalidad que permita el marcado y etiquetado de fotogramas capturados en vídeos, incluyendo además la generación de archivos en varios formatos (PNG para imágenes, PDF para informes o JSON para metadatos) ofreciendo evidencias de la tarea realizada por el alumno, realizando su compresión y entrega en la plataforma de forma automatizada.

Objetivo 3. Implementación de vídeos interactivos con anotaciones dinámicas: Desarrollar una funcionalidad de generación de vídeos interactivos que permitan consultar las anotaciones en tiempo real, con la posibilidad de pausar su reproducción y hacer clic sobre las marcas para desplegar la información detallada.

1.4. Estructura del documento

La presente memoria se ha organizado en seis capítulos que documentan de manera progresiva la investigación, el diseño, la implementación y la validación del proyecto.

El **capítulo 1** establece el contexto del trabajo, presenta los antecedentes tecnológicos y pedagógicos, y define los objetivos que guían todo el desarrollo. Sobre la base conceptual aquí introducida, el **capítulo 2**: "**Fundamentos Teóricos**" profundiza en las tecnologías clave, describiendo la plataforma Moodle y sus utilidades, los principios de la inteligencia artificial y las particularidades de los modelos generativos tanto en la nube como en entornos locales.

Una vez sentadas estas bases teóricas, el **capítulo 3**: "**Análisis Comparativo de Modelos de IA Generativa**" presenta una evaluación cuantitativa y cualitativa de los modelos seleccionados. En él se establecen los criterios de análisis y se justifica la elección final de la herramienta de IAG que asistirá el desarrollo. A partir de esta selección, el **capítulo 4**: "**Diseño e Implementación de la Anotación de Vídeos sobre Moodle**" detalla el proceso de creación del primer módulo funcional de la herramienta, abarcando desde el diseño de la arquitectura hasta la implementación de la interfaz de marcado.

Para complementar esta funcionalidad, el **capítulo 5**: "**Sistema de Generación y Entrega de Informes**" describe la extensión del sistema para documentar las anotaciones, incluyendo la generación de archivos PDF y ZIP, y su entrega automatizada en

Moodle. Como culminación del proyecto, el **capítulo 6**: "**Integración con H5P y Creación de Vídeos Interactivos**" aborda la fase final, explicando cómo las anotaciones individuales se transforman en un recurso de aprendizaje interactivo y reutilizable mediante la generación de paquetes H5P.

Finalmente, el documento concluye con un apartado de **Conclusiones**, donde se sintetizan los resultados y se proponen líneas de trabajo futuro, seguido del **Presupuesto** detallado y los **Anexos** que complementan la información presentada.

1.5. Uso de IA Generativas

En cumplimiento de las "Recomendaciones sobre uso de la IAG en la ULPGC" [20], aprobadas por el Consejo de Gobierno Extraordinario de la ULPGC el 6 de junio de 2024 [21], en este Trabajo de Fin de Grado se ha empleado la inteligencia artificial generativa desde una doble perspectiva: como herramienta de apoyo en la elaboración de la memoria y como objeto central del desarrollo práctico.

Como herramienta de apoyo instrumental, se han utilizado diversas plataformas de IAG para optimizar tareas específicas del proceso de investigación y documentación. Para la creación de diagramas de flujo y arquitecturas visuales, como los presentados en capítulos posteriores, se empleó la herramienta NapkinAI [22]. Asimismo, se recurrió a modelos como los integrados en ChatGPT para la elaboración de bocetos conceptuales de la interfaz de usuario, así como en la fase de redacción de esta memoria para la reformulación de frases y la búsqueda de sinónimos, con el fin de mejorar la claridad y el estilo académico del texto. Finalmente, para la optimización de los *prompts* utilizados en la fase de desarrollo, se empleó Claude con el fin de refinar y estructurar las instrucciones más complejas, asegurando una mayor efectividad.

De forma más significativa, y como eje central de este trabajo, la IAG no ha sido solo una herramienta auxiliar, sino el principal objeto de estudio y el motor del desarrollo. Tal y como se establece en el Objetivo 1 y se desarrolla en el Capítulo 3, se ha realizado un análisis comparativo sistemático de diferentes modelos de IAG. Posteriormente, el modelo seleccionado fue empleado de manera intensiva para la generación asistida del código fuente de la herramienta de anotación, cuyo proceso y resultados se documentan en los capítulos 4, 5 y 6.

Capítulo 2: Fundamentos Teóricos

En este capítulo se exponen los fundamentos teóricos que sustentan el proyecto. Se realiza una introducción a los conceptos clave de la inteligencia artificial, incluyendo los paradigmas de aprendizaje y las arquitecturas de los modelos generativos, seguido de un análisis de la plataforma Moodle y sus tecnologías de desarrollo.

2.1. Introducción a la Inteligencia Artificial

La Inteligencia Artificial (IA) [23] es una disciplina de la informática que busca desarrollar sistemas capaces de realizar tareas que, tradicionalmente, requieren inteligencia humana. Estas tareas incluyen el reconocimiento de patrones, la toma de decisiones, la resolución de problemas y el procesamiento del lenguaje natural (*Natural Language Processing, NLP*). En términos generales, la IA se fundamenta en la creación y aplicación de algoritmos avanzados, diseñados para analizar datos y mejorar su rendimiento a través de la experiencia. Asimismo, se basa en el diseño de modelos computacionales capaces de aprender, razonar y adaptarse a entornos cambiantes a partir de datos.

El concepto de IA abarca múltiples enfoques, entre los que destacan el aprendizaje automático (*Machine Learning*, *ML*) y el aprendizaje profundo (*Deep Learning*, *DL*). El aprendizaje automático se basa en algoritmos que mejoran su desempeño a partir de la experiencia, sin necesidad de programación explícita. Dentro de este, el aprendizaje profundo emplea redes neuronales artificiales (*Artificial Neural Networks, ANN*) de múltiples capas para procesar información de manera jerárquica y extraer representaciones complejas de los datos de entrada. A medida que las redes neuronales se entrenan con grandes volúmenes de datos, su precisión mejora progresivamente, convirtiéndose en herramientas muy útiles en informática e inteligencia artificial.

2.1.1. Tipos de Inteligencia Artificial

Desde una perspectiva funcional, la IA puede clasificarse en dos categorías principales [24]:

Por un lado, la **Inteligencia Artificial Débil o Específica *(Weak Artificial Intelligence, WAI o Narrow Artificial Intelligence, NAI)***, que es capaz de realizar tareas específicas y limitadas. No tiene capacidad de aprendizaje o adaptación por sí misma, y requiere ser programada para realizar una tarea en un determinado campo de especialización. Unos ejemplos de IA específica pueden ser los chatbots, el reconocimiento de voz o la traducción automática, presentes en la actualidad.

Por otro lado, la **Inteligencia Artificial Fuerte o General *(Strong Artificial Intelligence, SAI o Artificial General Intelligence, AGI)***, aunque aún hipotética, está diseñada para tener una amplia gama de habilidades cognitivas y capacidad de aprendizaje autónomo. Estos sistemas pueden realizar múltiples tareas y aprenden de forma autónoma a medida que interactúan con el entorno. La IA fuerte debe tener la capacidad de razonar, planificar y tomar decisiones complejas en un amplio espectro de situaciones.

La IA también puede clasificarse en cuatro categorías principales, según la naturaleza de los datos de entrada utilizados en el proceso de aprendizaje [25]: **aprendizaje supervisado**, **aprendizaje no supervisado**, **aprendizaje semisupervisado** y **aprendizaje por refuerzo**.

En primer lugar, el **aprendizaje supervisado** emplea algoritmos que trabajan con datos etiquetados, es decir, conjuntos de datos que incluyen tanto las variables de entrada como las respuestas esperadas. El objetivo es encontrar una función o modelo que relacione de manera precisa las variables de entrada con sus correspondientes salidas. El modelo se entrena utilizando un conjunto de datos históricos, aprendiendo patrones y relaciones que permiten predecir resultados para datos nuevos no vistos durante el entrenamiento.

Este enfoque es especialmente útil en tareas como la clasificación de texto o la predicción de variables numéricas, y resulta fundamental para modelos generativos que necesitan conocer las relaciones existentes en el código para crear nuevas soluciones.

El **aprendizaje no supervisado**, por otro lado, emplea algoritmos que no requieren datos etiquetados, lo que significa que trabajan únicamente con datos de entrada sin conocer previamente los valores de salida esperados. Su principal objetivo es descubrir patrones, relaciones o estructuras subyacentes en los datos, permitiendo así una organización o agrupación de la información de forma automática y sin intervención humana directa. Estos métodos resultan particularmente útiles cuando no se dispone de datos con respuestas conocidas, ya que pueden revelar segmentos ocultos o asociaciones inesperadas.

Aunque este enfoque es muy empleado en problemas como la agrupación de datos o el análisis exploratorio de conjuntos complejos, su aplicación directa en la generación de código con modelos generativos es limitada, ya que este tipo de modelos suelen requerir información explícita sobre las estructuras del lenguaje de programación que están tratando de replicar o ampliar.

Ocupando un lugar intermedio entre los dos paradigmas anteriores, se encuentra el **aprendizaje semisupervisado** [26]. Esta rama combina el uso de datos etiquetados y no etiquetados para entrenar modelos de IA para tareas de clasificación y regresión. El objetivo es aprovechar la estructura inherente de la totalidad de los datos para construir un modelo más robusto y preciso de lo que sería posible utilizando únicamente el conjunto reducido de datos etiquetados.

Este enfoque es particularmente valioso en dominios donde la adquisición de etiquetas es un proceso prohibitivamente difícil o costoso, pero grandes volúmenes de datos no etiquetados son relativamente fáciles de obtener.

Por último, el **aprendizaje por refuerzo** se basa en un agente que toma decisiones en un entorno dinámico, con el objetivo de maximizar una recompensa acumulada a lo largo del tiempo. Este tipo de aprendizaje se estructura como un ciclo de interacción: el agente toma una acción, el entorno responde con un nuevo estado y una señal de recompensa, y el agente ajusta su estrategia para mejorar su desempeño.

Pese a demostrar ser muy eficaz en áreas como la robótica, los videojuegos y la optimización de procesos industriales, no tiene una aplicación directa en la generación automática de código mediante modelos generativos.

La tabla 1 presenta una comparativa de los cuatro paradigmas de aprendizaje analizados, destacando sus principales diferencias.

Tipo de aprendizaje	Tipo de datos de entrada	Objetivo principal	Aplicaciones
Supervisado	Datos etiquetados (pares de entrada-salida).	Aprender una función que mapea entradas a salidas conocidas para realizar predicciones.	Clasificación de texto, regresión, reconocimiento de imágenes.
No supervisado	Datos no etiquetados.	Descubrir patrones, estructuras o agrupaciones (clústeres) inherentes en los datos.	Agrupación de clientes, detección de anomalías, reducción de dimensionalidad.
Semisupervisado	Combinación de datos etiquetados (pocos) y no etiquetados (muchos).	Mejorar la precisión del modelo aprovechando la estructura de los datos no etiquetados.	Clasificación de documentos o imágenes donde el etiquetado manual es costoso.
Por refuerzo	Interacción con un entorno (estado, acción, recompensa).	Aprender una política de acciones para maximizar una recompensa acumulada a lo largo del tiempo.	Robótica, control de sistemas autónomos, juegos.

Tabla 1. Comparativa de paradigmas de aprendizaje. Fuente: Elaboración propia

2.2. Introducción a la Inteligencia Artificial Generativa

La **Inteligencia Artificial Generativa** (IAG) [27] es una rama de la inteligencia artificial enfocada en la creación de contenido nuevo a partir de datos existentes. A diferencia de otros sistemas que se centran en la clasificación, detección o toma de decisiones, estos tienen la capacidad de producir autónomamente texto, imágenes, audio, vídeos y código de programación.

Se basa en modelos matemáticos y estadísticos diseñados para aprender patrones complejos a partir de grandes volúmenes de datos. Además, se fundamenta en la

capacidad de estos modelos para extraer características relevantes de los datos de entrenamiento y utilizarlas en la generación de contenido, que sigue la distribución aprendida sin replicar directamente los ejemplos utilizados durante el aprendizaje.

2.2.1. Arquitecturas de modelos generativos

Para lograr esto, la IAG emplea principalmente redes neuronales profundas (*Deep Neural Networks, DNN*), estructuras computacionales inspiradas en el funcionamiento del cerebro humano que permiten modelar relaciones complejas en los datos, compuestas por múltiples capas de neuronas artificiales interconectadas, cada una de las cuales aplica transformaciones matemáticas a los datos de entrada para extraer patrones y representaciones progresivamente más abstractas.

Un ejemplo destacado de modelo generativo es el de las **Redes Generativas Adversariales o Antagónicas** (*Generative Adversarial Networks, GAN*), introducidas por Goodfellow et al. en 2014. Se compone de dos redes neuronales que trabajan de forma conjunta: el generador, que crea datos sintéticos para imitar a los reales, y el discriminador, que evalúa si los datos generados se parecen a los datos reales.

A través de un proceso de entrenamiento adversarial, el generador mejora sus resultados para engañar al discriminador, mientras este se perfecciona para detectar las diferencias entre datos reales y generados, logrando así que el generador produzca contenidos cada vez más realistas. Este proceso se puede ver claramente en la figura 1, que proporciona un esquema visual de este modelo.

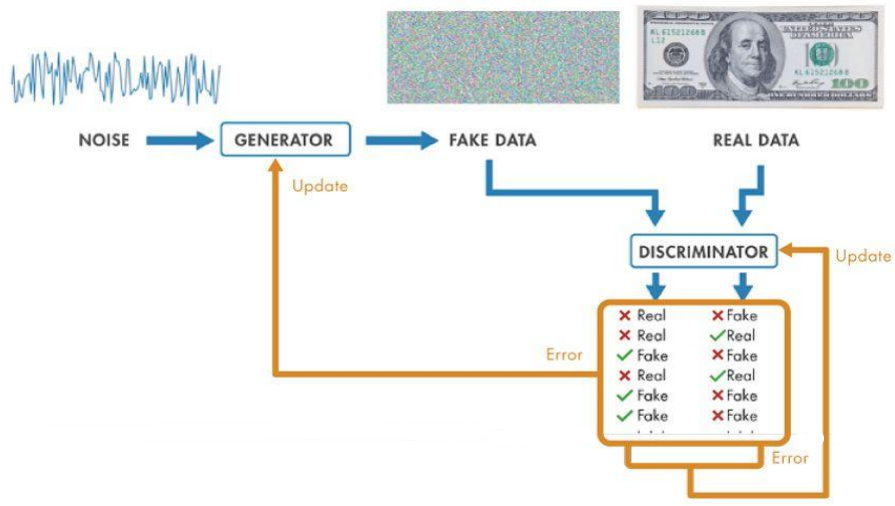


Figura 1. Flujo de entrenamiento de una GAN. Fuente: [28]

Este tipo de modelos ha encontrado aplicaciones en diversas áreas como la generación de imágenes sintéticas, la mejora de la resolución de imágenes y la conversión de estilos

visuales. Algunos modelos más avanzados, como StyleGAN, han permitido incluso controlar de forma precisa el estilo y la apariencia de las imágenes generadas.

Por otra parte, los **modelos autorregresivos** (*Autoregressive, AR, Models*) también representan una parte fundamental de la inteligencia artificial generativa. Estos modelos generan contenido de manera secuencial, prediciendo cada nuevo elemento a partir de los anteriores en la secuencia. En el ámbito del procesamiento del lenguaje natural, este enfoque es fundamental para la creación de texto coherente y fluido.

Un modelo representativo de esta categoría es GPT (*Generative Pre-trained Transformer*) [29], que ha supuesto un gran avance en la generación de texto gracias a su capacidad para aprender el contexto y las relaciones entre las palabras. Aprovechando la técnica de autoatención *(self-attention*), estos modelos pueden analizar simultáneamente distintas partes del contexto para generar secuencias más naturales y adaptadas. Este mecanismo es básico para entender cómo los modelos autorregresivos son capaces de mantener la coherencia y la fluidez del texto, o en el caso de este trabajo, de las líneas de código generadas.

La figura 2 muestra el flujo general de funcionamiento de un modelo de lenguaje autorregresivo. Se parte de una ventana de contexto formada por palabras anteriores, que se transforma en vectores mediante una capa de vectorización (*embedding*) y se procesa a través de capas ocultas de una red neuronal. El resultado final es una distribución de probabilidad sobre las posibles palabras siguientes, permitiendo al modelo seleccionar la palabra más probable para completar la secuencia.

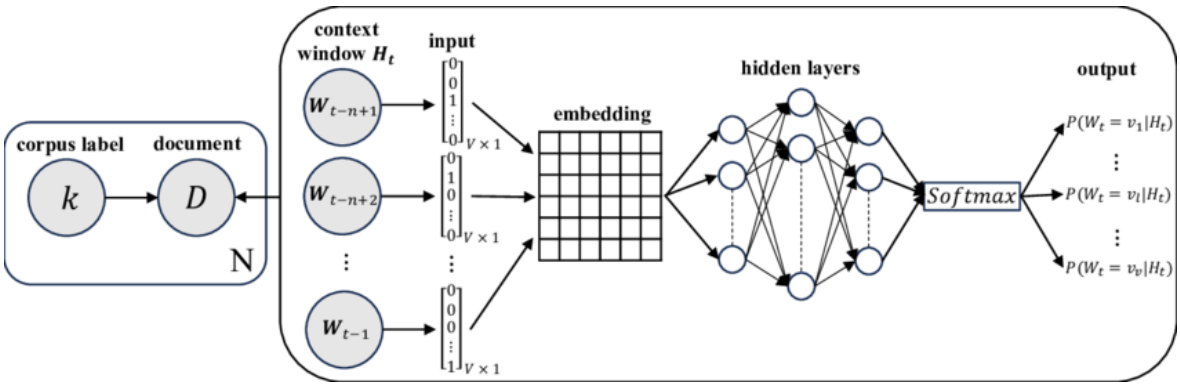


Figura 2. Flujo de un modelo autorregresivo clásico. Fuente [30]

Aunque los modelos autorregresivos también se utilizan en tareas como la traducción automática, su aplicación más relevante para este trabajo es la generación de código, ya que se basan en el mismo principio de predicción secuencial. Herramientas basadas en modelos como GPT-4, BERT o T5 están diseñadas para ofrecer sugerencias de código y

completar automáticamente fragmentos, apoyándose precisamente en esa capacidad de los modelos autorregresivos para generar secuencias adaptadas al contexto.

Por su parte, los **modelos variacionales** [27] (*Variational Autoencoders, VAE*) representan otro enfoque dentro de la inteligencia artificial generativa. Estos modelos aprenden a codificar los datos de entrada en una representación comprimida y probabilística, desde la cual son capaces de reconstruir una salida similar. A diferencia de los modelos autorregresivos, que generan secuencias paso a paso, los VAE se centran en capturar las características generales de los datos mediante una distribución latente.

A pesar de que su uso más común ha sido en la generación de imágenes, también se han aplicado a tareas como la compresión de datos, la reducción de ruido o la generación controlada de contenido, lo que los convierte en una opción interesante como complemento a otras técnicas generativas más conocidas.

2.2.2. Modelos de lenguaje para generación de código

La generación de código asistida por inteligencia artificial se fundamenta en el uso de modelos de lenguaje, los cuales pueden clasificarse según su tamaño y capacidad en **Modelos de Lenguaje Simples o Pequeños** y **Modelos de Lenguaje de Gran Escala**.

Por una parte, los **Modelos de Lenguaje Simples o Pequeños (*Small Language Models, SLM*)** [31] son modelos de inteligencia artificial diseñados a menor escala para comprender y generar lenguaje natural. Estos suelen tener entre unos pocos millones y unos pocos miles de millones de parámetros, lo que los hace más compactos, eficientes y aptos para entornos con recursos limitados, como dispositivos móviles, sistemas embebidos o ejecución offline sin conexión a red.

Aunque utilizan arquitecturas similares a los grandes modelos (como los *transformers*), se benefician de técnicas de compresión como *pruning*, cuantización y factorización de matrices, entre otros, permitiendo mantener una buena precisión con una menor carga computacional. Esta eficiencia no compromete necesariamente su rendimiento, ya que en tareas específicas como la generación de texto, resúmenes o incluso generación de código, algunos modelos como *GPT-4 mini* o *Granite-3B* han demostrado superar a modelos más grandes en rapidez y relevancia contextual.

Por otro lado, los **Modelos de Lenguaje de Gran Escala** (***Large Language Models, LLMs***) [32] son una clase de modelos fundacionales entrenados sobre grandes cantidades de datos textuales, diseñados para comprender y generar lenguaje natural con un alto nivel de coherencia y contexto. A diferencia de los modelos pequeños, estos cuentan con miles de millones de parámetros, lo que les permite identificar patrones lingüísticos complejos y

abordar una amplia gama de tareas sin necesidad de ser entrenados específicamente para cada una de ellas.

Durante el entrenamiento, los LLMs aprenden a predecir la siguiente palabra en función del contexto, operando bajo esquemas de aprendizaje no supervisado. El proceso implica la tokenización del texto, es decir, su segmentación en unidades mínimas llamadas **tokens** [33] (palabras, juegos de caracteres o combinaciones de palabras), que el modelo analiza para detectar patrones lingüísticos; la conversión en representaciones numéricas (incrustaciones o *embeddings*) y el ajuste de los pesos de la red neuronal a partir de *corpus* masivos que incluyen miles de millones de páginas. Gracias a esto, pueden generar contenido en formato de texto, responder preguntas, resumir información, analizar sentimientos o asistir en tareas de programación, como la generación y corrección de código, siendo así una herramienta útil en el desarrollo de asistentes inteligentes, aplicaciones de atención al cliente o automatización de procesos complejos.

2.2.3. Aplicaciones transversales y sectoriales

La inteligencia artificial generativa ha demostrado su versatilidad en múltiples tareas y sectores, transformando la manera en que las organizaciones operan, optimizan procesos y ofrecen servicios. Desde el ámbito financiero hasta la educación, su adopción está guiada por la necesidad de mejorar la eficiencia, personalizar la experiencia del usuario y potenciar la toma de decisiones basada en datos.

Aplicaciones transversales:

Algunas de las aplicaciones directas de la IAG incluyen [34]:

- **Asistencia y comunicación interpersonal**: Se cuenta entre los casos de uso más adoptados por las organizaciones, incluyendo chatbots y asistentes virtuales. Estos permiten interacciones fluidas y personalizadas en servicio al cliente y recursos humanos, optimizando la comunicación y el soporte en tiempo real.
- **Marketing y ventas impulsado por interacción**: La IA conversacional se utiliza para recopilar datos del cliente durante las interacciones. Esto permite comprender preferencias, adaptar estrategias de marketing y ventas, y ofrecer sugerencias personalizadas para optimizar la conversión.
- **Gestión y generación de conocimiento**: Se emplea para identificar, resumir, traducir y recuperar información específica de grandes volúmenes de datos. También facilita la creación de contenido y ofrece soluciones a problemas complejos, optimizando la accesibilidad y la toma de decisiones.

Adopción sectorial:

Por su parte, la IAG también ha tenido impacto sobre determinados sectores [35], tales como:

- **Sector financiero**: Las entidades bancarias están empleando la IA generativa para centralizar el conocimiento y capacitar a sus profesionales, a la vez que revolucionan las interfaces conversacionales. Esto permite ofrecer respuestas más naturales y manejar interacciones complejas, mejorando así la experiencia del cliente y la eficiencia operativa.
- **Sanidad**: Las instituciones sanitarias están aprovechando la IA generativa para resumir historiales clínicos extensos, optimizar el flujo de información entre profesionales y automatizar la generación de informes. Esta tecnología también contribuye a personalizar la atención al paciente y apoyar la investigación médica y el descubrimiento de fármacos, si bien persisten desafíos en cuanto a la fiabilidad de los datos y la precisión del modelo.
- **Manufactura**: En el sector manufacturero, los modelos generativos se emplean para acelerar el diseño de productos, optimizar los procesos de producción y reducir los desechos. Esto se traduce en una mejora significativa de la eficiencia y la sostenibilidad en la fabricación.

Sector educativo

El impacto de la IA generativa en el ámbito educativo ha sido objeto de un análisis significativo. En este sector, potencia la creación de experiencias de aprendizaje personalizadas que se adaptan a las necesidades individuales de cada estudiante, mejorando el compromiso y la eficacia. Asimismo, asiste a los docentes al automatizar tareas administrativas, desarrollar estrategias de enseñanza y generar materiales de curso en múltiples idiomas, permitiéndoles enfocar su tiempo en la interacción directa con los alumnos.

La figura 3 muestra cómo en la actualidad padres y profesores perciben la influencia de la IA en la educación y su papel en el futuro profesional de los estudiantes.

Figura 3. Percepción del impacto de la inteligencia artificial en la educación. Fuente: [36]

En cuanto al impacto de la IA en el futuro profesional de los estudiantes, ambos colectivos tienden a ser más optimistas. Aunque los padres mantienen una perspectiva muy favorable, los profesores también perciben un impacto positivo notable. No obstante, una parte significativa del profesorado aún manifiesta incertidumbre, lo que sugiere una mayor cautela o falta de información sobre este aspecto.

2.3. Generación de código con inteligencia artificial

La generación automática de código es una de las aplicaciones más relevantes de la inteligencia artificial generativa en el ámbito del desarrollo software. Esta tecnología, a la que se suele denominar ***vibe coding*** [37], se refiere a una metodología de desarrollo de software donde se utiliza la inteligencia artificial para generar, mejorar y depurar código a partir de instrucciones en lenguaje natural. Este proceso permite a los sistemas transformar descripciones en código funcional, lo que optimiza los procesos de programación y reduce los tiempos de desarrollo. Para comprender su funcionamiento, es fundamental analizar los modelos de lenguaje que hacen posible esta tarea.

2.3.1. Generación de código con LLMs

La aplicación de los modelos de lenguaje de gran escala en el ámbito de la programación ha supuesto un cambio profundo en la forma en que los desarrolladores diseñan, escriben y mantienen software. Gracias a su capacidad para comprender instrucciones en lenguaje natural y generar secuencias sintácticamente correctas, estas tecnologías han logrado integrarse en múltiples etapas del ciclo de vida del desarrollo [38].

Uno de los avances más destacados es el **autocompletado inteligente**, presente en herramientas como *GitHub Copilot*, que proporcionan sugerencias contextuales en tiempo

real dentro de entornos de desarrollo integrados (*Integrated Development Environments, IDEs*). Esta funcionalidad no solo acelera la escritura de código, sino que también reduce errores comunes y mejora la productividad.

Otra funcionalidad ampliamente utilizada es la **generación de código a partir de descripciones en lenguaje natural**. Basta con que el usuario escriba una instrucción o *prompt* describiendo lo que desea implementar, y el modelo es capaz de devolver la solución en el lenguaje de programación indicado. Por ejemplo, el usuario introduce "*crea una función en Python para sumar dos números y devuelve el resultado*", y la IA genera el código `def sumar(a, b): return a + b`. Esta característica resulta especialmente útil en fases de prototipado o cuando se requiere traducir ideas abstractas en soluciones funcionales rápidamente

La figura 4 muestra un ejemplo representativo de esta funcionalidad: a partir de un simple *prompt* en lenguaje natural, el sistema genera automáticamente una función en Python que responde con precisión a la instrucción dada.

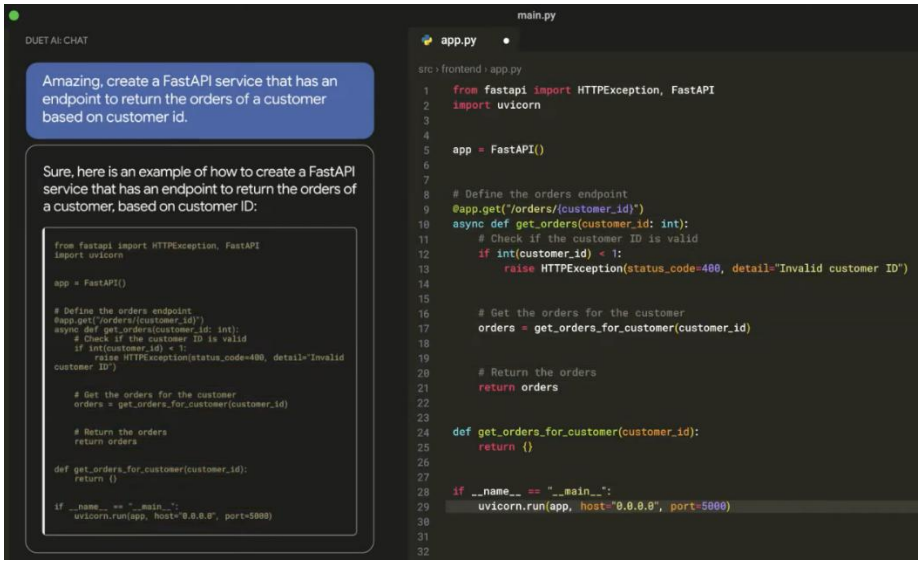


Figura 4. Ejemplo de generación de código a partir de un prompt. Fuente: [39]

Asimismo, estos sistemas también pueden llevar a cabo tareas de **refactorización y optimización**, identificando fragmentos de código ineficientes o redundantes y proponiendo alternativas más limpias, comprensibles o ajustadas a las mejores prácticas del lenguaje. Un ejemplo de esto puede ser que la IA analiza un fragmento de código JavaScript y sugiere reemplazar un bucle `for` complejo por un método `map` más conciso y legible.

En cuanto a la **detección y corrección de errores**, los modelos actúan como asistentes virtuales que analizan el código en busca de fallos lógicos o sintácticos, proponiendo correcciones fundamentadas. Herramientas como *Jules* [40], desarrollada por Google,

están orientadas precisamente a este tipo de soporte, proporcionando retroalimentación inmediata durante el proceso de depuración.

En su conjunto, los avances recientes han dado lugar a asistentes inteligentes capaces de acompañar al desarrollador desde la ideación hasta el mantenimiento del software. Estas herramientas además de aumentar la eficiencia también democratizan el acceso al desarrollo, permitiendo que usuarios sin conocimientos avanzados puedan interactuar con sistemas de programación mediante lenguaje natural.

Limitaciones de la generación de código con LLMs

Los LLMs han revolucionado la generación de código, pero presentan una serie de desafíos que los desarrolladores deben considerar antes de su implementación efectiva. Según Microsoft [41], los principales retos incluyen:

- **Limitación del conocimiento:** Debido al alto costo de entrenamiento, solo pueden generar código basado en la información disponible en su conjunto de datos de entrenamiento. No tienen acceso a información en tiempo real ni a datos privados.
- **Alucinaciones en la generación de código:** generan código basado en probabilidades estadísticas, lo que puede llevar a la producción de fragmentos incorrectos o no funcionales. Existen mecanismos para mejorar la precisión, sin embargo, no hay garantías de que el código generado sea completamente fiable.
- **Falta de transparencia en el razonamiento del modelo:** A diferencia de los sistemas tradicionales de programación, los LLMs no pueden explicar de manera clara por qué han generado un fragmento de código específico. Esto dificulta la depuración y validación de los resultados.
- **Ausencia de conocimiento especializado:** Si un LLM no ha sido entrenado en un dominio específico (por ejemplo, sistemas embebidos o ciberseguridad), no podrá generar código optimizado para esas áreas.

Técnicas para mejorar la generación de código

La mejora de la generación de código mediante LLMs requiere técnicas específicas que permitan adaptar el comportamiento del modelo, reducir errores y aumentar la calidad del código generado, lo cual se basa en gran medida en saber cómo formular los *prompts*.

En este sentido, **definir un rol explícito** [42] resulta muy eficaz. Indicar que, por ejemplo, "*actúa como ingeniero senior en Python*" o "*desarrollador experto en seguridad*" orienta la respuesta hacia un estilo técnico, preciso y adecuado al contexto. Esta técnica, conocida como *role prompting*, garantiza que el modelo adopte una perspectiva coherente con los requisitos del usuario, mejorando calidad y pertinencia.

En paralelo, **mantener los *prompts* breves y estructurados** [43] favorece la generación de código más claro y preciso. Cuando se limitan a menos de cincuenta palabras y se presentan de forma ordenada, se reduce considerablemente la aparición de respuestas inconsistentes o fragmentos de código inutilizable. Los estudios de la comunidad han encontrado mejoras de hasta un 40 % en la claridad y precisión al utilizar *prompts* más concisos.

Otra práctica muy útil consiste en **iterar sobre el propio código generado** [44]. Solicitar al modelo que revise, refactorice o corrija el fragmento previamente producido (por ejemplo: "*haz este código más limpio*" o "*corrige posibles errores*") permite pulir detalles lógicos, mejorar la estructura y elevar la calidad general del código. En foros técnicos, numerosos usuarios han constatado que múltiples rondas de autoevaluación conducen a soluciones más robustas.

Cuando se trata de corregir errores concretos, la técnica de ***lazy prompting*** [45] demuestra ser efectiva, especialmente en tareas de depuración. Consiste en copiar directamente los mensajes de error al modelo sin extensas explicaciones y permitir que el LLM infiera la intención y proponga soluciones. Esta aproximación funciona mejor cuando se itera rápidamente en interfaces web o de chat, como se indica en el análisis de Andrew Ng.

Finalmente, el ***meta-prompting*** [46] es una técnica avanzada que consiste en pedir al propio modelo que analice y optimice el *prompt* inicial. Por ejemplo: "*¿cómo reformularías esta instrucción para obtener un código más modular y eficiente?*". Al aprovechar la capacidad reflexiva del LLM, se consigue mejorar la claridad y efectividad del *prompt* original. Esta estrategia de revisión ha sido destacada en varias guías como una metodología eficaz para incrementar la calidad del código generado

En la figura 5 se muestra un ejemplo de cómo estructurar un *prompt* eficaz para la generación de código, desglosando elementos fundamentales como la descripción de la tarea, el contexto técnico, ejemplos de entrada y salida, y el historial de interacción.

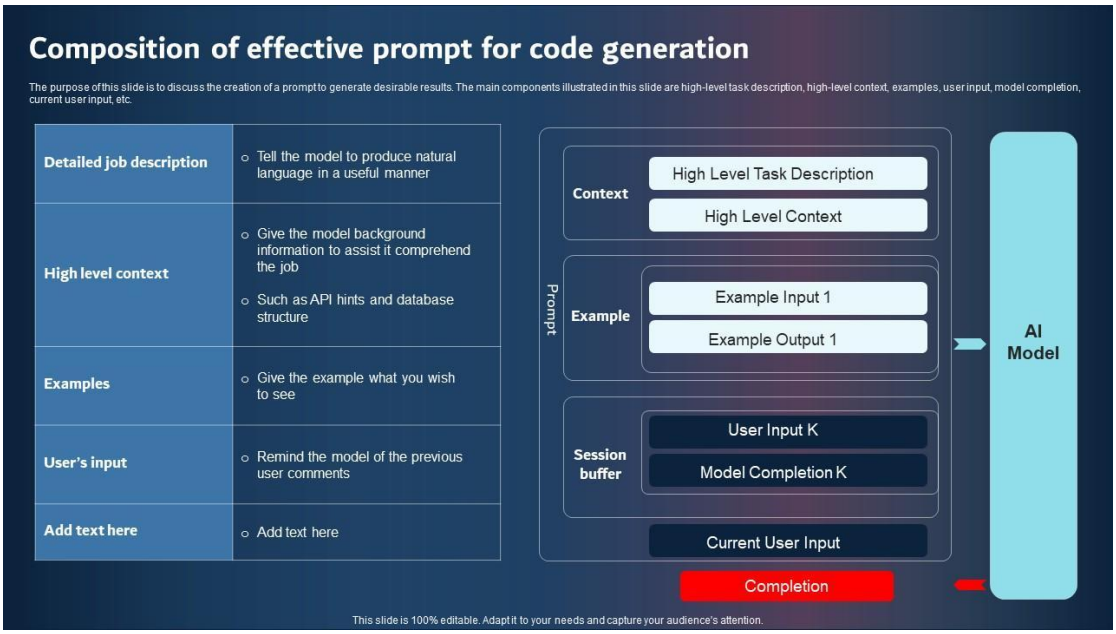


Figura 5. Componentes de un prompt eficaz para generación de código. Fuente: [47]

2.4. Inteligencia artificial generativa en la nube

La inteligencia artificial generativa en la nube [41] consiste en la implementación de modelos de IA a través de servidores remotos, accesibles mediante APIs o entornos web interactivos. Esta modalidad ha permitido el acceso a modelos avanzados sin la necesidad de contar con hardware especializado. Empresas tecnológicas han desarrollado plataformas que ofrecen modelos de IA generativa como servicio, permitiendo la generación de código, imágenes, texto y otros contenidos sin requerimientos de infraestructura local. Esta aproximación ha facilitado el desarrollo y despliegue de soluciones de IA a gran escala, optimizando costos y mejorando la accesibilidad para investigadores y desarrolladores.

Además, los modelos en la nube aprovechan arquitecturas de procesamiento distribuido, empleando unidades de procesamiento gráfico (*Graphics Processing Unit*, GPU) y unidades de procesamiento tensorial (*Tensor Processing Unit*, TPU) en centros de datos especializados, lo que permite ejecutar redes neuronales profundas con gran eficiencia. Para garantizar la seguridad y privacidad, estas plataformas implementan protocolos avanzados de protección de datos, fundamentales para industrias que manejan información sensible.

2.4.1. Modelos de ejecución en la nube

Dentro del ámbito de la inteligencia artificial generativa en la nube, los modelos de propósito general han evolucionado significativamente, ampliando sus capacidades más allá de la simple generación textual. En el contexto del desarrollo de software y la resolución de

problemas técnicos, destacan especialmente aquellos diseñados para realizar razonamientos complejos, ejecutar código, interpretar entradas multimodales o responder de forma eficiente en entornos productivos.

Estos permiten a los desarrolladores interactuar con sistemas capaces de comprender y procesar problemas complejos de manera estructurada. En esta categoría se encuentran **GPT-4.1** [48] y GPT-4o [49] (OpenAI), modelos de pago que destacan por su precisión en razonamiento lógico y su capacidad para seguir instrucciones complejas, lo que los convierten en una herramienta eficaz para tareas de desarrollo estructurado de software.

En esta misma línea se encuentra **DeepSeek R1 0528** [50] (mayo de 2025), un modelo gratuito accesible mediante la API de DeepSeek. Está diseñado con una arquitectura *Mixture of Experts* (MoE), la cual divide el modelo en subredes especializadas que abordan conjuntamente diferentes partes de los datos de entrada [51]. Gracias a este enfoque, DeepSeek R1 combina la eficiencia computacional con un rendimiento competitivo en la generación de código y en la resolución de problemas algorítmicos.

Por su parte, **Grok 3 mini Reasoning (*high*)** [52], desarrollado por xAI, representa una alternativa compacta y especializada en tareas de razonamiento estructurado, con respuestas concisas y precisas. Este modelo, accesible de forma gratuita, ha sido optimizado para reducir la latencia sin comprometer la calidad del razonamiento.

Entre los modelos que incorporan capacidades multimodales destacan las versiones **Gemini 2.5 Pro** y **Gemini 2.5 Flash** [53], desarrolladas por Google DeepMind y accesibles en la plataforma Vertex AI. Ambos modelos son de pago. Gemini 2.5 Pro está optimizado para la comprensión detallada de entradas complejas, lo que incluye texto, imagen y otros formatos sensoriales, y permite una mayor profundidad de razonamiento en entornos de desarrollo técnico. Por su parte, Gemini 2.5 Flash prioriza la agilidad en la generación de respuestas, lo que lo hace adecuado para tareas interactivas con requerimientos de baja latencia.

La familia **Claude 4** [54], de la empresa Anthropic, ha sido diseñada para entornos donde se combina razonamiento profundo con habilidades conversacionales. **Claude 4 Opus** y **Claude 4 Sonnet** son modelos de pago, integrados en plataformas como Claude.ai o Poe.com. Claude Opus destaca por su rendimiento en tareas que exigen razonamiento prolongado y manejo de contexto complejo, mientras que Claude Sonnet ofrece un equilibrio entre velocidad de respuesta y solidez en análisis técnico. Ambos modelos permiten ejecutar código y realizar explicaciones detalladas de procesos algorítmicos.

En el contexto de IA generativa orientada al desarrollo de código, resulta clave contar con modelos eficientes en coste y computacionalmente accesibles. En este segmento destaca **Qwen3 235B** [55], modelo de pago desarrollado por Alibaba Cloud y basado en arquitectura MoE, adecuado para tareas de análisis y generación de código multilingüe en entornos distribuidos.

Otro modelo representativo es **Mistral Medium** [56], también de pago, que combina capacidades multilingües con una ventana de contexto amplia. Su uso se ha extendido en entornos técnicos por su equilibrio entre coste operativo y razonamiento competente, especialmente en tareas con lógica de programación y documentación técnica.

Finalmente, **Llama 4 Maverick** [57], ofrecido como modelo de pago a través de NVIDIA cloud, ha sido diseñado para entornos productivos y se distingue por su estabilidad operativa en procesos prolongados de generación de código.

En la tabla 2 se presenta a modo de resumen los modelos de IAG en la nube mencionados anteriormente, clasificados según su enfoque y capacidades específicas.

Modelo	Descripción
GPT-4o / GPT-4.1 (OpenAI)	Interpretación multimodal de texto, imagen y audio. Elevada precisión en tareas de desarrollo, depuración y explicación de código estructurado.
Claude 4 (Opus / Sonnet / Sonnet Thinking) (Anthropic)	Interacción conversacional fluida combinada con razonamiento profundo. Soporta funciones como ejecución de código y análisis documental.
Gemini 2.5 Pro / Flash (Google DeepMind)	Capaces de integrar entradas textuales, visuales y auditivas. Incluyen un modo de reflexión previa (Deep Think) que mejora la coherencia contextual.
DeepSeek R1 0528 (DeepSeek)	Resolución de tareas estructuradas con uso optimizado de recursos. Basado en arquitectura *Mixture of Experts*.
Grok 3 mini Reasoning (high) (xAI)	Variante ligera de Grok con rendimiento lógico elevado. Respuestas rápidas con gestión eficiente del cómputo.
Qwen3 235B (Alibaba Cloud)	Modelo híbrido basado en MoE, ajustable según contexto. Razonamiento eficaz en consultas tanto simples como complejas.
Mistral Medium 3 (Mistral AI)	Preparado para entradas visuales y procesamiento de datos multilingües en escenarios corporativos con alta carga de información.
Llama 4 Maverick / Scout (Meta / NVIDIA)	Versiones industriales de Llama 4 orientadas a tareas complejas distribuidas y generación automatizada de código a gran escala.

Tabla 2. Resumen de modelos de IAG en la nube. Fuente: Elaboración propia

2.4.2. Ventajas y limitaciones de la IAG en la nube

El uso de plataformas de inteligencia artificial generativa en la nube ha ganado especial relevancia en los últimos años debido a su capacidad para ofrecer soluciones avanzadas sin necesidad de infraestructura local. Estas plataformas presentan múltiples ventajas, pero también ciertos desafíos que deben evaluarse cuidadosamente antes de su adopción [58].

Ventajas

Una de las principales ventajas de la IAG en la nube es su **escalabilidad**. Este tipo de plataformas permite ajustar dinámicamente los recursos computacionales en función de las necesidades de cada proyecto, sin requerir inversiones iniciales en hardware. Esta flexibilidad resulta especialmente útil para organizaciones que trabajan con cargas de trabajo variables o que desean experimentar con nuevas aplicaciones sin comprometer recursos a largo plazo.

En términos económicos, la nube permite operar bajo un modelo de pago por uso, lo que **reduce considerablemente los costes** asociados a la adquisición y mantenimiento de infraestructuras propias. Este enfoque hace que las tecnologías de IA sean más accesibles para pequeñas y medianas empresas, que de otro modo no podrían afrontar la inversión inicial que implican los entornos tradicionales.

Otra ventaja significativa es la **rapidez de despliegue**. Las plataformas en la nube permiten implementar modelos y servicios de inteligencia artificial de forma prácticamente inmediata, acelerando el desarrollo de soluciones y reduciendo el tiempo necesario para llevarlas al mercado. Esta agilidad es importante en entornos competitivos, donde las decisiones basadas en datos en tiempo real marcan una diferencia operativa.

Además, la **integración con otros servicios en la nube** favorece la colaboración entre equipos distribuidos geográficamente. Las plataformas permiten compartir recursos y coordinar esfuerzos de manera eficiente, lo que se traduce en una mejora de la productividad y en una mayor facilidad para desarrollar proyectos conjuntos en entornos híbridos o completamente remotos.

Limitaciones

A pesar de sus beneficios, la adopción de estas plataformas en la nube implica ciertos retos. Como hemos destacado anteriormente, uno de los más relevantes es la seguridad y privacidad de los datos. El hecho de almacenar información sensible en servidores externos puede suponer un riesgo si no se implementan adecuadamente mecanismos de protección y cumplimiento normativo, especialmente en sectores donde la confidencialidad es crítica.

La **dependencia de proveedores externos** también representa una limitación. El funcionamiento continuo de los sistemas depende sobre todo de la estabilidad, precios y condiciones del servicio ofrecido por terceros, lo que puede dificultar la migración entre plataformas o generar situaciones de dependencia tecnológica.

Otro aspecto a tener en cuenta es la **complejidad en la integración con infraestructuras** ya existentes. La adopción de soluciones en la nube puede requerir modificaciones en los sistemas actuales de una organización, lo que implica costes adicionales y posibles interrupciones operativas durante la fase de transición.

Por último, la **escasez de personal cualificado** en inteligencia artificial y aprendizaje automático puede limitar el aprovechamiento efectivo de estas tecnologías. La puesta en marcha de soluciones basadas en IAG requiere conocimientos técnicos específicos que no siempre están disponibles internamente, lo que obliga a recurrir a formación especializada o contratación externa.

2.5. Inteligencia artificial generativa en entornos locales

La inteligencia artificial generativa en entornos locales se refiere a la ejecución de modelos de IAG directamente en el hardware del usuario (por ejemplo, en un ordenador personal o servidor), en lugar de recurrir a servidores en la nube. Esto implica que los modelos se despliegan y procesan localmente, aprovechando la capacidad de los componentes clave del sistema. Por un lado, la **CPU** (*Central Processing Unit*) se encarga de las tareas generales del sistema. Por otro, la GPU es fundamental para la IAG, ya que su arquitectura está diseñada para realizar los cálculos paralelos masivos que estos modelos requieren [59].

El rendimiento de la ejecución local depende de la interacción entre estos procesadores y la memoria disponible: la **RAM** (*Random Access Memory*), que es la memoria principal del sistema, y la **VRAM** (*Video RAM*), que es la memoria dedicada y de alta velocidad integrada en la propia GPU [60]. A diferencia de la IAG en la nube, donde los datos deben enviarse a servicios remotos, la IAG local permite mantener los datos y el procesamiento *in situ*, sin necesidad de conexión externa.

Este enfoque ha ganado relevancia en los últimos años gracias a la liberación de modelos de lenguaje de código abierto altamente eficientes y a los avances en software de optimización que permiten ejecutar estos modelos en hardware de consumo. De este modo, incluso desarrolladores individuales o instituciones con recursos limitados pueden acceder a modelos avanzados sin depender de un proveedor remoto.

2.5.1. Modelos activos en local para generación de código, razonamiento y asistencia técnica

Todos los modelos evaluados en esta sección son gratuitos y de código abierto, disponibles para descarga directa o ejecución a través de herramientas como Ollama, LM Studio o repositorios especializados. Esta disponibilidad permite a desarrolladores y organizaciones experimentar con soluciones avanzadas de inteligencia artificial generativa sin necesidad de licencias comerciales ni acceso a plataformas en la nube.

El crecimiento de los modelos de IA ejecutables localmente ha permitido a investigadores y desarrolladores contar con potentes alternativas sin depender de la infraestructura cloud. Dicha capacidad de ejecución en hardware personal garantiza control total sobre los datos y puede funcionar *offline*, lo que resulta atractivo por motivos de privacidad y seguridad [13].

Uno de los referentes actuales en este ámbito es **DeepSeek Coder V2** [61], un modelo de código abierto con 16 mil millones (16B) de parámetros. El número de parámetros de un modelo, que pueden entenderse como las variables internas que la red neuronal ajusta durante el entrenamiento, es un indicador clave de su tamaño y capacidad potencial. Este modelo está entrenado con grandes corpus de código fuente y emplea una arquitectura basada en transformers optimizados para la eficiencia local. Destaca por su rendimiento competitivo frente a soluciones comerciales de alta gama como GPT-4 Turbo.

En la misma línea, **Qwen2.5 Coder 32B** [62], un modelo de 32 mil millones (32B) de parámetros desarrollado por Alibaba, ha sido entrenado exclusivamente con datos de programación. Este modelo incorpora capacidades avanzadas de comprensión semántica de código, documentación y razonamiento algorítmico, situándose como una opción robusta para entornos técnicos exigentes.

Modelos como **Mistral 7B** [63], con 7 mil millones (7B) de parámetros y desarrollado por Mistral AI, han ganado popularidad por su equilibrio entre tamaño y rendimiento. Diseñado para tareas de inferencia eficiente en local, ha demostrado un comportamiento competente en generación de código estructurado y respuestas coherentes ante instrucciones técnicas complejas.

Otro exponente representativo es **LLaMA 3 8B** [64], un modelo de 8 mil millones (8B) de parámetros de Meta. Es gratuito, con licencia abierta y está adaptado a tareas de programación y razonamiento conversacional técnico. Su ejecución es viable tanto en CPU como en GPU de gama media, lo que facilita su adopción por parte de desarrolladores independientes.

La familia **Phi-3** [65], desarrollada por Microsoft, ofrece versiones livianas diseñadas para ejecutarse eficientemente en entornos locales. **Phi-3 Medium**, con 14 mil millones (14B) de parámetros, proporciona capacidades suficientes para tareas de análisis de código y estructuras algorítmicas, mientras que **Phi-3 Mini**, con 3.8 mil millones (3.8B) de parámetros, se plantea como una opción base para dispositivos con capacidad limitada, manteniendo una razonable competencia técnica para tareas básicas.

Reka Flash 3 [66], un modelo de 21 mil millones (21B) de parámetros lanzado en septiembre de 2024 por Reka AI, presenta un enfoque orientado a reducir la latencia sin sacrificar precisión. Está diseñado para funcionar en estaciones de trabajo avanzadas, pero su eficiencia lo hace viable para entornos locales exigentes con conectividad limitada.

En paralelo, **Qwen3 14B** (*Reasoning*) [67], con 14 mil millones (14B) de parámetros, representa la evolución más reciente de la familia Qwen, con énfasis en tareas de razonamiento general y solución de problemas complejos. Este modelo destaca por su arquitectura equilibrada y alto rendimiento en *benchmarks* de razonamiento lógico.

Una alternativa flexible es **Jamba 1.6 Large** [68], creado por AI21 Labs. Este modelo conversacional presenta una innovadora arquitectura de MoE con un total de 398 mil millones (398B) de parámetros, de los cuales solo una fracción se activa durante la inferencia, optimizando su eficiencia. Su diseño lo hace adecuado para integraciones con interfaces conversacionales locales y puede ser cuantizado para reducir los requisitos computacionales sin degradar notablemente la calidad.

Finalmente, **Gemma 3 4B** [69], un modelo ligero de 4 mil millones (4B) de parámetros de Google, se posiciona como una de las propuestas más eficientes dentro de los modelos de código abierto recientes. Su diseño facilita tareas de inferencia estructurada y análisis básico en entornos técnicos, destacando por su rápida implementación.

Para ofrecer una visión comparativa y consolidada, la tabla 3 resume las características de cada modelo descrito, junto a las especificaciones de hardware recomendadas para su ejecución fluida. Estas estimaciones asumen el uso de versiones cuantizadas a 4 bits (formato GGUF), que es el estándar para el despliegue en hardware de consumo [70].

Modelo	Parámetros	Descripción	RAM del sistema recomendada	VRAM (GPU) recomendada
DeepSeek Coder V2	16B	Entrenado con grandes corpus de código, ofrece rendimiento competitivo y eficiencia operativa en tareas técnicas complejas.	32 GB	> 20 GB
Qwen2.5 Coder 32B	32B	Modelo focalizado en programación, entrenado exclusivamente con corpus técnico para análisis, documentación y generación estructurada.	32 GB	> 20 GB
Reka Flash 3	21B	Enfocado en rapidez de inferencia sin comprometer la precisión técnica. Compatible con ejecución offline eficiente.	32 GB	> 16 GB
Jamba 1.6 Large	398B	Diseñado para interacción técnica prolongada, con soporte para contexto extendido y generación asistida por recuperación de información (RAG).	64 GB	> 32 GB
Qwen3 14B (Reasoning)	14B	Potente en tareas de razonamiento lógico general, con buena gestión del contexto en conversaciones técnicas.	16 GB	> 10 GB
Phi-3 Medium	14B	Versiones diseñadas para portátiles o hardware limitado. Mantienen funcionalidad en codificación básica y análisis algorítmico.	16 GB	> 10 GB
LLaMA 3 8B	8B	Modelo abierto de Meta orientado a programación y razonamiento conversacional, adaptable y eficiente para ejecución local.	16 GB	> 8 GB
Mistral 7B	7B	Equilibrio entre tamaño y rendimiento. Ideal para tareas de codificación estructurada con instrucciones técnicas.	16 GB	> 8 GB
Phi-3 Mini	3.8B	Versiones diseñadas para portátiles o hardware limitado. Mantienen funcionalidad en codificación básica y análisis algorítmico.	8 GB	> 4 GB
Gemma 3 4B	4B	Modelo eficiente y de fácil implementación, adecuado para tareas básicas de asistencia técnica y generación estructurada de código.	8 GB	> 4 GB

Tabla 3. Resumen de modelos de IAG en local. Fuente: Elaboración propia.

2.5.2. Ventajas y limitaciones de la IAG en local

El uso de modelos generativos en entornos locales ofrece una alternativa real y cada vez más viable frente a los servicios en la nube. Esta opción presenta una serie de beneficios técnicos y operativos, especialmente en lo que respecta al control sobre los datos, los costes de operación y la independencia tecnológica. No obstante, también conlleva ciertos desafíos relacionados con la infraestructura y el mantenimiento que deben ser evaluados [71].

Ventajas

Una de las principales ventajas de la IAG en local es la **protección de la privacidad y la integridad de los datos**. Al ejecutarse los modelos directamente en el dispositivo o servidor del usuario, la información sensible no necesita salir del entorno controlado, lo que elimina la dependencia de terceros para el procesamiento. Esto permite cumplir con normativas específicas de protección de datos y aplicar medidas de seguridad de forma personalizada. Además, se dispone de un control total sobre la configuración del hardware y del modelo, pudiendo ajustar el rendimiento o implementar medidas de seguridad según las necesidades del entorno.

Desde una perspectiva económica, operar localmente puede implicar **costes reducidos a largo plazo**, especialmente cuando ya se cuenta con el hardware adecuado. En este escenario no se incurre en tarifas variables por número de solicitudes ni en el pago continuo por acceso a servicios o APIs, como ocurre en la nube. En su lugar, el gasto se limita a la inversión inicial en infraestructura y al consumo energético, lo que puede resultar favorable en usos intensivos o sostenidos.

Otra ventaja importante es la **baja latencia**. Al encontrarse el modelo en el mismo equipo o red interna, la comunicación se realiza sin necesidad de conexión externa, lo que reduce drásticamente los tiempos de respuesta. Esto resulta útil en aplicaciones que requieren inmediatez, como asistentes de codificación dentro de un entorno de desarrollo o sistemas conversacionales integrados, donde una respuesta fluida mejora significativamente la experiencia de uso.

La **posibilidad de operar sin conexión** es también un punto a favor. Una vez descargado el modelo, su ejecución no requiere acceso a Internet, permitiendo su uso en entornos aislados como ubicaciones remotas, laboratorios desconectados o contextos con conectividad limitada. Además, el usuario puede adaptar el modelo a sus propias necesidades, modificando parámetros de inferencia, ajustando la compatibilidad con el

hardware disponible o aplicando optimizaciones específicas que no serían posibles en soluciones en la nube.

Limitaciones

A pesar de las grandes ventajas que posee, también encontramos una serie de limitaciones, entre las que se encuentra principalmente la **necesidad de contar con recursos computacionales** adecuados. Los modelos generativos más avanzados, especialmente aquellos de gran tamaño, requieren cantidades elevadas de memoria RAM o VRAM, así como capacidad de procesamiento que generalmente solo pueden proporcionar GPUs de gama alta. Esto supone una inversión inicial considerable en hardware, que puede resultar inaccesible para determinados usuarios o quedar obsoleta rápidamente debido a la evolución acelerada de los modelos.

A este aspecto se suma la **complejidad técnica** que implica la implementación y el mantenimiento de la solución local. El usuario debe encargarse de instalar dependencias, configurar entornos, gestionar versiones de modelos, resolver incompatibilidades y aplicar actualizaciones de seguridad. Estas tareas requieren conocimientos técnicos especializados y tiempo de dedicación. Además, la escalabilidad de una solución local está limitada por el hardware disponible, lo que dificulta su adaptación a un mayor número de usuarios o a cargas de trabajo crecientes, en contraste con la elasticidad automática de los servicios en la nube.

Del mismo modo, un entorno local **carece de la robustez y redundancia** propias de los grandes proveedores de servicios. Ante fallos de hardware o errores de software, no existen mecanismos automáticos de recuperación, lo que puede interrumpir completamente el servicio hasta que se intervenga manualmente. Asimismo, el acceso a modelos de última generación (como GPT-4) suele estar restringido a través de APIs comerciales, lo que limita la disponibilidad inmediata de las versiones más recientes en entornos locales. Aunque la comunidad *open-source* avanza rápidamente, es posible que exista un desfase en cuanto a rendimiento o funcionalidades respecto a las soluciones comerciales más punteras.

Esta revisión de herramientas, plataformas y modelos generativos tanto en entornos locales como en la nube proporciona una base para comprender las posibilidades de despliegue de inteligencia artificial generativa. Estas soluciones varían en cuanto a eficiencia, requisitos de hardware, control sobre los datos y facilidad de integración. En el capítulo 3 se abordará la evaluación práctica de estas tecnologías, comparando su rendimiento en distintos contextos con el fin de identificar la alternativa más adecuada para el desarrollo del sistema propuesto en este proyecto

2.6. Introducción a la plataforma Moodle

Moodle (*Modular Object-Oriented Dynamic Learning Environment*) es un LMS de código abierto que permite desarrollar entornos formativos virtuales estructurados, escalables y flexibles. Los LMS son plataformas digitales orientadas a gestionar, distribuir y evaluar experiencias educativas tanto en modalidad presencial como en línea o mixta [72]. Moodle incorpora una arquitectura modular que facilita su extensión y adaptación a distintos escenarios docentes. Esta estructura ha contribuido a su consolidación como una de las plataformas de referencia en el ámbito universitario.

2.6.1. Estructura general y navegación

Cada curso en Moodle constituye una unidad autónoma en la que se integran los materiales, actividades y usuarios asociados a una asignatura concreta. Su estructura se organiza en secciones que pueden adoptar una disposición cronológica o temática. Dentro de estas secciones, el profesorado inserta los distintos recursos y actividades que conforman la experiencia de aprendizaje.

La interfaz se divide en dos zonas principales:

- Un panel lateral, con enlaces a las funcionalidades administrativas (participantes, calificaciones, informes, etc.).
- Una área central que muestra la secuencia de secciones del curso y sus contenidos.

Tal y como se muestra en la figura 6, las secciones pueden contraerse y expandirse de forma independiente, lo que favorece una navegación ágil y centrada en el contenido pertinente.

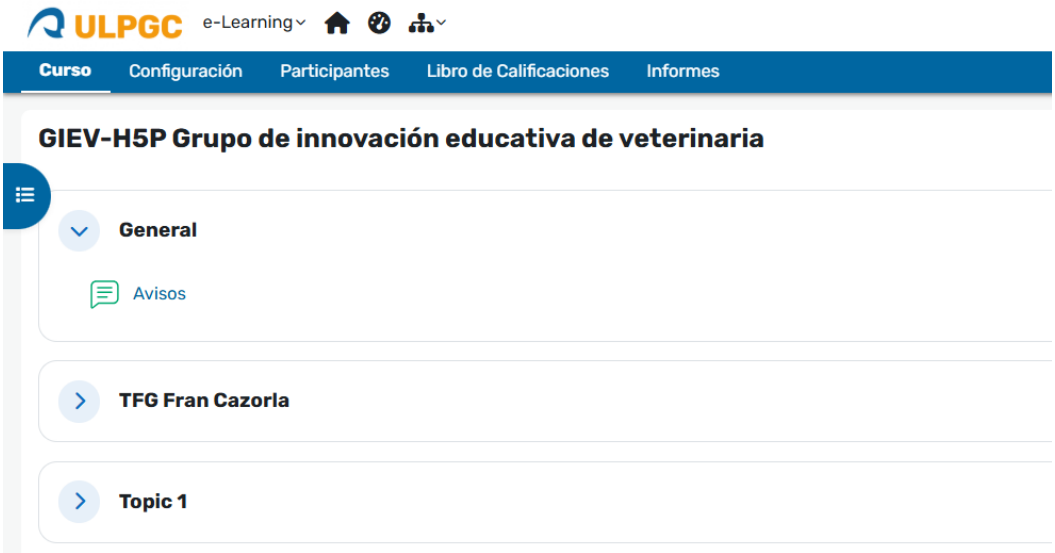


Figura 6. Vista general de un curso en Moodle - ULPGC.

La navegación es adaptable al dispositivo y al rol del usuario. Moodle ofrece diferentes vistas para el alumnado, el profesorado con o sin permiso de edición y perfiles administrativos, así como una interfaz optimizada para dispositivos móviles [73]. Esta versatilidad garantiza una experiencia de usuario coherente y accesible en diversos contextos de uso.

2.6.2. Roles y control de permisos

El sistema de permisos de Moodle se basa en la asignación de roles que determinan las acciones permitidas dentro de un curso. Cada rol agrupa un conjunto de capacidades, que pueden ampliarse o restringirse de forma granular [74]. Esta arquitectura de control permite definir con precisión los niveles de intervención y participación de cada tipo de usuario.

Los roles predeterminados incluyen:

- **Administrador**: acceso completo a la configuración del sitio.
- **Gestor**: administración de cursos y usuarios sin modificar el sistema.
- **Profesor**: edición de contenidos, calificación y seguimiento del alumnado.
- **Profesor sin edición**: evaluación sin capacidad de modificar el contenido.
- **Estudiante**: acceso a recursos y actividades.
- **Invitado**: visualización limitada sin interacción activa.

En entornos como el de la ULPGC es común encontrar variaciones de estos perfiles, como *Docente*, *Estudiante Demo* o *Invitado evaluable*, adaptados a las necesidades institucionales. En la figura 7 se visualiza un selector de rol con estas personalizaciones.

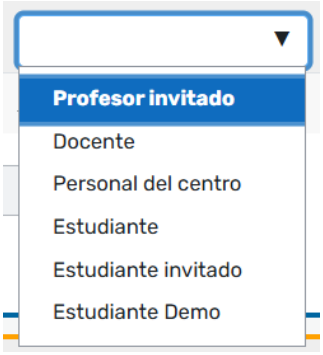


Figura 7. Selector de roles institucionales en Moodle - ULPGC.

La asignación y modificación de roles se realiza desde el apartado de "Participantes" del curso. Desde este menú también es posible establecer restricciones de acceso basadas

en fechas, grupos, criterios de evaluación o condiciones de finalización [75]. Esta flexibilidad facilita el diseño de experiencias docentes más controladas y personalizadas.

2.6.3. Inserción de recursos y actividades

Moodle permite insertar contenidos estructurados mediante un sistema de bloques, diferenciando entre recursos (información unidireccional) y actividades (interacción con el estudiante).

El acceso al menú de inserción se habilita al activar el modo de edición, accesible únicamente por los usuarios con el permiso para ello. En ese momento aparece el botón "Añadir una actividad o un recurso". Al hacer clic, se despliega un panel que presenta los elementos disponibles clasificados por tipo, como se muestra en la figura 8. Entre los **recursos** más habituales se encuentran:

- **Archivo** / **Carpeta**: subida de documentos y organización de materiales.
- **Página**: contenido textual con formato enriquecido.
- **H5P (*HTML5 Package*):** Facilita la integración de contenido interactivo y multimedia de forma estandarizada.
- **Etiqueta**: bloques informativos insertados directamente en la sección.
- **URL (*Uniform Resource Locator*)**: enlaces externos integrados.

Y entre las **actividades** destacadas:

- **Tarea**: envío de archivos o textos para ser evaluados.
- **Cuestionario**: pruebas automáticas con retroalimentación.
- **Foro**: espacios de debate asincrónico.
- **Taller**: evaluación entre iguales con fases definidas.

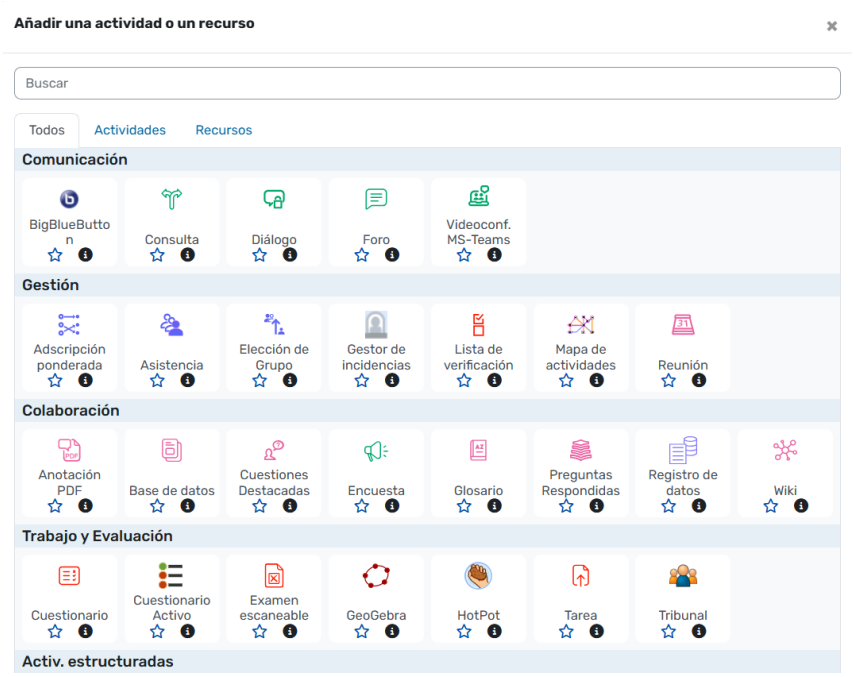


Figura 8. Menú de selección de recursos y actividades Moodle - ULPGC.

Cada elemento cuenta con configuraciones específicas relativas a su visibilidad, calificación, restricciones de acceso o condiciones para su consideración como "completado". Esto permite un diseño instruccional flexible y coherente con la progresión esperada del curso.

En lugar de modificar el núcleo de Moodle o desarrollar un *plugin* complejo, este proyecto adopta una estrategia de integración no invasiva que se apoya fundamentalmente en tres recursos nativos de la plataforma: ***Página***, ***Tarea*** y ***H5P***. La elección de estos elementos responde directamente a las limitaciones de Moodle y a los objetivos de portabilidad y facilidad de uso del proyecto.

La principal barrera para la innovación docente en Moodle es la imposibilidad de que un profesor cree nuevos tipos de actividades o instale librerías sin permisos de administrador. Para superar este obstáculo, el recurso ***Página*** se convierte en el pilar de la solución. Una Página actúa como un contenedor que permite la inserción de código HTML, CSS y JavaScript, posibilitando el despliegue de una aplicación web completa que se ejecuta íntegramente en el navegador del cliente. De este modo, la herramienta de anotación de este TFG se implementa como una aplicación cliente-ligera contenida dentro de este recurso, sin requerir intervención administrativa.

Complementariamente, la actividad ***Tarea*** proporciona la infraestructura estándar para la entrega de los informes generados por la herramienta. La solución interactúa

programáticamente con la interfaz de la *Tarea* para automatizar la subida de los archivos ZIP, integrándose de forma fluida en el flujo de evaluación del curso.

Finalmente, el formato ***H5P*** se utiliza como el estándar de salida para crear los vídeos interactivos. La herramienta genera un paquete `.h5p` estandarizado que el profesor puede subir utilizando el recurso *H5P* nativo ya existente, asegurando la máxima compatibilidad y reutilización del contenido.

Esta estrategia de aprovechamiento de recursos existentes es fundamental para el proyecto, ya que garantiza que la solución sea fácilmente portable, escalable y desplegable por cualquier docente con permisos de edición, sin comprometer la seguridad ni la estabilidad de la plataforma Moodle.

2.6.4. Tecnologías web utilizadas en el editor HTML de Moodle

Los recursos de Moodle incluyen un editor enriquecido que ofrece una vista en código fuente. A través de esta funcionalidad, es posible integrar directamente tecnologías web como HTML5, JavaScript y CSS, permitiendo ampliar las capacidades del recurso sin necesidad de instalar *plugins* ni modificar el núcleo de la plataforma.

HTML5 (*HyperText Markup Language 5*) [76] es la quinta versión del lenguaje estándar de marcado utilizado para estructurar contenido en la web. Introduce etiquetas semánticas que mejoran la accesibilidad y el posicionamiento, y permite incrustar directamente contenido multimedia como audio y vídeo sin necesidad de complementos externos. Además, admite elementos interactivos como formularios dinámicos y gráficos vectoriales, facilitando la creación de experiencias web ricas y adaptadas a distintos dispositivos.

JavaScript (JS) [77] es un lenguaje de programación interpretado orientado a objetos, diseñado para ejecutarse en el navegador del cliente. Se utiliza para desarrollar comportamientos dinámicos en las páginas web, como respuestas a eventos del usuario, validación de formularios, actualización de contenidos sin recargar la página o interacción con APIs externas. Su integración permite transformar páginas estáticas en aplicaciones web interactivas.

CSS (*Cascading Style Sheets*) [78] es el lenguaje que define la presentación visual del contenido HTML. Permite establecer estilos tipográficos, colores, márgenes, disposición de elementos y reglas de diseño responsivo que adaptan la interfaz a diferentes tamaños de pantalla. La separación entre estructura y presentación facilita la mantenibilidad del código y mejora la experiencia de usuario.

Estas tecnologías, combinadas, ofrecen una base sólida para desarrollar funcionalidades enriquecidas directamente desde el editor del recurso, como se muestra en la figura 9.

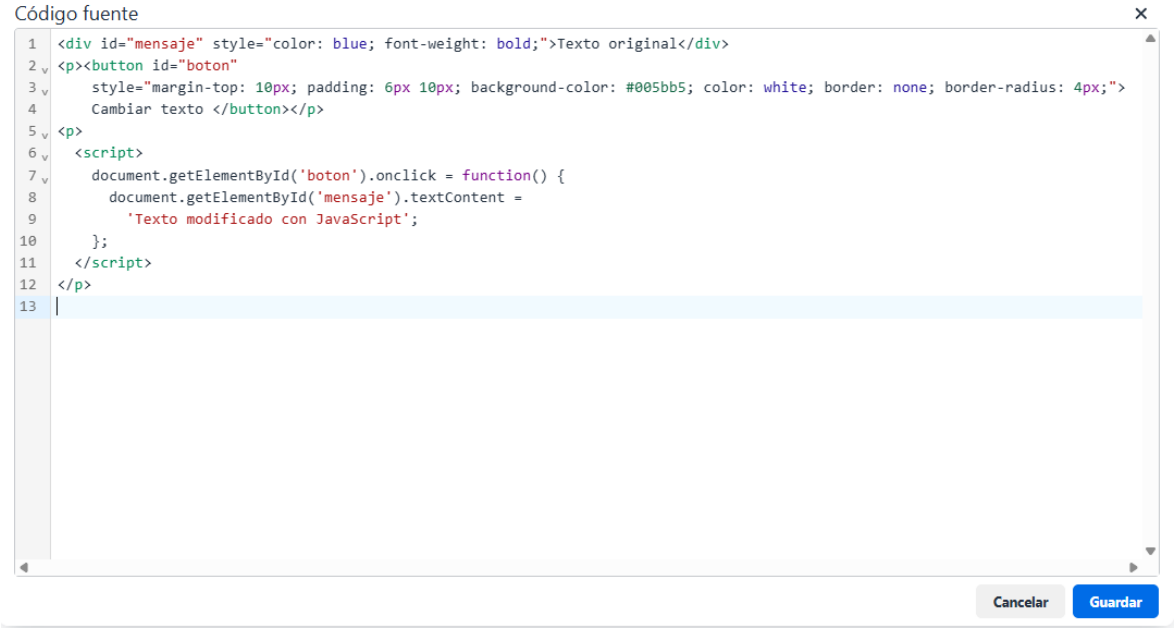


Figura 9. Editor de código fuente de un recurso tipo Página.

Capítulo 3: Análisis Comparativo de Modelos de IA Generativa

En este capítulo se realiza un análisis comparativo de diversos modelos de inteligencia artificial generativa, tanto en la nube como en ejecución local. Se establecen los criterios de evaluación, se presentan los resultados obtenidos en benchmarks estandarizados y se lleva a cabo una valoración final para seleccionar el modelo más adecuado para el desarrollo de la herramienta propuesta.

3.1. Criterios de evaluación y análisis comparativo de modelos

Para realizar el análisis comparativo entre los modelos de IAG ejecutados en la nube y en local, es necesario establecer un conjunto de criterios que sirvan como marco de referencia para evaluar su utilidad en el contexto del presente Trabajo de Fin de Grado. En este caso, el objetivo es identificar qué modelos resultan más adecuados para tareas de generación de código, especialmente aplicadas al desarrollo de la herramienta propuesta sobre la interfaz de usuario de Moodle, considerando tanto aspectos técnicos como prácticos.

Dado que el proyecto se realiza en un entorno académico y que la calidad del código generado es un factor determinante, se ha definido una serie de criterios que abarcan desde la capacidad de razonamiento del modelo hasta su eficiencia operativa, el coste de uso o la velocidad de respuesta. El objetivo es garantizar una evaluación objetiva, reproducible y alineada con las condiciones reales en las que se desarrollará la solución propuesta.

Los modelos seleccionados para este análisis han sido presentados previamente en el capítulo 2, optando por mantener la misma selección con el fin de asegurar coherencia metodológica y continuidad entre el marco conceptual y la fase evaluativa. Esta decisión responde tanto a razones de representatividad como de viabilidad práctica: en el caso de los modelos en la nube, se han incluido alternativas de última generación con alto rendimiento demostrado en tareas de codificación, junto con otras opciones gratuitas o de acceso abierto que permiten su integración en proyectos educativos sin barreras económicas ni licencias restrictivas.

En cuanto a los modelos locales, la selección se ha fundamentado en criterios de accesibilidad técnica. Se han priorizado modelos disponibles públicamente que puedan ejecutarse de forma eficiente en entornos sin aceleración por hardware especializado, facilitando así su uso en equipos personales o institucionales de gama media. Esta aproximación persigue la replicabilidad del proyecto en escenarios reales, en línea con los principios de sostenibilidad y democratización tecnológica que fundamentan el trabajo.

La variedad de arquitecturas, tamaños y condiciones de uso presente en la muestra permite comparar con rigor el comportamiento de los modelos bajo diferentes métricas, extrayendo conclusiones aplicables tanto al desarrollo actual como a posibles extensiones futuras en contextos educativos.

3.1.1. Nivel de inteligencia

En este análisis, se entiende el **nivel de inteligencia** de un modelo de IA generativa como su capacidad para razonar a partir de una instrucción, interpretarla en contexto y generar soluciones útiles en forma de código. Este aspecto es fundamental en el presente trabajo, centrado en la generación automatizada de fragmentos de código funcionales y adaptables. Para evaluarlo, se han utilizado métricas obtenidas de benchmarks estandarizados que simulan situaciones reales de programación y resolución de problemas.

La métrica empleada ha sido el *Artificial Analysis Intelligence Index*, un índice compuesto publicado por la plataforma Artificial Analysis [79], que agrupa resultados de varios bancos de pruebas especializados:

- **MMLU-Pro (*Multi-Task Language Understanding Benchmark, Pro version*)** [80]**:** Test de opción múltiple (10 opciones) que mide conocimientos avanzados en 12 áreas, como matemáticas, derecho o informática. Basado en una versión ampliada del MMLU original (*Hendrycks et al.*), con 12.032 preguntas. Las respuestas se extraen mediante expresiones regulares.
- **HLE (*Humanity's Last Exam*)** [81]**:** Evaluación avanzada creada por el *Centre for AI Safety*, con 2.684 preguntas difíciles en matemáticas, humanidades y ciencias. Elaborada de forma adversarial con modelos de última generación (GPT-4o, Gemini 1.5, Claude 3.5, etc.). Las respuestas se validan con un modelo verificador (GPT-4o).
- **GPQA Diamond** [82]**:** Subconjunto del benchmark GPQA con 198 preguntas complejas de biología, física y química. Formato de opción múltiple (4 opciones), con extracción de respuestas por regex.
- **MATH-500** [83]**:** Conjunto de 500 preguntas del *dataset* MATH original (*Hendrycks et al.*, 2021), seleccionadas al azar tras excluir las usadas en entrenamiento. Evaluación en dos fases: verificación con *SymPy* (PRM800K) y revisión con un modelo LLM (Llama 3.3 70B).
- **AIME 2024** [84]**:** Problemas matemáticos de la edición 2024 del AIME. Soluciones entre 1 y 999. Se utiliza el mismo sistema de evaluación que en MATH-500.
- ***HumanEval*** [85]**:** Tareas de programación en Python evaluadas con pruebas unitarias. Cada una consiste en generar una función. Se aplica la métrica *pass@1*, con cinco ejecuciones.
- ***SciCode*** [86]**:** *Benchmark* de código científico en Python, con contexto anotado por expertos. Evaluación por subproblema con *pass@1*.

- ***LiveCodeBench*** [87]**:** Conjunto de 315 problemas extraídos de diversas plataformas, publicados entre julio de 2024 y enero de 2025. Evaluación basada en *pass@1*, que indica el porcentaje de problemas resueltos correctamente en el primer intento del modelo. Por ejemplo, si de 100 problemas resuelve 72 a la primera, su *pass@1* sería del 72 %.

Para este estudio, se han priorizado los benchmarks que permiten medir con mayor precisión la inteligencia computacional en tareas asociadas al desarrollo de software, abarcando tanto la generación como la comprensión de código en lenguaje Python.

En el caso de ***HumanEval***, los modelos reciben instrucciones explícitas para implementar una función completa, dada su firma y descripción.

En ***SciCode***, se plantean tareas de computación científica estructuradas en pasos secuenciales. El modelo recibe la descripción del problema, una serie de pasos previos y el paso que debe resolver. Se exige que introduzca primero el conocimiento científico necesario en forma de comentario, seguido de la implementación en código. Se emplea el *prompt* original propuesto por *Tian et al.*, sin modificaciones, incluido en el anexo A2.2. Las directrices prohíben expresamente incluir código de pasos anteriores, ejemplos o pruebas, y obligan a ceñirse a las dependencias indicadas para cada tarea.

Para ***LiveCodeBench***, se ha utilizado el *prompt* base del benchmark, sin introducir *system prompts* personalizados. La naturaleza del enunciado varía según se proporcione o no una plantilla de código inicial. En los problemas con plantilla, el modelo debe completar el bloque dentro de delimitadores Python. En los casos sin plantilla, debe generar un programa autónomo que lea desde *stdin*, procese la información y escriba el resultado por *stdout*. La extracción del código generado se realiza mediante una expresión regular que captura exclusivamente el contenido dentro del bloque de código en formato Markdown. El *prompt* completo se incluye en el anexo A2.3.

Este conjunto de instrucciones y mecanismos de validación garantiza que las tareas de generación de código se evalúen de forma estructurada, homogénea y orientada a casos de uso reales, lo que permite comparar de forma fiable el rendimiento de los modelos en diversos contextos de programación.

HumanEval

En las figuras 10 y 11 se muestra el rendimiento de los modelos evaluados en el benchmark ***HumanEval*** mediante la métrica *pass@1*.

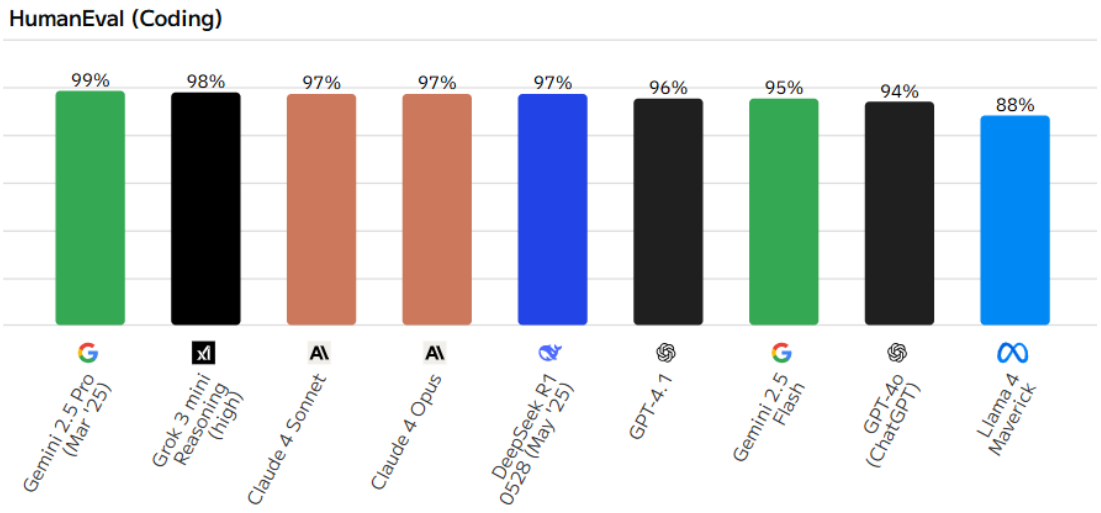


Figura 10. Evaluación de modelos en la nube de HumanEval (pass@1). Fuente: [88]

Estos resultados evidencian una clara ventaja de los modelos en la nube en generación de código funcional. Destacan Gemini 2.5 Pro (99 %) y Grok 3 mini Reasoning (98 %) como los más precisos, mientras que LLaMA 4 Maverick (88 %) representa el valor más bajo dentro del entorno cloud, aunque aún en un rango funcional aceptable.

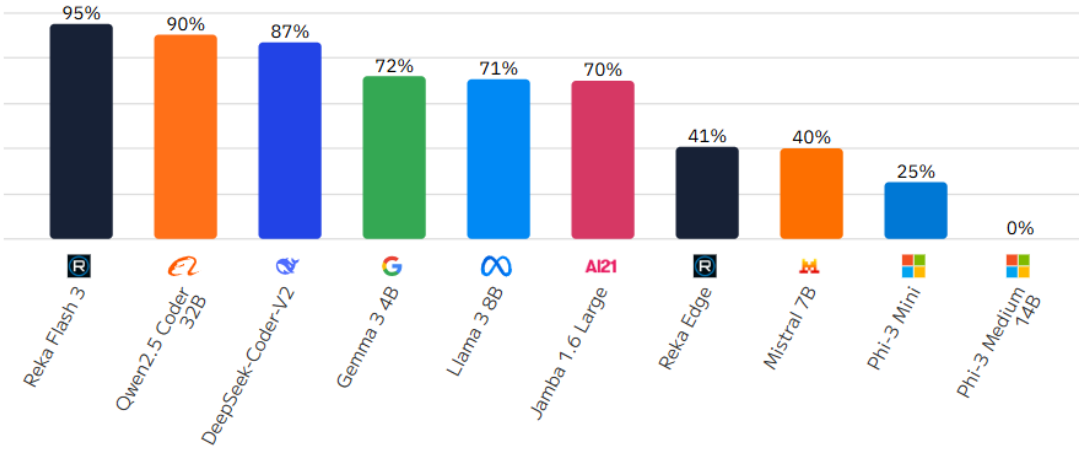


Figura 11. Evaluación de modelos en local de HumanEval (pass@1). Fuente: [88]

En ejecución local, Reka Flash 3 (95 %) se posiciona como el modelo más competente, seguido de Qwen2.5 Coder 32B (90 %) y DeepSeek Coder V2 (87 %), todos ellos con rendimientos comparables a los mejores modelos cloud. En contraste, Phi-3 Medium 14B (0 %) y Phi-3 Mini (25 %) registran los valores más bajos del entorno local, mostrando claras limitaciones para tareas que requieren precisión en la primera inferencia.

SciCode

Por otro lado, en las figuras 12 y 13 se representan los resultados obtenidos por los modelos en el benchmark ***SciCode***, orientado a la resolución de problemas científicos en lenguaje Python con contexto técnico predefinido.

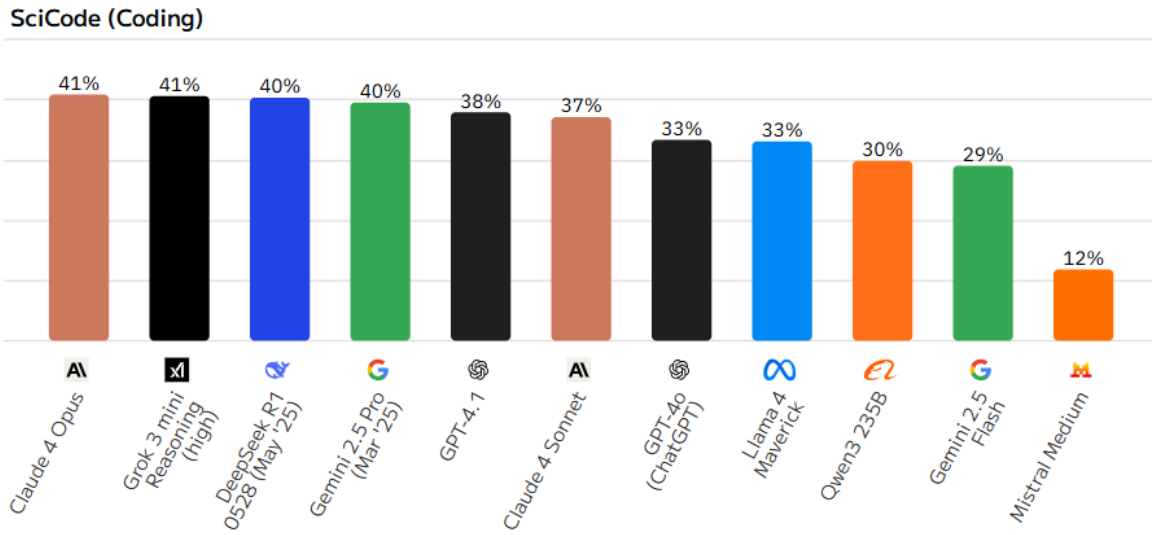


Figura 12. Evaluación de modelos en la nube de SciCode (pass@1). Fuente: [88]

Estos resultados muestran un rendimiento más moderado que en otros benchmarks, debido a la complejidad del contexto científico estructurado que exige razonamiento técnico preciso. En la nube, los mejores resultados corresponden a Claude 4 Opus y Grok 3 mini Reasoning (41 %), seguidos muy de cerca por DeepSeek R1 0528 y Gemini 2.5 Pro (40 %). En el otro extremo, Gemini 2.5 Flash (29 %) y Qwen3 235B (30 %) presentan los valores más bajos del entorno cloud, manteniéndose en una franja funcional media.

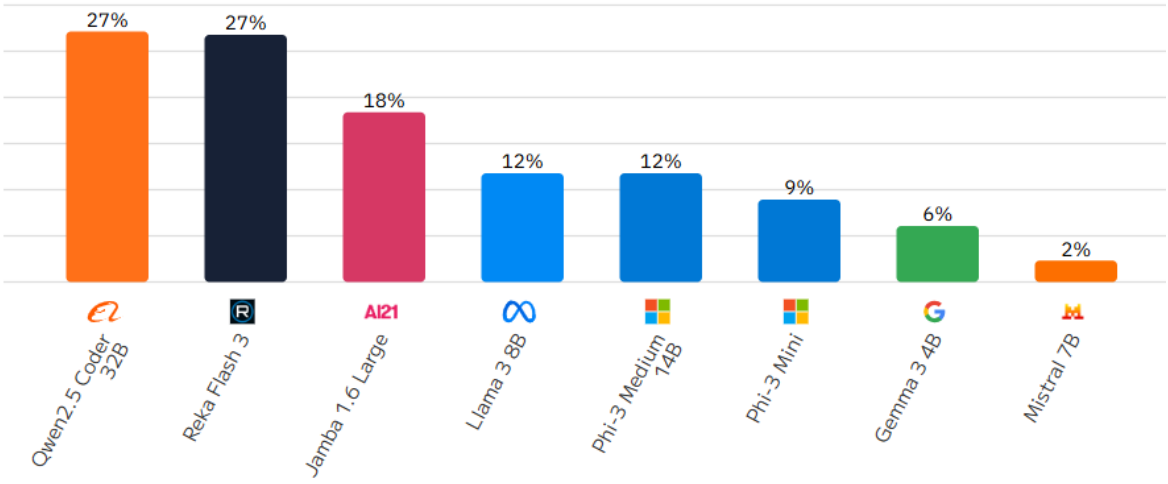


Figura 13. Evaluación de modelos en local de SciCode (pass@1). Fuente: [88]

En ejecución local, los únicos modelos con resultados destacables son Qwen2.5 Coder 32B y Reka Flash 3 (ambos con 27 %), que se aproximan al umbral de utilidad práctica en este tipo de tareas. En contraste, modelos como Phi-3 Mini (9 %), Gemma 3 4B (6 %) o Mistral 7B (2 %) muestran rendimientos claramente insuficientes, lo que confirma las limitaciones de las arquitecturas ligeras para tareas científicas complejas y con fuerte dependencia contextual.

LiveCodeBench

En las figuras 14 y 15 se presentan los resultados del benchmark ***LiveCodeBench***, diseñado para evaluar la capacidad de los modelos para resolver problemas de programación extraídos de plataformas reales como *LeetCode* [89], *Codeforces* [90] o *AtCoder* [91].

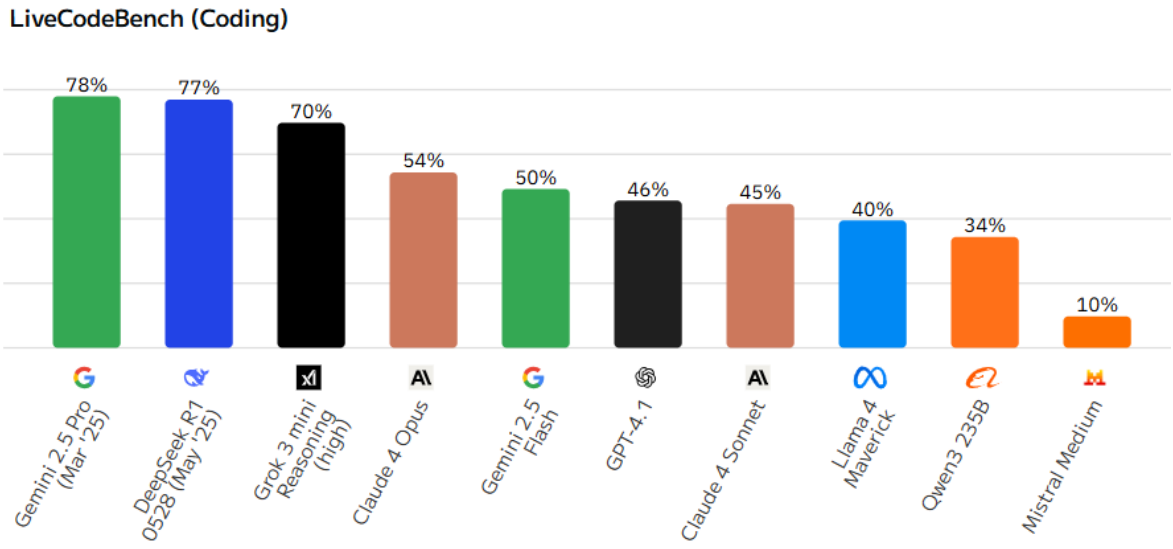


Figura 14. Evaluación de modelos en la nube de LiveCodeBench (pass@1). Fuente: [88]

LiveCodeBench evalúa la capacidad de los modelos para resolver problemas reales de programación, con estructuras abiertas y condiciones variadas. En la nube, destacan Gemini 2.5 Pro (78 %) y DeepSeek R1 0528 (77 %) como las opciones más competentes, seguidos por Grok 3 mini Reasoning (70 %), que mantiene un rendimiento sólido. En el otro extremo, Qwen3 235B (34 %) y Mistral Medium (10 %) obtienen los resultados más bajos del entorno cloud, reflejando una menor eficacia en tareas prácticas con alta variabilidad sintáctica.

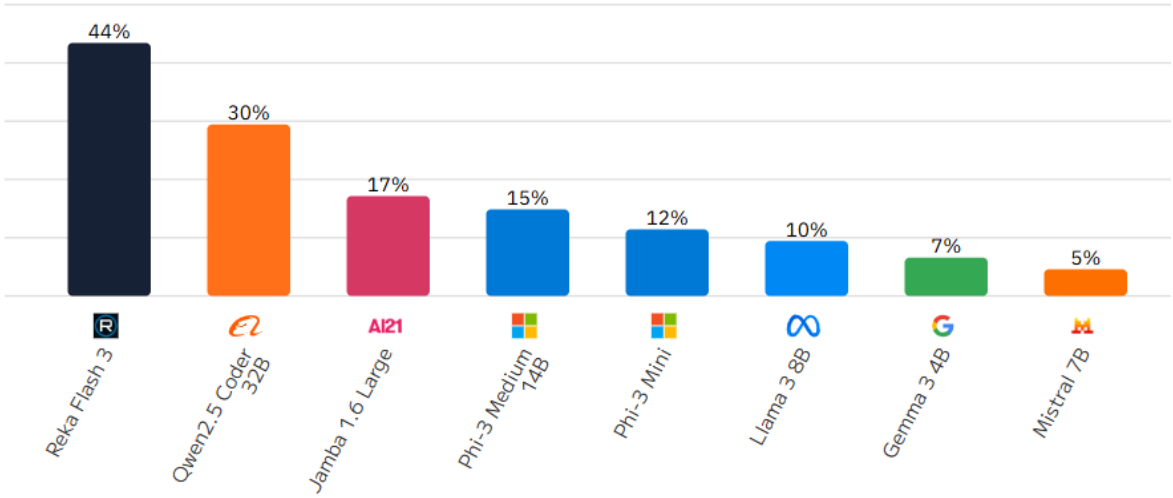


Figura 15. Evaluación de modelos en local de LiveCodeBench (pass@1). Fuente: [88]

En ejecución local, Reka Flash 3 alcanza el mejor resultado (44 %), posicionándose como una alternativa funcional frente a modelos cloud de gama media. También destaca Qwen2.5 Coder 32B (30 %) con un rendimiento aceptable. En contraste, modelos como Gemma 3 4B (7 %) y Mistral 7B (5 %) se sitúan entre los de menor precisión, mostrando una aplicabilidad muy limitada en contextos reales de desarrollo.

En conjunto, los resultados obtenidos en los distintos benchmarks de generación de código confirman una tendencia clara: los modelos en la nube presentan un rendimiento significativamente superior en términos de precisión, robustez contextual y consistencia funcional. No obstante, algunos modelos en local, como Reka Flash 3 y Qwen2.5 Coder 32B, logran posicionarse como alternativas viables en tareas técnicas exigentes, especialmente cuando se prioriza la ejecución privada o sin conexión. Por el contrario, los modelos más ligeros, tanto en la nube como en local, muestran limitaciones importantes en escenarios que requieren razonamiento estructurado o adaptación a contextos complejos.

3.1.2. Índice de eficiencia entre inteligencia, tokens y costes

Además del nivel bruto de inteligencia de un modelo, en entornos reales de desarrollo es fundamental considerar la relación entre la calidad de sus respuestas y el coste de obtenerlas. Este criterio combina la puntuación obtenida en el *Artificial Analysis Intelligence Index* con dos parámetros principales: el número de tokens generados y el coste económico asociado al uso del modelo [88]. En conjunto, este análisis permite identificar qué modelos ofrecen un mejor equilibrio entre rendimiento técnico y sostenibilidad operativa.

Inteligencia - Número de tokens usados

En la figura 16 se representa la distribución de los modelos en la nube según su puntuación en **inteligencia frente al número de tokens generados**. El cuadrante superior izquierdo indica los modelos más atractivos, al conjugar alta inteligencia con bajo consumo de tokens.

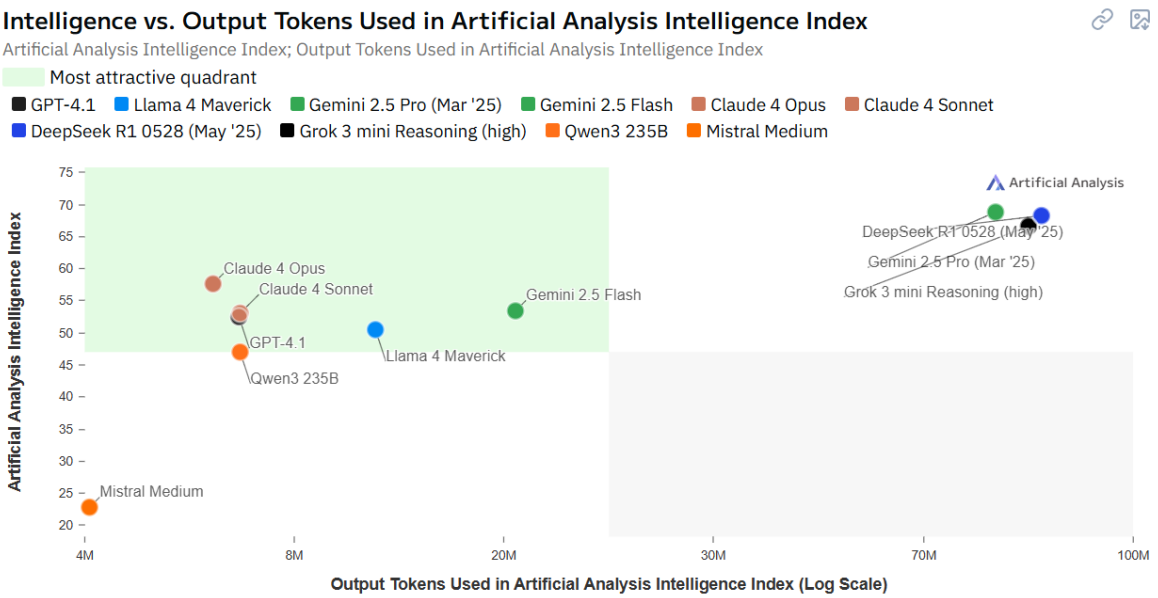


Figura 16. Relación entre inteligencia y número de tokens en modelos en la nube. Fuente: [88]

DeepSeek R1 0528, Gemini 2.5 Pro y Grok 3 mini Reasoning presentan las puntuaciones de inteligencia más altas (superiores a 70), pero a costa de un consumo muy elevado de tokens (más de 85 millones), lo que puede implicar mayores costes y latencias. Por el contrario, Gemini 2.5 Flash y LLaMA 4 Maverick se sitúan en el cuadrante más eficiente, al combinar un rendimiento aceptable (~50) con un consumo contenido, entre 11 y 20 millones de tokens. Claude 4 Opus, Claude 4 Sonnet, GPT-4.1 y Qwen3 235B muestran un equilibrio intermedio, mientras que Mistral Medium, con baja puntuación (~20), queda fuera del rango recomendable a pesar de su bajo consumo

La figura 17 muestra el mismo análisis aplicado a los modelos ejecutados localmente.

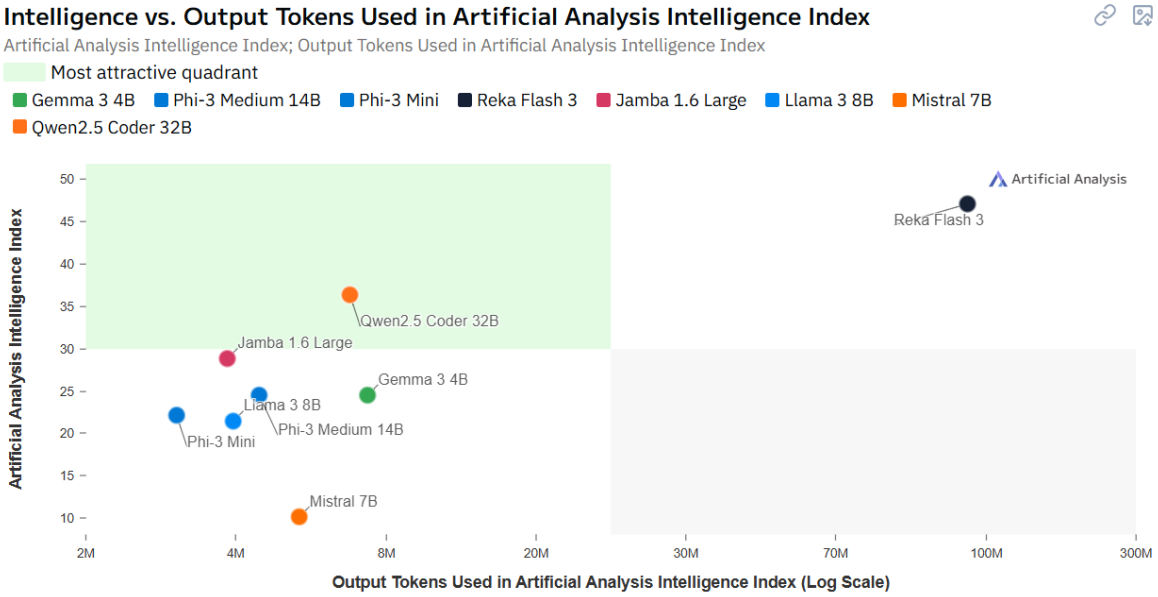


Figura 17. Relación entre inteligencia y número de tokens en modelos locales. Fuente: [88]

En este conjunto, Qwen2.5 Coder 32B destaca como el modelo local más equilibrado, con una puntuación cercana a 36 y consumo moderado. También Jamba 1.6 Large muestra una relación favorable entre rendimiento y eficiencia, situándose dentro del cuadrante atractivo. Aunque Reka Flash 3 alcanza una puntuación técnica muy alta (superior a 50), su altísimo consumo de tokens (alrededor de 100 millones) compromete su viabilidad en entornos sensibles al coste. El resto de modelos locales como Phi-3 Medium 14B, Phi-3 Mini, LLaMA 3 8B o Gemma 3 4B presentan puntuaciones bajas sin destacar en eficiencia. Mistral 7B, con un rendimiento inferior a 15, resulta aplicable solo en pruebas ligeras donde el bajo coste sea el único requisito relevante.

Costes de uso

En el ámbito económico, es importante evaluar la eficiencia de los modelos en términos de **relación entre su rendimiento técnico y el coste total asociado a su uso**. En este análisis, se considera el coste acumulado (en *United States Dollar, USD*) necesario para ejecutar todas las evaluaciones del *Artificial Analysis Intelligence Index*, incluyendo el coste de tokens de entrada, procesamiento (*reasoning*) y salida. Este parámetro es especialmente relevante en escenarios de uso intensivo o presupuestos limitados.

La figura 18 muestra esta relación aplicada a los modelos en la nube.

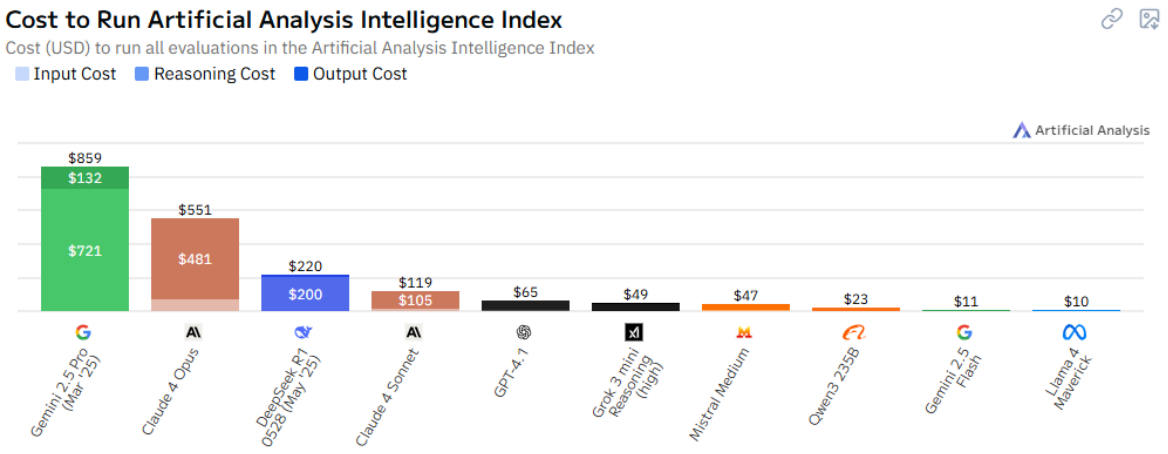


Figura 18. Relación entre inteligencia y coste económico en modelos en la nube. Fuente: [88]

Gemini 2.5 Pro (859 USD) y Claude 4 Opus (551 USD) alcanzan los mayores niveles de inteligencia, pero con los costes más elevados. DeepSeek R1 0528, con un coste intermedio (220 USD), representa una alternativa equilibrada. En el extremo opuesto, modelos como Gemini 2.5 Flash (11 USD) y LLaMA 4 Maverick (10 USD) destacan por su excelente relación coste-rendimiento, mientras que opciones como Mistral Medium (47 USD) y Grok 3 mini Reasoning (49 USD) ofrecen costes bajos con un rendimiento más contenido.

La figura 19 representa el mismo análisis aplicado a los modelos de ejecución local. Es importante señalar que, si bien estos modelos no suelen incurrir en costes directos por licencia o por uso de token (muchos son de código abierto), los valores mostrados reflejan los costes operativos asociados a la infraestructura de *hardware* (por ejemplo, GPUs) y al consumo energético necesarios para ejecutar de forma autónoma todas las evaluaciones del índice.

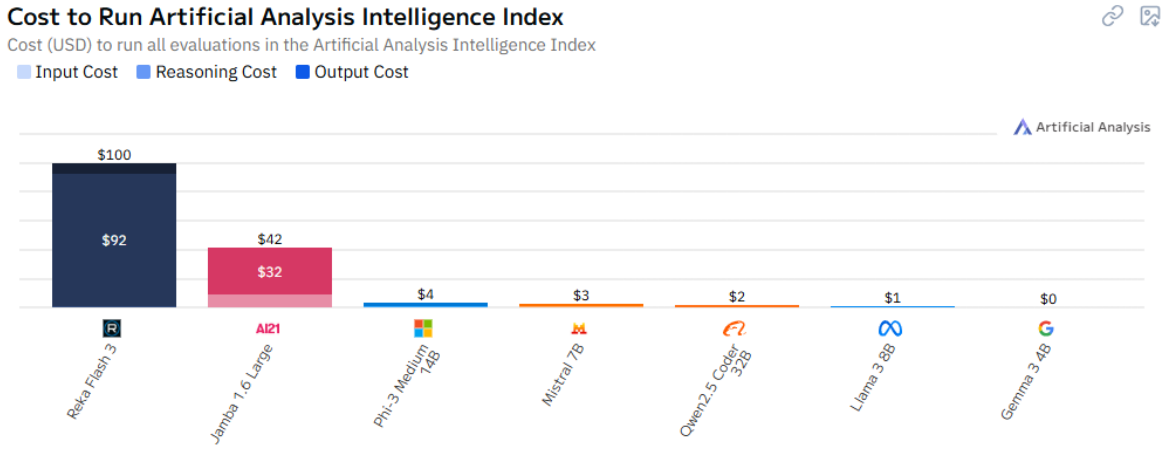


Figura 19. Relación entre inteligencia y coste económico en modelos locales. Fuente: [88]

Reka Flash 3 es el más costoso con diferencia (100 USD), aunque también el más potente en términos de inteligencia. Jamba 1.6 Large (42 USD) mantiene una posición media en ambas dimensiones. Por el contrario, modelos como Qwen2.5 Coder 32B (2 USD) y Phi-3 Medium 14B (4 USD) logran un equilibrio notable entre funcionalidad básica y eficiencia económica. LLaMA 3 8B (1 USD) y Gemma 3 4B (0 USD) sobresalen por su bajo coste, aunque con capacidades limitadas para tareas complejas.

Costes por token de entrada y salida

En esta línea, las figuras 20 y 21 permiten desglosar con mayor precisión los costes específicos asociados al uso de cada modelo, diferenciando entre el **precio por tokens de entrada (*input*) y el de salida (*output*)**, expresado en USD por millón de tokens procesados. Esta perspectiva complementa el análisis de eficiencia al ofrecer una visión más concreta de la viabilidad económica de cada alternativa.

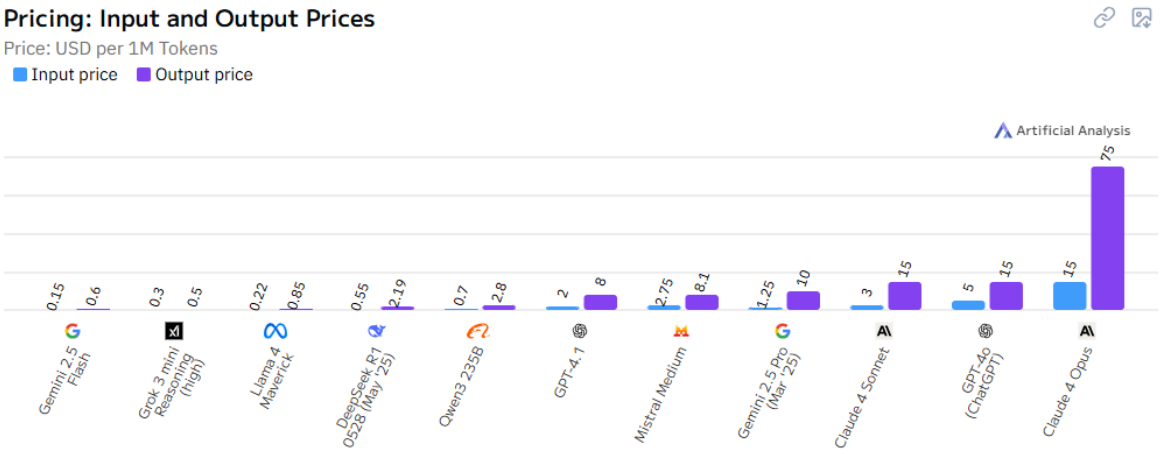


Figura 20. Coste por millón de tokens en modelos en la nube. Fuente: [88]

En el entorno cloud, Claude 4 Opus presenta los costes más elevados (15 USD por input y 75 USD por output), seguido por GPT-4o (5 + 15 USD) y Claude 4 Sonnet (0 + 15 USD), orientados a escenarios donde se prioriza el rendimiento. En una franja intermedia se sitúa Gemini 2.5 Pro (1.25 + 10 USD), mientras que opciones como DeepSeek R1 0528 (0.55 + 2.19 USD) y Qwen3 235B (0.7 + 2.8 USD) ofrecen un coste más ajustado. Modelos como Gemini 2.5 Flash (0.15 + 0.6 USD), Grok 3 mini Reasoning (0.3 + 0.5 USD) o LLaMA 4 Maverick (0.22 + 0.85 USD) destacan por su eficiencia tarifaria dentro de este entorno.

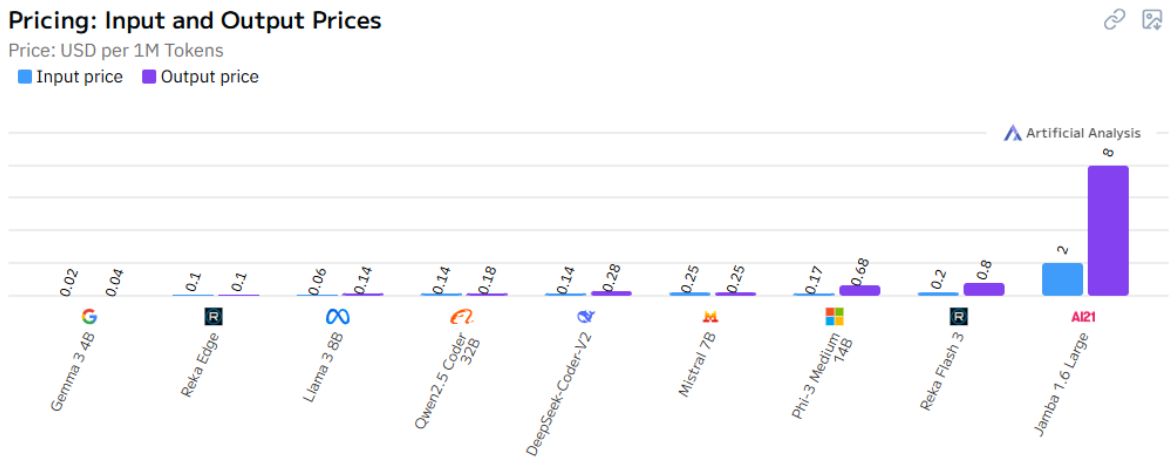


Figura 21. Coste por millón de tokens en modelos en local. Fuente: [88]

Para los modelos de ejecución local, estos valores reflejan el coste computacional o la eficiencia de procesamiento por millón de tokens, asumiendo una infraestructura estándar para su ejecución. No representan una tarifa monetaria directa por token, sino una métrica de los recursos computacionales y energéticos necesarios para su operación a esa escala. En este caso, destacan Gemma 3 4B (0.02 + 0.04 USD), Reka Edge (0.1 + 0.1 USD) y LLaMA 3 8B (0.06 + 0.14 USD) como opciones altamente económicas. Otros modelos como Qwen2.5 Coder 32B, Mistral 7B y DeepSeek Coder V2 se mantienen en el rango de 0.14 a 0.28 USD por cada tipo de token. Aunque Phi-3 Medium 14B (0.17 + 0.68 USD) y

Reka Flash 3 (0.2 + 0.8 USD) presentan un coste algo más elevado, siguen siendo viables para muchos entornos técnicos. La excepción es Jamba 1.6 Large, cuyo coste de salida (8 USD) limita su eficiencia a largo plazo, pese a su input moderado (2 USD).

En conjunto, el análisis de eficiencia evidencia que, si bien los modelos en la nube tienden a ofrecer mayores niveles de inteligencia, esta ventaja suele ir acompañada de un consumo significativamente más alto de tokens y costes operativos elevados. Sin embargo, existen excepciones notables como Gemini 2.5 Flash o LLaMA 4 Maverick, que logran un equilibrio competitivo entre rendimiento y sostenibilidad. En el entorno local, modelos como Qwen2.5 Coder 32B y Jamba 1.6 Large destacan por su buena relación entre funcionalidad técnica y bajo coste, mientras que Reka Flash 3, pese a su alto consumo, sigue siendo una de las opciones más potentes disponibles fuera de la nube. Estos resultados permiten identificar candidatos prometedores según las restricciones de cada entorno, especialmente cuando se prioriza la eficiencia en tareas recurrentes o con limitaciones presupuestarias.

3.1.3. Capacidad de contexto o ventana de memoria

La **ventana de contexto** define el número máximo de tokens (entrada y salida) que un modelo puede procesar simultáneamente. En generación de código, esta capacidad resulta crítica para mantener coherencia, referencias previas y continuidad funcional en sesiones largas o fragmentadas [88].

La figura 22 muestra el **límite de tokens de contexto admitido** por los modelos ejecutados en la nube.

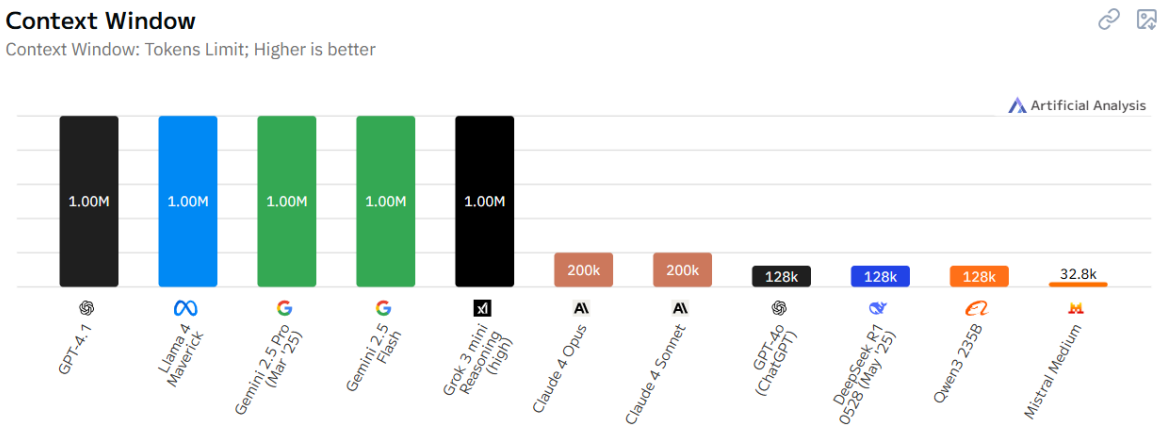


Figura 22. Límite de tokens de contexto en modelos en la nube. Fuente: [88]

Cinco modelos cloud alcanzan el límite máximo actual de 1 millón de tokens: GPT-4.1, LLaMA 4 Maverick, Gemini 2.5 Pro, Gemini 2.5 Flash y Grok 3 mini Reasoning. Este valor permite manejar documentos extensos, múltiples archivos o instrucciones complejas sin interrupciones. En un nivel intermedio se sitúan Claude 4 Opus y Sonnet (200 mil tokens),

y con 128 mil tokens aparecen GPT-4o, DeepSeek R1 0528 y Qwen3 235B, valores aún adecuados para muchas tareas técnicas. Mistral Medium, con solo 32.8 mil tokens, presenta una capacidad considerablemente limitada.

La figura 23 representa los límites de contexto de los modelos ejecutables en local.

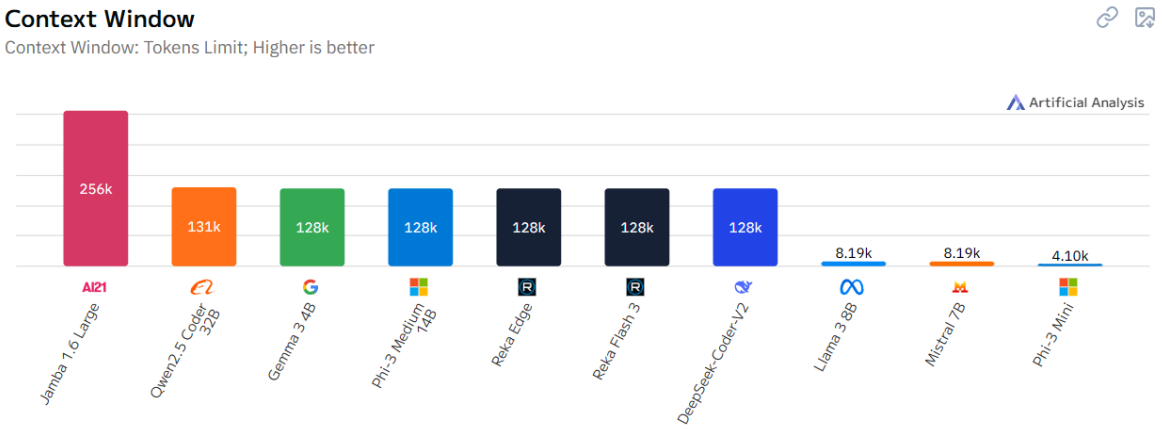


Figura 23. Límite de tokens de contexto en modelos locales. Fuente: [88]

Entre los modelos locales, Jamba 1.6 Large lidera con 256 mil tokens, seguido por Qwen2.5 Coder 32B (131 mil). Un grupo relevante formado por Gemma 3 4B, Phi-3 Medium 14B, Reka Edge, Reka Flash 3 y DeepSeek Coder V2 mantiene una ventana de 128 mil tokens, suficiente para tareas estructuradas. En cambio, modelos ligeros como LLaMA 3 8B y Mistral 7B (8.19 mil), o Phi-3 Mini (4.10 mil), ofrecen un contexto muy limitado, lo que restringe su aplicabilidad a tareas simples o sesiones breves.

Inteligencia - Ventana de contexto

A continuación, en la figura 24, se representa la **relación entre la puntuación en el *Artificial Analysis Intelligence Index* y el límite de contexto en tokens** para los modelos en la nube. Esta comparación permite evaluar qué modelos logran un equilibrio efectivo entre capacidad de razonamiento y retención de contexto, lo que es clave en tareas de desarrollo que requieren seguimiento de instrucciones largas o análisis de estructuras extensas.

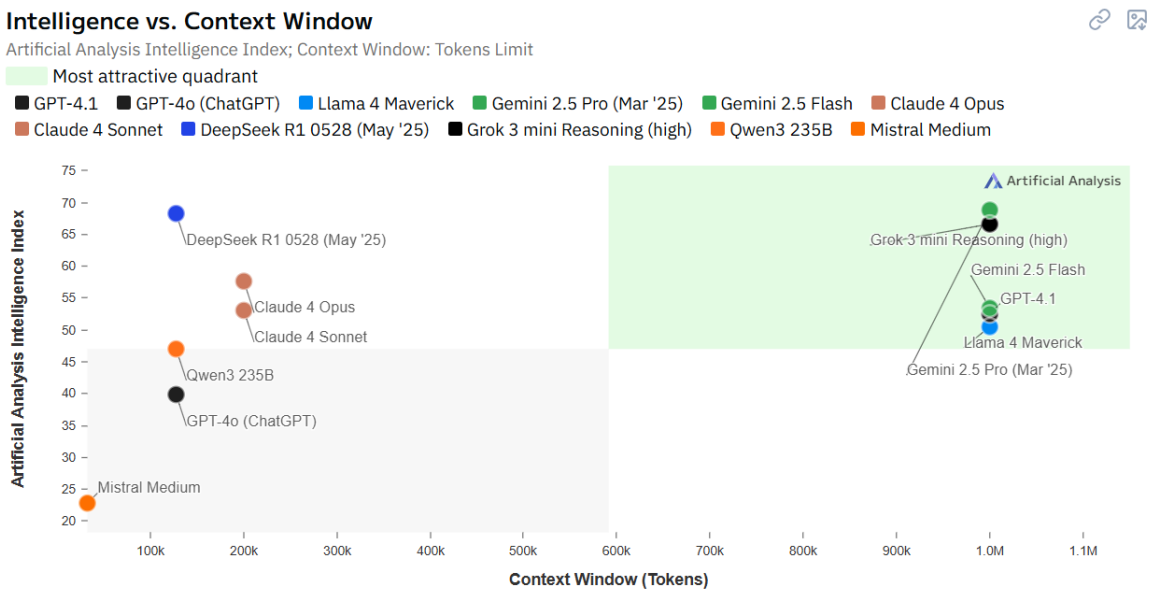


Figura 24. Comparativa inteligencia–contexto en modelos en la nube. Fuente: [88]

Grok 3 mini Reasoning, Gemini 2.5 Flash, GPT-4.1, LLaMA 4 Maverick y Gemini 2.5 Pro se sitúan en el cuadrante más atractivo, al combinar puntuaciones altas (superiores a 55) con una ventana de contexto máxima (1 millón de tokens). Por su parte, Claude 4 Opus, Claude 4 Sonnet y DeepSeek R1 0528 mantienen una inteligencia competitiva (entre 50 y 67), aunque con un límite contextual menor (200 y 128 mil tokens), lo que requiere mayor control en sesiones extensas. GPT-4o, Qwen3 235B y Mistral Medium, con valores bajos en al menos una de las dos dimensiones, quedan fuera del rango óptimo para tareas avanzadas.

La figura 25 aplica este mismo análisis a los modelos de ejecución local.

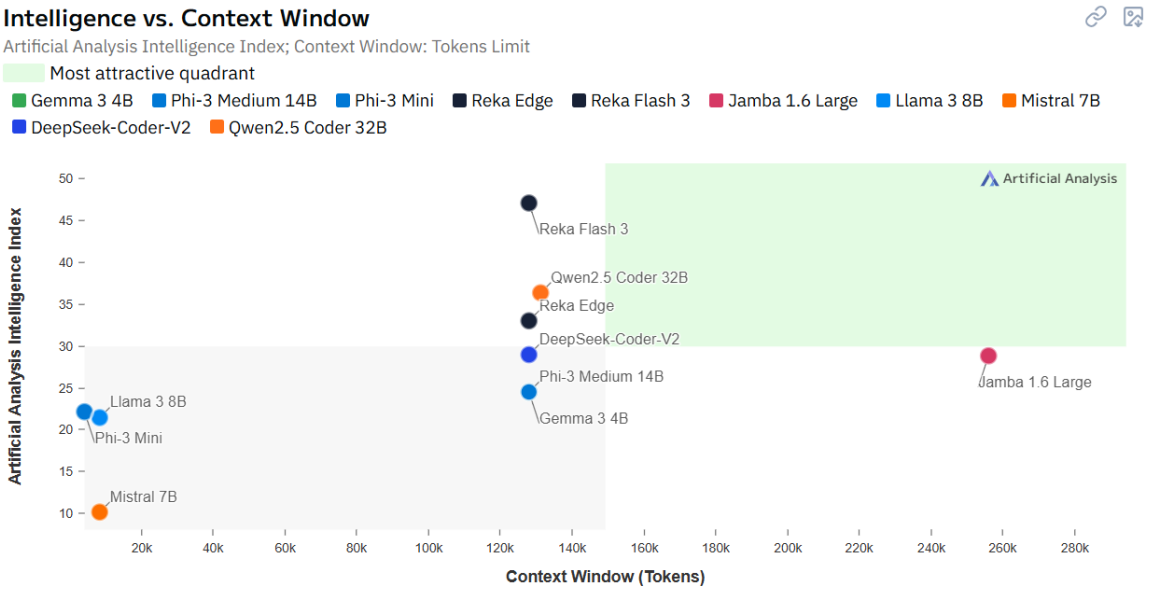


Figura 25. Comparativa inteligencia–contexto en modelos locales. Fuente: [88]

Reka Flash 3 destaca como el único modelo local que combina una ventana amplia (128 mil) con una puntuación de inteligencia elevada (~45), posicionándose en el cuadrante ideal para tareas exigentes sin conexión. Le siguen Qwen2.5 Coder 32B, Reka Edge y DeepSeek Coder V2, todos con ventanas de 128–131 mil y un rendimiento razonable, adecuados para flujos de trabajo estructurados. Jamba 1.6 Large presenta la mayor ventana (256 mil), pero su menor nivel de inteligencia reduce su aplicabilidad en tareas complejas. Modelos como Phi-3 Mini, Mistral 7B o Gemma 3 4B, con capacidades limitadas tanto en contexto como en razonamiento, quedan descartados para usos avanzados.

En conjunto, el análisis de la capacidad de contexto confirma que los modelos en la nube lideran con claridad, al combinar límites de tokens excepcionalmente amplios con altos niveles de inteligencia, lo que los hace idóneos para tareas complejas y sesiones prolongadas sin fragmentación. No obstante, algunos modelos locales (como Reka Flash 3 o Qwen2.5 Coder 32B) comienzan a posicionarse como alternativas viables en entornos donde se requiere coherencia contextual sin depender de servicios externos. La ventana de memoria, por tanto, se consolida como un factor diferencial en el diseño de flujos de trabajo complejos, especialmente en aplicaciones que exigen mantener múltiples referencias a lo largo de la interacción.

3.1.4. Tiempo de respuesta

En tareas de generación de código en entornos interactivos, como asistentes en plataformas educativas o herramientas de desarrollo integradas, la fluidez de la interacción es un factor crítico. Por ello, es importante analizar tanto la velocidad de generación de tokens por segundo como el tiempo total de respuesta extremo a extremo, dos métricas que ofrecen perspectivas complementarias sobre el rendimiento del modelo [88].

Velocidad de generación de tokens (Output Speed)

La métrica ***output speed*** representa la velocidad con la que un modelo genera tokens una vez que ha comenzado a emitir la respuesta. Esta velocidad se mide en tokens por segundo (t/s) y varía en función del número de tokens de entrada, ya que los modelos pueden ralentizarse ante contextos más largos.

La figura 26 muestra la media de tokens generados por segundo para distintos tamaños de entrada (100, 1000, 10 mil y 100 mil tokens) en modelos en la nube.

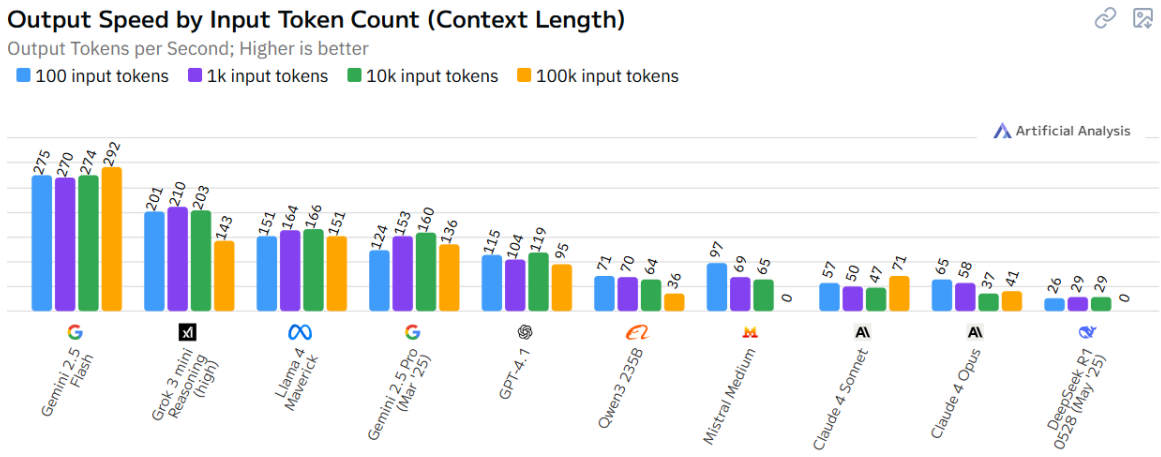


Figura 26. Velocidad de generación (tokens/s) en modelos en la nube. Fuente: [88]

Gemini 2.5 Flash lidera con diferencia, alcanzando hasta 275 tokens/s con contextos breves y manteniéndose por encima de 240 t/s incluso con entradas de 100 mil tokens. Le siguen Grok 3 mini Reasoning y LLaMA 4 Maverick, ambos con velocidades superiores a 150 t/s en la mayoría de escenarios. Gemini 2.5 Pro también muestra un rendimiento ágil, aunque algo inferior a su versión Flash. En posiciones más modestas se sitúan GPT-4.1 y Qwen3 235B, con velocidades que descienden notablemente a medida que aumenta el contexto. Claude 4 Sonnet y Claude 4 Opus presentan generación lenta (37–71 t/s), mientras que DeepSeek R1 0528 destaca por su bajo rendimiento, siendo incapaz de generar salida a partir de 100 mil tokens.

En la figura 27 se muestran los resultados del mismo análisis aplicado a los modelos de ejecución local.

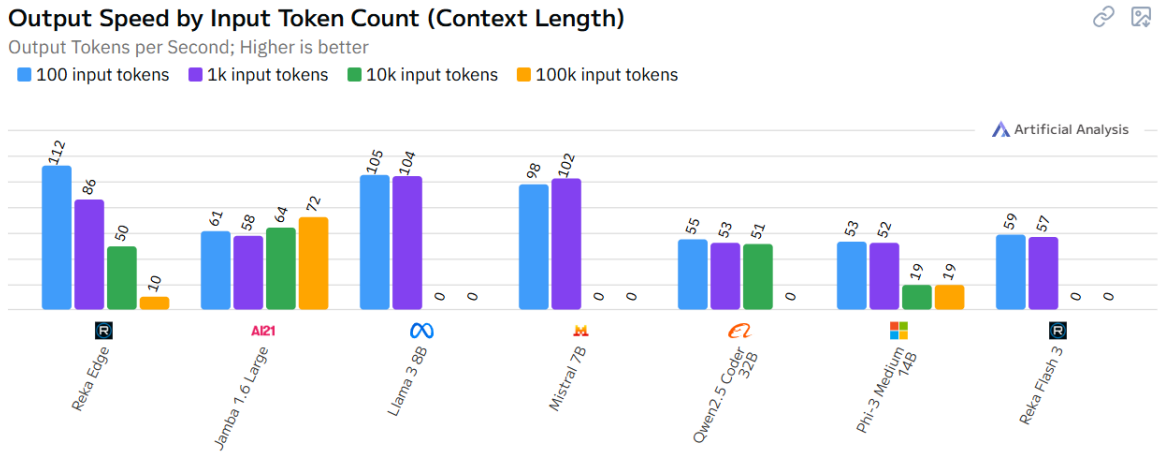


Figura 27. Velocidad de generación (tokens/s) en modelos locales. Fuente: [88]

LLaMA 3 8B y Mistral 7B ofrecen las velocidades más altas (superiores a 100 t/s) con entradas breves, aunque no generan salida cuando el contexto supera los 10 mil tokens. Reka Edge alcanza 121 t/s con 100 tokens, pero también se ve limitado en contextos

largos. Jamba 1.6 Large destaca por su estabilidad (58–72 t/s en todos los rangos), lo que lo convierte en una opción versátil para sesiones prolongadas. Modelos como Qwen2.5 Coder 32B y Phi-3 Medium 14B mantienen un rendimiento medio, funcional en tareas de media escala, aunque sin soporte para contextos muy extensos. Reka Flash 3 repite el patrón de buen rendimiento inicial con pérdida total a partir de 10 mil tokens, lo que restringe su uso a interacciones cortas.

Tiempo de respuesta extremo a extremo

Esta métrica mide el tiempo total necesario para obtener una respuesta completa de 500 tokens, integrando el procesamiento inicial, el tiempo de razonamiento (si aplica) y la generación final. Es un parámetro clave en aplicaciones interactivas, donde la inmediatez condiciona la experiencia del usuario [88].

La figura 28 presenta los tiempos medidos en los modelos en la nube.

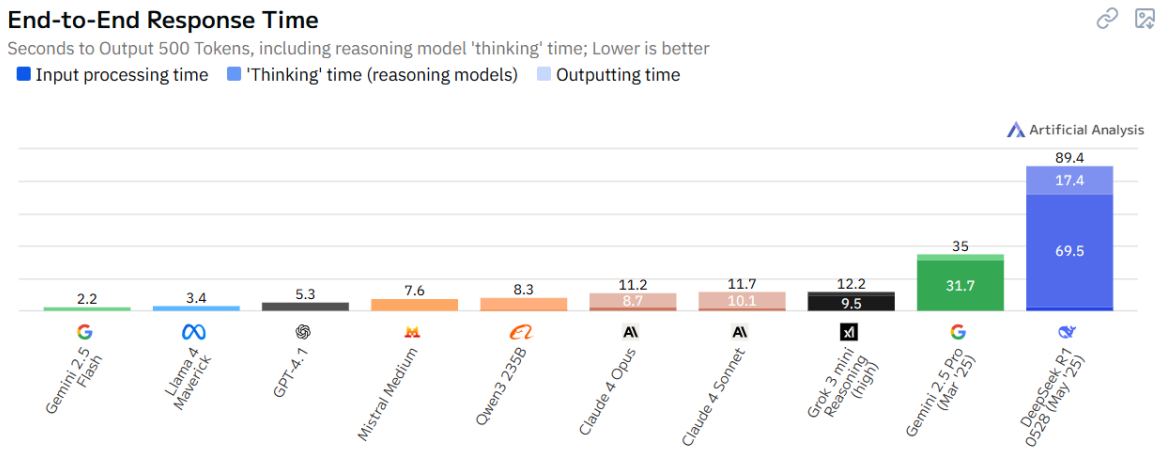


Figura 28. Tiempo de respuesta extremo a extremo en modelos en la nube. Fuente: [88]

Gemini 2.5 Flash se posiciona como el modelo más rápido, con un tiempo total de solo 2.2 s, ideal para tareas en tiempo real. Le siguen LLaMA 4 Maverick (3.4 s) y GPT-4.1 (5.3 s), ambos con tiempos bajos y sin penalización por razonamiento. Modelos con tiempo de razonamiento, como Mistral Medium (7.6 s) y Qwen3 235B (8.3 s), mantienen un rendimiento aceptable. En cambio, Claude 4 Opus, Claude 4 Sonnet y Grok 3 mini Reasoning superan los 11 s, debido a procesos de razonamiento más prolongados. Gemini 2.5 Pro (35 s) y especialmente DeepSeek R1 0528 (89.4 s) presentan tiempos excesivos, que los descartan para flujos interactivos.

La figura 29 muestra los tiempos de respuesta registrados en los modelos de ejecución local.

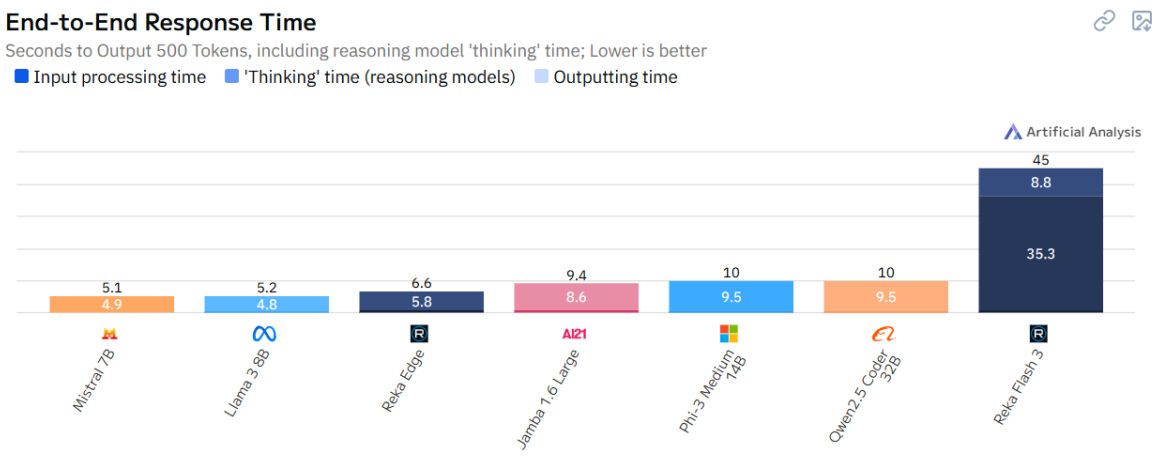


Figura 29. Tiempo de respuesta extremo a extremo en modelos locales. Fuente: [88]

En este entorno, los tiempos son generalmente menores. Mistral 7B (5.1 s), LLaMA 3 8B (5.2 s) y Reka Edge (6.6 s) ofrecen una respuesta ágil. Jamba 1.6 Large alcanza los 9.4 s debido a un tiempo de razonamiento elevado, mientras que Phi-3 Medium 14B y Qwen2.5 Coder 32B llegan a los 10 s, con reparto homogéneo entre las fases. Reka Flash 3, aunque potente en otros criterios, alcanza 45 s, lo que limita su idoneidad para entornos que requieran fluidez constante.

En conjunto, el análisis del rendimiento revela diferencias sustanciales entre los modelos evaluados en términos de agilidad operativa. En la nube, Gemini 2.5 Flash se consolida como la opción más eficiente, combinando la mayor velocidad de generación con el menor tiempo de respuesta extremo a extremo. Le siguen Grok 3 mini Reasoning y LLaMA 4 Maverick, que mantienen un buen equilibrio entre rapidez y estabilidad. En cambio, modelos como DeepSeek R1 0528 o Gemini 2.5 Pro, pese a su alto nivel de inteligencia, presentan latencias inasumibles para aplicaciones interactivas.

En local, Mistral 7B, LLaMA 3 8B y Reka Edge destacan por su rapidez en contextos breves, mientras que Jamba 1.6 Large ofrece un rendimiento sostenido incluso con entradas largas. Sin embargo, modelos como Reka Flash 3 o Phi-3 Medium 14B muestran limitaciones evidentes al aumentar la carga contextual. Estos resultados confirman que la elección de un modelo debe contemplar no solo su capacidad cognitiva, sino también su capacidad de respuesta, especialmente en entornos donde la inmediatez es determinante.

3.2. Valoración y selección final de modelos para el desarrollo

Una vez realizados los análisis comparativos según los criterios establecidos en este trabajo (nivel de inteligencia general, eficiencia en el uso de tokens y recursos, tiempo de respuesta extremo a extremo y rendimiento funcional en tareas de codificación), se procede

a la **selección final** de los modelos de inteligencia artificial generativa que se utilizarán durante el desarrollo práctico de la herramienta de anotación integrada en Moodle.

Esta decisión no se fundamenta únicamente en los resultados técnicos obtenidos, sino también en los principios establecidos como el **Objetivo de Desarrollo Sostenible 4 (Educación de calidad)**, al promover el uso de herramientas educativas inclusivas, escalables y de acceso abierto; y se alinea de forma complementaria con el **ODS 9 (Industria, innovación e infraestructuras)**, al fomentar una integración práctica y realista de tecnologías innovadoras en sectores clave como el educativo, sin depender de infraestructuras exclusivas o restrictivas.

Evaluación comparativa de los modelos en la nube

Tal y como se recoge en el análisis previo del apartado 2.5, la inteligencia artificial generativa en la nube presenta ventajas clave en términos de escalabilidad, velocidad de despliegue, acceso a modelos de última generación y costes nulos de infraestructura. En el caso concreto de este proyecto, los modelos evaluados han demostrado rendimientos sobresalientes en tareas de codificación asistida, incluso en contextos complejos que requieren razonamiento, consistencia sintáctica y comprensión semántica.

Entre todos los modelos analizados, **Gemini 2.5 Flash** se posiciona como la opción más equilibrada, al ofrecer:

- **Rendimiento técnico**, pues alcanza valores cercanos al 95 % de acierto en *HumanEval*, mostrando también resultados sólidos en *SciCode* y *LiveCodeBench*, lo que lo acredita como una solución plenamente válida para la generación de código funcional.
- **Eficiencia operativa**, dado que combina un bajo tiempo de respuesta (alrededor de 2,2 s) con una velocidad de generación superior a los 240 tokens/s y una ventana de contexto de 1 millón de tokens. Esta última característica permite gestionar instrucciones extensas, estructuras condicionales o bloques anidados, aspecto esencial para el desarrollo de interfaces interactivas como la herramienta descrita.
- **Accesibilidad gratuita**: su disponibilidad sin coste mediante Google AI Studio [92] elimina las barreras económicas, permitiendo su uso por cualquier institución educativa o desarrollador sin necesidad de licencias, credenciales empresariales o modelos de suscripción.

Frente a modelos de alta gama como GPT-4o, Claude 4 Opus o Qwen3 235B, que requieren suscripción o están restringidos a entornos comerciales, Gemini 2.5 Flash ofrece

un punto de equilibrio especialmente alineado con el carácter abierto, replicable y sostenible del proyecto.

Evaluación comparativa de los modelos en local

Los modelos generativos ejecutables en local ofrecen ventajas notables en términos de privacidad, autonomía, operación sin conexión y control sobre el entorno de ejecución, tal y como se analizó en el apartado 2.6. No obstante, este análisis ha permitido constatar que, cuando se aplica una evaluación rigurosa basada en los cuatro criterios establecidos, las limitaciones técnicas de los modelos locales reducen significativamente su aplicabilidad práctica para el desarrollo completo de esta herramienta.

Modelos como Reka Flash 3 o Qwen2.5 Coder 32B ofrecen un rendimiento excelente (95 % y 90 % en *HumanEval*, respectivamente), pero requieren configuraciones hardware avanzadas, incluyendo GPUs de gama alta y amplia memoria VRAM. Este nivel de exigencia contradice la premisa básica de accesibilidad universal que guía este proyecto, haciendo inviable su uso generalizado en contextos educativos de infraestructura media o limitada.

En el extremo opuesto, modelos más ligeros como Phi-3 Mini 4K, Mistral 7B o Gemma 3 4B presentan un perfil claramente accesible, ejecutable incluso en CPU sin GPU dedicada. Sin embargo, su rendimiento en tareas de generación de código resulta insuficiente para un uso completo y sostenido en sesiones complejas. En particular, Phi-3 Mini 4K obtiene solo un 25 % en *HumanEval* y una media inferior al 10 % en benchmarks como *SciCode*, lo que impide confiar en sus capacidades para la generación precisa, coherente y funcional de interfaces de anotación avanzadas como la propuesta.

Por tanto, aunque se reconoce la utilidad de estos modelos ligeros para tareas específicas (como la refactorización de código, la revisión de estructuras o la asistencia en iteraciones breves de diseño), **no resultan viables para liderar el proceso de desarrollo funcional completo**. Del mismo modo, los modelos de alto rendimiento quedan descartados por su incompatibilidad con los principios de bajo coste, replicabilidad y ejecución en dispositivos convencionales.

Otros modelos locales evaluados, como Qwen2.5 Coder 32B o LLaMA 3 8B, presentan un mayor consumo de memoria o un equilibrio menos favorable entre rendimiento y eficiencia, lo que limita su aplicabilidad práctica en entornos de infraestructura media. Modelos más ligeros como Phi-3 Mini 4K o Mistral 7B ofrecen tiempos de respuesta rápidos y buena

integración, pero no alcanzan un nivel de razonamiento suficientemente robusto para tareas complejas de codificación.

Selección final

A la luz del análisis conjunto de criterios técnicos (capacidad de razonamiento, eficiencia, tiempos de respuesta) y operativos (accesibilidad, escalabilidad, facilidad de integración), se determina que el modelo **Gemini 2.5 Flash** será el único empleado en el proceso de desarrollo de la herramienta. Su rendimiento elevado, su acceso gratuito y su capacidad de ejecución inmediata en la nube sin configuración adicional lo convierten en la opción óptima para garantizar la calidad técnica del producto final sin comprometer la replicabilidad y sostenibilidad del proyecto.

Esta decisión busca maximizar el impacto del trabajo, permitiendo que cualquier lector o institución con conocimientos básicos y sin recursos técnicos avanzados pueda reproducir, adaptar o escalar la solución propuesta.

Capítulo 4: Diseño e Implementación de la Anotación de Vídeos sobre Moodle

En este capítulo se detalla el diseño y la implementación del sistema de anotación de vídeos en Moodle. Se describe la arquitectura de la solución, el flujo operativo del usuario y el desarrollo técnico de sus componentes funcionales, como el selector de contenido, el panel de anotaciones y la capa gráfica interactiva, documentando el proceso de desarrollo asistido por IA.

4.1. Diseño de la arquitectura e interfaz

El desarrollo de una solución de anotación sobre vídeos integrada en Moodle requiere una planificación técnica centrada en la compatibilidad, la escalabilidad y la facilidad de uso. En este apartado se detallan los principios que orientan su arquitectura lógica, la composición de sus módulos funcionales y las decisiones que definen la interfaz orientada al usuario. El objetivo es ofrecer un sistema plenamente operativo, integrado sin fricciones en el entorno educativo digital, que respete las limitaciones de la plataforma y mejore la experiencia de aprendizaje.

Dado que Moodle no está concebido como un entorno para aplicaciones interactivas complejas, el diseño deberá adaptarse a sus restricciones sin comprometer la funcionalidad ni la accesibilidad. Esta adaptación contempla tanto la estructura modular de la plataforma como la heterogeneidad de dispositivos y contextos de acceso por parte del alumnado.

Desde el punto de vista arquitectónico, se ha optado por una solución completamente cliente, basada en tecnologías web estándar (HTML, CSS y JavaScript). Esta elección permite ejecutar la herramienta en cualquier dispositivo con acceso al campus virtual, sin necesidad de instalar extensiones ni depender de servicios del servidor. Además, se aprovechan las capacidades del navegador para gestionar eventos, renderizar elementos gráficos sobre contenidos multimedia y almacenar temporalmente la información generada por el usuario.

4.1.1. Componentes funcionales del sistema de anotación

La propuesta plantea una interfaz accesible para el marcado visual y la anotación semántica de imágenes estáticas extraídas de material docente. Su diseño modular permite una integración fluida en Moodle, sin requerir conocimientos técnicos específicos por parte del profesorado.

Desde el punto de vista estructural, el sistema estará compuesto por los siguientes bloques:

- **Selector de contenido:** menú desplegable que mostrará al alumno una lista cerrada de recursos previamente definidos por el profesorado. Esta configuración estará vinculada a un repositorio interno del curso, lo que garantiza un uso controlado y coherente con la actividad planteada.

- **Área de reproducción:** sección central que integrará un reproductor de vídeo HTML5 con controles básicos y un botón de marcado. Este botón permitirá detener la

reproducción en el instante deseado e iniciar el proceso de anotación sobre la imagen detenida.

- **Panel lateral de anotación:** bloque persistente que permitirá al alumno introducir los datos asociados a cada marca visual que trace sobre el fotograma (por ejemplo, nombre del elemento, color de referencia y descripción textual). Su contenido se ajustará dinámicamente en función de los parámetros definidos por el profesorado, adaptándose así a diversos contextos temáticos (como anatomía, estructuras técnicas o componentes mecánicos).

- **Capa gráfica interactiva:** superposición visual sobre la imagen pausada donde se representarán las anotaciones mediante círculos coloreados. Al pasar el cursor sobre una de estas marcas, se desplegará una caja flotante con la información asociada, lo que facilita una revisión inmediata e intuitiva del trabajo realizado.

- **Parámetros de configuración adaptables:** el comportamiento general de la interfaz se podrá definir mediante un fichero auxiliar situado junto al repositorio de vídeos. Este archivo establecerá tanto los contenidos disponibles como los campos del panel de anotación, permitiendo ajustar la herramienta a diferentes prácticas o asignaturas.

El conjunto de estos elementos, funcionando íntegramente en el cliente, conformará una solución ligera, versátil y extensible, apta para su integración en múltiples contextos docentes dentro del ecosistema Moodle.

4.1.2. Flujo operativo

La solución propuesta se integrará como un recurso de tipo *Página* dentro del curso. El funcionamiento diferenciará claramente entre dos perfiles: el **profesorado** (encargado de preparar el entorno) y el **alumnado** (responsable de completar la tarea). A continuación se detalla el flujo previsto para cada caso, ilustrado con ejemplos visuales.

Preparación del entorno por parte del profesorado

El profesorado será responsable de organizar los recursos que se emplearán durante la actividad. Para ello, se utilizará un recurso de tipo *Carpeta*, ubicado en una sección oculta del curso, en el que se agruparán los materiales por bloques temáticos. Este repositorio servirá como origen de los contenidos accesibles desde la interfaz de anotación.

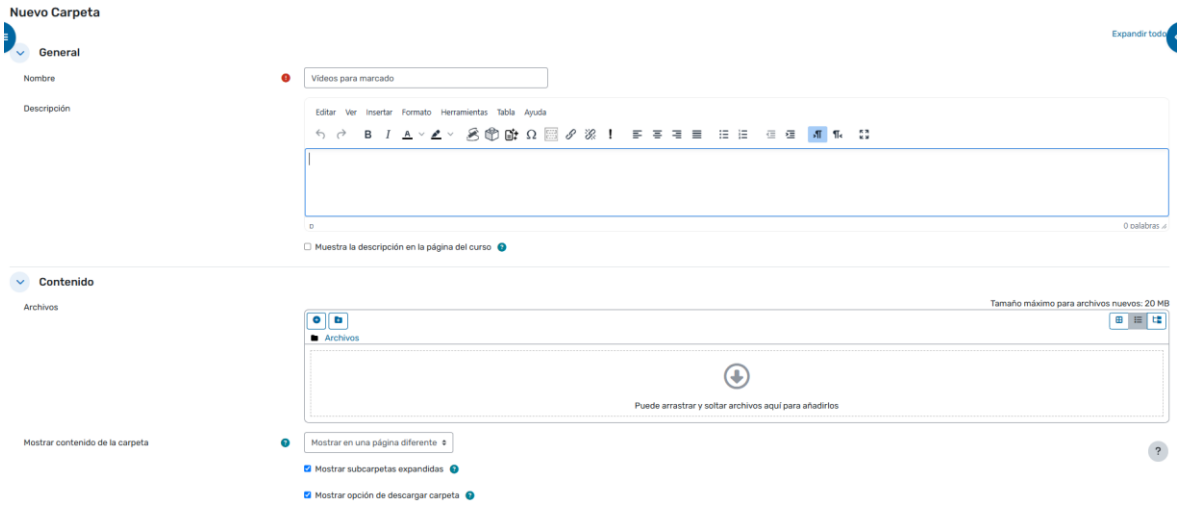


Figura 30. Vista de configuración del repositorio de vídeos en Moodle por parte del profesorado.

Junto a los materiales multimedia, el sistema requerirá un fichero auxiliar ubicado en la misma carpeta. Este fichero permitirá definir qué campos deberán cumplimentarse al generar una anotación (por ejemplo, "*Elemento identificado*", "*Órgano identificado*" o "*Descripción breve*"), facilitando la adaptación de la interfaz a las necesidades concretas de cada práctica o asignatura.

Publicación y configuración del recurso de anotación

Una vez preparados los materiales, el docente añadirá una nueva *Página* al curso, donde se integrará la herramienta de anotación. Esta interfaz detectará automáticamente los contenidos habilitados y los parámetros definidos, generando un formulario adaptado al tipo de vídeo seleccionado.

Figura 31. Desplegable de selección de vídeo al iniciar el recurso de anotación. Fuente: Generado con ChatGPT.

Acceso del alumno y selección de vídeo

Al acceder al recurso, el alumno visualizará un menú desplegable que incluirá los vídeos habilitados por el docente. Para comenzar una anotación, deberá hacer clic sobre el desplegable y seleccionar uno de los vídeos disponibles. La tarea requerirá realizar

anotaciones sobre todos los vídeos mostrados, repitiendo el proceso de forma individual para cada uno.

Reproducción del vídeo y activación del modo de marcado

Una vez seleccionado el recurso, se cargará en un reproductor embebido dentro de la misma página. El alumno podrá utilizar los controles de reproducción (*play*, pausa, avance/retroceso) para visualizar el contenido. Cuando localice el momento clave solicitado en la actividad, deberá hacer clic en el botón "***Marcar***", situado junto al reproductor. Esta acción pausará el vídeo y activará el modo de anotación sobre el fotograma congelado.

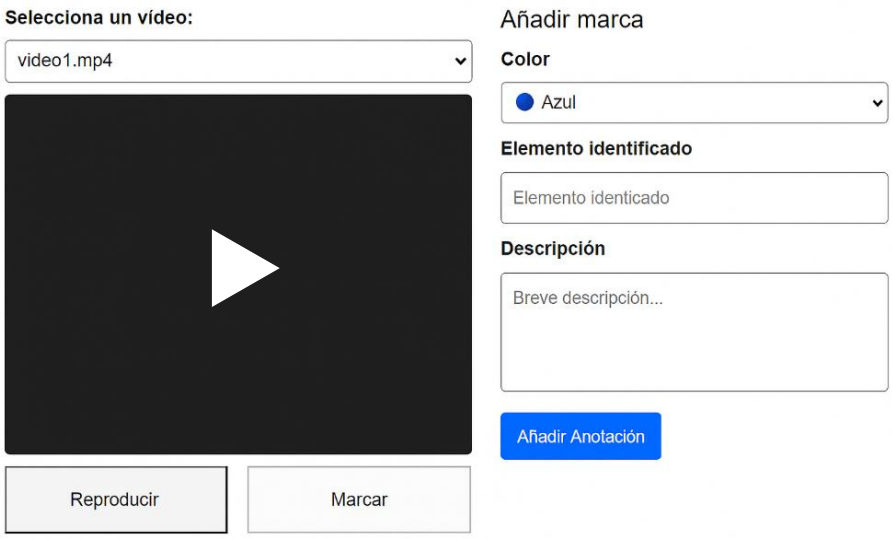


Figura 32. Reproductor de vídeo con botones de interacción y formulario lateral para registrar la anotación. Fuente: Generado con ChatGPT.

Creación de marcas circulares sobre el fotograma

Con el modo de anotación activado, se mostrará una interfaz lateral con el formulario correspondiente, visible en todo momento mientras dure la sesión de marcado. El alumno podrá introducir primero los datos requeridos (por ejemplo, seleccionar el color de la marca, indicar el elemento identificado y escribir una descripción), o bien realizar primero el trazado y luego completar los campos, en el orden que prefiera.

Para trazar una marca circular, deberá situar el cursor sobre el área deseada del fotograma, mantener pulsado el botón izquierdo del ratón y arrastrarlo hacia fuera para definir el radio del círculo. Al soltar el botón, la marca quedará fijada sobre la imagen congelada. Para fijar la marca sobre el fotograma, deberá hacer click sobre el botón "***Añadir anotación***", situado bajo el formulario.

El alumno podrá realizar múltiples marcas dentro del mismo fotograma, cada una asociada a los datos introducidos en el formulario.

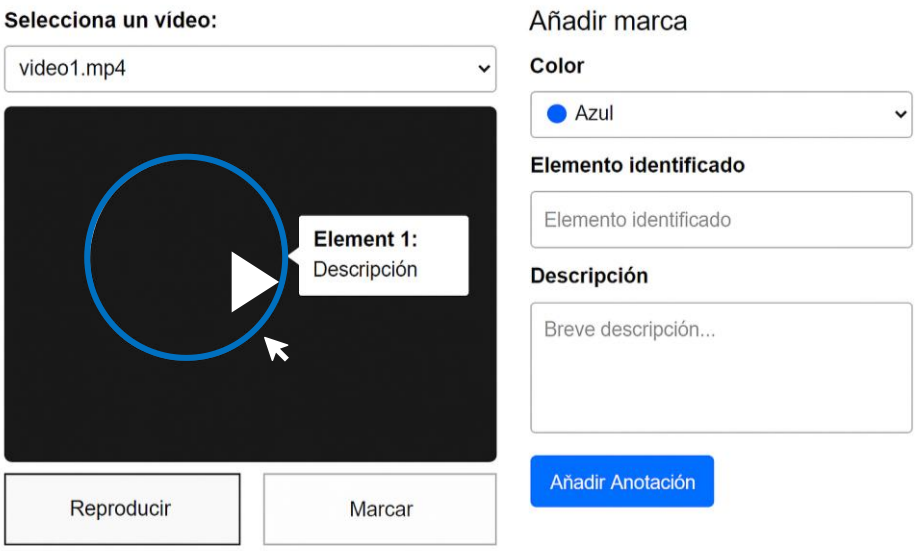


Figura 33. Vista de una marca generada sobre el vídeo, con el contenido semántico mostrado en una caja flotante. Fuente: Generado con ChatGPT.

Confirmación y visualización final

Una vez completadas todas las marcas, se mostrará una vista final del fotograma anotado. Cada marca se mantendrá visible sobre la imagen, y al pasar el cursor sobre una de ellas, se desplegará una caja flotante con el título y la descripción introducida.

La herramienta estará diseñada para que **solo se permita el marcado sobre un único fotograma por vídeo**, evitando que el estudiante realice anotaciones en diferentes momentos del mismo recurso. Esta limitación tiene como objetivo reforzar la capacidad de observación crítica y fomentar la selección precisa de un instante visual relevante, en función de las indicaciones específicas de la práctica.

Tal y como se resume en la figura 34, el proceso completo de anotación se estructura en seis etapas consecutivas, desde la configuración inicial del profesorado hasta la confirmación final por parte del estudiante.

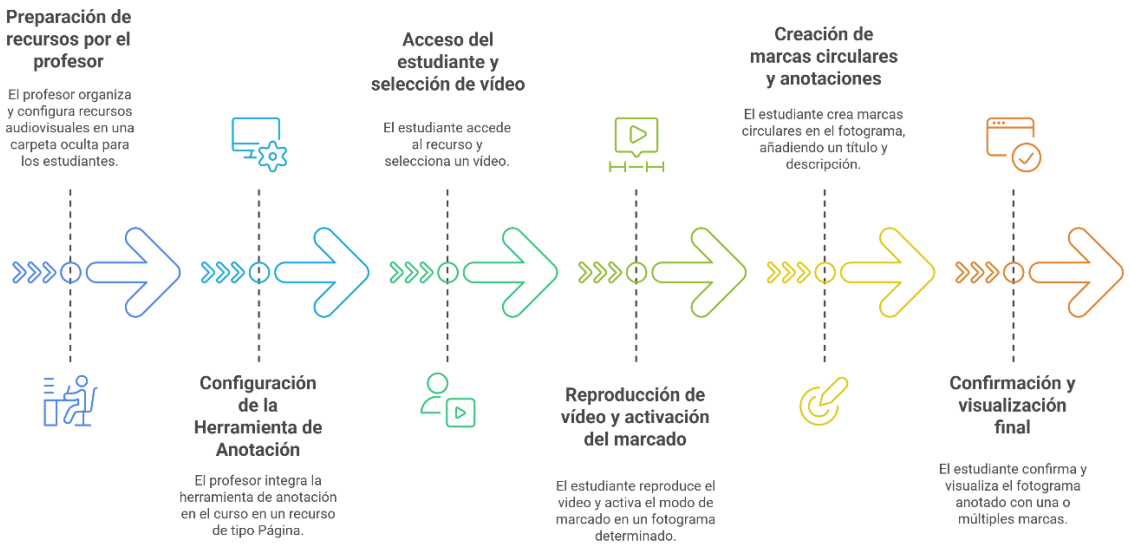


Figura 34. Flujo completo de uso de la herramienta de anotación desde la preparación del profesorado hasta la entrega del estudiante. Fuente: Generado con NapkinAI

4.1.3. Diagrama de la arquitectura

La herramienta de anotación propuesta se articula sobre una arquitectura cliente ligera, especialmente diseñada para integrarse en Moodle sin necesidad de modificar la infraestructura del servidor ni establecer conexiones con bases de datos. Este enfoque garantiza la compatibilidad con una amplia gama de dispositivos y navegadores, preservando una experiencia de usuario fluida y sin fricciones.

Tal y como se representa en la figura 35, el sistema se organiza en dos entornos claramente diferenciados: el entorno Moodle, encargado de proporcionar los recursos y configuraciones necesarias, y el navegador web del estudiante, que asume completamente la ejecución de la herramienta en el cliente.

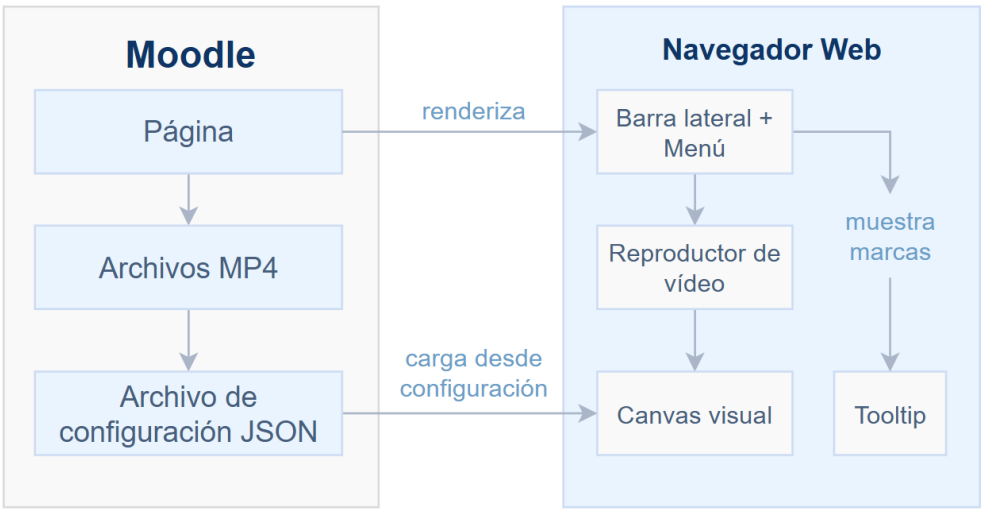


Figura 35. Diagrama de arquitectura general del sistema. Fuente: elaboración propia.

Los componentes se distribuyen del siguiente modo:
Moodle actúa como entorno de distribución de contenidos:

- El recurso ***Página*** incorpora el código HTML, CSS y JavaScript que define la interfaz de usuario y su lógica.
- Los **archivos de vídeo** (formato MP4) se alojan en el curso como parte de un repositorio accesible desde la interfaz.
- Un **archivo de configuración JSON** especifica las rutas de los vídeos y los campos personalizados que deben mostrarse en el formulario, adaptándose a cada contexto educativo.

Navegador del estudiante: al acceder a la herramienta, el navegador interpreta y ejecuta el código embebido en la *Página*. El comportamiento general se divide en los siguientes módulos funcionales:

- **Selector lateral y menú contextual**: se genera dinámicamente a partir del archivo JSON y permite elegir el vídeo deseado.
- **Reproductor de vídeo**: muestra el contenido seleccionado y permite pausarlo en el instante a marcar.
- **Canvas interactivo**: superpuesto al vídeo, permite representar visualmente las marcas realizadas por el alumno.
- **Tooltip o cuadro informativo**: aparece al hacer clic sobre una marca, mostrando las anotaciones asociadas.

En esta arquitectura, la interacción es completamente unidireccional: Moodle proporciona los recursos (vídeos y configuración), que son interpretados y desplegados por el navegador. A partir de ahí, todas las acciones de marcado, visualización y gestión de anotaciones se realizan íntegramente en el cliente, sin comunicación posterior con el servidor.

Este modelo garantiza una solución ligera, portable y fácilmente replicable, adaptable a distintos contextos docentes sin necesidad de modificaciones técnicas complejas.

4.2. Desarrollo de la funcionalidad de anotación

Una vez definidos los componentes funcionales y la lógica de uso de la herramienta, se abordará a continuación su desarrollo e implementación técnica, apoyada de forma selectiva por un modelo de inteligencia artificial generativa. En lugar de construir

directamente todo el código fuente, se opta por un enfoque asistido que permite evaluar el valor que este modelo puede aportar en el desarrollo de interfaces educativas interactivas.

Con este propósito, se empleará **Gemini 2.5 Flash**, desplegado en la nube a través de la plataforma **Google AI Studio** [https://aistudio.google.com/prompts/new_chat]. Esta elección responde tanto a su rendimiento técnico como a su accesibilidad, tal y como se ha justificado en el apartado 3.2 de esta memoria.

Para cada bloque funcional que compone la herramienta (desde el selector inicial de vídeos hasta la gestión visual de marcas circulares sobre el fotograma) se ha diseñado una serie de *prompts* que permiten generar código fuente de manera estructurada, validada e integrada. Estos abordan aspectos progresivos del desarrollo, desde la generación inicial de estructuras hasta la refactorización y mejora de componentes específicos.

A lo largo del capítulo se documenta el proceso de interacción con **Gemini 2.5 Flash**, incluyendo:

- El código final y validado generado, explicando los fragmentos más destacables.
- Una valoración crítica de los resultados obtenidos, atendiendo a criterios como adecuación al contexto educativo, claridad del código, buenas prácticas de programación y facilidad de integración en Moodle.

Todos los *prompts* empleados y las respuestas generadas se recogen en el **Anexo I**, permitiendo su consulta detallada. Además, el código final completo de la herramienta estará disponible en un repositorio accesible desde el mismo anexo, lo que facilita su reutilización o adaptación en otros contextos docentes.

Al cierre del capítulo se incluirá un resumen cuantitativo del proceso, que recogerá datos relevantes como el número total de *prompts*, el volumen de *tokens* procesados o los tiempos estimados de respuesta, todos ellos proporcionados por la propia plataforma.

4.2.1. Selector de contenido y área de reproducción

Como primer bloque funcional de la herramienta, se implementa un componente de selección de contenido que permita al estudiante escoger el vídeo sobre el que desea realizar el marcado. Esta funcionalidad se construye a partir de un conjunto de recursos y configuraciones previamente definidos en el entorno Moodle, y constituye la base interactiva desde la que se inicializa el flujo de anotación.

Para ello, se crea un recurso de tipo ***Página***, vinculado a la ejecución del sistema en la nube mediante el modelo Gemini 2.5 Flash, que actúa como contenedor de la herramienta

de anotación generada con asistencia de IA. Esta integración permite ofrecer al usuario una interfaz fluida e interactiva sin necesidad de desplegar infraestructura adicional, tal y como se muestra en la figura 36.

Figura 36. Recursos en Moodle para pruebas con el modelo seleccionado.

Se habilita un repositorio compartido dentro del curso, bajo la carpeta "*Vídeos para marcado*", donde se incorporan los recursos suministrados por el tutor. Los archivos están en formato `.mp4` y corresponden a vídeos de órganos pertenecientes a tres tipos de animales: bóvidos (Bo), cánidos (Ca) y équidos (Eq). Para evitar accesos por parte del alumnado, el tema donde se aloja esta carpeta se configura como oculto mediante el menú contextual de opciones, seleccionando la acción **"*Ocultar Tema*"**, tal y como se muestra en la figura 37.

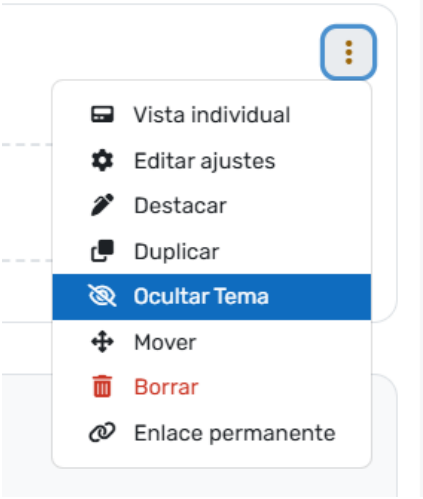


Figura 37. Menú contextual en Moodle para ocultar un tema del curso.

La codificación del nombre del archivo identifica también el órgano representado: aquellos cuyo nombre comienza por "H" muestran el hígado, y los que comienzan por "B" muestran el bazo. Esta organización permite estructurar la selección posterior de forma semántica y accesible para el usuario final.

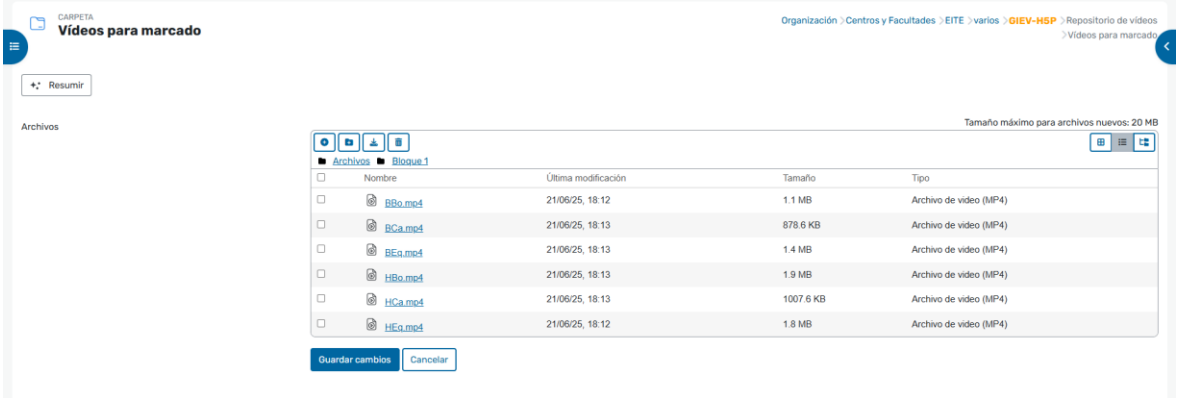


Figura 38. Repositorio de vídeos en Moodle.

Con el objetivo de alimentar dinámicamente el selector de vídeos, se crea un archivo de configuración en formato JSON, nombrado data.json. La estructura interna de estos archivos es sencilla y normalizada, como se ilustra en el siguiente fragmento:

```
{
  "videos": [
    {
      "nombre": "Hígado (Équidos)",
      "url": "HEq.mp4"
    },
    {
      "nombre": "Hígado (Cánidos)",
      "url": "HCa.mp4"
    },
    {
      "nombre": "Hígado (Bóvidos)",
      "url": "HBo.mp4"
    }
  ]
}
```

Este archivo se guarda en la misma carpeta que el repositorio de vídeos. Puede editarse para modificar las opciones del desplegable y las referencias a los archivos cargados. Para acceder a ellos desde el selector HTML, sus URLs directas de descarga se extraen de Moodle mediante la opción "*Abrir enlace en una ventana nueva*". Esto permite construir referencias absolutas accesibles desde el código embebido.

Evolución del desarrollo

A lo largo de siete iteraciones consecutivas se generaron y refinaron instrucciones para construir progresivamente un bloque funcional compuesto por:

- Un selector desplegable (`<select>`) que se alimenta de forma dinámica con los vídeos disponibles.
- Un reproductor de vídeo HTML5 que se activa tras la selección, cargando el recurso correspondiente.

- Un diseño visual limpio, adaptado al contexto de Moodle y compatible con navegadores modernos.

Cada interacción con el modelo se centró en resolver aspectos concretos: desde la estructura base en HTML hasta el comportamiento dinámico del reproductor, incluyendo ajustes de estilo y condiciones de visualización inicial.

La evolución del componente puede dividirse en tres grandes etapas:

1. **Estructuración HTML y compatibilidad con Moodle**: Se definió una estructura semántica simple, centrada en un contenedor general, un selector (`<select>`) y un área reservada para el reproductor. Se prescindió de bibliotecas externas, y se verificó que todos los estilos utilizados fueran compatibles con los recursos embebidos como *Página* en Moodle.

2. **Carga dinámica de contenido**: A partir de un archivo `videos.json`, el modelo generó la lógica necesaria para recorrer los elementos y poblar el desplegable. Se incluyó una opción inicial deshabilitada como mensaje orientativo ("*Seleccione un vídeo*"), y se aseguró que la ruta del recurso fuera accesible desde la misma carpeta del curso.

3. **Visualización condicional y comportamiento interactivo**: Se implementó la lógica que permite mostrar el reproductor HTML5 únicamente tras una selección válida. El vídeo cargado conserva controles estándar y se ajusta visualmente con respecto al selector.

Fragmento de código representativo

A continuación, se presenta el bloque principal generado para gestionar la carga del selector y la visualización del vídeo. Este fragmento, producido tras la séptima interacción con el modelo, sintetiza el comportamiento dinámico central del componente:

```
<select id="videoSelector">
  <option value="" disabled selected>Seleccione un vídeo</option>
</select>

<div id="videoContainer" style="display: none; margin-top: 20px;">
  <video id="videoPlayer" width="640" controls>
    <source id="videoSource" src="" type="video/mp4">
    Tu navegador no soporta el elemento de vídeo.
  </video>
</div>

<script>
  async function cargarVideos() {
    const response = await fetch('videos.json');
    const data = await response.json();
    const selector = document.getElementById('videoSelector');

    data.videos.forEach(video => {
      const option = document.createElement('option');
      option.value = video.url;
      option.textContent = video.nombre;
      selector.appendChild(option);
    });
  }

  document.getElementById('videoSelector').addEventListener('change', function () {
    const selectedUrl = this.value;
    const videoSource = document.getElementById('videoSource');
    const videoPlayer = document.getElementById('videoPlayer');
    const videoContainer = document.getElementById('videoContainer');

    videoSource.src = selectedUrl;
    videoPlayer.load();
    videoContainer.style.display = 'block';
  });

  cargarVideos();
</script>
```

Este bloque puede descomponerse en tres elementos clave:

1. **Estructura HTML base**:
   - El elemento `<select>` actúa como menú desplegable de selección de vídeos. Se inicializa con una opción deshabilitada como guía para el usuario.
   - El `<div>` contenedor del reproductor (`#videoContainer`) se encuentra inicialmente oculto (`display: none`) y solo se muestra tras una selección válida.
   - El reproductor de vídeo HTML5 se compone de un contenedor `<video>` con controles activados (`controls`) y una única fuente dinámica (`<source id="videoSource">`), que será actualizada al vuelo según la selección.

2. **Carga dinámica del listado de vídeos**:
   - La función `cargarVideos()` se ejecuta al cargarse la página, utilizando `fetch()` para recuperar un archivo `videos.json` alojado en el mismo directorio.

- Una vez parseado el contenido JSON, se recorre el array de objetos videos, generando dinámicamente cada opción del selector (`<option>`) mediante `document.createElement`.
- El atributo value de cada opción contiene la URL del vídeo, mientras que el `textContent` define el nombre visible.

3. **Gestión del evento de cambio (`change`)**:
   - Al seleccionar una opción, se actualiza el atributo `src` del elemento `<source>` dentro del reproductor, asignándole la URL seleccionada.
   - La llamada a `videoPlayer.load()` fuerza la recarga del recurso multimedia.
   - Finalmente, se modifica el estilo del contenedor para hacerlo visible (`display: block`), asegurando así que el reproductor solo se muestre cuando exista contenido.

En la figura 39 se muestra la vista inicial de la herramienta, donde el selector aparece centrado y aún no se ha cargado ningún vídeo. Una vez realizada una selección, se habilita el reproductor en la parte inferior del mismo contenedor, como se observa en la figura 40.

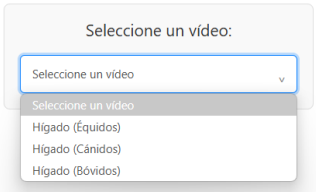


Figura 39. Vista inicial del componente con el selector desplegado.

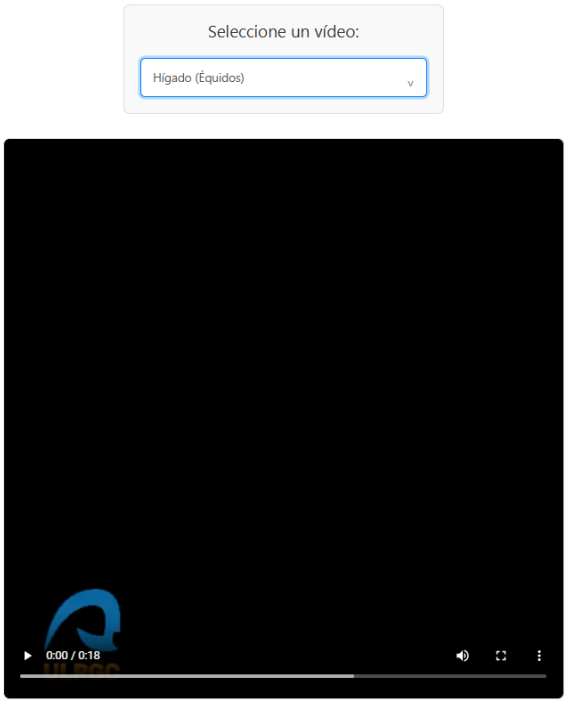


Figura 40. Visualización de vídeo activa tras la selección.

4.2.2. Panel lateral de anotación

El segundo bloque funcional de la herramienta corresponde al formulario que acompaña al reproductor de vídeo y permite al estudiante registrar la información asociada a cada marca visual. Este panel lateral permanece visible durante todo el proceso de anotación y se adapta dinámicamente a los campos definidos en el archivo de configuración JSON, lo que facilita su reutilización en contextos docentes variados.

Antes de proceder al desarrollo asistido por modelos de inteligencia artificial, se realiza una modificación en la estructura del archivo JSON, incorporando una nueva clave denominada `camposFormulario`. Esta define los elementos que deberán mostrarse en el panel lateral para cada anotación, incluyendo su etiqueta visible y el tipo de campo HTML correspondiente (por ejemplo, texto libre o área de descripción). A modo de ejemplo, la siguiente estructura configura dos entradas básicas: una para indicar el nombre del elemento identificado y otra para añadir una descripción breve.

```
{
  "videos": [
    {
      "nombre": "Hígado (Équidos)",
      "url": "HEq.mp4"
    },
    {
      "nombre": "Hígado (Cánidos)",
      "url": "HCa.mp4"
    },
    {
      "nombre": "Hígado (Bóvidos)",
      "url": "HBo.mp4"
    }
  ],
  "camposFormulario": [
    {
      "etiqueta": "Elemento identificado:",
      "tipo": "text"
    },
    {
      "etiqueta": "Breve descripción:",
      "tipo": "textarea"
    }
  ]
}
```

Gracias a esta modificación, el formulario se construirá de forma automática a partir de la información contenida en el archivo de configuración, sin necesidad de alterar el código fuente para cada actividad.

Evolución del desarrollo

En las primeras iteraciones, el modelo generó una estructura básica del formulario y sugirió la integración de campos dinámicos a partir de una definición JSON. Esta definición se

carga conjuntamente con los vídeos desde el fichero `data.json`, y se almacena en una variable que luego se utiliza para generar los campos de forma programática en el bloque `#dynamicFormFields`.

En pasos posteriores se abordó la gestión de altura del formulario. Inicialmente se intentó mediante estilos CSS fijos, pero esta estrategia generaba inconsistencias visuales cuando el contenido del formulario era escaso. Por ello, se propuso una solución basada en JavaScript, que mide dinámicamente la altura del contenedor del vídeo (`videoPlayerCard`) y la aplica al contenedor del formulario (`markingFormCard`). Este ajuste se realiza al pulsar *Marcar* y se revierte al reanudar la reproducción con el botón *Reproducir*.

Por último, se detectó que el botón *Añadir Anotación* se estiraba verticalmente al ocupar todo el espacio disponible del panel. El modelo propuso eliminar la propiedad `flex-grow` de su clase CSS y encapsularlo en una nueva clase específica (`annotation-button`) con estilo centrado, logrando así un diseño más consistente.

Fragmento de código representativo

A continuación se muestra un fragmento clave del código JavaScript correspondiente al manejo del botón *Marcar*, que ilustra la generación dinámica del formulario y la sincronización visual con el reproductor de vídeo:

```
markButton.addEventListener('click', function() {
    if (mainVideoPlayer.src) {
        mainVideoPlayer.pause(); // Pausa el vídeo
        generateFormFields();    // Carga campos dinámicos desde JSON

        videoPlayerCard.classList.add('shrink-player');
        markingFormCard.style.display = 'flex';

        // Ajuste dinámico de altura sincronizada
        const resizeObserver = new ResizeObserver(entries => {
            for (let entry of entries) {
                if (entry.target === videoPlayerCard) {
                    const playerHeight = videoPlayerCard.offsetHeight;
                    markingFormCard.style.minHeight = `${playerHeight}px`;
                }
            }
        });
        resizeObserver.observe(videoPlayerCard);
    }
});
```

Este fragmento se puede descomponer en los siguientes bloques funcionales:

1. **Gestión del evento click sobre el botón `markButton`:**
   Se asocia un *listener* al evento click, que actúa como punto de entrada al proceso de anotación. Antes de proceder, se comprueba que el reproductor tiene un vídeo cargado

(`mainVideoPlayer.src`). Esta verificación evita errores cuando el botón se pulsa antes de haber seleccionado contenido.

2. **Pausa del vídeo y carga del formulario:**
   La llamada a `mainVideoPlayer.pause()` detiene la reproducción en el fotograma actual, que actúa como referencia temporal para la anotación. A continuación, `generateFormFields()` invoca una función que crea los campos del formulario de forma dinámica. Esta función recorre la estructura definida en el archivo `data.json` y construye los elementos del formulario dentro del contenedor lateral `markingFormCard`.

3. **Ajuste del diseño visual mediante clases y estilos:**
   Se aplica la clase CSS `shrink-player` al contenedor del vídeo (`videoPlayerCard`), que modifica su ancho al 49 %, permitiendo mostrar el panel lateral en la misma línea. Al mismo tiempo, se modifica el estilo del panel (`markingFormCard.style.display = 'flex'`) para hacerlo visible.

4. **Sincronización de altura con `ResizeObserver`:**
   La principal mejora para la estabilidad de la interfaz fue el uso de *ResizeObserver* [93]. Esta API, perteneciente al Modelo de Objetos del Documento (*Document Object Model*, *DOM*) [94], está diseñada para reaccionar a los cambios de tamaño de un elemento.

   Su implementación resuelve un problema clave: se monitoriza el contenedor del video (`videoPlayerCard`) para detectar cualquier ajuste en su altura. De inmediato, la nueva altura se establece como la altura mínima (`minHeight`) del panel de anotaciones, garantizando que ambos componentes permanezcan alineados verticalmente y eliminando cualquier desajuste o solapamiento visual. El resultado visual del componente puede observarse en la figura 41, donde se muestra el panel lateral de anotación activo junto al reproductor tras haber pulsado el botón *Marcar*.

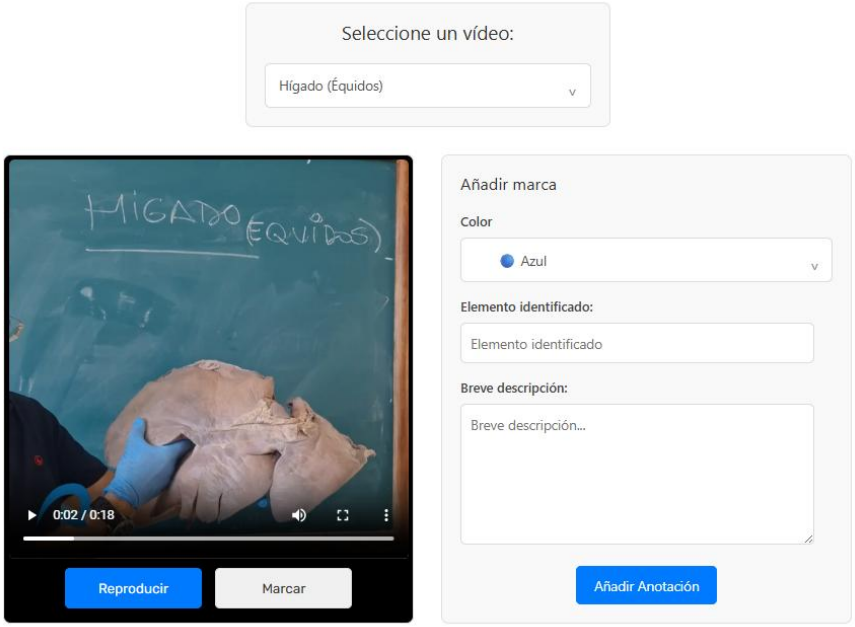


Figura 41. Visualización del formulario tras pausar el vídeo.

4.2.3. Capa gráfica interactiva

El tercer bloque funcional de la herramienta implementa el mecanismo de marcado visual mediante círculos interactivos sobre el vídeo. A través del uso de un elemento `canvas` superpuesto, el estudiante puede indicar una o varias regiones de interés mediante el arrastre del cursor. Estas marcas visuales, de forma circular, se almacenan asociadas al instante de pausa del vídeo y a los datos semánticos definidos en el formulario lateral.

Adicionalmente, se incorpora un sistema de revisión flotante mediante `tooltips` que permite consultar el contenido de cada anotación posicionando el ratón sobre las marcas previamente añadidas.

Este componente actúa como puente entre la interacción visual (`canvas`) y la anotación semántica (formulario), facilitando una experiencia de marcado intuitiva y no intrusiva. Para ello, se requiere la gestión concurrente de varios círculos, el trazado en tiempo real durante el arrastre, el almacenamiento persistente de cada anotación y su recuperación dinámica mediante eventos de puntero.

Evolución del desarrollo

En una primera etapa, se configuró el elemento `canvas` para superponerse al reproductor de vídeo, activándose al pausar mediante el botón *Marcar*. El modelo generativo sugirió encapsular la lógica de dibujo dentro de tres controladores principales: `handleMouseDown`, `handleMouseMove` y `handleMouseUp`. Esta estructura permite iniciar el dibujo al presionar

el botón izquierdo del ratón, actualizar el radio del círculo en función de la distancia al punto de inicio y completar la marca al soltar el botón.

Posteriormente se abordó la necesidad de almacenar múltiples anotaciones sin sobrescribir las anteriores. Para ello, se introdujo un array `videoAnnotations` que recoge las coordenadas, el radio, el color y los valores del formulario para cada marca. Además, se separó la lógica de renderizado en una función específica `drawAllCircles`, invocada tras cada interacción para actualizar el `canvas` con todas las marcas almacenadas.

El modelo propuso también una validación mínima del radio (por ejemplo, `> 5 px`) para evitar marcas accidentales, y el uso de una variable `currentCircle` que representa el círculo temporal en proceso de creación antes de ser confirmado.

En iteraciones posteriores se introdujo una mejora clave: la posibilidad de revisar el contenido semántico de cada marca sin necesidad de pulsaciones adicionales. Para ello, se diseñó un sistema de `tooltips` flotantes que se activan al posicionar el cursor sobre un círculo previamente almacenado. Esta funcionalidad se basa en la detección de proximidad entre el puntero y el centro de cada marca, utilizando una función auxiliar `getCircleAtPoint`. En caso de coincidencia, se genera un contenedor flotante junto al cursor (`annotationTooltip`), con los datos formateados del campo correspondiente.

Fragmento de código representativo

A continuación, se muestra el fragmento central que gestiona la visualización del `tooltip` flotante sobre el `canvas`. Este bloque combina detección geométrica, posicionamiento dinámico y actualización de contenido semántico:

```
function handleHover(event) {
    const mousePos = getMousePos(markingCanvas, event);
    const hoveredAnnotation = getCircleAtPoint(mousePos.x, mousePos.y);

    if (hoveredAnnotation) {
        // Mostrar tooltip
        annotationTooltip.style.display = 'block';
        annotationTooltip.style.left = `${event.clientX + 10}px`;
        annotationTooltip.style.top = `${event.clientY + 10}px`;

        // Construir el contenido del tooltip a partir de los campos del formulario
        let tooltipContent = '<strong>Anotación:</strong><br>';
        hoveredAnnotation.formFields.forEach((field, index) => {
            tooltipContent += `${field.etiqueta} ${field.valor}<br>`;
        });

        annotationTooltip.innerHTML = tooltipContent;
    } else {
        annotationTooltip.style.display = 'none';
    }
}
```

Este bloque se puede descomponer de la siguiente forma:

1. **Obtención de posición del cursor relativa al canvas:**
   La función `getMousePos` convierte las coordenadas del evento de ratón en valores relativos al lienzo, independientemente de su ubicación en pantalla.

2. **Identificación de marca bajo el cursor:**
   Se invoca `getCircleAtPoint`, que recorre las marcas existentes y calcula la distancia entre el puntero y el centro de cada círculo. Si esta es inferior al radio de la marca, se considera una colisión válida.

3. **Visualización dinámica del `tooltip`:**
   En caso de colisión, se actualiza la posición del elemento `annotationTooltip` mediante CSS absoluto (`left`, `top`) y se rellena su contenido con los campos etiqueta: valor de la anotación correspondiente.

4. **Ocultamiento automático fuera de foco:**
   Si no se detecta colisión, el `tooltip` se oculta mediante `display: none`, evitando mostrar información residual o errónea.

El resultado visual del componente puede observarse en la figura 42, donde se representa el estado del lienzo con varias marcas activas y la visualización emergente del `tooltip` al pasar el ratón por encima de una de ellas.

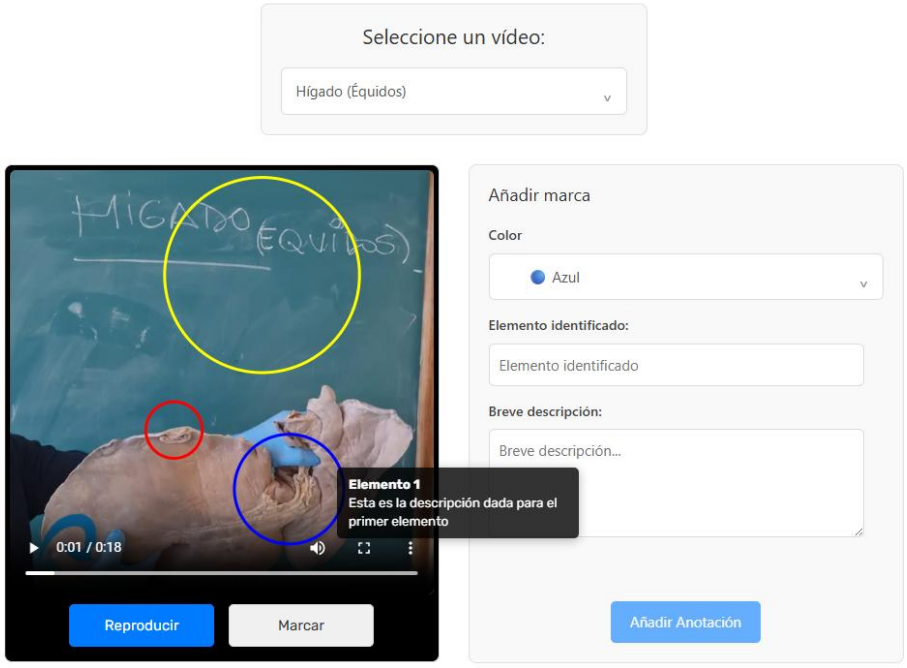


Figura 42. Visualización de marcas múltiples sobre el vídeo con tooltip flotante activo.

4.2.4. Valoración de métricas para la funcionalidad de anotación

Esta sección detalla los resultados cuantitativos y cualitativos obtenidos durante el desarrollo de la herramienta de anotación, centrándose en la interacción con el modelo empleado: Gemini 2.5 Flash. El objetivo es evaluar la eficacia de esta asistencia en la generación de código para los distintos componentes funcionales descritos, a través de métricas clave como la tasa de aceptación de respuestas, los tiempos de respuesta y el consumo de tokens. La tabla 4 resume las principales métricas por cada bloque funcional: Selector de Contenido (SC), Panel de Anotaciones (PA) y Capa Gráfica (CG).

Métrica/ Componente	Selector de Contenido (SC)	Panel de Anotaciones (PA)	Capa Gráfica (CG)
Prompts totales	7	6	6
Tokens totales de entrada (*input*)	2.093	1.317	1.274
Tokens totales de salida (*output*)	10.731	17.760	22.055
Tokens promedio por entrada	299,00	219,50	212,33
Tokens promedio por salida	1.533,00	2.960,00	3.675,83
Respuestas aceptadas	2	2	5
Respuestas no aceptadas	3	4	1
Tasa de aceptación (%)	40 % (2/5)	33 % (2/6)	83 % (5/6)
Tiempo medio de respuesta (s)			
- Respuestas < 1000 tokens	10,37 (4 respuestas)	14,80 (1 respuesta)	(Ninguna)
- Respuestas 1000-1500 tokens	15,70 (2 respuestas)	14,80 (1 respuesta)	(Ninguna)
- Respuestas > 1500 tokens	17,02 (1 respuesta)	18,11 (5 respuestas)	20,15 (6 respuestas)
Complejidad del prompt (Tokens entrada)	Media	Baja	Baja
Complejidad de la respuesta (Tokens salida)	Media	Alta	Muy Alta

Tabla 4. Análisis de Interacción con Gemini 2.5 Flash por bloque funcional. Fuente: Elaboración propia

Análisis general de la asistencia por IAG

El proceso de desarrollo, documentado a través de siete interacciones para el Selector de Contenido, seis para el Panel de Anotaciones y seis para la Capa Gráfica, evidencia la capacidad de Gemini 2.5 Flash para generar código funcional de manera iterativa. Se observa una tendencia general de aumento en los **tokens de salida promedio por**

respuesta a medida que la complejidad de la funcionalidad se incrementa (de 1533 tokens para el Selector a prácticamente 3676 para la Capa Gráfica), lo que sugiere que las soluciones más elaboradas requieren mayor volumen de código generado. Paralelamente, los **tiempos medios de respuesta** también aumentan con la complejidad y el volumen de tokens generados, confirmando las tendencias observadas en el capítulo 3 y la importancia de la eficiencia en entornos interactivos.

Resultados específicos por bloque funcional

Selector de Contenido (SC): La implementación del selector de vídeos se abordó mediante una serie de siete prompts, cuyo objetivo principal fue establecer la estructura HTML básica, la carga dinámica de los datos desde un archivo JSON (`videos.json`) y la gestión de la visualización condicional del reproductor. Este componente tuvo una **tasa de aceptación** del 40% (2 de 5 respuestas funcionales). Aunque las primeras interacciones, iniciadas con prompts directos como "*Necesito que generes el código necesario para crear un selector...*", demostraron una buena comprensión inicial de los requisitos, los prompts posteriores evidenciaron que los ajustes visuales y de posicionamiento precisos en el entorno Moodle (ej. "*No está mal, pero me gustaría que estuviera centrado en la página*") representaron un desafío recurrente para el modelo, requiriendo una formulación iterativa para resolver las inconsistencias de diseño. No obstante, el modelo demostró capacidad para ampliar la funcionalidad progresivamente y el código final, incluyendo el fragmento presentado en 4.2.1, fue el producto de esta interacción iterativa.

Panel de Anotaciones (PA): El panel lateral para la introducción de anotaciones fue construido a lo largo de seis interacciones. El enfoque principal fue la generación dinámica de campos de formulario a partir de la estructura `camposFormulario` definida en el JSON de configuración, así como la sincronización visual de su altura con el reproductor de vídeo. Este componente mostró una **tasa de aceptación** del 33% (2 de 6 respuestas válidas), indicando una mayor dificultad en la implementación asistida, principalmente debido a los desafíos en la gestión del diseño y la interacción visual compleja. Aunque la lógica funcional básica (pausar vídeo, generar campos) fue bien interpretada, los detalles de CSS y JavaScript para una integración visual fluida (como el ajuste dinámico de altura o el dimensionamiento de botones) requirieron varias correcciones. La complejidad para el modelo de interpretar y corregir comportamientos visuales se ilustra con prompts como "*El contenedor con el formulario sigue teniendo el mismo tamaño de antes, ¿qué puede estar pasando?".* Sin embargo, la solución del `ResizeObserver`, una técnica moderna y robusta para la sincronización de altura, fue generada por el modelo cuando la instrucción fue suficientemente clara, demostrando su capacidad para producir código de alta calidad.

Capa Gráfica Interactiva (CG): La funcionalidad de marcado visual mediante círculos interactivos y la visualización de `tooltips` se implementó en seis prompts. Este proceso abarcó desde la superposición del `canvas` sobre el vídeo hasta la gestión de múltiples anotaciones, el trazado en tiempo real y la visualización de información al pasar el cursor. Este componente exhibió la **tasa de aceptación más alta**, con un 83% (5 de 6 respuestas funcionales), lo que sugiere que Gemini 2.5 Flash se desempeñó de manera más efectiva en la generación de lógica interactiva y manipulación del DOM para elementos gráficos. A pesar de un problema puntual de visualización inicial, la corrección fue efectiva, permitiendo avanzar rápidamente hacia la implementación de características avanzadas como el `tooltip`. La capacidad de generar código para manejar múltiples círculos, detectar colisiones de puntero y mostrar información contextual de forma dinámica (como el fragmento presentado en el apartado 4.2.3) resalta la fortaleza del modelo en tareas de interacción visual compleja, a pesar del mayor volumen de tokens de salida asociado.

Valoración de resultados del proceso

El uso de **Gemini 2.5 Flash** como asistente de desarrollo ha demostrado ser un catalizador significativo para la eficiencia en la creación de la herramienta. Si bien la **tasa de aceptación global** (promedio ponderado de los tres módulos) no alcanzó un 100%, las interacciones "*No aceptadas sin correcciones*" no significaron un fallo completo, sino que requirieron iteraciones adicionales y prompts más depurados. Esto refuerza la idea de que los modelos de IA generativa, en el contexto actual, actúan como “copilotos de programación”, acelerando el proceso mediante la generación de estructuras iniciales y soluciones a problemas bien definidos, pero aún requieren la guía y el ajuste crítico por parte del desarrollador humano.

La capacidad del modelo para comprender el contexto de Moodle (al integrar código sin necesidad de *plugins*) y generar soluciones basadas en estándares web (HTML, CSS, JavaScript) fue destacable. La eficiencia en **tiempo de respuesta** fue generalmente buena para la complejidad de las tareas, y el consumo de tokens, si bien varía, se mantuvo dentro de los rangos de viabilidad económica (como se concluyó en el apartado 3.1.2).

En definitiva, este proceso asistido por IA validó la elección del modelo y demostró su potencial para el desarrollo de interfaces educativas interactivas, facilitando la implementación de la herramienta de marcado de vídeos de forma robusta y alineada con los principios de accesibilidad y sostenibilidad del proyecto.

4.3. Casos de uso de la aplicación

Una vez detallados el diseño y la implementación de la herramienta de anotación de vídeos en Moodle, es importante ilustrar su utilidad práctica y las características que garantizan su solidez en diversos contextos educativos. Esta aplicación ha sido concebida para transformar la visualización pasiva de vídeos en una experiencia de aprendizaje activa y personalizada, permitiendo a los estudiantes interactuar directamente con los contenidos para reforzar la comprensión y el análisis crítico.

La versatilidad de la herramienta se manifiesta en su aplicabilidad a una amplia gama de disciplinas. Por ejemplo, en **estudios de ciencias de la salud** (como veterinaria o medicina), los estudiantes podrían utilizarla para identificar y anotar estructuras anatómicas, procedimientos quirúrgicos o patologías específicas en vídeos clínicos, marcando directamente sobre el fotograma el elemento de interés y añadiendo descripciones detalladas. En el ámbito de la **ingeniería o la tecnología**, la herramienta facilitaría el análisis de procesos de fabricación, la identificación de componentes en diagramas técnicos o el seguimiento de secuencias de ensamblaje, permitiendo a los alumnos registrar sus observaciones y reflexiones sobre el funcionamiento de sistemas complejos. Estos escenarios demuestran cómo la capacidad de marcado visual y anotación semántica en un entorno controlado puede enriquecer significativamente las actividades didácticas.

Las siguientes subsecciones exploran los mecanismos técnicos que le confieren flexibilidad operativa, manejo de errores y compatibilidad en distintos entornos, asegurando su rendimiento y adaptabilidad en la práctica educativa.

4.3.1. Versatilidad mediante configuraciones JSON

Este modelo operativo confiere a la herramienta una gran versatilidad didáctica, adaptable a diversas disciplinas. Para validar esta flexibilidad, se configuró y se alteró experimentalmente el archivo `data.json`, modificando sus contenidos para simular diferentes contextos pedagógicos.

Por ejemplo, las entradas de vídeo se ajustaron para cambiar el listado visible en el desplegable de selección, incluyendo los vídeos correspondientes a un bloque específico de análisis de bazo (`BEq.mp4`, `BCa.mp4`, `BBo.mp4`). Simultáneamente, las definiciones de los camposFormulario fueron alteradas para transformar el campo "*Elemento identificado*" en "*Tipo de tejido identificado*" en el panel de anotación. Un fragmento de esta configuración JSON, diseñada para el estudio del bazo, se muestra en la figura 43.

```
{
  "videos": [
    {
      "nombre": "Bazo (Équidos)",
      "url": "BEq.mp4"
    },
    {
      "nombre": "Bazo (Cánidos)",
      "url": "BCa.mp4"
    },
    {
      "nombre": "Bazo (Bóvidos)",
      "url": "BBo.mp4"
    }
  ],
  "camposFormulario": [
    {
      "etiqueta": "Tipo de tejido identificado:",
      "tipo": "text"
    },
    {
      "etiqueta": "Breve descripción:",
      "tipo": "textarea"
    }
  ]
}
```

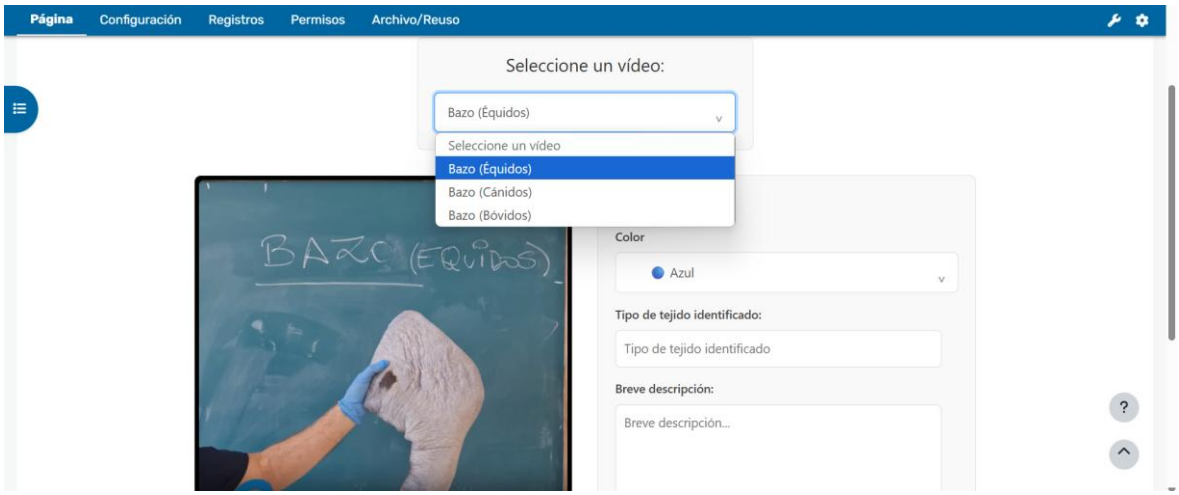


Figura 43. Interfaz de anotación con configuración JSON adaptada (ejemplo: bazo).

En cada modificación, la interfaz de la herramienta se adaptó instantáneamente al recargar en Moodle, mostrando las opciones de vídeo y los campos del formulario tal como se especificaron en el JSON editado. Esta prueba valida cómo la edición de un solo archivo JSON permite adecuar el proceso de marcado a diversos contextos, optimizando el tiempo de preparación del profesorado y asegurando una experiencia de aprendizaje del alumnado siempre contextualizada y directamente relevante.

4.3.2. Manejo de errores en la carga de recursos

La robustez de la aplicación se manifiesta en su capacidad para gestionar situaciones anómalas, especialmente aquellas relacionadas con la carga de recursos multimedia y la

interpretación de la configuración. En un entorno educativo digital, es fundamental que el sistema reaccione de manera controlada ante ficheros ausentes, rutas incorrectas o archivos malformados, evitando así interrupciones en la experiencia del usuario. Esta funcionalidad se implementa directamente en el cliente mediante JavaScript, permitiendo que el navegador capture y procese los errores antes de que afecten gravemente la interfaz o la funcionalidad central.

Para validar la eficacia de estos mecanismos, se simularon deliberadamente diversas condiciones de error. Por ejemplo, al modificar intencionadamente el archivo `data.json` para incluir una ruta de vídeo inexistente (por ejemplo, `video1.mp4`), la herramienta mostró un mensaje claro de alerta en la interfaz como el que se muestra en la figura 44, indicando la imposibilidad de cargar el recurso específico, en lugar de un fallo genérico o la interrupción de la aplicación.

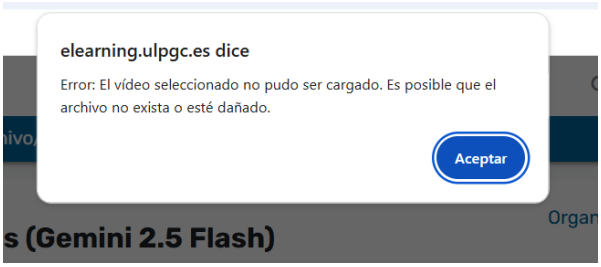


Figura 44. Mensaje de error por imposibilidad de cargar el vídeo seleccionado.

De forma similar, la introducción de un `data.json` con una sintaxis incorrecta (ej. una coma faltante o un corchete mal cerrado) desencadenó una notificación que guiaba al usuario sobre la naturaleza del error de formato, visible en la figura 45.

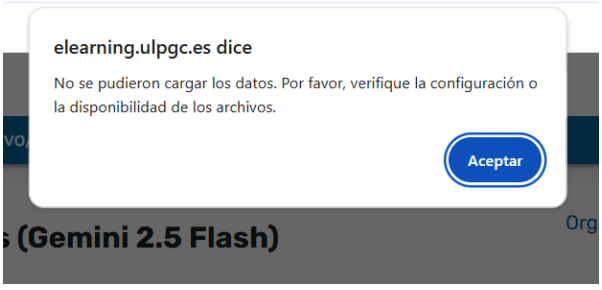


Figura 45. Notificación de error por configuración JSON inválida o inaccesible.

Estas pruebas confirmaron que la aplicación es capaz de reaccionar de manera controlada, proporcionando retroalimentación comprensible que facilita la depuración por parte del profesorado y asegura una continuidad operativa, incluso ante configuraciones erróneas.

4.3.3. Adaptación a múltiples navegadores

Finalmente, para validar esta amplia compatibilidad y asegurar una experiencia uniforme, la aplicación fue sometida a pruebas en las últimas versiones de los navegadores más prevalentes en el ámbito educativo.

Durante la fase de desarrollo, **Google Chrome** fue el navegador principal utilizado como entorno de trabajo y validación continua. Para asegurar una amplia compatibilidad y una experiencia uniforme para el alumnado, la aplicación fue sometida posteriormente a pruebas exhaustivas en las últimas versiones de otros navegadores prevalentes en el ámbito educativo.

Las verificaciones en **Microsoft Edge** confirmaron el correcto funcionamiento de todas las funcionalidades, desde la carga dinámica de contenidos hasta las interacciones de marcado y la visualización de `tooltips`.

De manera análoga, al realizar pruebas en **Mozilla Firefox**, se observó un rendimiento consistente y sin incidencias, confirmando la fluidez del proceso de anotación y la fidelidad visual de la interfaz.

Estas pruebas cruzadas validaron que el diseño basado en estándares permite a la herramienta ofrecer una experiencia de usuario robusta y accesible, independientemente del navegador utilizado por el alumnado.

Capítulo 5: Sistema de Generación y Entrega de Informes

En este capítulo se aborda el diseño e implementación de la funcionalidad de generación y entrega de informes. Se explica la arquitectura ampliada con los nuevos componentes, el proceso para generar archivos en formatos estándar como PDF y ZIP, y el mecanismo para la entrega automatizada de dichos informes en una actividad de Moodle.

5.1. Adaptaciones de la arquitectura del sistema

Tras el desarrollo de la interfaz de marcado de vídeos en Moodle, la herramienta permite a los alumnos interactuar activamente con los vídeos y enriquecerlos con anotaciones visuales y semánticas. Sin embargo, para que esta experiencia de aprendizaje activo trascienda la mera interacción y adquiera un valor pedagógico completo, es importante que el trabajo del estudiante pueda ser documentado, revisado y evaluado por el profesorado.

Este capítulo presenta la funcionalidad de generación y entrega automatizada de informes de anotación, una extensión fundamental cuyo principal objetivo es transformar las marcas circulares y los datos semánticos que el alumno ha creado en un resultado final o "evidencia" estructurada del proceso.

Su incorporación justifica su extensión no solo como un módulo adicional, sino como un paso lógico y fundamental para la utilidad práctica de la herramienta en el entorno educativo, permitiendo al alumno obtener un registro de su trabajo para revisión personal o para compartir, y, de forma conjunta, facilitar la entrega formal de este informe en Moodle, abriendo la vía para la calificación y el *feedback* docente.

5.1.1. Componentes funcionales adicionales

Para dotar a la herramienta de la capacidad de generar y gestionar informes, se ha diseñado e integrado un nuevo componente principal que actúa como la interfaz directa para el usuario en esta fase, y que orquesta el funcionamiento de varios módulos internos. Este componente se añade a los ya existentes descritos en el capítulo 4, garantizando una expansión coherente del sistema sin comprometer su naturaleza cliente-ligera

La **interfaz de opciones de informe** es el componente principal de esta ampliación, siendo la única parte directamente visible y accesible para el alumno. Se presenta como una ventana modal (un tipo de *pop-up* ligero que se superpone a la interfaz principal de la herramienta) que se activa cuando el alumno decide generar o enviar un informe. Dentro de esta interfaz, el alumno puede previsualizar el contenido consolidado de su trabajo y seleccionar las diversas opciones de descarga (PDF, imagen, ZIP) o, en su caso, iniciar el proceso de entrega del informe a la tarea de Moodle.

Para ofrecer estas capacidades, este componente se apoya en y orquesta la interacción de varios módulos internos:

- **Módulo de gestión de anotaciones en sesión:** administra y retiene en la memoria del navegador las marcas circulares y todos sus datos semánticos asociados (coordenadas, radio, color, contenido de los campos de formulario) mientras el alumno

trabaja activamente en el modo de marcado. Esto asegura que todas las anotaciones realizadas en un fotograma dado estén disponibles de forma coherente para su posterior compilación y visualización en el informe.

- **Módulo de composición visual de fotogramas:** toma el fotograma exacto del vídeo en el instante de la anotación y superpone sobre él las marcas circulares con sus respectivas numeraciones. El resultado es una imagen compuesta que proporciona una evidencia visual clara y precisa del trabajo de marcado del alumno, lista para ser incrustada en el informe final, manteniendo la fidelidad de la posición y el tamaño de las marcas.
- **Módulo de ensamblaje de archivos:** recopila toda la información textual (como los datos del alumno, del vídeo, el instante de tiempo de la anotación y los detalles específicos de cada marca) y las imágenes compuestas generadas. A partir de estos elementos, orquesta la creación de diferentes tipos de archivos estándares, como documentos PDF, imágenes individuales del fotograma marcado, archivos de texto plano (.txt) y paquetes comprimidos (.zip) que integran y presentan todos los contenidos generados.
- **Módulo de nombramiento de archivos inteligente:** genera nombres de archivo únicos y descriptivos para los informes, combinando de manera estructurada identificadores extraídos dinámicamente del entorno Moodle (como el ID del recurso actual), una versión abreviada del título del vídeo que está siendo anotado y las iniciales del nombre del usuario. Esta convención simplifica enormemente la gestión por parte del profesorado al recibir y clasificar múltiples entregas.

Estos nuevos componentes operan de forma coordinada con la interfaz de usuario y los flujos ya implementados hasta ahora (como la selección de vídeos o el marcado gráfico), permitiendo una transición fluida desde la creación de anotaciones hasta la documentación y entrega de los resultados.

5.1.2. Secuencia operativa del proceso

La generación y entrega de informes se articula como una secuencia lógica que comienza tras la finalización del marcado por parte del estudiante. Este flujo, representado visualmente en la figura 46, está diseñado para ser intuitivo, guiando al usuario desde la consolidación de su trabajo hasta la producción de un entregable final, ya sea para su descarga o para su envío a una tarea de Moodle. La secuencia operativa se desglosa en los siguientes pasos:

Inicio de la generación del informe

Una vez que el estudiante ha finalizado el proceso de anotación en el fotograma, se habilita un botón de acción contextual en la interfaz. El texto y la función de este botón se adaptan al tipo de recurso Moodle:

- En un recurso de tipo ***Página***, el botón se mostrará como ***Opciones de descarga***, indicando que el resultado será un archivo para uso personal.
- En una actividad de tipo ***Tarea***, el botón se mostrará como ***Entregar informe***, señalando el inicio del proceso de envío para evaluación.

Activación de la interfaz de informe

Al hacer clic en el botón de acción, el sistema activará la **interfaz de opciones de informe**. Esta se materializa como una ventana modal que se superpone a la herramienta de marcado, centrando la atención del usuario en la fase de revisión y finalización sin abandonar el contexto de la página.

Compilación y previsualización de datos:

De forma automática y transparente para el usuario, el sistema compilará toda la información generada durante la sesión de marcado y la presentará en la ventana modal a modo de previsualización. Este proceso incluye:

- La recuperación de metadatos de la sesión: el nombre del usuario de Moodle, el título del vídeo y el instante de tiempo exacto del marcado.
- La generación de dos imágenes del fotograma: una imagen original sin alteraciones y una segunda imagen compuesta, que muestra el fotograma con todas las marcas circulares y su numeración correspondiente superpuestas.
- La consolidación de todas las anotaciones textuales en una lista estructurada y ordenada numéricamente.

Revisión por parte del estudiante

La interfaz modal presentará al estudiante un resumen completo de su trabajo, permitiéndole verificar la exactitud de las marcas visuales y los datos semánticos introducidos antes de proceder con la acción final.

Selección de la acción final

En el pie de la ventana modal, se presentarán al estudiante los botones de acción definitivos, cuyo repertorio dependerá nuevamente del contexto del recurso Moodle:

- **Contexto de *Página*:** Se ofrecerán tres opciones de descarga:

- **Descargar PDF:** Genera un informe completo en formato PDF.
- **Descargar Imagen:** Descarga únicamente el fotograma con las marcas.
- **Descargar ZIP:** Genera un paquete comprimido con las dos imágenes (original y marcada) y dos archivos de texto (un `.txt` para lectura y un `.json` estructurado para procesamiento automático).

- **Contexto de *Tarea*:** Se mostrará un único botón:
  - **Confirmar entrega:** Inicia el proceso de empaquetado del informe en un archivo ZIP y su posterior subida automatizada a la bandeja de entrega de Moodle.

Ejecución y finalización

Al seleccionar una de las opciones, el sistema ejecutará la acción correspondiente de forma íntegra en el cliente. En el caso de la entrega en una *Tarea*, se realizará una interacción programática con la interfaz de Moodle para simular la subida del fichero, notificando al estudiante del resultado del proceso.

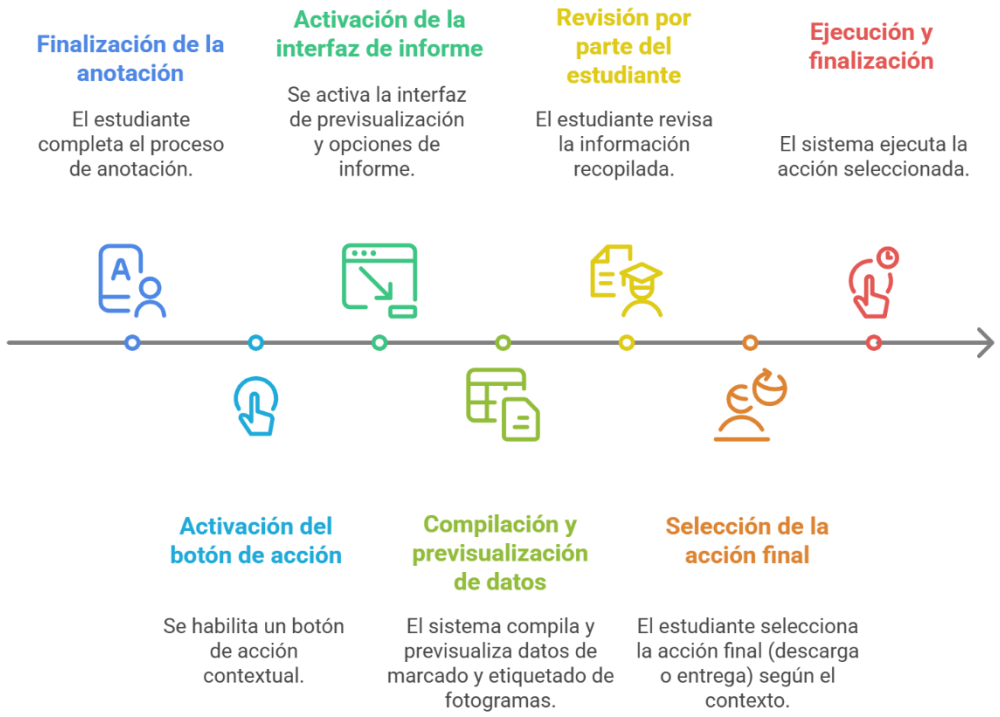


Figura 46. Flujo operativo para la generación y entrega de informes. Fuente: Generado con NapkinAI.

Este flujo operativo garantiza que el trabajo del estudiante se documente de manera robusta y coherente, proporcionando una evidencia de aprendizaje estructurada y fácilmente gestionable por el profesorado.

5.1.3. Representación de la arquitectura final

Para reflejar la incorporación del módulo de informes, la arquitectura del sistema, presentada inicialmente en la figura 35, se expande para incluir los nuevos componentes y flujos de interacción. Esta arquitectura expandida, mostrada en la figura 47, mantiene el principio de una solución ligera y centrada en el cliente, añadiendo una capa de funcionalidad para la consolidación y entrega de evidencias.

La expansión arquitectónica introduce los siguientes elementos dentro del entorno del navegador del estudiante:

- **Módulo de generación de informes:** Actúa como el componente orquestador que se activa por acción del usuario y coordina la producción de los archivos finales.
- **Librerías de terceros (*Client-Side*):** La arquitectura integra dos dependencias externas para la generación de formatos estándar:
  - **jsPDF:** Para la creación de documentos PDF a partir de texto e imágenes.
  - **html2canvas:** Para la captura del estado visual del canvas como una imagen
  - **JSZip:** Para la creación de archivos ZIP.
- **Flujo de salida dual:** El sistema ahora contempla un flujo de salida desde el cliente con dos destinos posibles:
  1. **Descarga local:** El usuario puede descargar los archivos generados directamente.
  2. **Envío a Moodle:** Se establece un flujo de interacción para enviar el informe a la actividad propuesta.

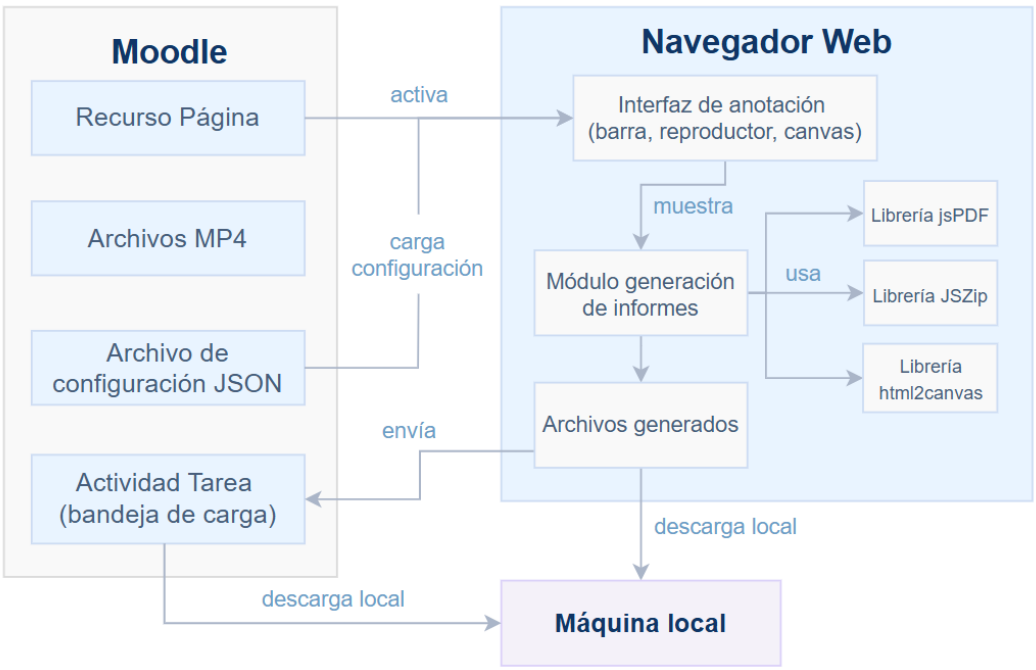


Figura 47. Arquitectura final expandida del sistema. Fuente: Elaboración propia.

Esta arquitectura demuestra cómo la herramienta no solo consume recursos de Moodle, sino que también es capaz de producir resultados estructurados y devolverlos al ecosistema educativo, sin requerir modificaciones en la infraestructura del servidor.

5.2. Implementación técnica del módulo

La implementación de la funcionalidad de informes se aborda como una extensión del sistema de anotación, manteniendo la arquitectura cliente-ligera y aprovechando la asistencia del modelo de IAG Gemini 2.5 Flash, tal y como hemos hecho hasta ahora. El desarrollo se estructura en bloques funcionales progresivos, comenzando por la activación de la interfaz y culminando con la generación de los archivos finales y su entrega.

Para contextualizar la funcionalidad de entrega, es necesario configurar una actividad de tipo *Tarea* en Moodle. A diferencia de un recurso *Página*, esta habilita una bandeja de entrega donde el alumnado puede subir archivos para su evaluación. La configuración de esta actividad es estándar en la plataforma, como se muestra en la figura 48, y no requiere ajustes específicos más allá de los parámetros habituales como las fechas de disponibilidad o los tipos de entrega permitidos.

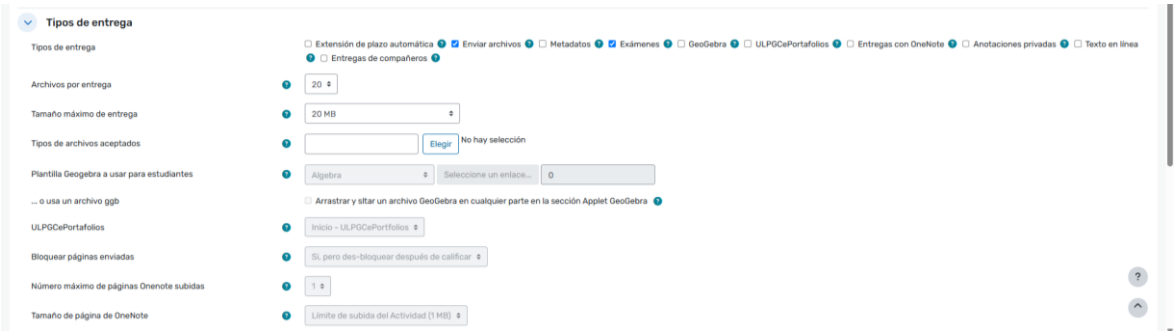


Figura 48. Configuración de una actividad de tipo Tarea en Moodle para la entrega del informe.

Una vez configurada, la herramienta de anotación se embebe en la descripción de la *Tarea*. Desde la perspectiva del alumno, la interfaz inicial es idéntica a la de un recurso *Página*, pero el sistema detectará este nuevo contexto para adaptar su comportamiento final. La figura 49 ilustra la vista que el alumno tendría de la actividad antes de iniciar el proceso de marcado y entrega.

Figura 49. Vista del alumno de la Tarea antes de la entrega.

5.2.1. Activación contextual e interfaz modal de informes

El primer componente a implementar es el mecanismo que activa la funcionalidad de informes. Se basará en un botón de acción cuya visibilidad y comportamiento se adaptarán al contexto del recurso Moodle. Esta dualidad es fundamental para ofrecer una experiencia de usuario coherente, ya sea para una descarga personal en un recurso *Página* o para una entrega formal en una *Tarea*.

Al finalizar el proceso de marcado, se habilitará un botón que, dependiendo de una variable de configuración manual en el código (`MOODLE_RESOURCE_TYPE`), mostrará el texto "*Opciones de descarga*" o "*Entregar informe*", según el tipo de recurso. Al pulsar este botón, se activará una ventana modal superpuesta, que actuará como el centro de operaciones del módulo de informes. Su contenido se generará dinámicamente al ser invocada,

compilando toda la información de la sesión: metadatos del usuario y del vídeo, las anotaciones textuales y una previsualización visual del trabajo realizado mediante la captura de dos fotogramas del vídeo, uno original y otro con las marcas superpuestas.

Evolución del desarrollo

Las primeras iteraciones se centran en definir la estructura HTML y los estilos CSS necesarios para la ventana modal, asegurando que se presente como una superposición limpia y profesional.

Posteriormente, se aborda la lógica para la captura de los fotogramas del vídeo. Para ello, se emplea el elemento <canvas> de HTML5. El proceso consiste en dibujar el fotograma actual del vídeo (`mainVideoPlayer`) en un canvas temporal usando `drawImage()`. Esta técnica permitirá obtener una representación estática del vídeo en el instante preciso del marcado. Para la imagen con las marcas, se repetirá este proceso, pero superponiendo adicionalmente los círculos de las anotaciones guardadas.

Un desafío técnico a considerar es la política de seguridad de origen cruzado (*Cross-Origin Resource Sharing, CORS*) de los navegadores, que podría impedir la captura de fotogramas si el vídeo se sirve desde un dominio diferente sin las cabeceras adecuadas [95]. Para ello, se aplica un manejo de errores que notificará al usuario si la captura no es posible.

Finalmente, se implementa la lógica para la numeración de las marcas y la correcta presentación de los datos en la modal. Se diseña un sistema de escalado para que las coordenadas de las marcas, almacenadas de forma relativa, se representen correctamente en los fotogramas, manteniendo su posición y tamaño proporcionales sin distorsión. En la figura 50 podemos ver la vista previa de la información recopilada, previa a su descarga en el formato deseado.

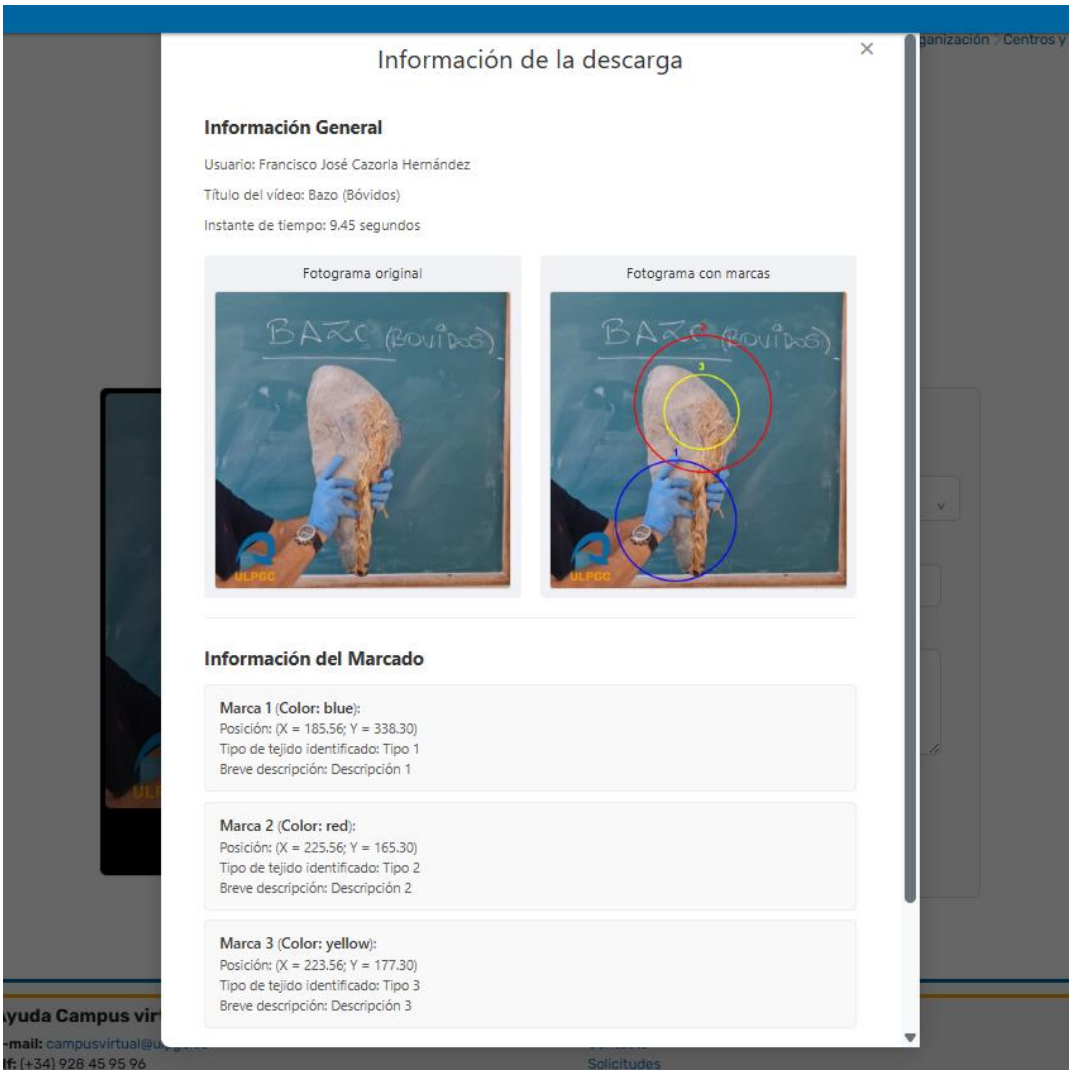


Figura 50. Vista de la interfaz modal de informes con datos compilados y fotogramas.

Fragmento de código representativo

El núcleo de esta funcionalidad reside en la función `showReportModal()`. Este bloque de código JavaScript es responsable de recopilar todos los datos de la sesión de marcado, generar el contenido dinámico de la ventana modal y gestionar su visualización.

```
// Función para mostrar la ventana modal con el informe compilado
async function showReportModal() {
    // 1. Configurar el título de la modal según el contexto de Moodle
    modalTitle.textContent = (MOODLE_RESOURCE_TYPE === 'page')
        ? 'Información de la descarga'
        : 'Entregar informe';

    // 2. Rellenar la información general (usuario, vídeo, tiempo)
    reportUser.textContent = getMoodleUserName();
    reportVideoTitle.textContent = currentVideoTitle;
    reportMarkTime.textContent = `${mainVideoPlayer.currentTime.toFixed(2)} segundos`;

    // 3. Capturar y mostrar los fotogramas (original y con marcas)
    if (mainVideoPlayer.readyState >= 2) { // Asegura que el vídeo esté listo
        const originalFrameDataUrl = captureFrame(mainVideoPlayer, false);
        const markedFrameDataUrl = captureFrame(mainVideoPlayer, true);
        originalFrameImage.src = originalFrameDataUrl || '';
        markedFrameImage.src = markedFrameDataUrl || '';
    }

    // 4. Llenar la lista de anotaciones y configurar los botones del pie de página
    populateMarksList();
    setupModalFooterButtons();

    // 5. Mostrar la ventana modal
    reportModalOverlay.style.display = 'flex';
}
```

Este bloque se descompone en cinco pasos:

1. **Configuración del título y botones:** Se establece el texto del encabezado y de los botones de la modal de forma condicional, basándose en la variable `MOODLE_RESOURCE_TYPE`.
2. **Recopilación de metadatos:** Se invocan funciones auxiliares para obtener el nombre del usuario (`getMoodleUserName()`) y se recuperan el título del vídeo y el instante de tiempo de las variables de estado de la aplicación.
3. **Captura de fotogramas:** Se verifica que el vídeo tenga datos disponibles (`readyState >= 2`) antes de llamar a la función `captureFrame()`. Esta función, no mostrada en el fragmento, se encarga de dibujar el vídeo en un `canvas` y devolver una URL de datos (`data:URL`) de la imagen, que se asigna al atributo `src` de los elementos `<img>` correspondientes.
4. **Población de contenido dinámico:** Se llaman a las funciones `populateMarksList()` y `setupModalFooterButtons()` (no mostradas) que se encargan de generar dinámicamente la lista de anotaciones y los botones de acción (descarga o entrega) en el pie de la modal.
5. **Visualización:** Finalmente, se modifica el estilo CSS de la superposición de la modal para hacerla visible.

5.2.2. Generación de informes en formatos PDF y ZIP

Una vez que la interfaz modal presenta la información compilada, el siguiente paso es implementar las funcionalidades de descarga. Para ello, se integran librerías JavaScript del lado del cliente que permiten generar dinámicamente archivos en formatos estándar: `jsPDF` para la creación de documentos PDF y `JSZip` para el empaquetado de múltiples ficheros.

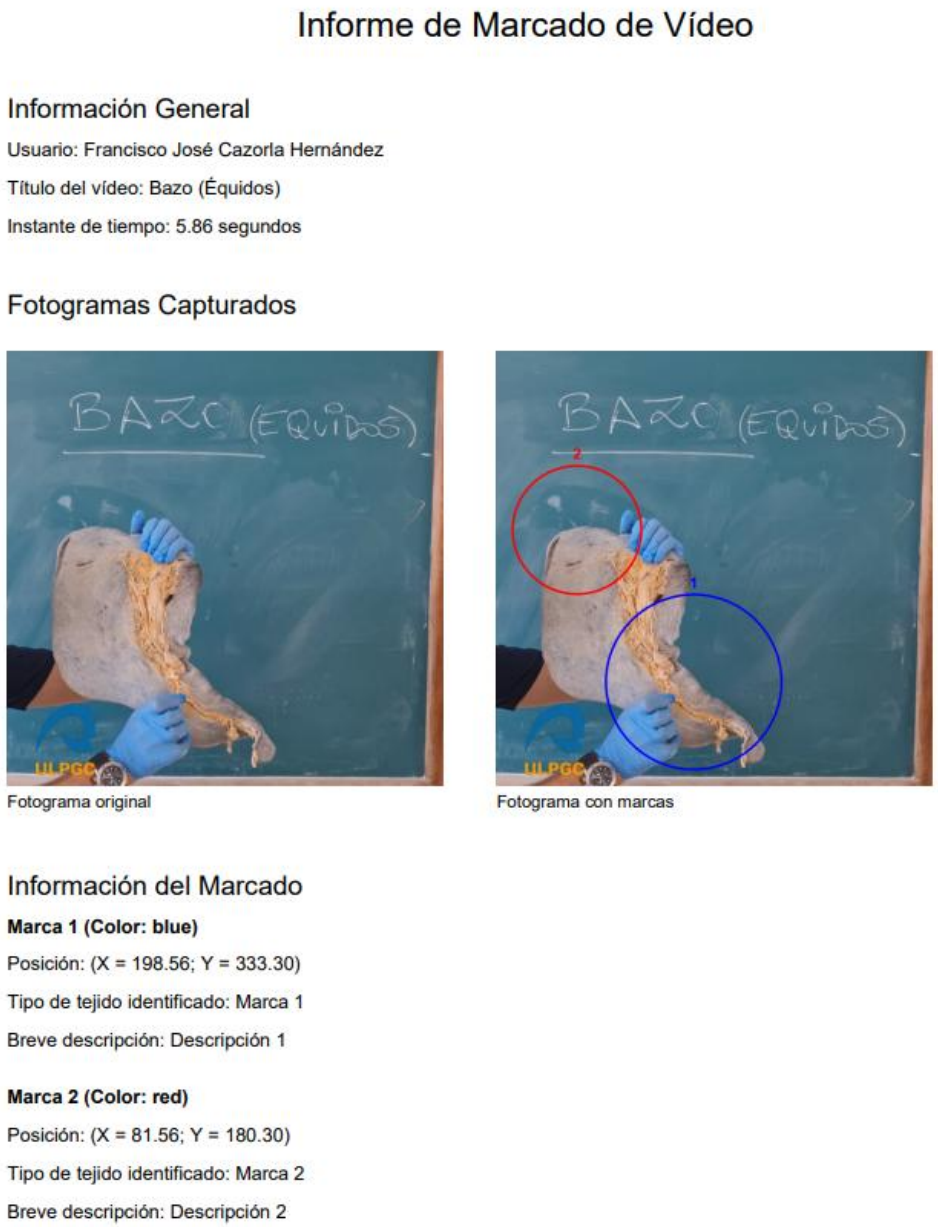
La implementación se centra en los botones del pie de la modal. Al pulsar "*Descargar PDF*", se genera un informe estructurado que incluye los metadatos de la sesión, los fotogramas capturados y un listado detallado de todas las anotaciones. El botón "*Descargar Imagen*" permite obtener únicamente el fotograma con las marcas superpuestas. Finalmente, la opción "*Descargar ZIP*" empaqueta la imagen original, la imagen marcada, un informe en formato de texto plano (`.txt`) y un archivo `informacion.json` con todos los datos de la sesión en un formato estructurado para su posible procesamiento automatizado.

Evolución del desarrollo

El desarrollo de esta funcionalidad comienza con la integración de las librerías externas. Para asegurar su disponibilidad, se incluyen sus respectivos servidores de CDN (*Content Delivery Network*) [https://aws.amazon.com/es/what-is/cdn/] en la cabecera del documento HTML de la herramienta, ofreciendo una mayor facilidad y velocidad de acceso.

La generación del PDF presenta el mayor desafío técnico. Se diseña una función, `downloadPdf()`, que utiliza la API de `jsPDF` para construir el documento de forma programática. Se define una estructura con márgenes, tamaños de fuente y saltos de línea para asegurar la legibilidad. Un aspecto a tener en cuenta es la inserción de las imágenes de los fotogramas; para evitar la distorsión visual observada en las primeras pruebas, se calcula la altura de la imagen de forma proporcional a su anchura fija, manteniendo así la relación de aspecto original del vídeo. El texto de las anotaciones se formatea para destacar los títulos de las marcas en negrita y se alinea correctamente, como se puede apreciar en el resultado final de la figura 51.

Para la funcionalidad del archivo ZIP, se implementa la función `downloadZip()`. Esta utiliza `JSZip` para crear un paquete en memoria. Se añaden las imágenes (original y marcada) a partir de sus `data:URL` y se generan los ficheros de texto (`.txt` y `.json`), antes de compilar y ofrecer el `.zip` para su descarga, como se ve reflejado en la figura 52.



Informe de Marcado de Vídeo

Información General

Usuario: Francisco José Cazorla Hernández

Título del vídeo: Bazo (Équidos)

Instante de tiempo: 5.86 segundos

Fotogramas Capturados

Fotograma original

Fotograma con marcas

Información del Marcado

Marca 1 (Color: blue)

Posición: (X = 198.56; Y = 333.30)

Tipo de tejido identificado: Marca 1

Breve descripción: Descripción 1

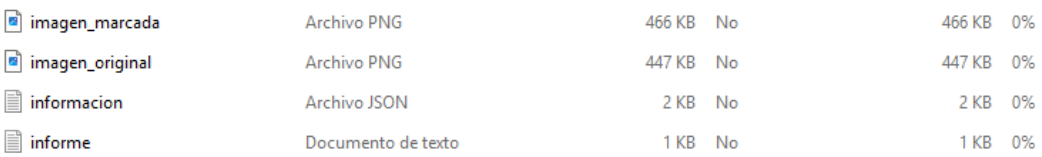
Marca 2 (Color: red)

Posición: (X = 81.56; Y = 180.30)

Tipo de tejido identificado: Marca 2

Breve descripción: Descripción 2

Figura 51. Ejemplo de informe generado en formato PDF.



imagen_marcada	Archivo PNG	466 KB	No	466 KB	0%
imagen_original	Archivo PNG	447 KB	No	447 KB	0%
informacion	Archivo JSON	2 KB	No	2 KB	0%
informe	Documento de texto	1 KB	No	1 KB	0%

Figura 52. Contenido del archivo ZIP generado, mostrando los ficheros de imagen, TXT y JSON.

Fragmento de código representativo

Las funciones `downloadPdf()` y `downloadZip()` encapsulan la lógica de generación de archivos. El siguiente fragmento muestra la estructura de `downloadZip()`, que orquesta la creación de los distintos ficheros y su empaquetado.

```
// Función para descargar un archivo ZIP con todos los contenidos
async function downloadZip(submitToMoodle = false) {
    if (typeof JSZip === 'undefined') {
        alert('La librería JSZip no está cargada.');
        return;
    }

    const zip = new JSZip();
    const baseFileName = generateBaseFileName();

    // 1. Añadir imágenes (original y marcada) al ZIP
    const originalImgSrc = originalFrameImage.src;
    if (originalImgSrc && originalImgSrc.startsWith('data:')) {
        zip.file('imagen_original.png', originalImgSrc.split(',')[1], { base64: true
});
    }

    const markedImgSrc = markedFrameImage.src;
    if (markedImgSrc && markedImgSrc.startsWith('data:')) {
        zip.file('imagen_marcada.png', markedImgSrc.split(',')[1], { base64: true });
    }

    // 2. Añadir archivo de texto plano (.txt)
    const infoContentTxt = generateReportTextContent();
    zip.file('informe.txt', infoContentTxt);

    // 3. Añadir archivo de datos estructurados (.json)
    const infoContentJson = generateReportJsonContent();
    zip.file('informacion.json', JSON.stringify(infoContentJson, null, 2));

    // 4. Generar el archivo ZIP y ofrecerlo para descarga o entrega
    const content = await zip.generateAsync({ type: "blob" });

    if (submitToMoodle) {
        // Lógica para la entrega en Tarea de Moodle (se verá en el siguiente apartado)
        await submitFileToMoodle(content, `${baseFileName}.zip`);
    } else {
        saveAs(content, `${baseFileName}.zip`); // Usa FileSaver.js
        alert('Archivo ZIP generado y descargado.');
    }
}
```

Este bloque de código se puede descomponer en cuatro acciones principales:

1. **Inclusión de imágenes**: Se obtienen las imágenes de los fotogramas desde sus `data:URL`, se extrae la parte codificada en `Base64` y se añaden al zip como archivos `.png`.
2. **Generación del informe TXT**: Se invoca la función `generateReportTextContent()` para crear una cadena de texto formateada con toda la información de la sesión, que se guarda en `informe.txt`.
3. **Generación del informe JSON**: De manera análoga, la función `generateReportJsonContent()` construye un objeto JavaScript estructurado que se convierte a una cadena JSON y se guarda en `informacion.json`.

4. **Compilación y descarga**: Finalmente, se genera el archivo `.zip` como un objeto tipo fichero de datos planos inmutables o *Blob* y se utiliza la librería `FileSaver.js` (a través de la función `saveAs`) para iniciar su descarga en el navegador del usuario, o se pasa a la función de entrega si el contexto es una *Tarea*.

5.2.3. Entrega automatizada en informes de Moodle.

La funcionalidad más avanzada del módulo de informes es la capacidad de entregar el trabajo del alumno directamente en una actividad de tipo *Tarea* de Moodle. Dado que la herramienta opera íntegramente en el lado del cliente y no tiene acceso a las API de Moodle, esta entrega se implementa mediante una simulación de la interacción del usuario con la interfaz de subida de archivos de la plataforma.

Cuando el usuario hace clic en el botón "*Confirmar entrega*" dentro de la ventana modal, se activa una secuencia de comandos que localiza los elementos del DOM de Moodle responsables de la subida de archivos y los manipula programáticamente para cargar el `.zip` generado. Este proceso está diseñado para ser lo más robusto posible, aunque su éxito depende de la estructura HTML de la interfaz de Moodle.

Evolución del desarrollo

La implementación de esta funcionalidad requiere un análisis detallado del DOM de la página de entrega de tareas de Moodle. El primer paso es identificar los selectores CSS de los elementos clave: el botón para añadir un fichero, el campo de entrada de tipo `file` (que normalmente está oculto) y el botón final para confirmar la subida del archivo seleccionado.

Una vez identificados, se diseña la función `submitFileToMoodle()`. La principal complicación es la naturaleza asíncrona de la interfaz de Moodle, que carga el selector de archivos dinámicamente. Para solventarlo, se implementa un bucle con `setTimeout` que intenta localizar el campo de entrada de archivo a intervalos regulares hasta que esté disponible en el DOM.

Una vez localizado, es necesario crear un objeto `DataTransfer` para simular un arrastre y suelta, y luego disparar manualmente un evento `change` sobre el `input`. Esto es fundamental para que los scripts de Moodle detecten la selección del archivo y procedan a mostrar el botón de subida final. Un segundo bucle de espera se encarga de localizar este último botón para, finalmente, simular un clic sobre él y completar el proceso. Durante todo el flujo, se muestran alertas al usuario para informarle del progreso, como se ve en la figura 53.

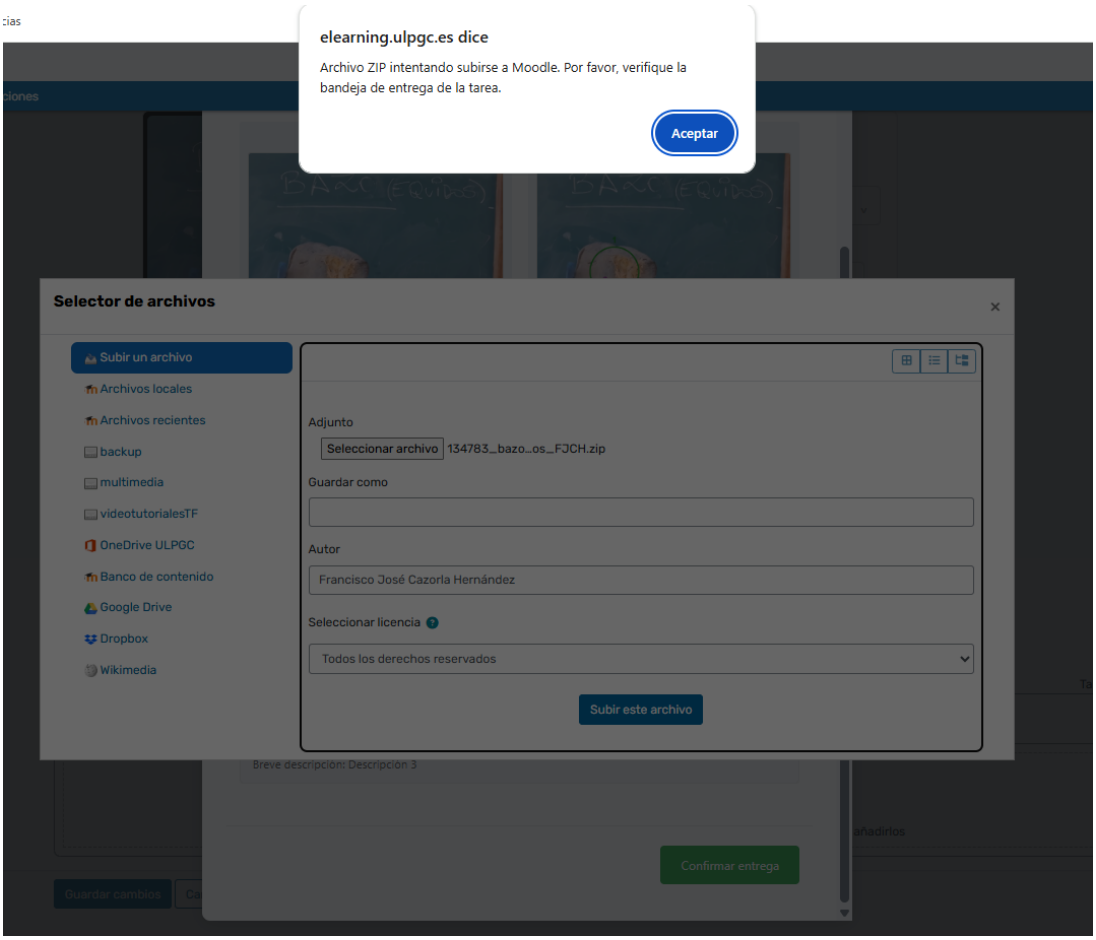


Figura 53. Notificación de la herramienta durante el proceso de subida automatizada del informe a Moodle.

Fragmento de código representativo

La función `submitFileToMoodle()` encapsula toda la lógica de interacción con la interfaz de Moodle. El siguiente fragmento ilustra los pasos clave de este proceso de automatización.

```
// Lógica para subir un archivo (Blob) a la bandeja de entrega de Moodle
async function submitFileToMoodle(fileBlob, fileName) {
    const file = new File([fileBlob], fileName, { type: "application/zip" });

    // 1. Simular clic en el botón "Añadir fichero" de Moodle
    const addFileButton = document.querySelector('.fp-btn-add');
    if (!addFileButton) {
        alert('Error: No se encontró el botón "Añadir fichero" de Moodle.');
        return;
    }
    addFileButton.click();

    // 2. Esperar y localizar el input de archivo de Moodle
    const fileInput = await waitForElement('input[name="repo_upload_file"]');
    if (!fileInput) {
        alert('Error: No se encontró el campo de subida de archivo de Moodle.');
        return;
    }

    // 3. Inyectar el archivo en el input y disparar evento de cambio
    const dataTransfer = new DataTransfer();
    dataTransfer.items.add(file);
    fileInput.files = dataTransfer.files;
    const changeEvent = new Event('change', { bubbles: true });
    fileInput.dispatchEvent(changeEvent);

    // 4. Esperar y hacer clic en el botón "Subir este archivo"
    const uploadButton = await waitForElement('.fp-upload-btn');
    if (!uploadButton) {
        alert('Error: No se encontró el botón "Subir este archivo" de Moodle.');
        return;
    }

    reportModalOverlay.style.display = 'none'; // Ocultar nuestra modal
    uploadButton.click();
    alert('Archivo ZIP intentando subirse a Moodle. Por favor, verifique la bandeja de
entrega.');
}

// Función auxiliar para esperar a que un elemento aparezca en el DOM
function waitForElement(selector, timeout = 2000) {
    return new Promise(resolve => {
        const interval = 100;
        const endTime = Date.now() + timeout;

        const check = () => {
            const element = document.querySelector(selector);
            if (element) {
                resolve(element);
            } else if (Date.now() < endTime) {
                setTimeout(check, interval);
            } else {
                resolve(null); // Timeout
            }
        };
        check();
    });
}
```

Este bloque de código se puede descomponer en cuatro pasos principales, orquestados de forma asíncrona:

1. **Activación del selector de archivos:** Se localiza y se hace clic en el botón que Moodle utiliza para iniciar el proceso de subida de ficheros.
2. **Localización del campo de entrada:** Mediante una función auxiliar `waitForElement`, el script espera activamente a que Moodle genere y añada al DOM el elemento `<input type="file">`.
3. **Inyección del archivo**: Se crea un objeto `File` a partir del *Blob* del ZIP. Se utiliza la interfaz `DataTransfer` para asignarlo al `input` localizado y se dispara un evento `change` para notificar a Moodle de la "selección".
4. **Confirmación de la subida:** El script espera nuevamente a que aparezca el botón de confirmación de subida y simula un clic sobre él, cerrando el ciclo de entrega.

5.2.4. Valoración de métricas para la generación y entrega de informes

La implementación del módulo de informes representa una evolución en la complejidad funcional de la herramienta. Esta sección cuantifica y valora el proceso de desarrollo asistido por IA, analizando cómo el modelo Gemini 2.5 Flash gestionó tareas de mayor sofisticación, como la generación de archivos y la interacción con interfaces externas. La tabla 5 resume las métricas clave de este proceso, desglosadas por los bloques funcionales definidos: Activación e Interfaz Modal (AIM), Generación de Archivos (GA) y Entrega a Moodle (EM).

Métrica/ Componente	Activación e Interfaz Modal (AIM)	Generación de Archivos (GA)	Entrega a Moodle (EM)
Prompts totales	6	4	1
Tokens totales de entrada (*input*)	1.994	1.111	466
Tokens totales de salida (*output*)	25.727	28.540	7.064
Tokens promedio por entrada	332,33	277,75	466,00
Tokens promedio por salida	4.287,83	7.135,00	7.064,00
Respuestas aceptadas	3	3	1
Respuestas no aceptadas	3	1	0
Tasa de aceptación (%)	50% (3/6)	75% (3/4)	100% (1/1)
Tiempo medio de respuesta (s)			
- Respuestas < 5000 tokens	33.01 (4 respuestas)	(Ninguna)	(Ninguna)
- Respuestas 5000-7000 tokens	44,88 (2 respuestas)	47,4 (2 respuestas)	(Ninguna)
- Respuestas > 7000 tokens	(Ninguna)	53,2 (2 respuestas)	51,82 (1 respuesta)
Complejidad del prompt (Tokens entrada)	Media	Media	Alta
Complejidad de la respuesta (Tokens salida)	Alta	Muy Alta	Muy Alta

Tabla 5. Análisis de Interacción con Gemini 2.5 Flash para el Módulo de Informes. Fuente: Elaboración propia

Análisis general de la asistencia por IAG

El desarrollo del módulo de informes, documentado a través de 11 interacciones, demuestra la viabilidad de emplear la IAG para construir funcionalidades complejas de forma iterativa. Un análisis de los datos revela una correlación directa entre la sofisticación de la tarea y los recursos computacionales requeridos. El **volumen de tokens de salida** y el **tiempo de respuesta** aumentan notablemente a medida que el código generado se vuelve más complejo. Como se detalla en la tabla 5, el tiempo medio de respuesta para soluciones ligeras (menos de 5000 tokens) es de aproximadamente 33 segundos, mientras que para respuestas de más de 7000 tokens, este tiempo supera los 52 segundos.

La **tasa de aceptación** variable (entre el 50% y el 100%) refleja la naturaleza del desarrollo iterativo: las respuestas no aceptadas no constituyeron fallos, sino versiones iniciales que necesitaron prompts de seguimiento para su refinamiento. Esto refuerza el rol de la IAG como un aistente que acelera la creación de código base, pero que depende de una guía precisa del desarrollador para alcanzar la solución final.

Resultados específicos por bloque funcional

- **Activación e Interfaz Modal (AIM):** Este bloque inicial, enfocado en la interfaz de usuario, presentó la **tasa de aceptación más baja (50%)**. Esto se debe a que las tareas de diseño visual y posicionamiento preciso en la interfaz son subjetivas y requirieron múltiples iteraciones de corrección para lograr el resultado deseado. Aunque el modelo generó la estructura HTML/CSS y la lógica JavaScript fundamental con eficiencia, los ajustes finos necesitaron una guía constante.
- **Generación de Archivos (GA):** Con una **tasa de aceptación del 75%**, este bloque demostró una mayor eficacia del modelo. La tarea de integrar librerías externas como jsPDF y JSZip fue gestionada con éxito. La única respuesta no aceptada correspondió a un refinamiento en el formato del PDF, una tarea con un componente visual que, de nuevo, requirió una iteración adicional. La capacidad del modelo para generar artefactos de datos estructurados (`.txt`, `.json`) fue notablemente alta.
- **Entrega a Moodle (EM):** Esta funcionalidad, implementada en un único y complejo prompt, alcanzó una **tasa de aceptación del 100%**. A pesar de ser la tarea más exigente, el prompt fue formulado con un alto grado de detalle técnico, especificando el flujo de interacción con el DOM de Moodle. La capacidad del modelo para generar una solución funcional y robusta en un solo intento subraya su potencial para abordar problemas de ingeniería complejos cuando se le proporciona una instrucción clara y bien definida.

5.3. Aplicabilidad del sistema de informes

Una vez implementado el módulo de generación y entrega de informes, es fundamental validar su robustez y aplicabilidad en escenarios de uso real. Este apartado detalla las pruebas de integración y los mecanismos implementados para asegurar que el sistema funcione de manera fiable, gestione los errores de forma controlada y facilite la labor tanto del alumnado como del profesorado.

5.3.1. Nomenclatura dinámica y estructurada de archivos

Una de las claves para la aplicabilidad del sistema en un contexto docente es la capacidad de gestionar múltiples entregas de forma organizada. Para ello, se implementa un algoritmo de nomenclatura dinámica para todos los archivos generados. Este sistema extrae tres piezas de información para construir un nombre de archivo único y descriptivo:

- **ID del recurso:** Se localiza el enlace del recurso actual (*Tarea* o *Página*) en la ruta de navegación de Moodle y se extrae su identificador numérico único.

- **Título del vídeo:** Se toma el título del vídeo que está siendo anotado y se abrevia, eliminando espacios y caracteres especiales para hacerlo compatible con los nombres de archivo.
- **Iniciales del usuario:** Se obtiene el nombre completo del usuario que ha iniciado sesión en Moodle y se generan sus iniciales.

El resultado es un nombre de archivo estandarizado que permite al profesorado identificar de un vistazo la tarea, el contenido y el alumno correspondiente, simplificando enormemente la clasificación y evaluación de las entregas. Esta convención se aplica a todos los artefactos descargables, garantizando la coherencia. La figura 54 muestra un ejemplo del archivo ZIP generado, donde el nombre *134783_bazocnidos_FJCH.zip* refleja el ID del recurso (134783), el vídeo ("Bazo Cánidos") y las iniciales del usuario ("Francisco José Cazorla Hernández").

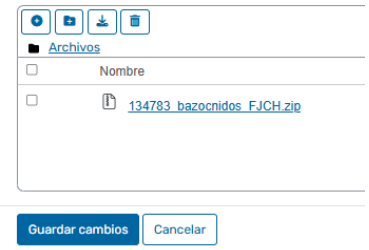


Figura 54. Ejemplo de archivo ZIP con nomenclatura dinámica y estructurada.

La implementación de esta nomenclatura automática no solo mejora la experiencia del profesorado, sino que también establece una buena base para futuros sistemas de procesamiento por lotes, donde los archivos podrían ser analizados o clasificados basándose en la información contenida en su nombre.

5.3.2. Gestión de errores en la generación de informes

La fiabilidad del sistema depende de su capacidad para gestionar fallos, especialmente aquellos relacionados con dependencias externas. La generación de archivos PDF y ZIP se apoya en librerías (*jsPDF*, *JSZip*) que se cargan desde un CDN. Si por algún motivo, como problemas de red o un bloqueo del CDN, estas librerías no están disponibles en el momento en que el usuario intenta generar un informe, la herramienta debe reaccionar de forma controlada.

Para validar este comportamiento, se simuló un escenario en el que las librerías no se cargan correctamente. Al intentar generar un informe, el sistema captura el error y, en lugar de fallar silenciosamente o bloquear la interfaz, muestra una notificación clara al usuario, como se observa en la figura 55. Este mecanismo de alerta informa del problema específico

y permite al usuario o al administrador del curso tomar las medidas oportunas, garantizando la continuidad de la experiencia sin interrupciones.

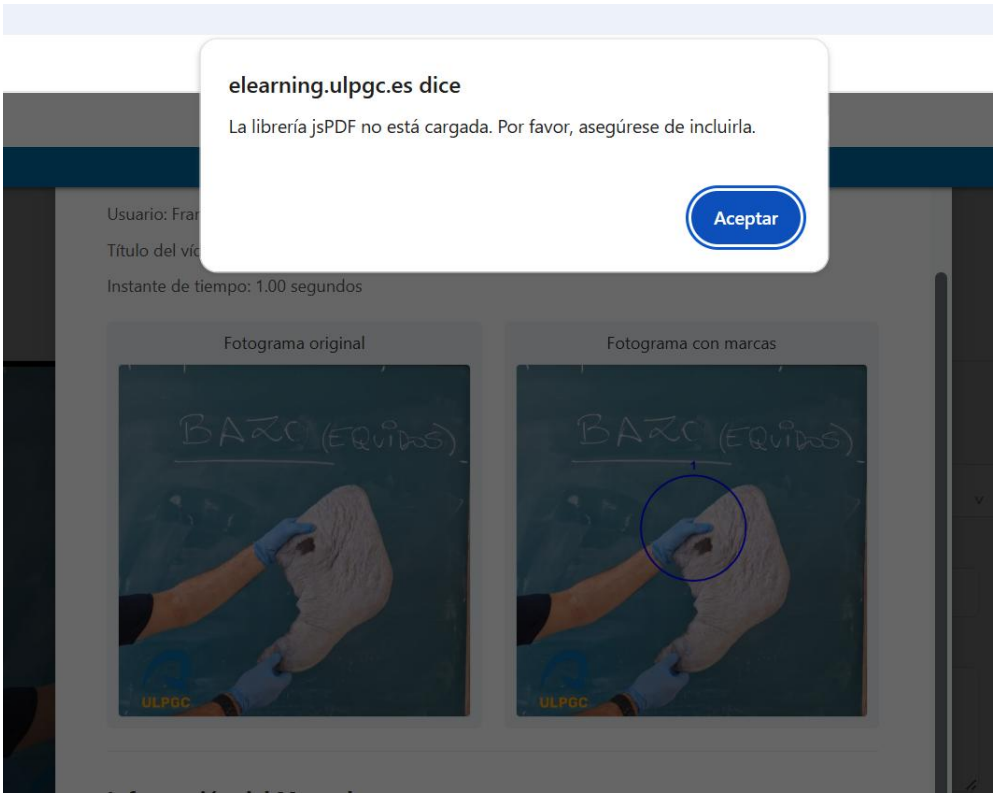


Figura 55. Notificación de error por dependencia de librería no cargada.

Esta capacidad de gestionar errores es importante para la robustez de la aplicación, ya que asegura una experiencia de usuario continua, incluso ante imprevistos técnicos. La retroalimentación clara que se proporciona al usuario es un pilar fundamental para la usabilidad y el mantenimiento de la herramienta en un entorno educativo.

5.3.3. Verificación de la integridad de los datos entregados

Finalmente, se valida que los datos generados y empaquetados para la entrega sean completos y estructuralmente coherentes. El objetivo es asegurar que la información no solo sea útil para la revisión manual por parte del profesorado, sino que también sea directamente consumible por otras herramientas para enriquecer la experiencia de aprendizaje.

El archivo `informacion.json`, incluido en el paquete ZIP, es el elemento central de esta validación. Como se muestra en la figura 56, este archivo contiene toda la información de la sesión de marcado en un formato JSON bien definido. Incluye una sección `informacionGeneral` con los metadatos de la sesión y un array anotaciones, donde cada objeto representa una marca con su número, tiempo, color, posición, radio y los datos de los campos del formulario.

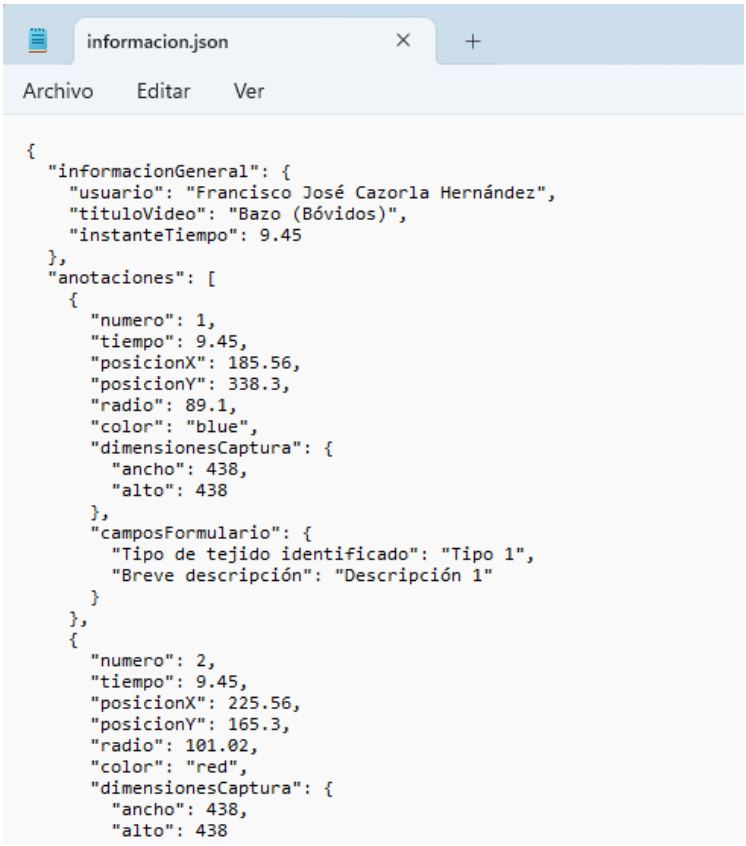


Figura 56. Ejemplo de archivo informacion.json estructurado con la información completa.

La estructura normalizada y en español de este archivo es fundamental, pues no solo garantiza la integridad de los datos, sino que constituye la base para la siguiente fase del proyecto. La información contenida en este fichero JSON será utilizada para alimentar y configurar dinámicamente un vídeo interactivo mediante la herramienta H5P. De este modo, las anotaciones individuales de los estudiantes se transforman en un recurso de aprendizaje colaborativo, tal y como se detallará en el próximo capítulo.

Capítulo 6: Integración con H5P y Creación de Vídeos Interactivos

En este capítulo se describe el desarrollo de la funcionalidad para generar paquetes H5P a partir de las anotaciones de los usuarios. Se analiza la estructura interna de este formato, se detalla el diseño y la implementación del generador de paquetes y se presentan los resultados de la validación del contenido interactivo creado.

6.1. Metodología y diseño del generador de paquetes H5P

Una vez consolidada la capacidad del sistema para documentar el trabajo del estudiante mediante la generación de informes estructurados, el proyecto aborda una fase final de integración cuyo objetivo es transformar dicha evidencia individual en un recurso de aprendizaje interactivo y reutilizable. Este paso es fundamental para cerrar el ciclo pedagógico, permitiendo que las anotaciones del alumno ofrezcan un material educativo original y sirvan como base para el estudio y la colaboración de otros compañeros.

Para materializar este objetivo, se seleccionó el estándar H5P (*HTML5 Package*) como formato de destino, dada su amplia adopción en plataformas de gestión del aprendizaje (LMS), incluyendo Moodle, y su naturaleza basada en tecnologías web abiertas. Asimismo, se empleó la herramienta de escritorio Lumi [96] para la validación y revisión de los paquetes generados, facilitando un ciclo de desarrollo ágil sin depender de subidas constantes a la plataforma.

La implementación de esta funcionalidad se articula mediante el desarrollo de un **generador de paquetes H5P**, una herramienta web autónoma que opera íntegramente en el lado del cliente. Esta aplicación se ha diseñado para tomar como entrada el archivo de vídeo original (`.mp4`) y los ficheros `informacion.json` (que contienen las anotaciones estructuradas generadas en el capítulo anterior) y producir como salida un fichero `.h5p` funcional y estandarizado.

El diseño y desarrollo de este generador se fundamenta en un proceso de ingeniería inversa y adaptación. Por ello, el primer paso metodológico consiste en un análisis detallado de la estructura interna de un paquete H5P de tipo *Vídeo Interactivo*, con el fin de identificar los componentes clave y, en particular, el formato del fichero `content/content.json` que gobierna las interacciones.

6.1.1. Análisis de la estructura de paquetes H5P

El primer paso metodológico para el desarrollo del generador consiste en un proceso de ingeniería inversa para comprender en detalle la estructura y los requisitos de un paquete H5P de tipo *Vídeo Interactivo*. Para ello, se genera un paquete de referencia utilizando Lumi como herramienta de autoría. En este proceso, se configura un vídeo interactivo simple, al que se le añade una única interacción de tipo texto. Esta acción es fundamental para asegurar que el paquete resultante contenga los recursos multimedia y las librerías base, además de la estructura de datos específica que define las anotaciones sobre el vídeo.

Un fichero `.h5p` es, en esencia, un archivo comprimido en formato ZIP que encapsula todos los recursos necesarios (HTML, JavaScript, CSS, imágenes, vídeos y ficheros de configuración) para la visualización y funcionamiento del contenido interactivo. Para examinar su estructura interna, se modifica la extensión del archivo de referencia de `.h5p` a `.zip` y se procede a su descompresión.

El análisis de los ficheros y directorios resultantes que podemos ver en la figura 57 muestra una arquitectura estandarizada que incluye los siguientes componentes clave:

- **Fichero `h5p.json`**: Actúa como el manifiesto principal del paquete. Define metadatos esenciales, como el título, la librería principal que se debe ejecutar (`H5P.InteractiveVideo`) y un listado de todas las librerías dependientes con sus respectivas versiones.
- **Directorio `content`**: Contiene los activos específicos de esta instancia de contenido, como los vídeos o imágenes utilizados. Su elemento más importante es el fichero `content.json`.
- **Directorios de librerías**: Cada librería H5P (ej. `H5P.InteractiveVideo-1.26`, `H5P.Video-1.6`, `H5P.JoubelUI-1.3`) reside en su propio directorio, que contiene su código fuente (JavaScript), hojas de estilo (CSS) y otros activos necesarios para su funcionamiento.

De todos los componentes, el fichero `content/content.json` se identifica como el núcleo de la lógica de la herramienta. Este fichero JSON contiene la configuración detallada de la instancia de vídeo interactivo, incluyendo la ruta al fichero de vídeo (relativa a la carpeta `content`) y un array denominado `interactions`, que almacena un conjunto de objetos donde cada uno representa una de las interacciones (anotaciones) añadidas al vídeo, con sus coordenadas, temporización, apariencia y contenido.

Una vez identificados estos componentes, el desafío técnico se centra en la generación programática de esta estructura de archivos y, en particular, en la correcta construcción del fichero `content/content.json` a partir de los datos de anotación del alumno.

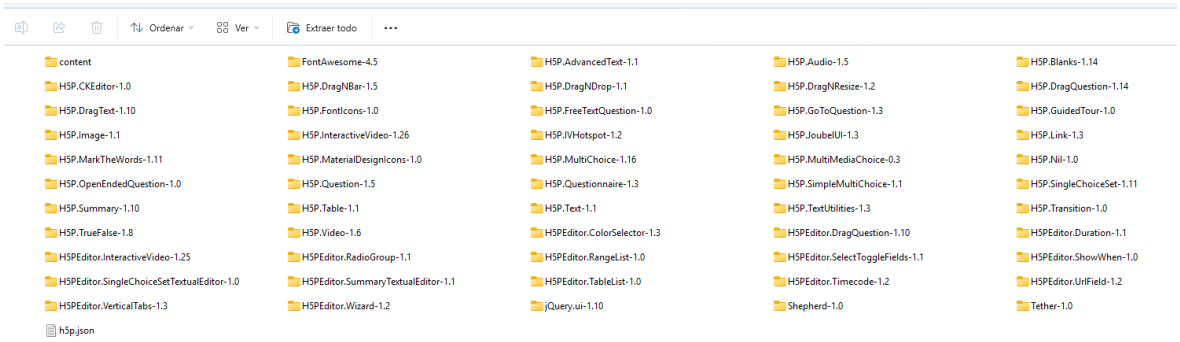


Figura 57. Estructura de directorios y ficheros de un paquete H5P de Vídeo Interactivo base.

6.1.2. Diseño conceptual de la herramienta de generación

Tras el análisis de la estructura de los paquetes H5P, se define el diseño conceptual de la herramienta de generación. El sistema se proyecta como una aplicación web autónoma, encapsulada en una única página HTML, que opera exclusivamente en el lado del cliente. Esta decisión arquitectónica es coherente con el resto del proyecto, garantizando que no se requieran dependencias de servidor, lo que maximiza la portabilidad, la privacidad (los archivos del usuario no se suben a ningún servidor externo) y la facilidad de uso.

El flujo de interacción con el usuario se planifica para ser intuitivo y guiado. Para ello, se elabora un boceto de la interfaz como el de la figura 58, que define los componentes funcionales necesarios y su disposición:

- **Área de entrada de datos:** Se diseña una sección principal donde el usuario (el profesorado) proporcionará los dos tipos de recursos necesarios:
    1. **El archivo de vídeo base (`.mp4`):** Un selector de fichero para el vídeo sobre el cual se montarán las interacciones.
    2. **Los archivos de anotaciones (`.json`):** Un segundo selector, con capacidad de selección múltiple, para cargar uno o varios ficheros informacion.json generados previamente por la herramienta de anotación.
- **Botón de acción principal:** Se plantea un botón claramente visible, "*Generar Paquete H5P*", que iniciará el proceso de compilación. Su estado inicial será deshabilitado para prevenir errores, activándose únicamente cuando se cumplan las condiciones de entrada (un vídeo y al menos un JSON seleccionados).
- **Panel de retroalimentación:** Se reserva un área en la interfaz para mostrar un registro (`log`) de las operaciones. Este panel informará al usuario en tiempo real sobre el estado del proceso, desde la carga de archivos hasta la finalización del paquete, proporcionando transparencia y ayudando a diagnosticar posibles incidencias.

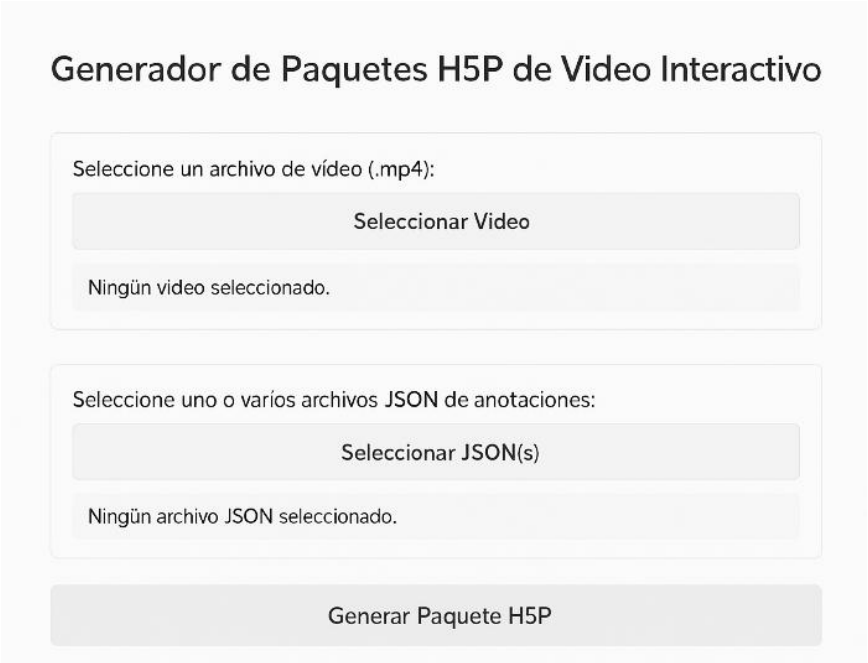


Figura 58. Boceto conceptual de la interfaz de usuario para el generador de paquetes H5P. Fuente: Generado con ChatGPT.

Este diseño conceptual, centrado en la usabilidad y la claridad del proceso, sienta las bases para la implementación técnica de la herramienta, asegurando que todos los requisitos funcionales estén contemplados antes de escribir el código.

6.1.3. Flujo de interacción del usuario

La herramienta de generación de paquetes H5P ha sido diseñada para ser utilizada por el profesorado a través de un flujo de trabajo sencillo y guiado, detallando a continuación los pasos que el usuario debe seguir para crear un vídeo interactivo a partir de las anotaciones de los alumnos.

Preparación de los archivos

Antes de utilizar la herramienta, el profesor debe tener localizados los siguientes ficheros en su equipo:

- El **archivo de vídeo original** (en formato `.mp4`) que se utilizó en la actividad de marcado.
- Uno o varios **archivos de datos** (en formato `.json`) que contienen las anotaciones. Estos son los ficheros `informacion.json` que se obtienen al descargar el ZIP generado por la herramienta de marcado del capítulo anterior.

Carga de los recursos en la herramienta

Con la página del generador abierta en el navegador, el usuario interactúa con los dos selectores de la interfaz:

- **Cargar el vídeo:** Hacer clic en el botón "*Seleccionar Video*" y elegir el archivo `.mp4` correspondiente. El nombre del fichero aparecerá en el área de estado para confirmar su selección.
- **Cargar las anotaciones:** Hacer clic en el botón "Seleccionar JSON(s)" y elegir uno o varios archivos `informacion.json`. La herramienta permite la selección múltiple, y los nombres de todos los ficheros cargados se mostrarán en su respectiva área de estado.

Generación del paquete H5P

Cuando se han cargado tanto el vídeo como al menos un archivo JSON, el botón "*Generar Paquete H5P*" se habilitará automáticamente. Al hacer clic en este botón:

- El sistema iniciará el proceso de compilación, mostrando el progreso en el panel de registro.
- Una vez finalizado, el navegador iniciará automáticamente la descarga de un único fichero con la extensión `.h5p`.

Recomendación de verificación y uso

El fichero `.h5p` descargado es un recurso autónomo y listo para ser utilizado. Se recomienda realizar una verificación final:

- Abrir el fichero `.h5p` con una herramienta compatible como Lumi para previsualizar el vídeo y confirmar que las interacciones (anotaciones) se muestran correctamente.
- Una vez verificado, el fichero puede ser subido directamente a Moodle o a cualquier otra plataforma compatible con H5P para ser compartido con los estudiantes.

Este flujo de trabajo, mostrado visualmente en la figura 59, simplifica al máximo el proceso de consolidación, permitiendo al profesorado crear recursos de aprendizaje enriquecidos sin necesidad de conocimientos técnicos sobre el formato H5P.

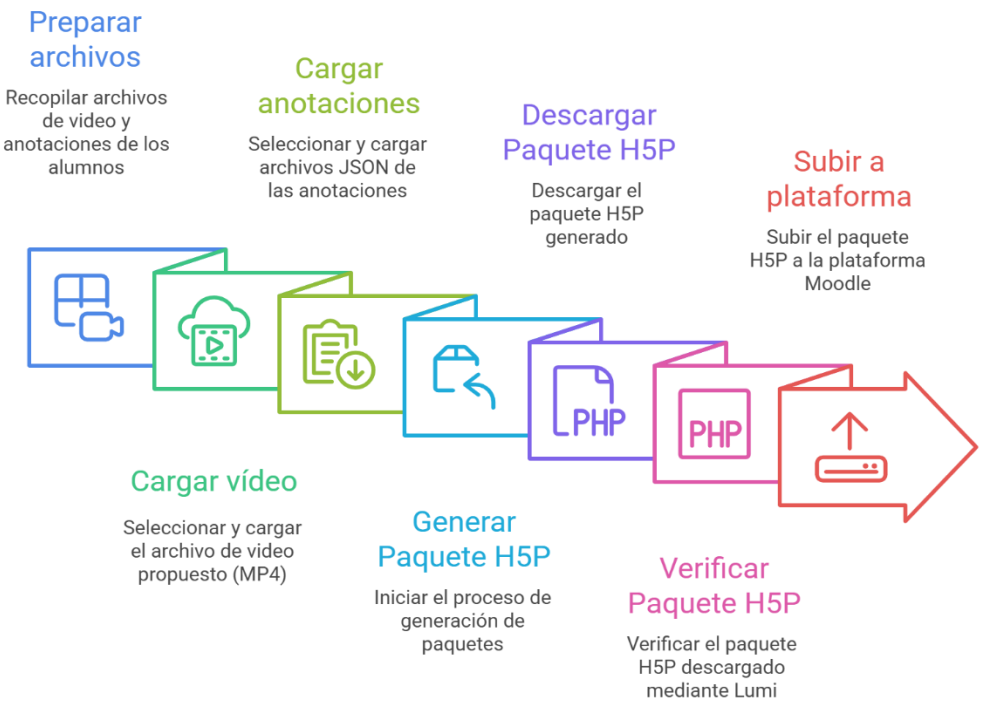


Figura 59. Flujo completo de uso de la herramienta de generación de paquetes H5P. Fuente: Generado con NapkinAI

6.2. Implementación técnica del generador

La implementación de la herramienta se lleva a cabo siguiendo el diseño conceptual y se basa enteramente en tecnologías del lado del cliente como se ha tratado hasta ahora. Para la manipulación y creación de archivos, empleamos dos de las librerías externas utilizadas en el sistema de generación de informes, a través de sus respectivas CDN:

- **JSZip:** componente central para empaquetar todos los recursos en un fichero `.h5p`.
- **FileSaver.js:** que nos permite guardar archivos en el cliente

El proceso de desarrollo se aborda de forma modular, comenzando por la replicación de la estructura de la plantilla H5P.

6.2.1. Replicación de la plantilla H5P y gestión de archivos

El primer paso en la lógica de generación consiste en construir una réplica en memoria de la estructura de archivos y directorios de un paquete H5P de *Vídeo Interactivo* funcional. Para ello, la herramienta se basa en una constante interna (`H5P_TEMPLATE_FILES`) que contiene un listado completo de todas las rutas de fichero que componen la plantilla base.

Para garantizar la integridad del paquete, este listado se generó de forma programática en lugar de manualmente, evitando posibles omisiones. Se utilizó un script de PowerShell [97] ejecutado sobre un directorio local que contenía una versión funcional de la plantilla H5P:

```
$basePath = Get-Location; Get-ChildItem -Path $basePath -Recurse | Where-Object { -
not $_.PSIsContainer } | ForEach-Object { $_.FullName.Replace($basePath,
"").Replace("\", "/") }
```

Este comando realiza lo siguiente:

```
$basePath = Get-Location
```

- Guarda en la variable `$basePath` la ruta del directorio actual desde el que se está ejecutando el script.

```
Get-ChildItem -Path $basePath -Recurse
```

- Obtiene todos los elementos (archivos y carpetas) dentro del directorio actual y de forma recursiva en todos los subdirectorios.

```
| Where-Object { -not $_.PSIsContainer }
```

- Filtra los resultados anteriores para excluir las carpetas, dejando solo los archivos.

```
| ForEach-Object { $_.FullName.Replace($basePath, "").Replace("\", "/") }
```

- Para cada archivo:
    1. Toma la ruta completa (`FullName`)
    2. Reemplaza la parte que corresponde al path base (`$basePath`) por una cadena vacía, dejando solo la ruta relativa
    3. Reemplaza las barras invertidas `\` (de Windows) por barras normales `/` (como en URLs)

Finalmente, devuelve una lista de rutas relativas de todos los archivos dentro del directorio actual y sus subdirectorios, como podemos ver en la figura 60.

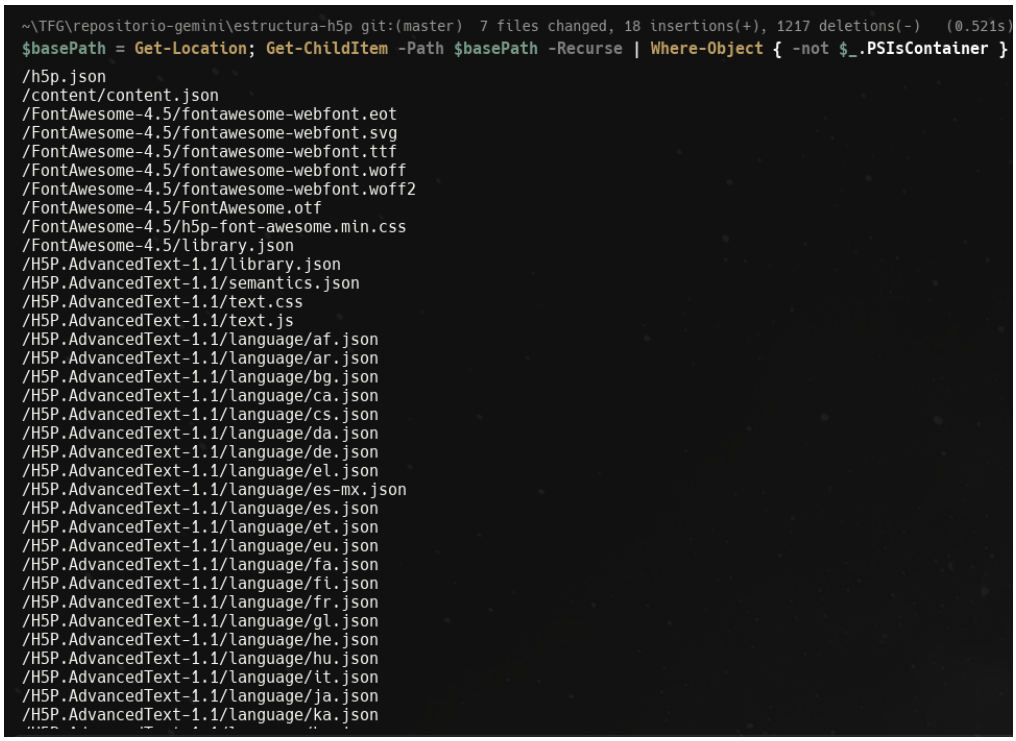


Figura 60. Salida de consola mostrando el listado de rutas relativas de los ficheros de la plantilla H5P.

En lugar de empaquetar estos ficheros junto a la herramienta, lo que aumentaría su tamaño y dificultaría su actualización, se opta por alojarlos en un repositorio público de GitHub [98]. Al iniciar el proceso de generación, la aplicación recorre el listado de rutas y recupera cada fichero de la plantilla de forma remota mediante la API fetch de JavaScript.

Para optimizar el tiempo de carga y evitar sobrecargar el servidor de GitHub con un gran número de peticiones simultáneas, la implementación no se realiza de forma secuencial. En su lugar, se aplica una estrategia de **carga concurrente por lotes**. Los ficheros se agrupan en pequeños bloques (por ejemplo, de 10 peticiones) que se ejecutan en paralelo. Una vez que un lote finaliza, se introduce un breve retardo antes de procesar el siguiente. Esto reduce significativamente el tiempo total de descarga frente a un método secuencial, sin comprometer la estabilidad.

Durante este proceso, la interfaz informa al usuario en tiempo real del estado de cada descarga. Los ficheros recuperados (texto como cadenas de caracteres y binarios como objetos *Blob*) se almacenan en memoria, listos para ser modificados y empaquetados en la fase posterior.

Fragmento de código representativo

El siguiente fragmento de código ilustra la función clave de esta lógica de carga:

```
// Función para cargar un único archivo de la plantilla desde GitHub
async function loadFile(filePath) {
    const fullUrl = GITHUB_RAW_BASE_URL + filePath;
    try {
        const response = await fetch(fullUrl);
        if (response.ok) {
            // Distingue entre archivos de texto y binarios para su correcto
almacenamiento
            const isTextFile = ['.json', '.js', '.css'].some(ext =>
filePath.endsWith(ext));
            const content = isTextFile ? await response.text() : await
response.blob();
            templateFilesData[filePath] = content; // Almacena en memoria
        } else {
            // Manejo de errores si el fichero no se encuentra o hay un problema de
servidor
            throw new Error(`Estado HTTP: ${response.status}`);
        }
    } catch (error) {
        // Manejo de errores de red o de la petición
        console.error(`Fallo al cargar ${filePath}:`, error);
        // Se podría añadir lógica para reintentos o para notificar el fallo
    }
}

// Lógica principal para procesar los archivos por lotes
const fileQueue = [...H5P_TEMPLATE_FILES];
const CONCURRENCY_LIMIT = 10;

while (fileQueue.length > 0) {
    const batch = fileQueue.splice(0, CONCURRENCY_LIMIT);
    const promises = batch.map(filePath => loadFile(filePath));

    await Promise.allSettled(promises); // Espera a que el lote actual termine

    if (fileQueue.length > 0) {

        await delay(50); // Pequeño retardo para no saturar el servidor

    }

}
```

Este bloque de código se puede descomponer en:

1. **Función `loadFile(filePath)`:** Es la unidad de trabajo fundamental. Recibe la ruta de un fichero de la plantilla, construye la URL completa para acceder al archivo *raw* en GitHub y ejecuta una petición `fetch`.

    - **Detección de tipo de archivo:** Se realiza una comprobación de la extensión del fichero para determinar si debe ser tratado como texto (`.json`, `.js`, etc.) o como un archivo binario. Esta distinción es importante, ya que los archivos de texto se leen con `response.text()`, mientras que los binarios (como fuentes o imágenes) se procesan como un *Blob* con `response.blob()` para preservar su integridad.

- **Almacenamiento en memoria:** El contenido recuperado, ya sea como texto o *Blob*, se almacena en un objeto `Map` global (`templateFilesData`), usando la ruta del fichero como clave.

2. **Función `loadTemplateConcurrently()`:** Orquesta el proceso de carga.

   - **Cola de trabajo:** Se crea una copia de la lista de ficheros (`fileQueue`) para poder manipularla sin alterar la constante original.
   - **Procesamiento por lotes:** Un bucle `while` se ejecuta mientras queden ficheros en la cola. En cada iteración, se extrae un lote de ficheros (definido por `CONCURRENCY_LIMIT`) y se crea un array de promesas, donde cada promesa corresponde a la ejecución de `loadFile` para un fichero del lote.
   - **Ejecución concurrente:** `Promise.allSettled(promises)` ejecuta todas las peticiones del lote de forma concurrente. `allSettled` se elige sobre `Promise.all` porque espera a que todas las promesas se resuelvan (ya sea con éxito o con fallo), asegurando que el proceso continúe incluso si la descarga de un fichero falla.
   - **Retardo (`delay`):** Tras procesar cada lote, se introduce una breve pausa para evitar realizar un número excesivo de peticiones a GitHub en un corto período de tiempo, lo que podría resultar en un bloqueo temporal por exceder los límites de tasa del servidor.

Una vez que esta función completa su ejecución, la aplicación dispone de una réplica en memoria de toda la plantilla H5P, preparada para la siguiente fase de personalización.

6.2.2. Transformación de anotaciones y modificación del `content.json`

Una vez que la plantilla H5P se encuentra cargada en memoria, el siguiente paso es personalizarla con los datos proporcionados por el usuario. El centro de esta operación es la modificación del fichero `content.json`, que define la estructura y el comportamiento del vídeo interactivo. Este proceso se realiza en dos etapas principales: la actualización de la ruta del vídeo y la inserción de las interacciones.

Primeramente, la herramienta lee el contenido del fichero `content.json` de la plantilla y lo convierte en un objeto JavaScript. A continuación, se genera un nombre de fichero único y aleatorio para el vídeo proporcionado por el usuario (ej. `videos/file-xxxxxxxx.mp4`), y esta nueva ruta se inyecta en la estructura del objeto JSON.

La fase más compleja es la transformación de las anotaciones. Para facilitar este proceso, la herramienta de marcado de vídeos fue adaptada para que su salida, el archivo ZIP con el fichero `informacion.json`, contenga directamente un array de objetos de interacción en el formato exacto que requiere H5P.

Fragmento de código representativo

El siguiente fragmento de código, extraído de dicha herramienta, ilustra esta función de transformación adaptada:

```
function generateReportJsonContent() {
    // 1. El método .map() itera sobre cada anotación guardada por el alumno
    const h5pInteractions = videoAnnotations.map(annotation => {
        // 2. Extrae los valores de los campos del formulario para la interacción
        const fields = annotation.camposFormulario || {};
        const briefDescription = fields['Breve descripción'] || '';
        const identifiedType = fields['Tipo de tejido identificado'] || '';
        const h5pX = (annotation.posicionX / annotation.dimensionesCaptura.ancho) *
100;
        const h5pY = (annotation.posicionY / annotation.dimensionesCaptura.alto) *
100;

        // Construye y retorna el objeto de interacción con la estructura H5P
        return {
            x: parseFloat(h5pX.toFixed(14)),
            y: parseFloat(h5pY.toFixed(14)),
            width: 10,
            height: 10,
            duration: {
                from: annotation.tiempo,
                to: annotation.tiempo
            },
            libraryTitle: "Text",
            action: {
                library: "H5P.Text 1.1",
                params: { text: `<p>${briefDescription}</p>` },
                subContentId: crypto.randomUUID(), // Genera un identificador único
                metadata: {
                    contentType: "Text",
                    license: "U",
                    title: identifiedType || "Untitled Text",
                    authors: [],
                    changes: []
                }
            },
            pause: true,
            displayType: "button",
            buttonOnMobile: false,
            visuals: {
                backgroundColor: colorNameToRgb(annotation.color),
                boxShadow: true
            },
            goto: { /* ... estructura fija ... */ },
            label: `<p>${identifiedType}</p>`
        };
    });
    return h5pInteractions;
}
```

Este bloque de código es el responsable de la traducción de los datos y su funcionamiento se divide en los siguientes pasos:

1. **Iteración sobre las anotaciones**: La función utiliza el método `Array.prototype.map()` para iterar sobre `videoAnnotations`, el array que almacena todas las marcas creadas por el alumno. Este método crea un nuevo array (`h5pInteractions`) que contendrá los objetos transformados.

2. **Extracción de datos semánticos**: Para cada anotación, se accede a la clave `camposFormulario` para extraer la descripción breve y el tipo de tejido identificado. Estos datos conformarán el contenido visible de la interacción en H5P.

3. **Conversión de coordenadas**: Este es un paso más complejo. Las coordenadas de la anotación original (`posicionX`, `posicionY`) están expresadas en píxeles y son relativas a las dimensiones del canvas en el momento de la captura (`dimensionesCaptura`). Para que H5P pueda posicionar la interacción correctamente sobre el vídeo independientemente de su tamaño, estas coordenadas absolutas deben convertirse a valores porcentuales. La fórmula `(píxel / dimensión_total) * 100` se aplica tanto para el eje X como para Y. El resultado se formatea con una precisión decimal (`toFixed(14)`) para mantener la exactitud.

4. **Construcción del objeto H5P**: Se devuelve un nuevo objeto JavaScript que se adhiere rigurosamente a la estructura de una interacción de H5P. Se asignan los valores transformados a las claves correspondientes (`x`, `y`, `action.params.text`, `label`). Además, se definen valores fijos para propiedades estándar de H5P, como el tipo de librería (`H5P.Text 1.1`) o el comportamiento (`pause: true`). Se utiliza la función `crypto.randomUUID()` para generar un `subContentId` único para cada interacción, un requisito indispensable de H5P para evitar conflictos internos.

5. **Retorno del array transformado**: La función `generateReportJsonContent` devuelve directamente el array `h5pInteractions`. Esto significa que el fichero `informacion.json` generado por la herramienta de marcado ya contiene los datos en el formato exacto que la herramienta de generación de paquetes necesita.

Gracias a esta adaptación, la herramienta de generación de paquetes H5P simplemente lee los ficheros `informacion.json` seleccionados, concatena los arrays de interacciones que contienen y los inyecta directamente en la clave

`interactiveVideo.assets.interactions` del objeto `content.json` en memoria. Las figura 61 y 62 ilustran esta correspondencia, mostrando los datos de una anotación original y el objeto de interacción H5P que la función anterior genera a partir de ellos.

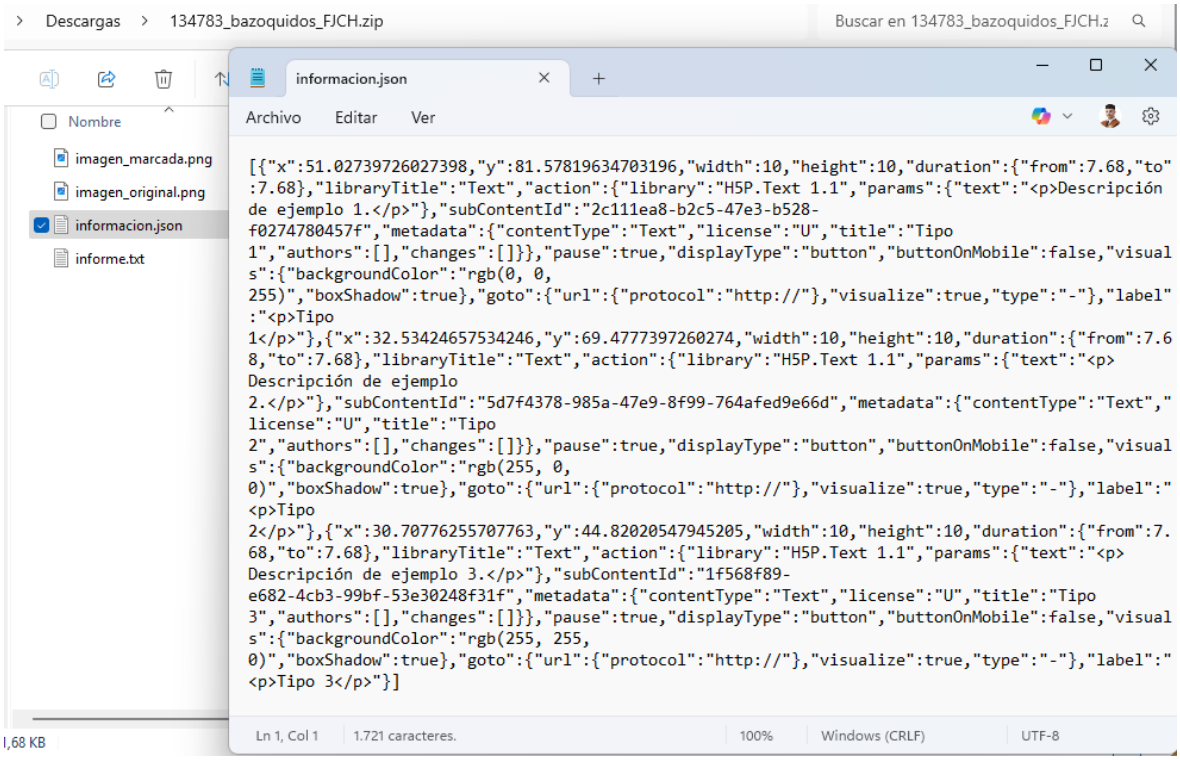


Figura 61. Formato de anotaciones generadas por la herramienta de marcado. Fuente: Elaboración propia.

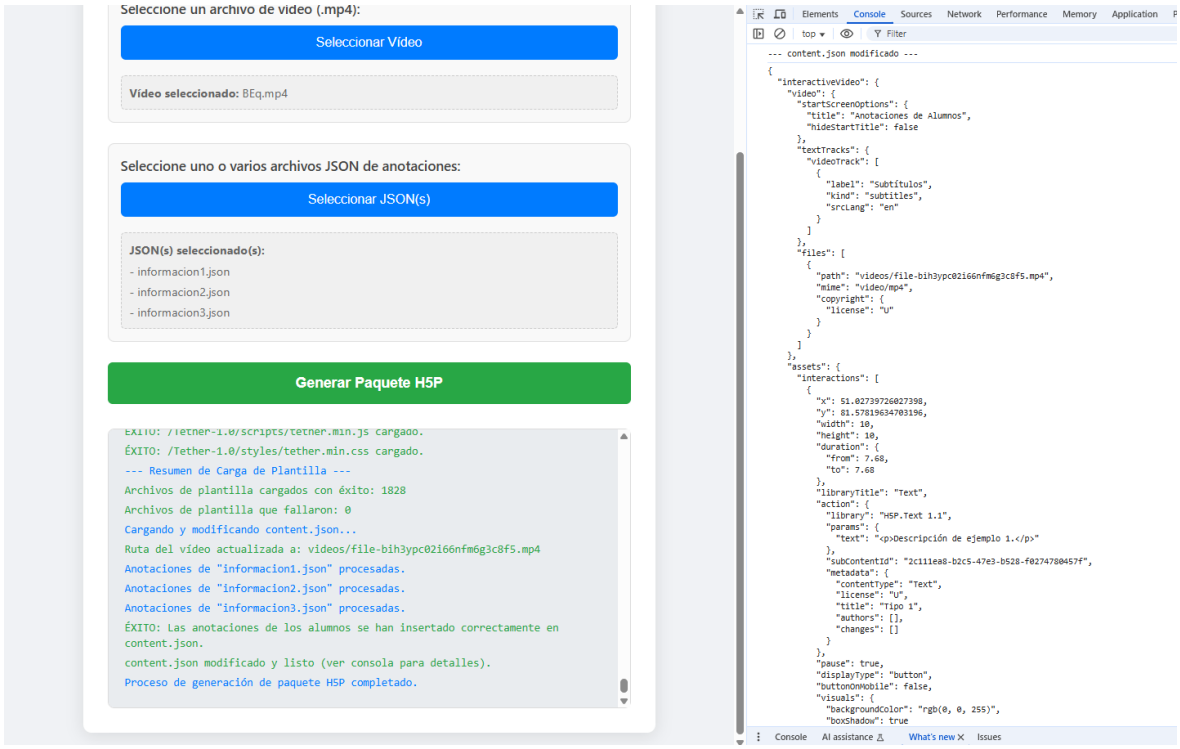


Figura 62. Objeto de interacción H5P generado. Fuente: Elaboración propia.

6.2.3. Ajuste de coordenadas y generación del paquete final

El último paso de la implementación técnica consiste en la compilación de todos los recursos en memoria (los ficheros de la plantilla y el contenido personalizado) en un único paquete `.h5p` descargable. Este proceso, gestionado por la librería JSZip, requiere una atención especial a la estructura del archivo ZIP resultante para garantizar su compatibilidad con las plataformas H5P.

Una de las principales complejidades detectadas durante el desarrollo es que plataformas como Moodle o Lumi son muy estrictas con el formato del paquete `.h5p`. Específicamente, rechazan los paquetes que contienen entradas explícitas para directorios. Un archivo ZIP válido para H5P no debe incluir entradas marcadas como carpetas (por ejemplo, `content/`), sino que la estructura de directorios debe inferirse implícitamente de las rutas de los ficheros que contiene (ej. `content/content.json`).

Aunque JSZip generalmente maneja esto de forma correcta, en el contexto del navegador se observó que, en ocasiones, se generaban estas entradas de directorio no deseadas. Para tratar de solventar este problema y asegurar la máxima compatibilidad, se implementó una estrategia de **reconstrucción del ZIP**.

Fragmento de código representativo

En lugar de generar el archivo final directamente, el proceso se divide en dos fases, como se ilustra en el siguiente fragmento de código:

```
// 1. Creación de un ZIP temporal
const tempZip = new JSZip();
// [Se añaden todos los archivos de plantilla y de usuario a tempZip]

// 2. Filtrado y reconstrucción en un ZIP final
const finalZip = new JSZip();
const copyPromises = [];

tempZip.forEach((relativePath, zipEntry) => {
    // 3. Condición de filtrado: se procesan solo las entradas que no son directorios
    if (!zipEntry.dir) {
        // 4. Se obtiene el contenido de la entrada como un Blob de forma asíncrona
        const promise = zipEntry.async('blob').then(content => {
            // Se añade el fichero con su contenido al ZIP final
            finalZip.file(relativePath, content);
        });
        copyPromises.push(promise);
    }
});

// 5. Se espera a que todas las operaciones de copia asíncrona finalicen
await Promise.all(copyPromises);

// 6. Se genera el Blob del ZIP final y se inicia la descarga
const zipBlob = await finalZip.generateAsync({ type: "blob", compression: "DEFLATE"
});
saveAs(zipBlob, `H5P_InteractiveVideo_Custom_${new Date().toISOString().slice(0,
10)}.h5p`);
```

Este enfoque de dos fases se descompone en los siguientes pasos lógicos:

1. **Creación de un ZIP temporal:** Se instancia un primer objeto JSZip, denominado `tempZip`. En este objeto se añaden todos los ficheros necesarios (la plantilla base, el `content.json` modificado y el vídeo del usuario) utilizando el método `tempZip.file()`. En esta fase no se aplican filtros, permitiendo que JSZip genere su estructura interna, incluyendo las posibles entradas de directorio no deseadas.

2. **Inicialización del ZIP final:** Se crea un segundo objeto JSZip, `finalZip`, que contendrá la estructura limpia y definitiva del paquete.

3. **Iteración y filtrado:** Se utiliza el método `tempZip.forEach()` para recorrer cada una de las entradas del ZIP temporal. Para cada entrada (`zipEntry`), se evalúa su propiedad booleana `.dir`. Esta propiedad es `true` si la entrada es un directorio explícito y `false` si es un fichero. La condición `if (!zipEntry.dir)` actúa como un filtro, asegurando que solo los ficheros pasen al siguiente paso.

4. **Copia asíncrona de contenido:** Para cada fichero que supera el filtro, se invoca el método `zipEntry.async('blob')`. Este método lee el contenido del fichero de forma asíncrona y lo devuelve como un objeto Blob, preservando su contenido binario. La promesa resultante se añade a un array (`copyPromises`). Dentro del `.then()` de la

promesa, una vez que el contenido está disponible, se añade al `finalZip` con su ruta correspondiente.

5. **Sincronización de operaciones:** Dado que la lectura y escritura de cada fichero es asíncrona, es fundamental esperar a que todas las operaciones de copia hayan concluido antes de generar el archivo final. `Promise.all(copyPromises)` cumple esta función, pausando la ejecución hasta que todas las promesas del array `copyPromises` se hayan resuelto.

6. **Generación y descarga:** Una vez garantizado que `finalZip` contiene todos los ficheros y ninguna carpeta explícita, se invoca `finalZip.generateAsync({ type: "blob", ... })`. Esta llamada compila el contenido en un único *Blob* binario, aplicando compresión `DEFLATE` para optimizar su tamaño. Finalmente, este Blob se pasa a la función `saveAs()` de la librería `FileSaver.js`, que gestiona la descarga en el navegador del usuario con la extensión `.h5p`.

El resultado es un fichero descargado con un nombre estandarizado que incluye la fecha de generación, como se puede observar en la figura 63. Con el paquete ya generado, el siguiente paso es validar su integridad y correcto funcionamiento.

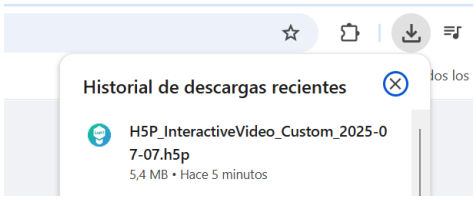


Figura 63. Notificación de descarga del navegador mostrando el paquete H5P generado por la herramienta.

6.3. Resultados y validación del proceso

Una vez implementada la funcionalidad de generación de paquetes, la fase final del desarrollo se centra en la validación de los ficheros `.h5p` resultantes. El objetivo de esta etapa es asegurar que los paquetes no solo se generen sin errores técnicos, sino que también sean estructuralmente correctos, funcionalmente operativos y compatibles con las plataformas de destino.

Este proceso de validación permitirá confirmar la viabilidad de la herramienta en un contexto educativo real. Un paquete H5P debe cumplir con estándares estrictos para ser reconocido e interpretado correctamente por un LMS como Moodle. Por ello, la validación se diseña en dos etapas secuenciales que simulan un flujo de trabajo realista para el profesorado:

1. **Verificación en un entorno de autoría local:** Se utiliza una herramienta de escritorio para una primera inspección visual y funcional del paquete, permitiendo una revisión rápida y ágil.
2. **Prueba de integración en el LMS:** Se procede a la subida del paquete al entorno final (Moodle) para comprobar su comportamiento en un escenario de producción y analizar las interacciones con la plataforma.

6.3.1. Verificación en Lumi

La primera etapa de validación se realiza en un entorno de autoría local, para lo cual se selecciona la herramienta de código abierto Lumi, específicamente diseñada para la creación, edición y visualización de contenido H5P, lo que la convierte en el entorno ideal para una primera inspección del paquete generado sin la necesidad de interactuar con un LMS en línea.

El flujo de trabajo propuesto para el profesorado se simplifica así en un proceso de dos pasos: primero, utilizar la herramienta web desarrollada en este proyecto para generar el paquete `.h5p` a partir de un vídeo y los ficheros de anotaciones; segundo, abrir dicho paquete directamente en Lumi para su revisión.

Esta verificación intermedia es clave para confirmar de manera inmediata y visual varios aspectos del resultado de la generación:

- **Integridad del paquete**: La capacidad de Lumi para abrir el fichero `.h5p` sin errores confirma que la estructura de directorios y ficheros del ZIP es correcta y que el manifiesto `h5p.json` está bien formado.
- **Integración del vídeo**: Se comprueba que el vídeo base seleccionado por el usuario se ha incrustado correctamente y se reproduce sin problemas de códec o de ruta.
- **Funcionalidad de las interacciones**: Es el punto más importante de la validación. Se verifica que las interacciones, correspondientes a las anotaciones de los alumnos, aparecen en los instantes de tiempo correctos. Además, se comprueba que su posición sobre el fotograma y el contenido textual que muestran al ser activadas se corresponden con los datos de los ficheros `informacion.json` de entrada.

Como se puede apreciar en la figura 64, al abrir un paquete generado por la herramienta, Lumi renderiza el vídeo interactivo de forma exitosa. La imagen de la izquierda muestra el reproductor de vídeo con los botones de interacción superpuestos en las coordenadas ajustadas.

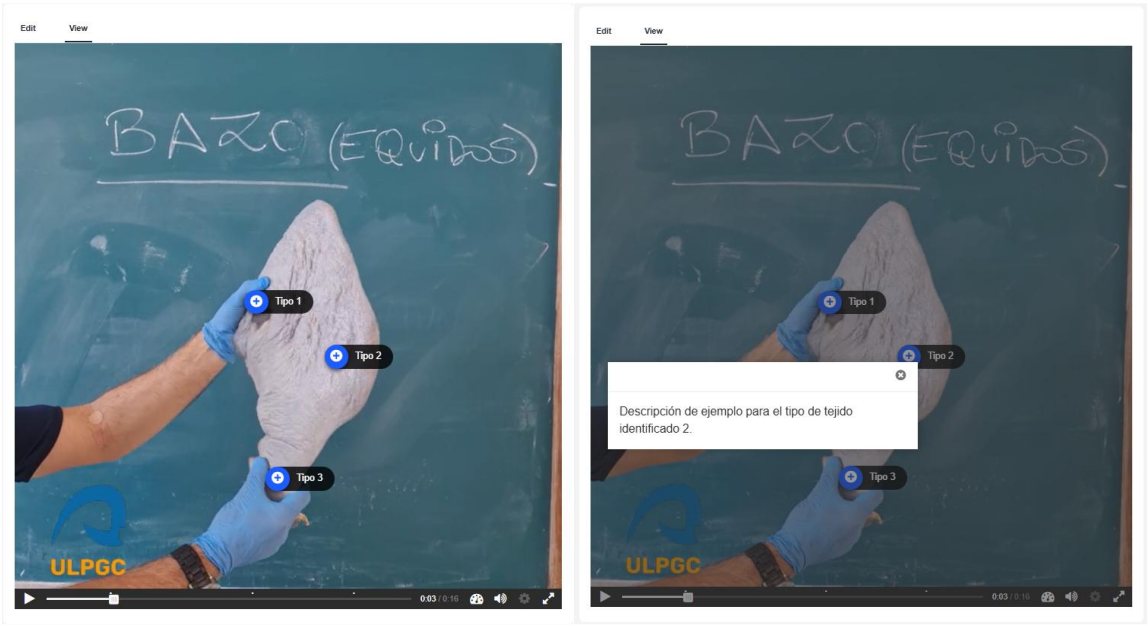


Figura 64. Visualización de un paquete H5P generado en la herramienta Lumi.

La imagen de la derecha muestra el cuadro de diálogo que se despliega al hacer clic en una de estas interacciones, mostrando el título y la descripción correspondientes a esa marca. Este resultado valida que el proceso de transformación de datos y empaquetado ha sido exitoso.

6.3.2. Integración en Moodle y consideraciones administrativas

La validación final del proceso se lleva a cabo en el entorno de destino: la plataforma Moodle. El objetivo de esta prueba es confirmar que un paquete `.h5p` generado por la herramienta puede ser subido y reconocido correctamente por una actividad H5P estándar dentro de un curso.

Para realizar la prueba, se accede a un curso de Moodle con el rol de profesor y se intenta subir el fichero `.h5p` a una nueva actividad H5P. Durante este proceso, la plataforma analiza el contenido del paquete, incluyendo sus dependencias, y lo compara con las librerías H5P que ya están instaladas y permitidas en el sitio. El resultado confirma que la plataforma reconoce y procesa el contenido de forma satisfactoria, permitiendo su uso interactivo sin incidencias, como se observa en la figura 65.

Figura 65. Vídeo interactivo generado cargado en Moodle.

Se concluye que la herramienta desarrollada genera paquetes `.h5p` que son técnica y estructuralmente válidos, y completamente funcionales en el entorno Moodle estándar.

Es importante considerar que el ecosistema tecnológico (Moodle, H5P y Lumi) está en constante evolución. Para garantizar la compatibilidad a largo plazo y el correcto funcionamiento ante futuras actualizaciones, es importante mantener actualizadas las librerías del paquete H5P, cuya instalación de estas pueden ser requeridas por la plataforma.

Adicionalmente, para optimizar la experiencia de usuario y asegurar una visualización adecuada, el administrador del curso de Moodle deberá ajustar el dimensionado de los vídeos subidos a través de la configuración del recurso. Aunque esta dependencia administrativa queda fuera del alcance técnico de este proyecto, la validación confirma la viabilidad de la solución y su correcta adherencia al estándar H5P.

6.3.3. Valoración de métricas para la generación de paquetes H5P

El desarrollo del generador de paquetes H5P constituye la culminación funcional del proyecto, abordando la tarea de mayor complejidad algorítmica: la manipulación de estructuras de datos anidadas, la gestión de ficheros binarios en memoria y la depuración de formatos de archivo estrictos. Esta sección final cuantifica y valora el rendimiento del modelo Gemini 2.5 Flash en este escenario avanzado. La tabla 6 resume las métricas clave del proceso, desglosadas por los bloques funcionales definidos: Replicación de Plantilla (RP), Transformación y Modificación (TM) y Generación y Depuración del Paquete (GD).

Métrica/ Componente	Replicación de Plantilla (RP)	Transformación y Modificación (TM)	Generación y Depuración (GD)
Prompts totales	6	6	6
Tokens totales de entrada (*input*)	14.795	3.192	789
Tokens totales de salida (*output*)	42.600	42.490	7.043
Tokens promedio por entrada	2.465,83	532,00	131,50
Tokens promedio por salida	7.100,00	7.081,67	1.173,83
Respuestas aceptadas	3	3	3
Respuestas no aceptadas	3	3	3
Tasa de aceptación (%)	50% (3/6)	50% (3/6)	50% (3/6)
Tiempo medio de respuesta (s)			
- Respuestas < 5000 tokens	31,04 (4 respuestas)	44,93 (1 respuesta)	22,28 (6 respuestas)
- Respuestas 5000-7000 tokens	(Ninguna)	55,88 (4 respuestas)	(Ninguna)
- Respuestas > 7000 tokens	140,15 (2 respuestas)	144,07 (2 respuestas)	(Ninguna)
Complejidad del prompt (Tokens entrada)	Alta	Media	Baja
Complejidad de la respuesta (Tokens salida)	Muy Alta	Muy Alta	Media

Tabla 6. Análisis de Interacción con el modelo de IAG para la de generación de paquetes H5P. Fuente: Elaboración propia.

Análisis general de la asistencia por IAG

El desarrollo de este módulo, documentado a través de 18 interacciones, confirma la capacidad de Gemini 2.5 Flash para actuar como un asistente eficaz en tareas de programación complejas y multifacéticas. El análisis de los datos de la tabla 6 muestra una correlación directa entre la naturaleza de la tarea y los recursos requeridos. El bloque de "*Replicación de Plantilla (RP)*", que implicó la gestión de un gran listado de rutas de fichero y la implementación de lógica de carga concurrente, destaca por su elevado consumo de tokens tanto de entrada como de salida, así como por los tiempos de respuesta más prolongados para las soluciones de mayor complejidad.

La **tasa de aceptación del 50%** en todos los bloques funcionales es un indicador consistente que refleja la naturaleza iterativa del desarrollo asistido. Al igual que en los módulos anteriores, las respuestas no aceptadas no representaron fallos del modelo, sino soluciones iniciales que requirieron refinamiento y depuración. Este hecho consolida la

visión del modelo de IA como ayudante en la generación de código base funcional, pero cuya optimización y ajuste final dependen de la dirección técnica y la validación por parte del desarrollador. La capacidad de la IAG para proponer soluciones a problemas bien definidos, como la implementación de una cola de promesas para la carga concurrente o la lógica de reconstrucción de un ZIP, ha sido fundamental para acelerar el desarrollo.

Resultados específicos por bloque funcional

- **Replicación de Plantilla (RP):** Este bloque presentó la mayor complejidad en los prompts de entrada, dado que se le proporcionaron listados extensos de ficheros y se le solicitaron implementaciones de carga asíncrona. Aunque la tasa de aceptación fue del 50%, las respuestas aceptadas demostraron una notable capacidad para estructurar código complejo, como la gestión de lotes de promesas, validando su utilidad para tareas algorítmicas.
- **Transformación y Modificación (TM):** Con una complejidad de prompt media, este bloque se centró en la manipulación de objetos JSON. El modelo gestionó con gran eficacia la tarea de mapear la estructura de datos de las anotaciones al formato requerido por H5P, incluyendo la conversión de coordenadas y la generación de UUIDs. Las iteraciones adicionales se centraron principalmente en ajustes finos del formato de salida, una tarea que requirió una guía muy específica.
- **Generación y Depuración del Paquete (GD):** A pesar de tener la menor complejidad en los prompts de entrada, este bloque fue técnicamente exigente, ya que abordó la depuración de la estructura del archivo ZIP. La solución final, que implicó la reconstrucción del paquete filtrando las entradas de directorio, fue propuesta por el modelo tras una descripción detallada del problema (`zipEntry.dir === true`). Esto subraya la capacidad del modelo para ofrecer soluciones técnicas sofisticadas cuando se le presenta un problema bien diagnosticado.

En su conjunto, el desarrollo completo de la herramienta, desde la anotación inicial hasta la generación del paquete H5P, ha validado la elección de Gemini 2.5 Flash como un modelo de IAG equilibrado y potente. Su rendimiento ha sido sólido tanto en tareas de diseño de interfaz como en la implementación de lógica compleja, demostrando ser un impulsor eficaz para la creación de soluciones educativas interactivas y alineadas con los principios de accesibilidad y sostenibilidad del proyecto.

Capítulo 7: Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones finales derivadas del desarrollo y la evaluación del proyecto. Se realiza una reflexión sobre el impacto del uso de la inteligencia artificial generativa en el ámbito del desarrollo de software, se evalúa el grado de cumplimiento de los objetivos planteados y, finalmente, se proponen diversas líneas de trabajo futuro para la mejora y expansión de la herramienta.

7.1. Conclusiones sobre la metodología y su impacto formativo

La realización de este Trabajo de Fin de Grado ha permitido, además de desarrollar una solución tecnológica funcional, validar una metodología de desarrollo que integra la colaboración con la inteligencia artificial generativa. Este enfoque ha demostrado ser un impulsor de las capacidades de un desarrollador, ampliando el alcance de lo que es factible lograr en un proyecto de esta naturaleza.

Una de las conclusiones más relevantes es que el uso de un modelo como Gemini 2.5 Flash ha hecho posible abordar tareas de una complejidad técnica considerable, como la generación programática de paquetes H5P o la simulación de interacciones con el DOM de Moodle, que normalmente habrían requerido una profunda especialización y un ciclo de desarrollo mucho más extenso. Por tanto, la IAG ha demostrado la capacidad de reducir las barreras técnicas de entrada, permitiendo que un único desarrollador pueda crear soluciones robustas y sofisticadas dentro del marco de un proyecto académico.

Este proyecto evidencia además que la inteligencia artificial no reemplaza al programador, sino que redefine su rol, desplazando el foco de la escritura de código repetitivo hacia actividades de mayor valor añadido. El trabajo se ha centrado en tres áreas clave:

- **Diseño arquitectónico y especificación de módulos:** La habilidad para descomponer un problema complejo en componentes funcionales y formular instrucciones precisas, detalladas y contextualizadas ha sido fundamental para guiar al modelo hacia la solución deseada.
- **Validación y refinamiento crítico:** El código generado, si bien a menudo funcional, ha requerido una supervisión constante para asegurar su calidad, robustez y adherencia a las buenas prácticas de programación.
- **Integración arquitectónica:** La labor principal ha consistido en ensamblar los distintos bloques de código generados dentro de una arquitectura coherente y funcional, asegurando la correcta comunicación entre los módulos.

Desde una perspectiva formativa, la principal conclusión es que esta metodología de trabajo facilita enormemente la creación de herramientas para la educación. Permite desarrollar soluciones que se adaptan a necesidades pedagógicas específicas de una forma mucho más rápida y eficiente. En su conjunto, este Trabajo de Fin de Grado demuestra que es totalmente factible mejorar la enseñanza con herramientas tecnológicas a medida, sin necesidad de grandes equipos de desarrollo o extensos plazos de tiempo.

7.2. Conclusiones generales y cumplimiento de objetivos

A lo largo del desarrollo de este trabajo, se ha diseñado e implementado una solución de software que valida la viabilidad de integrar herramientas de anotación interactivas en plataformas LMS como Moodle, con la asistencia de IAG. La evaluación del trabajo se ha realizado en función de los objetivos específicos planteados en la introducción de esta memoria, cuyo grado de cumplimiento se detalla a continuación.

Objetivo 1: Evaluación de herramientas de IAG. Este objetivo se ha cumplido satisfactoriamente a través del análisis comparativo presentado en el capítulo 3. Se establecieron criterios técnicos y operativos para evaluar una selección representativa de modelos generativos, tanto en la nube como locales. El estudio concluyó con la selección de Gemini 2.5 Flash como la herramienta óptima para el proyecto, basándose en su excelente equilibrio entre rendimiento en la generación de código, eficiencia operativa, amplia ventana de contexto y accesibilidad gratuita. Esta decisión fundamentó y garantizó la viabilidad técnica del resto del desarrollo.

Objetivo 2: Desarrollo de un sistema de anotación en vídeos sobre Moodle. Se ha alcanzado plenamente este objetivo mediante la implementación de una herramienta funcional, como se documenta en los capítulos 4 y 5. Se ha diseñado e implementado una interfaz que permite el marcado visual y el etiquetado semántico de fotogramas. Además, se ha desarrollado con éxito el módulo de generación de informes, que permite al usuario exportar su trabajo en formatos estándar como PDF e imagen, así como empaquetarlo en un archivo ZIP para su posterior entrega automatizada en una actividad de Moodle.

Objetivo 3: Implementación de vídeos interactivos con anotaciones dinámicas. Este objetivo se ha cumplido con éxito, representando la culminación del ciclo pedagógico propuesto. Como se detalla en el capítulo 6, se ha desarrollado un generador de paquetes H5P que transforma las anotaciones estáticas de los estudiantes en un recurso de aprendizaje interactivo y reutilizable. La herramienta es capaz de construir programáticamente un paquete H5P válido, que puede ser visualizado en herramientas de autoría como Lumi e integrado en Moodle, haciendo que las anotaciones sean visibles y accesibles durante la reproducción del vídeo.

En su conjunto, se puede concluir que el proyecto ha alcanzado su objetivo principal. Se ha demostrado que es factible desarrollar, mediante una arquitectura cliente-ligera y el uso de un modelo generativo preseleccionado, una herramienta de anotación de vídeos que se integra de forma no invasiva en Moodle. La solución final, además de ser completamente

funcional, enriquece la experiencia educativa al transformar un recurso pasivo en una herramienta de aprendizaje activo.

7.3. Propuestas de mejora y líneas de trabajo futuro

Aunque el trabajo ha alcanzado todos los objetivos propuestos, su enfoque innovador abre la puerta a numerosas extensiones y mejoras que podrían aumentar significativamente su impacto y utilidad en un entorno educativo real.

1. Empaquetado como *plugin* nativo de Moodle: La mejora más significativa para garantizar la sostenibilidad y escalabilidad de la herramienta sería su transformación en un *plugin* de Moodle de tipo "Actividad" o "Recurso". Esto eliminaría la necesidad de que el profesorado inserte y gestione manualmente el código en una página HTML, o que dependa de un desarrollador para ello. Un *plugin* nativo permitiría una instalación centralizada por parte del administrador del sistema, automatizaría la gestión de dependencias (como las librerías H5P) y ofrecería una integración más profunda con el ecosistema de Moodle.

2. Ampliación de las funcionalidades de anotación: La herramienta actual se centra en marcas circulares, pero podría enriquecerse con nuevas capacidades de interacción:

- **Diversificación de formas de anotación:** Implementar la capacidad de dibujar otras formas geométricas (como rectángulos o flechas) para adaptarse a diferentes tipos de análisis visual.
- **Anotación colaborativa:** Desarrollar una funcionalidad que permita a múltiples usuarios trabajar sobre el mismo vídeo, visualizando y comentando las anotaciones de sus compañeros. Esto fomentaría el aprendizaje social y el debate en grupo.

3. Desarrollo de un *backend* para la persistencia y colaboración: Actualmente, la herramienta es completamente cliente. Para habilitar funcionalidades más avanzadas, como la anotación colaborativa o el análisis de datos de todo un curso, sería necesario desarrollar un *backend* simple. Este se encargaría de almacenar todas las anotaciones de forma centralizada en una base de datos, permitiendo guardar el progreso entre sesiones, además de sentar las bases para la interacción en tiempo real y analíticas de aprendizaje.

3. Integración de IAG para la asistencia docente y retroalimentación: Un paso futuro de gran impacto sería integrar la IAG para facilitar la labor del docente y enriquecer el proceso de retroalimentación posterior.

- **Análisis semántico de las entregas**: Implementar un sistema que, utilizando un modelo de lenguaje, analice el conjunto de anotaciones de todos los estudiantes.

Podría generar resúmenes automáticos para el profesorado, identificando patrones recurrentes, conceptos bien comprendidos o, más importante, dudas y errores comunes en todo el grupo.

- **Generación de feedback formativo**: Desarrollar una función donde, una vez entregada la tarea, la IAG pueda generar un primer borrador de feedback personalizado para cada estudiante. Este podría plantear preguntas reflexivas basadas en las propias anotaciones del alumno (ej: *"Has identificado correctamente el tejido X, ¿qué relación tiene con el proceso Y que se explica más adelante en el vídeo?"*), guiando así su aprendizaje.

4. Panel de analíticas de aprendizaje para el profesorado: Se podría diseñar y desarrollar un panel de control (*dashboard*) para el profesorado. Esta vista centralizaría y visualizaría datos agregados sobre el uso de la herramienta, ofreciendo métricas como los instantes de tiempo más anotados, las dudas más frecuentes extraídas de las descripciones o el tiempo medio dedicado a la tarea. Este sistema de analíticas proporcionaría información valiosa para evaluar tanto el desempeño de los estudiantes como la efectividad del material didáctico.

Bibliografía

En esta sección se detallan las referencias bibliográficas utilizadas a lo largo del trabajo, en el debido formato establecido por el Instituto de Ingenieros Eléctricos y Electrónicos (*Institute of Electrical and Electronics Engineers, IEEE*).

[1] MoodleDocs. (2024, octubre 5). Acerca de Moodle [En línea]. Disponible en: https://docs.moodle.org/all/es/Acerca_de_Moodle. Accedido: 22 de febrero de 2025.

[2] MoodleDocs. (2024, octubre 27). 45/Video [En línea]. Disponible en: https://docs.moodle.org/all/es/45/Video. Accedido: 22 de febrero de 2025.

[3] IBM. (s.f.). IA en el desarrollo de software [En línea]. Disponible en: https://www.ibm.com/mx-es/think/topics/ai-in-software-development. Accedido: 22 de febrero de 2025.

[4] GitHub, GitHub Copilot [Software]. San Francisco, CA: GitHub, Inc., 2021.

[5] OpenAI, ChatGPT [Software]. San Francisco, CA: OpenAI, 2022.

[6] Anthropic, Claude [Software]. San Francisco, CA: Anthropic, 2024.

[7] DeepSeek, DeepSeek [Software]. Hangzhou, CN-ZJ-01: DeepSeek AI, 2025.

[8] Google, Gemini [Software]. Mountain View, CA: Google LLC, 2023.

[9] IBM. (s.f.). Gestión de riesgos de la IA [En línea]. Disponible en: https://www.ibm.com/es-es/think/insights/ai-risk-management. Accedido: 24 de febrero de 2025.

[10] LMStudio, LMStudio [Software]. [S.l.]: LMStudio, 2024.

[11] Nomic AI, GPT4All [Software]. Brooklyn, NY: Nomic AI, 2025.

[12] Ollama Inc., Ollama [Software]. [S.l.]: Ollama Inc., 2023.

[13] Optimia Solution. (s.f.). IA local para empresas [En línea]. Disponible en: https://optimiasolution.com/ia-local-para-empresas/. Accedido: 24 de febrero de 2025.

[14] Mozilla. (2024, diciembre 21). El modelo de estándares web [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn_web_development/Getting_started/Web_standards/The_web_standards_model. Accedido: 3 de marzo de 2025.

[15] Cloudflare. (s.f.). ¿Cómo funciona el video HTML5? [En línea]. Disponible en: https://www.cloudflare.com/es-es/learning/video/how-html5-video-works/. Accedido: 3 de marzo de 2025.

[16] Mozilla. (2024, diciembre 17). Canvas API [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/API/Canvas_API. Accedido: 3 de marzo de 2025.

[17] jsPDF. (2019, abril 2). jsPDF Documentation [En línea]. Disponible en: https://artskydj.github.io/jsPDF/docs/jsPDF.html. Accedido: 3 de marzo de 2025.

[18] PDFKit. (s.f.). PDFKit [En línea]. Disponible en: https://pdfkit.org/. Accedido: 3 de marzo de 2025.

[19] H5P.org. (s.f.). Documentation [En línea]. Disponible en: https://h5p.org/documentation. Accedido: 3 de marzo de 2025.

[20] Universidad de Las Palmas de Gran Canaria. (2024, mayo). "Recomendaciones sobre el uso de la IA en la ULPGC", Servicio de Informática, Las Palmas de Gran Canaria, España, Doc. Interno, 2024.

[21] Universidad de Las Palmas de Gran Canaria. (2024, junio 6). "Acta Nº 241 de 06 junio 2024 (Extraordinario)", Consejo de Gobierno, Las Palmas de Gran Canaria, España, Acta, 2024.

[22] Napkin AI, Napkin [Software]. [S.l.]: Osmo AI, 2015.

[23] S. J. Russell y P. Norvig, *Artificial Intelligence: A Modern Approach, Global Edition*, 4ª ed. Harlow, Reino Unido: Pearson Education Limited, 2022.

[24] A. Radford, K. Simonyan, I. Sutskever, J. Wu, S. L. Amodei, D. Bahdanau, y J. LeCun, "Generative Adversarial Networks", *arXiv preprint arXiv:1406.2661*, enviado para publicación, junio 2014.

[25] Telefónica Tech. (2021). "El impacto de la inteligencia artificial en el sector telco", Telefónica Tech, Madrid, España, Doc. Técnico, 2021.

[26] IBM. (s.f.). *Aprendizaje semi-supervisado* [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/semi-supervised-learning. Accedido: 12 de marzo de 2025.

[27] MyTips.es. (2023, mayo 4). *¿Qué es la inteligencia artificial generativa (IAG)?* [En línea]. Disponible en: https://www.mytips.es/que-es-la-inteligencia-artificial-generativa-iag/. Accedido: 15 de marzo de 2025.

[28] MathWorks. (s.f.). *Generative Adversarial Networks (GANs)* [En línea]. Disponible en: https://www.mathworks.com/discovery/generative-adversarial-networks.html. Accedido: 15 de marzo de 2025.

[29] AWS Amazon. (s.f.). *¿Qué son los transformadores en inteligencia artificial?* [En línea]. Disponible en: https://aws.amazon.com/es/what-is/transformers-in-artificial-intelligence/#seo-faqpairs#how-do-transformers-work. Accedido: 21 de marzo de 2025.

[30] M. S. Zaki, C. K. D. Aslam, M. C. J. M. Khan, y M. M. S. Hassan, "A Two-Sample Test of Text Generation Similarity", *ResearchGate*, enviado para publicación, mayo 2025.

[31] R. D. Caballar. (2024, octubre 31). "*What are small language models?*". IBM Think. [En línea]. Disponible: https://www.ibm.com/think/topics/small-language-models. Accedido: 28 de marzo de 2025

[32] IBM. (2023, noviembre 2). *¿Qué son los grandes modelos de lenguaje (LLM)?* [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/large-language-models. Accedido: 28 de marzo de 2025.

[33] Microsoft Learn. (2025, mayo 29). *Descripción de los tokens* [En línea]. Disponible en: https://learn.microsoft.com/es-es/dotnet/ai/conceptual/understanding-tokens. Accedido: 2 de junio de 2025.

[34] IBM. (2024, febrero 23). *Casos de uso de la IA conversacional* [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/conversational-ai-use-cases. Accedido: 4 de abril de 2025.

[35] IBM. (2024, abril 18). *Ejemplos de inteligencia artificial general (AGI)* [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/artificial-general-intelligence-examples. Accedido: 8 de abril de 2025.

[36] Actualidad Docente. (2024, enero 24). *El 69% de los padres y madres en España ha utilizado alguna vez la inteligencia artificial (IA) y el 78% quiere aprender más sobre ella* [En línea]. Disponible en: https://actualidaddocente.cece.es/destacados-a-fondo/el-69-de-los-padres-y-madres-en-espana-ha-utilizado-alguna-vez-la-inteligencia-artificial-ia-y-el-78-quiere-aprender-mas-sobre-ella/. Accedido: 14 de abril de 2025.

[37] IBM. (2025, abril 8). *Vibe Coding* [En línea]. Disponible en: https://www.ibm.com/think/topics/vibe-coding. Accedido: 17 de abril de 2025.

[38] IBM. (2024, octubre 7). *IA en el desarrollo de software* [En línea]. Disponible en: https://www.ibm.com/mx-es/think/topics/ai-in-software-development. Accedido: 17 de abril de 2025.

[39] Google Cloud. (2023, junio 7). *Drive developer productivity with code assistance* [Video en línea]. Disponible en: https://www.youtube.com/watch?v=3nR2hUaqqrU. Accedido: 17 de abril de 2025.

[40] Google, Jules [Software]. Mountain View, CA: Google LLC, 2023.

[41] Microsoft Learn. (2025, enero 20). *Conceptos clave y consideraciones para crear soluciones de IA generativas* [En línea]. Disponible en: https://learn.microsoft.com/es-es/azure/developer/ai/gen-ai-concepts-considerations-developers. Accedido: 17 de abril de 2025.

[42] Learn Prompting. (2025, febrero 18). *Roles en los Prompts* [En línea]. Disponible en: https://learnprompting.org/docs/basics/roles. Accedido: 21 de abril de 2025.

[43] Reddit. (2024, julio). *Shorter prompts lead to 40% better code generation* [En línea]. Disponible en: https://www.reddit.com/r/PromptEngineering/comments/1do6wx4/shorter_prompts_lead_to_40_better_code_generation/. Accedido: 21 de abril de 2025.

[44] Reddit. (2024, septiembre). *Improving code iteration with LLM APIs* [En línea]. Disponible en: https://www.reddit.com/r/ClaudeAI/comments/1fpyp8d/improving_code_iteration_with_llm_apis_beyond/. Accedido: 21 de abril de 2025.

[45] Yahoo Tech. (2025, abril 4). *Andrew Ng says giving AI 'lazy' prompts is sometimes OK. Here's why.* [En línea]. Disponible en: https://tech.yahoo.com/articles/andrew-ng-says-giving-ai-124856839.html. Accedido: 21 de abril de 2025.

[46] PromptHub.us. (2025, marzo 27). *A Complete Guide to Meta-Prompting* [En línea]. Disponible en: https://www.prompthub.us/blog/a-complete-guide-to-meta-prompting. Accedido: Accedido: 21 de abril de 2025.

[47] SlideGeeks. (2024, enero). *Composition Of Effective Prompt For Code Generation Diagrams PDF* [En línea]. Disponible en: https://www.slidegeeks.com/composition-of-effective-prompt-for-code-generation-diagrams-pdf. Accedido: 21 de abril de 2025.

[48] OpenAI. (2025). *Presentación de ChatGPT y GPT-4.1* [En línea]. Disponible en: https://openai.com/index/gpt-4-1/. Accedido: 21 de abril de 2025.

[49] OpenAI. (2024). *GPT-4o y más herramientas para ChatGPT (gratis)* [En línea]. Disponible en: https://openai.com/es-ES/index/gpt-4o-and-more-tools-to-chatgpt-free/.

[50] DeepSeek. (2025). *Lanzamiento de DeepSeek-R1*, DeepSeek API Docs [En línea]. Disponible en: https://github.com/deepseek-ai/DeepSeek-R1. Accedido: 21 de abril de 2025.

[51] IBM. (2024, abril 5). *Mixture of Experts* [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/mixture-of-experts. Accedido: 23 de abril de 2025

[52] xAI. (2025). *Presentación de Grok*, xAI [En línea]. Disponible en: https://x.ai/news/grok-3. Accedido: 23 de abril de 2025

[53] Google DeepMind. (2025). *Presentación de Gemini 2.5*, DeepMind [En línea]. Disponible en: https://deepmind.google/technologies/gemini/. Accedido: 23 de abril de 2025

[54] Anthropic. (2025). *Familia de modelos Claude 4*, Anthropic Documentation [En línea]. Disponible en: https://www.anthropic.com/news/claude-4. Accedido: 23 de abril de 2025

[55] Alibaba Cloud. (2025). *Serie de modelos Qwen3*, ModelScope by Alibaba [En línea]. Disponible en: https://github.com/QwenLM/Qwen3. Accedido: 23 de abril de 2025

[56] Mistral AI. (2025). *Mistral Medium*, Mistral [En línea]. Disponible en: https://mistral.ai/news/mistral-medium-3. Accedido: 23 de abril de 2025

[57] Meta AI & NVIDIA. (2025). *Visión general de Llama 4*, Meta AI Research [En línea]. Disponible en: https://ai.meta.com/llama / https://catalog.ngc.nvidia.com/. Accedido: 23 de abril de 2025

[58] A. S. Raut, S. S. Patil, P. K. Singh, y P. S. Singh, "IoT Based Smart Waste Management System", *International Research Journal of Modern Engineering & Technology Sciences (IRJMETS)*, vol. 6, no. 12, pp. 1-6, diciembre 2024. DOI: https://www.doi.org/10.56726/IRJMETS65475. Accedido: 23 de abril de 2025

[59] Intel. (s.f.). *CPU vs. GPU: ¿cuáles son las diferencias?* [En línea]. Disponible en https://www.intel.la/content/www/xl/es/products/docs/processors/cpu-vs-gpu.html. Accedido: 24 de abril de 2025.

[60] Corsair. (2024, julio 20). *Explicación de los acrónimos de PC: RAM vs VRAM.* [En línea]. Disponible en: https://www.corsair.com/es/es/explorer/diy-builder/memory/pc-acronyms-explained-ram-vs-vram/. Accedido: 24 de abril de 2025.

[61] Ollama. (2025). *DeepSeek Coder V2*, Ollama Library [En línea]. Disponible en: https://ollama.com/library/deepseek-coder-v2. Accedido: 24 de abril de 2025.

[62] Alibaba Cloud. (2025). *Presentación de Qwen2.5 Coder Instruct* [En línea]. Disponible en: https://www.alibabacloud.com/blog/introducing-qwen2-5-coder-32b-instruct-%7C-qwen_601781. Accedido: 24 de abril de 2025.

[63] Mistral AI. (2024). *Anuncio de Mistral 7B*, Mistral [En línea]. Disponible en: https://mistral.ai/news/announcing-mistral-7b. Accedido: 24 de abril de 2025.

[64] Meta. (2025). *LLaMA 3: Modelos de Fundación Abierta de Próxima Generación*, Meta AI Blog [En línea]. Disponible en: https://ai.meta.com/llama. Accedido: 24 de abril de 2025.

[65] Microsoft Azure. (2024, abril 23). *Presentación de Phi-3: Redefiniendo lo que es posible con SLMs* [En línea]. Disponible en: https://azure.microsoft.com/en-us/blog/introducing-phi-3-redefining-whats-possible-with-slms/. Accedido: 24 de abril de 2025.

[66] Reka AI. (2024). *Presentación de Reka Flash, Arctic y Edge*, Reka Blog [En línea]. Disponible en: https://reka.ai/blog/reka-flash-arctic-edge. Accedido: 24 de abril de 2025.

[67] Alibaba Cloud. (2025). *Visión general de la serie de modelos Qwen3*, ModelScope by Alibaba [En línea]. Disponible en: https://modelscope.cn/models/qwen/Qwen3-14B-Chat. Accedido: 24 de abril de 2025.

[68] AI21 Labs. (2025). *Jamba: Una nueva familia de modelos de lenguaje multimodales*, AI21 Research [En línea]. Disponible en: https://www.ai21.com/research/jamba. Accedido: 24 de abril de 2025.

[69] Google. (2024). *Gemma: Modelos abiertos de Google*, Gemma Overview [En línea]. Disponible en: https://ai.google.dev/gemma. Accedido: 24 de abril de 2025.

[70] IBM. (2024, julio 3). *GGUF frente a GGML* [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/gguf-versus-

ggml#:~:text=GGUF%20es%20un%20formato%20binario,aumentado%20la%20popularidad%20del%20formato. Accedido: 24 de abril de 2025.

[71] DataCamp. (2023, mayo 23). *Los pros y los contras de usar LLM en la nube versus ejecutar LLM localmente* [En línea]. Disponible en: https://www.datacamp.com/blog/the-pros-and-cons-of-using-llm-in-the-cloud-versus-running-llm-locally. Accedido: 29 de abril de 2025.

[72] Moodle.com. (2022, enero 31). *¿Qué es un LMS? Explicación de los sistemas de gestión del aprendizaje* [En línea]. Disponible en: https://moodle.com/es/noticias/que-es-un-lms-learning-management-systems-explicated/. Accedido: 5 de mayo de 2025.

[73] Moodle Docs 4.1. (2022, diciembre 26). *Características* [En línea]. Disponible en: https://docs.moodle.org/401/en/Features. Accedido: 5 de mayo de 2025.

[74] Moodle Docs 4.1. (2020, octubre 21). *Añadir usuarios* [En línea]. Disponible en: https://docs.moodle.org/401/en/Add_users. Accedido: 5 de mayo de 2025.

[75] Moodle Docs 4.1. (2023, agosto 31). *Añadir un nuevo curso* [En línea]. Disponible en: https://docs.moodle.org/401/en/Adding_a_new_course. Accedido: 5 de mayo de 2025.

[76] Workana. (s.f.). *¿Qué es HTML5?* [En línea]. Disponible en: https://i.workana.com/glosario/que-es-html-5/. Accedido: 5 de mayo de 2025.

[77] MDN Web Docs. (2025, marzo 31). *¿Qué es JavaScript?* [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn_web_development/Core/Scripting/What_is_JavaScript. Accedido: 5 de mayo de 2025.

[78] MDN Web Docs. (2025, marzo 28). *CSS: Hojas de Estilo en Cascada* [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/CSS. Accedido: 5 de mayo de 2025.

[79] Artificial Analysis. (2025, febrero). *Metodología: Evaluación comparativa de inteligencia* [En línea]. Disponible en: https://artificialanalysis.ai/methodology/intelligence-benchmarking. Accedido: 9 de mayo de 2025.

[80] Y. Wang *et al.*, "MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark", *arXiv preprint arXiv:2406.01574*, enviado para publicación, junio 2024.

[81] P. Bradshaw *et al.*, "Humanity's Last Exam", *arXiv preprint arXiv:2501.14249v2*, enviado para publicación, febrero 2025.

[82] D. Rein *et al.*, "GPQA: A Graduate-Level Google-Proof Q&A Benchmark", *arXiv preprint arXiv:2311.12022*, enviado para publicación, noviembre 2023.

[83] D. Hendrycks *et al.*, "Measuring Mathematical Problem Solving With the MATH Dataset", *arXiv preprint arXiv:2103.03874*, enviado para publicación, marzo 2021.

[84] AI-MO, *AI-MO/aimo-validation-aime* [Dataset]. Hugging Face Datasets / ModelScope, 2024.

[85] M. Chen *et al.*, "Evaluating Large Language Models Trained on Code", *arXiv preprint arXiv:2107.03374*, enviado para publicación, julio 2021.

[86] M. Tian *et al.*, "SciCode: A Research Coding Benchmark Curated by Scientists", *arXiv preprint arXiv:2407.13168*, enviado para publicación, julio 2024.

[87] N. Jain *et al.*, "LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code", *arXiv preprint arXiv:2403.07974*, enviado para publicación, marzo 2024.

[88] Artificial Analysis. (2025). *Modelos* [En línea]. Disponible en: https://artificialanalysis.ai/models. Accedido: 9 de mayo de 2025.

[89] LeetCode. (2025). *LeetCode* [En línea]. Disponible en: https://leetcode.com. Accedido: 15 de mayo de 2025.

[90] Codeforces. (2025). *Codeforces* [En línea]. Disponible en: https://codeforces.com. Accedido: 15 de mayo de 2025.

[91] AtCoder. (2025). *AtCoder* [En línea]. Disponible en: https://atcoder.jp. Accedido: 15 de mayo de 2025.

[92] Google. (2025). *Google AI Studio* [En línea]. Disponible en: https://aistudio.google.com. Accedido: 15 de mayo de 2025.

[93] Mozilla. (2025, abril 3). *Resize Observer API* [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/API/Resize_Observer_API. Accedido: 4 de junio de 2025.

[94] Mozilla. (2025, marzo 31). *Modelo de Objetos del Documento (DOM)* [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model/Introduction. Accedido: 4 de junio de 2025.

[95] AWS Amazon. (s.f.). *¿Qué es el CORS?* [En línea]. Disponible en: https://aws.amazon.com/es/what-is/cross-origin-resource-sharing. Accedido: 12 de junio de 2025.

[96] Lumi Education. (2025, marzo). *Documentación de Lumi Education* [En línea]. Disponible en: https://docs.lumi.education/. Accedido: 17 de junio de 2025.

[97] Microsoft Learn. (2025, mayo 24). *¿Qué es PowerShell?* [En línea]. Disponible en: https://learn.microsoft.com/es-es/powershell/scripting/overview?view=powershell-7.5. Accedido: 18 de junio de 2025.

[98] GitHub. (2025). *GitHub* [En línea]. Disponible en: https://github.com/. Accedido: 20 de junio de 2025.

Presupuesto

En esta sección se realiza la valoración económica del proyecto, detallando los costes asociados a los recursos materiales y humanos. Siguiendo las orientaciones del Colegio Oficial de Graduados e Ingenieros Técnicos de Telecomunicación (COITT), se desglosan los gastos de amortización, el trabajo tarifado y los impuestos aplicables para establecer el presupuesto final de ejecución.

P1. Introducción

Para la elaboración del presente presupuesto, y dado el carácter académico y formativo de este Trabajo de Fin de Grado, se han tomado como referencia las orientaciones y precios publicados por el Colegio Oficial de Graduados e Ingenieros Técnicos de Telecomunicación (COITT). No obstante, es preciso señalar que, en el ejercicio libre de la profesión, los colegios profesionales ya no publican baremos de honorarios de carácter obligatorio, atendiendo a las directivas europeas. Tal y como recordaba el Ministerio de Economía y Hacienda, se "debían eliminar los baremos orientativos de honorarios que tradicionalmente veníamos publicando".[1]

La estimación de gastos de este proyecto se fundamenta en el análisis detallado de los siguientes conceptos:

- Recursos materiales
- Trabajo tarifado por tiempo empleado
- Costes asociados a la redacción del Trabajo Fin de Grado
- Derechos del visado del COITT
- Gastos de tramitación y envíos
- Gastos derivados de los impuestos

A efectos de los cálculos, se ha establecido una dedicación total de **300 horas**, distribuidas a lo largo de los **5 meses** de duración del proyecto.

P2. Recursos materiales

Entre los recursos materiales empleados para la ejecución de este trabajo se distingue entre el hardware necesario para el desarrollo y el software utilizado para la implementación y redacción del proyecto. Para calcular el coste asociado a estos activos, se aplica un método de amortización lineal sobre un período de vida útil de cuatro años (48 meses), asumiendo que tras dicho período los equipos conservan un valor residual.

La fórmula para calcular el coste de amortización mensual es la siguiente:

$$\text{Coste de amortización mensual} = \frac{(\text{Coste de adquisición} - \text{Valor residual})}{\text{Meses de vida útil}}$$

P2.1. Amortización del material hardware

Para el desarrollo del proyecto se ha hecho uso de los siguientes equipos informáticos:

- Ordenador portátil: HP ENVY x360 13-bf0003ns con procesador Intel Core i7 y 16GB de RAM.
- Monitor externo: LG UltraGear 24GS50F-B de 23.7".

El cálculo de la amortización para los cinco meses de duración del proyecto se desglosa en la tabla 7:

Recurso	Valor de adquisición	Valor residual	Coste de amortización
Ordenador portátil	1.350,00 €	340,00 €	105,21 €
Monitor	160,00 €	40,00 €	12,50 €
Total			**117,71 €**

Tabla 7. Tabla de amortización de hardware

Por lo tanto, el coste total correspondiente a la amortización del hardware asciende a **ciento diecisiete euros con setenta y un céntimos (117,71 €)**.

P2.2. Amortización del material software

Para la implementación y documentación del proyecto se han utilizado los siguientes recursos de software:

- **Microsoft Office 365:** Proporcionado bajo la licencia educativa de la Universidad de Las Palmas de Gran Canaria, por lo que su coste de amortización es nulo.
- **Software de código abierto y gratuito:** Se han empleado diversas herramientas libres como Moodle, H5P, draw.io, Cursor y GitHub, cuyo coste de licencia y amortización es cero.
- **Herramientas de Inteligencia Artificial Generativa:** Los modelos de IA utilizados (Gemini, NapkinAI, Perplexity, etc.) se han empleado a través de sus versiones gratuitas o de las capas de acceso sin coste, por lo que no han supuesto ningún gasto directo para el proyecto.

Dado que ninguno de los recursos de software ha requerido la adquisición de licencias de pago, el coste total de amortización de este apartado es nulo.

P3. Trabajo tarifado por tiempo empleado

Para cuantificar el coste de los recursos humanos, se aplican las recomendaciones del Colegio Oficial de Graduados e Ingenieros Técnicos de Telecomunicación (COITT) para el cálculo de honorarios profesionales. La estimación se basa en la aplicación de la siguiente expresión matemática:

$$H = Ct \cdot 74{,}88 \cdot Hn + Ct \cdot 96{,}72 \cdot He$$

Donde los términos se definen como:

- **H:** Representa el importe total de los honorarios a percibir.
- **Ct:** Es un factor de corrección que se aplica en función de las horas totales empleadas.
- **Hn:** Corresponde al número de horas trabajadas dentro del horario laboral ordinario.
- **He:** Corresponde al número de horas extras, trabajadas fuera del horario laboral.

El valor del factor de corrección (Ct) se determina según el total de horas dedicadas al proyecto, tal y como se especifica en la tabla 8 publicada por el COITT.

Horas	Factor de corrección
hasta 36	1
de 36 a 72	0,9
de 72 hasta 108	0,8
de 108 hasta 144	0,7
de 144 hasta 180	0,65
de 180 hasta 360	0,60
de 360 hasta 510	0,55

Tabla 8. Factor de corrección por horas empleadas

Para un total de 300 horas, se obtiene un factor de corrección (Ct) de 0,60. Considerando que la totalidad del trabajo se ha desarrollado en horario laboral (He = 0), el cálculo de los honorarios es el siguiente:

$$H = 0{,}60 \cdot 74{,}88 \cdot 300 + 0{,}60 \cdot 96{,}72 \cdot 0 = \mathbf{13.478{,}40}\ €$$

En consecuencia, el coste derivado de los honorarios profesionales, sin incluir impuestos, se estima en **trece mil cuatrocientos setenta y ocho euros con cuarenta céntimos (13.478,40 €)**.

P4. Costes de redacción y documentación

Para determinar el coste asociado a la elaboración de la presente memoria técnica, se empleará la siguiente expresión de cálculo:

$$R = 0{,}07 \cdot P \cdot Cn$$

Donde cada variable representa:

- **R:** El coste final por la redacción del proyecto.
- **P:** El presupuesto de ejecución material, correspondiente a la suma de los costes calculados en los apartados previos.
- **Cn:** El coeficiente de ponderación, cuyo valor depende de la cuantía del presupuesto.

El presupuesto de ejecución material (P) asciende a **13.596,11 €**, resultado de sumar los costes de Recursos Materiales y Trabajo Tarifado. Dado que este importe es inferior al umbral de 30.050,00 € establecido en las orientaciones de referencia, el coeficiente de ponderación (Cn) aplicable es 1.

Aplicando estos valores a la fórmula, se obtiene:

$$R = 0{,}07 \cdot 13.596{,}11 \cdot 1 = \mathbf{951{,}73}\ €$$

Por consiguiente, los costes correspondientes a la redacción y documentación de este proyecto, antes de impuestos, se estiman en **novecientos cincuenta y un euros con setenta y tres céntimos (951,73 €)**.

P5. Aplicación de Impuestos y Presupuesto Total

Para obtener el coste final de ejecución del proyecto, al subtotal acumulado se le debe añadir la carga fiscal aplicable. En este caso, corresponde a la aplicación del Impuesto General Indirecto Canario (IGIC) del 7%.

La tabla 9 resume y consolida todos los conceptos económicos detallados en este capítulo, presentando el presupuesto total del proyecto.

Concepto	Importe (€)
Recursos Materiales	117,71
Trabajo Tarifado por tiempo empleado	13.478,40
Costes asociados a la redacción del documento	951,73
Subtotal	**14.547,84**
Impuestos (7% IGIC)	1.018,35
Total	**15.566,19 €**

Tabla 9. Presupuesto general del proyecto

De este modo, el presupuesto total para la realización de este Trabajo de Fin de Grado, incluyendo los impuestos aplicables, asciende a **quince mil quinientos sesenta y seis euros con diecinueve céntimos (15.566,19 €)**.

Fdo. Francisco José Cazorla Hernández

Objetivos de Desarrollo Sostenible (ODS)

En esta sección se justifica la alineación del proyecto con varios de los Objetivos de Desarrollo Sostenible (ODS) adoptados por la Organización de las Naciones Unidas (ONU) en el año 2015 para lograr un futuro mejor y más sostenible para todos.

Grado de relación con los ODS

Este Trabajo de Fin de Grado, además de cumplir con sus objetivos técnicos, se alinea con varios Objetivos de Desarrollo Sostenible de las Naciones Unidas. A continuación, se justifica la relación del proyecto con los ODS identificados.

ODS	Grado de relación con los ODS			
	0 No procede	1 Bajo	2 Medio	3 Alto
ODS 1 Fin de la Pobreza	◉	○	○	○
ODS 2 Hambre cero	◉	○	○	○
ODS 3 Salud y Bienestar	◉	○	○	○
ODS 4 Educación de calidad	○	○	◉	○
ODS 5 Igualdad de género	◉	○	○	○
ODS 6 Agua limpia y saneamiento	◉	○	○	○
ODS 7 Energía Asequible y no contaminante	◉	○	○	○
ODS 8 Trabajo decente y crecimiento económico	◉	○	○	○
ODS 9 Industria, Innovación e Infraestructuras	○	◉	○	○
ODS 10 Reducción de las desigualdades	○	◉	○	○
ODS 11 Ciudades y comunidades sostenibles	◉	○	○	○
ODS 12 Producción y consumo sostenibles	◉	○	○	○
ODS 13 Acción por el clima	◉	○	○	○
ODS 14 Vida submarina	◉	○	○	○
ODS 15 Vida de ecosistemas terrestres	◉	○	○	○
ODS 16 Paz, justicia e instituciones sólidas	◉	○	○	○
ODS 17 Alianzas para lograr objetivos	◉	○	○	○

Tabla 10. Grado de alineación del Trabajo de Fin de Grado con los ODS de las Naciones Unidas.

ODS 4: Educación de calidad

El presente trabajo se alinea de manera fundamental con el ODS 4 al desarrollar una herramienta tecnológica diseñada para enriquecer la experiencia de aprendizaje en plataformas de *e-learning* como Moodle. La solución convierte el consumo pasivo de vídeo en un proceso de aprendizaje activo, permitiendo al estudiante analizar, anotar y reflexionar sobre el contenido audiovisual. La posterior transformación de estas anotaciones en un recurso interactivo H5P fomenta la reutilización del conocimiento y el estudio colaborativo, contribuyendo directamente a mejorar la calidad de la educación superior y a fomentar competencias de análisis crítico.

ODS 9: Industria, Innovación e Infraestructuras

El presente trabajo contribuye a este objetivo desde dos perspectivas. En primer lugar, investiga y aplica una de las innovaciones más significativas en la industria del software: la inteligencia artificial generativa. El análisis comparativo de modelos y su uso para el desarrollo de código se enmarcan en la vanguardia de la innovación tecnológica. En segundo lugar, la propia herramienta desarrollada constituye una pieza de infraestructura digital que se integra de forma no invasiva en Moodle, mejorando sus capacidades sin requerir modificaciones en el servidor. Este enfoque de desarrollo ligero y adaptable representa una innovación en sí mismo.

ODS 10: Reducción de las desigualdades

Este trabajo aborda la reducción de desigualdades al promover el acceso a tecnología educativa avanzada. La elección de un modelo de IAG de acceso gratuito (Gemini 2.5 Flash) y el desarrollo de una solución que no requiere licencias de pago ni infraestructura de servidor costosa, democratiza el acceso a herramientas de aprendizaje innovadoras. Cualquier institución o docente con una instancia de Moodle puede implementar esta solución sin necesidad de afrontar costes adicionales, reduciendo la brecha tecnológica entre centros educativos con diferentes capacidades económicas y promoviendo un acceso más equitativo a la educación de calidad.

Anexos

En esta sección se agrupa el material complementario que amplía y justifica la metodología y las herramientas empleadas en el proyecto. Se incluyen las guías de configuración del entorno de desarrollo, los *prompts* utilizados para los benchmarks y el diseño visual, así como la estructura detallada del repositorio de código fuente.

A1. Guía para el usuario de Google AI Studio

Todo el desarrollo de software asistido por IAG documentado en esta memoria se ha llevado a cabo utilizando **Google AI Studio**. Esta plataforma web actúa como un entorno de desarrollo interactivo para la experimentación con los modelos de la familia Gemini, proporcionando un control preciso sobre su comportamiento y un conjunto de herramientas para optimizar el proceso de creación.

A continuación, se detalla la configuración del entorno y el flujo de trabajo seguido en este proyecto.

A.1.1. Estructura de la interfaz

Como se muestra en la figura 66, la interfaz de Google AI Studio se organiza en tres áreas funcionales principales, que han sido clave para el desarrollo iterativo de la herramienta:

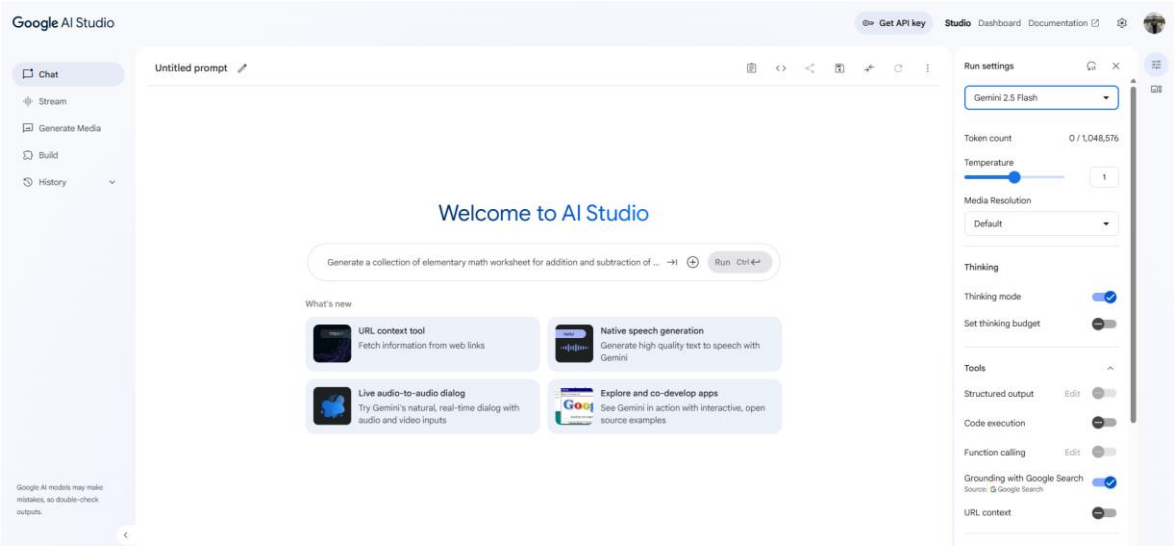


Figura 66. Interfaz principal de Google AI Studio.

- **Panel de navegación (izquierda):** Para la gestión de las sesiones de trabajo.
- **Área de trabajo central:** Donde se redactan, ejecutan y visualizan los *prompts* y las respuestas.
- **Panel de configuración (derecha):** Para ajustar el comportamiento del modelo y sus herramientas.

A.1.2. Gestión de sesiones (panel izquierdo)

El panel izquierdo ha sido fundamental para la organización del proyecto:

- **Nuevo chat:** Permite iniciar una nueva conversación. Cada chat es una sesión independiente, sin conocimiento de las anteriores. Esta función se utilizó para abordar

cada nuevo componente de la herramienta (selector de vídeo, panel de anotaciones, etc.) desde cero, evitando la contaminación contextual entre módulos.

- **Historial (*History*):** Esta sección almacena todas las conversaciones pasadas. Ha sido un recurso indispensable para recuperar *prompts* efectivos, comparar diferentes versiones de un mismo fragmento de código y mantener un registro cronológico del proceso de refinamiento de cada funcionalidad.

A.1.3. Interacción y ejecución (área central y barra superior)

El área central contiene el editor de *prompts* y la respuesta del modelo. La barra superior de esta área ofrece herramientas de productividad que se usaron constantemente:

- **Ejecutar (*Run*):** Envía la instrucción al modelo para que genere una respuesta.
- **Copiar:** Permite transferir el código generado directamente a un editor externo (como Visual Studio Code) para su validación y prueba.
- **Guardar prompt:** Facilita el almacenamiento de *prompts* complejos o bien formulados para su reutilización, una práctica útil para tareas repetitivas como la refactorización de código.

A.1.4. Configuración del modelo y herramientas (panel derecho)

El panel derecho es el centro de control que permite adaptar el modelo a las necesidades del proyecto. La configuración empleada fue la siguiente:

- **Modelo:** Se seleccionó Gemini 2.5 Flash, justificado en el capítulo 3 por su equilibrio entre rendimiento y eficiencia.
- **Temperatura:** Se mantuvo en 0.7, de un máximo de 2. Un valor alto fomenta la diversidad de soluciones, mientras que un valor bajo produce resultados más deterministas y predecibles.
- ***Thinking mode*:** Esta opción se mantuvo desactivada. Se optó por un modo de respuesta directo, donde el modelo genera el código sin un paso de razonamiento previo explícito, priorizando la inmediatez en la respuesta.

La sección ***Tools* (Herramientas)** fue crítica para definir las capacidades del modelo:

- ***Structured output*:** Fuerza al modelo a responder en un formato de datos específico (ej. JSON). Se mantuvo desactivada, ya que el objetivo era obtener código fuente en formato de texto plano, no datos estructurados.

- ***Code execution***: Permite al modelo ejecutar código internamente para autoevaluarlo. Se mantuvo desactivada, ya que la validación del código se realizó externamente en un entorno real (navegador web con Moodle) para garantizar su correcto funcionamiento en el contexto final.
- ***Function calling***: Habilita la integración del modelo con APIs externas. Se mantuvo desactivada al no ser un requisito para este proyecto
- ***Grounding with Google Search:*** Esta herramienta se mantuvo desactivada. Esta decisión es metodológicamente importante, ya que implica que el modelo no tuvo acceso a información en tiempo real de la web. Toda la información necesaria, como ejemplos de uso de librerías o estructuras de código específicas, tuvo que ser proporcionada directamente en el contexto del prompt.
- ***URL context:*** Se mantuvo desactivada en consonancia con el punto anterior, asegurando que el modelo no accediera a fuentes de información externas.

Este control detallado sobre la configuración y las herramientas del modelo ha proporcionado un entorno de desarrollo altamente eficaz, que ha sido clave para la implementación exitosa de la herramienta de software descrita en esta memoria.

A.2. Prompts utilizados para los benchmarks de evaluación

Para garantizar la transparencia y la replicabilidad de los resultados presentados en el capítulo 3, en este apartado se detallan los *prompts* exactos utilizados para cada uno de los benchmarks de generación de código. Estos *prompts* se han extraído de las fuentes oficiales de cada prueba y se presentan en su formato original.

A.2.1. Prompt para HumanEval

Para el benchmark HumanEval, se utiliza una instrucción directa que precede a la firma de la función y su correspondiente *docstring*. El objetivo es que el modelo complete únicamente el cuerpo de la función.

```
Read the following function signature and docstring, and fully implement the function
described. Your response should only contain the code for this function.
```

A.2.2. Prompt para SciCode

El prompt de SciCode es estructurado y diseñado para guiar al modelo a través de la resolución de problemas científicos que requieren conocimiento de dominio específico.

```
PROBLEM DESCRIPTION:
You will be provided with the main description of the problem, previous steps, and
the next step. Your task will be to generate the disciplinary knowledge necessary for
solving the next step and then develop a Python solution focused on this step.

PREVIOUS STEPS DESCRIPTION:
{problem_steps_str}

NEXT STEP - PROBLEM DESCRIPTION AND FUNCTION HEADER:
This part will describe the next step in the problem-solving process. First, provide
the necessary scientific background knowledge as a comment at the beginning of your
response, starting with 'Background: '. Then, a function header will be provided, and
your task is to develop the Python code for this next step based on the provided
description and function header.

{next_step_str}

DEPENDENCIES:
Use only the following dependencies in your solution. Do not include these
dependencies at the beginning of your code.
{dependencies}

RESPONSE GUIDELINES:
1. Start with the scientific background required for the next step, formatted as a
comment.
2. Then write the complete and executable Python program for the next step in a
single block.
3. Your response should focus exclusively on implementing the solution for the next
step, adhering closely to the specified function header and the context provided by
the initial steps.
4. DO NOT include previous function code, example usage or test code in your
response.
5. Ensure your response is in the format of ```python``` and includes the necessary
background as a comment at the top.

Example:
```python
# Background: [Here, insert the necessary scientific knowledge required for the next
step.]

[Insert the Python code here based on the provided function header and dependencies.]
```

A.2.3. Prompts para LiveCodeBench

LiveCodeBench utiliza dos plantillas de *prompt* diferentes, dependiendo de si el problema proporciona un código de inicio o no.

Caso 1: Problemas con código de inicio (starter code)

```
### Question:
{question.question_content}

### Format: You will use the following starter code to write the solution to the
problem and enclose your code within delimiters.
```python
{question.starter_code}
```

Answer: (use the provided format with backticks)
```
```

Caso 2: Problemas sin código de inicio

```
### Question:
{question.question_content}

### Format: Read the inputs from stdin solve the problem and write the answer to
stdout (do not directly test on the sample inputs). Enclose your code within
delimiters as follows. Ensure that when the python program runs, it reads the inputs,
runs the algorithm and writes output to STDOUT.
```python
# YOUR CODE HERE
```

Answer: (use the provided format with backticks
```
```

La presentación de estos *prompts* originales garantiza la total transparencia y replicabilidad de los resultados de los benchmarks analizados en el capítulo 3 de esta memoria.

A.3. Prompts para la Creación de Diagramas y Bocetos

Además de la generación de código, la IAG se utilizó como herramienta de apoyo para la conceptualización visual del proyecto. A continuación, se detallan los *prompts* empleados para generar algunas de las figuras y bocetos que ilustran el diseño de la interfaz de usuario a lo largo de esta memoria.

Prompt para la figura 31: Desplegable de selección de vídeo

- **Título de la figura:** *Figura 31. Desplegable de selección de vídeo al iniciar el recurso de anotación.*
- **Prompt utilizado:**

```
Genera una maqueta de interfaz de usuario que muestre un menú desplegable para la
selección de vídeo. La etiqueta sobre el desplegable debe ser "Seleccione un vídeo:".

El desplegable debe mostrar cuatro opciones: "Seleccione un vídeo", "video1.mp4",
"video2.mp4" y "video3.mp4". Resalta "Seleccione un vídeo" como la opción
seleccionada con una barra azul.

El diseño debe ser minimalista y limpio, con un fondo blanco y sombras suaves para
simular la vista de un navegador. No incluyas elementos adicionales, céntrate
únicamente en el desplegable y su etiqueta.
```

Prompt para la figura 32: Reproductor de vídeo con panel de anotación

- **Título de la figura:** *Figura 32. Reproductor de vídeo con botones de interacción y formulario lateral para registrar la anotación.*
- **Prompt utilizado:**
```

```
Diseña una interfaz web para una herramienta de anotación de vídeo. En el lado
izquierdo, incluye una sección con la etiqueta "Seleccione un vídeo:" seguida de un
desplegable que muestre "video1.mp4" como la opción seleccionada.

Debajo, añade un marcador de posición rectangular para el reproductor de vídeo con un
gran botón de reproducción en el centro.

Bajo el vídeo, sitúa dos botones: "Reproducir" y "Marcar".

En el lado derecho, muestra un formulario con la etiqueta "Añadir marca". Debe
incluir un desplegable etiquetado "Color" con la opción "Azul" seleccionada, un campo
de entrada etiquetado "Elemento identificado" con texto de marcador de posición, y un
área de texto etiquetada "Descripción" con un marcador de posición.

Al final del formulario, coloca un botón azul con la etiqueta "Añadir Anotación".

Mantén la interfaz limpia, blanca y con un diseño modular.
```

Prompt para la figura 33: Marca circular sobre el vídeo con caja flotante

- **Título de la figura:** *Figura 33. Vista de una marca generada sobre el vídeo, con el contenido semántico mostrado en una caja flotante.*
- **Prompt utilizado:**

```
Crea una maqueta web de un fotograma de vídeo pausado con una anotación.

En el lado izquierdo, muestra un reproductor de vídeo con "video1.mp4" seleccionado
en el desplegable superior.

Dentro del marco del vídeo, dibuja un círculo azul que represente una marca visual,
con un cursor al lado. Junto al círculo, añade una etiqueta flotante con el texto en
negrita "Elemento 1:" seguido de la palabra "Descripción" en una fuente más clara.

Debajo del vídeo, incluye los botones "Reproducir" y "Marcar". En el lado derecho,
muestra el panel "Añadir marca" con un desplegable de color (seleccionado: Azul), un
campo de entrada para "Elemento identificado", un área de texto para "Descripción" y
un botón azul de "Añadir Anotación".

Utiliza un diseño blanco y moderno con un contraste claro entre los componentes.
```

Prompt para la figura 58: Boceto de la interfaz de generación H5P

- **Título de la Figura:** *Figura 58. Boceto conceptual de la interfaz de usuario para el generador de paquetes H5P.*
- **Prompt utilizado:**

```
Necesito un boceto visual para la interfaz de una herramienta web. La finalidad de la
herramienta es la creación de paquetes H5P a partir de ficheros de vídeo y datos. El
diseño debe ser minimalista y funcional.

Prioriza la claridad y la facilidad de uso, empleando una tipografía legible. La
disposición de los elementos debe ser centrada y ordenada, con suficiente espacio en
blanco.

Estructura de la Interfaz:
 1. Un título principal que identifique la herramienta: "Generador de Paquetes H5P
    de Video Interactivo".
 2. Dos bloques de entrada de archivos, visualmente separados pero con un diseño
    coherente:
    ▪ Primer bloque: Dedicado a la selección de un único archivo de vídeo (.mp4), con
      su respectivo botón de carga y un indicador de estado.
    ▪ Segundo bloque: Destinado a la selección de uno o varios archivos de datos
      (.json), igualmente con su botón y área de estado.
 3. Un botón final, claramente diferenciado y ubicado en la parte inferior, que
    iniciará el proceso de generación del paquete H5P.
```

La inclusión de estos *prompts* ilustra la metodología sistemática empleada para la conceptualización visual del proyecto. Se demuestra así que las herramientas de IAG no solo han sido un pilar en el desarrollo del código, sino también un recurso valioso para el diseño de interfaces y la creación de material gráfico de apoyo.

A.4. Metodología para la estimación de requisitos de hardware

La estimación de los requisitos de hardware para la ejecución de modelos de lenguaje (LLM) en entornos locales, presentada en la tabla 3, se basa en la interacción de tres factores clave:

- **Parámetros del modelo:** El número de parámetros (ej. 7B para 7 mil millones) define el tamaño y la capacidad teórica del modelo. A mayor número de parámetros, mayor es el requisito de memoria para almacenarlos.
- **Cuantización (*Quantization*):** Es la técnica que reduce la precisión numérica de los parámetros (generalmente de 16 a 4 bits) para disminuir el tamaño del modelo. Todas las estimaciones de este trabajo se basan en modelos cuantizados a 4 bits en formato GGUF.
- **Memoria (VRAM y RAM):** La memoria de la GPU (VRAM) es el recurso crítico para una ejecución fluida. Si el modelo no cabe en la VRAM, utiliza la RAM del sistema, lo que provoca una notable ralentización. El objetivo es que el modelo y su contexto quepan en la VRAM.

Fórmula de cálculo de requisitos

El tamaño base de un modelo cuantizado se puede estimar con la siguiente fórmula:

$$\text{Tamaño (GB)} = (\text{Parámetros en miles de millones} \cdot \text{Bits por parámetro}) / 8$$

A este tamaño base se le debe sumar un margen de memoria (*overhead*) para el contexto y los cálculos, que suele estimarse en 1-2 GB.

Ejemplo de cálculo para LLaMA 3 8B (4 bits):

- **Tamaño base:** $(8 \cdot 4) / 8 = 4\ \text{GB}$
- **VRAM Recomendada:** $4\ \text{GB (modelo)} + \sim 2\ \text{GB (overhead)} \approx 6\ \text{GB}$.

Este cálculo justifica la recomendación de una GPU con **8 GB de VRAM** para asegurar un margen operativo suficiente. Esta misma metodología se ha aplicado para estimar los requisitos del resto de los modelos listados.

A.5. Repositorio de prompts, respuestas y código generado

Con el fin de ofrecer transparencia, replicabilidad y acceso completo a todos los materiales generados durante el desarrollo, se crea un repositorio público en la plataforma GitHub. Este repositorio centraliza todos los elementos clave del proyecto, incluyendo los *prompts* enviados al modelo, las respuestas obtenidas y el código fuente final de la herramienta.

A.5.1. Creación y Configuración Inicial

El primer paso consiste en la creación de un nuevo repositorio público en GitHub, denominado *Repositorio-TFG*, como se muestra en la figura 67.

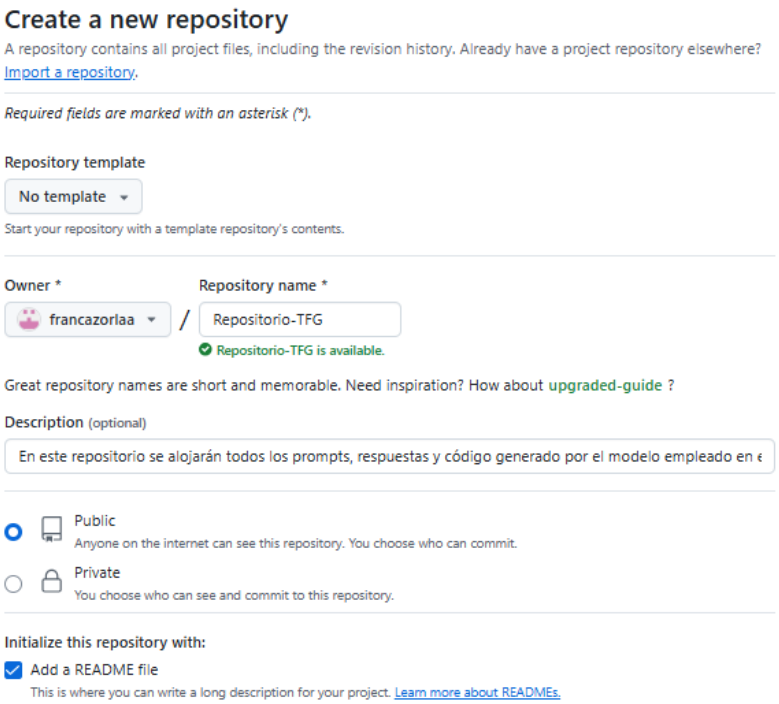


Figura 67. Interfaz de creación de repositorios en GitHub.

A.5.2. Estructura y sincronización local

Posteriormente, se prepara el entorno de desarrollo local. Se crea una estructura de directorios para organizar los componentes del proyecto en cuatro carpetas principales: *generacion-informes*, *marcado-videos*, *video-interactivo y estructura-h5p*.

A continuación, se inicializa un repositorio Git en este directorio y se enlaza con su contraparte remota en GitHub mediante los comandos `git init` y `git remote add origin`.

```
~\TFG\repositorio-gemini git:(main)  (0.122s)
git init
Initialized empty Git repository in C:/Users/franc/TFG/repositorio-gemini/.git/

~\TFG\repositorio-gemini git:(master)  (0.128s)
git remote add origin https://github.com/francazorlaa/Repositorio-TFG.git
```

Figura 68. Comandos para la inicialización y sincronización del repositorio.

A.5.3. Publicación de Contenidos

Con el repositorio ya configurado, se añaden todos los ficheros del proyecto al área de preparación (*staging*) y se realiza un *commit* inicial para registrar el estado de todos los artefactos. Finalmente, se ejecuta el comando `git push` para sincronizar el contenido local con el repositorio remoto, subiendo todos los directorios y archivos a la rama `master`, como se aprecia en las figura 69 y 70.

```
~\TFG\repositorio-gemini git:(master)  106 files changed, 68519 insertions(+)  (0.214s)
git commit -m "Commit inicial"

[master 4675e45] Commit inicial
 106 files changed, 68519 insertions(+)
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoPagina1.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoPagina2.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoPagina3.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoPagina4.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoPagina5.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoPagina6.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoTarea1.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoTarea2.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoTarea3.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoTarea4.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoTarea5.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMCodigoTarea6.html
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMPrompt1.txt
 create mode 100644 generacion-informes/activacion-interfaz-modal/AIMPrompt2.txt
```

Figura 69. Proceso de confirmación (commit) de los archivos.

```
~\TFG\repositorio-gemini git:(master)  (2.943s)
git push origin master

Enumerating objects: 118, done.
Counting objects: 100% (118/118), done.
Delta compression using up to 8 threads
Compressing objects: 100% (117/117), done.
Writing objects: 100% (117/117), 213.94 KiB | 2.93 MiB/s, done.
Total 117 (delta 61), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (61/61), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/francazorlaa/Repositorio-TFG/pull/new/master
remote:
To https://github.com/francazorlaa/Repositorio-TFG.git
 * [new branch]      master -> master
```

Figura 70. Proceso de subida (push) de los archivos a GitHub.

Este proceso finaliza con la publicación de todo el material de desarrollo, haciendo que el código fuente, los *prompts* y las respuestas sean públicamente accesibles.

A.5.4. GitHub: Estructura de directorios y ficheros

A continuación, se detalla la estructura de directorios y ficheros del repositorio, organizados por cada uno de los módulos funcionales del proyecto.

Directorio: marcado-videos

URL: https://github.com/francazorlaa/Repositorio-TFG/tree/master/marcado-videos

Este directorio contiene todos los elementos relacionados con el desarrollo del módulo de anotación de vídeos. Se subdivide en tres componentes funcionales, cada uno con sus respectivos *prompts*, respuestas y código generado. Los ficheros de cada subdirectorio siguen una nomenclatura consistente, donde las iniciales (SC, PA, CG) se corresponden con el componente.

- **selector-area-reproduccion/**
  - SCCodigo1.html
  - SCCodigo2.html
  - SCCodigo3.html
  - SCCodigo4.html
  - SCCodigo5.html
  - SCCodigo6.html
  - SCCodigo7.html
  - SCPrompt1.txt
  - SCPrompt2.txt
  - SCPrompt3.txt
  - SCPrompt4.txt
  - SCPrompt5.txt

  - SCPrompt6.txt
  - SCPrompt7.txt
  - SCRespuesta1.txt
  - SCRespuesta2.txt
  - SCRespuesta3.txt
  - SCRespuesta4.txt
  - SCRespuesta5.txt
  - SCRespuesta6.txt
  - SCRespuesta7.txt

- **panel-anotacion/**
  - PACodigo1.html
  - PACodigo2.html
  - PACodigo3.html
  - PACodigo4.html
  - PACodigo5.html
  - PACodigo6.html
  - PAPrompt1.txt
  - PAPrompt2.txt
  - PAPrompt3.txt
  - PAPrompt4.txt
  - PAPrompt5.txt
  - PAPrompt6.txt
  - PARespuesta1.txt
  - PARespuesta2.txt
  - PARespuesta3.txt
  - PARespuesta4.txt
  - PARespuesta5.txt
  - PARespuesta6.txt

- **capa-grafica/**
  - CGCodigo1.html
  - CGCodigo2.html
  - CGCodigo3.html
  - CGCodigo4.html
  - CGCodigo5.html
  - CGCodigo6.html
  - CGPrompt1.txt

- CGPrompt2.txt
- CGPrompt3.txt
- CGPrompt4.txt
- CGPrompt5.txt
- CGPrompt6.txt
- CGRespuesta1.txt
- CGRespuesta2.txt
- CGRespuesta3.txt
- CGRespuesta4.txt
- CGRespuesta5.txt
- CGRespuesta6.txt

Directorio: generacion-informes

URL: https://github.com/francazorlaa/Repositorio-TFG/tree/master/generacion-informes

Este directorio alberga los elementos correspondientes al módulo de generación y entrega de informes. Se organiza en tres subdirectorios que reflejan sus componentes funcionales. La nomenclatura de los ficheros utiliza las iniciales de cada componente (AIM, GA, EM).

- **activacion-interfaz-modal/**
  - AIMCodigoPagina1.html
  - AIMCodigoPagina2.html
  - AIMCodigoPagina3.html
  - AIMCodigoPagina4.html
  - AIMCodigoPagina5.html
  - AIMCodigoPagina6.html
  - AIMCodigoTarea1.html
  - AIMCodigoTarea2.html
  - AIMCodigoTarea3.html
  - AIMCodigoTarea4.html
  - AIMCodigoTarea5.html
  - AIMCodigoTarea6.html
  - AIMPrompt1.txt
  - AIMPrompt2.txt
  - AIMPrompt3.txt
  - AIMPrompt4.txt
  - AIMPrompt5.txt
  - AIMPrompt6.txt

  - AIMRespuesta1.txt
  - AIMRespuesta2.txt
  - AIMRespuesta3.txt
  - AIMRespuesta4.txt
  - AIMRespuesta5.txt
  - AIMRespuesta6.txt

- **generacion-archivos/**
  - GACodigoPagina7.html
  - GACodigoPagina8.html
  - GACodigoPagina10.html
  - GACodigoPagina11.html
  - GACodigoTarea7.html
  - GACodigoTarea8.html
  - GACodigoTarea10.html
  - GACodigoTarea11.html
  - GAPrompt7.txt
  - GAPrompt8.txt
  - GAPrompt10.txt
  - GAPrompt11.txt
  - GARespuesta7.txt
  - GARespuesta8.txt
  - GARespuesta10.txt
  - GARespuesta11.txt

- **entrega-moodle/**
  - EMCodigoPagina9.html
  - EMCodigoTarea9.html
  - EMPrompt9.txt
  - EMRespuesta9.txt

Directorio: video-interactivo

URL: https://github.com/francazorlaa/Repositorio-TFG/tree/master/video-interactivo

Este directorio contiene los elementos del módulo final, encargado de la creación de paquetes H5P interactivos. Se estructura en tres subdirectorios que reflejan las fases del proceso: replicación de la plantilla, transformación de datos y depuración del paquete final. La nomenclatura de los ficheros utiliza las iniciales de cada componente (RP, TM, GD).

- **replicacion-plantilla/**
  - RPCodigo1.html
  - RPCodigo2.html
  - RPCodigo3.html
  - RPCodigo5.html
  - RPCodigo8.html
  - RPPrompt1.txt
  - RPPrompt2.txt
  - RPPrompt3.txt
  - RPPrompt4.txt
  - RPPrompt5.txt
  - RPPrompt8.txt
  - RPRespuesta1.txt
  - RPRespuesta2.txt
  - RPRespuesta3.txt
  - RPRespuesta4.txt
  - RPRespuesta5.txt
  - RPRespuesta8.txt
- **transformacion-modificacion/**
  - TMCodigo6.html
  - TMCodigo7.html
  - TMCodigoPagina1.html
  - TMCodigoPagina2.html
  - TMCodigoPagina3.html
  - TMCodigoPagina4.html
  - TMCodigoTarea1.html
  - TMCodigoTarea2.html
  - TMCodigoTarea3.html
  - TMCodigoTarea4.html
  - TMPrompt1.txt
  - TMPrompt2.txt
  - TMPrompt3.txt
  - TMPrompt4.txt
  - TMPrompt6.txt
  - TMPrompt7.txt
  - TMRespuesta1.txt

  - TMRespuesta2.txt
  - TMRespuesta3.txt
  - TMRespuesta4.txt
  - TMRespuesta6.txt
  - TMRespuesta7.txt

- **generacion-depuracion/**
  - GDCodigo9.html
  - GDCodigo10.html
  - GDCodigo12.html
  - GDCodigo13.html
  - GDPrompt9.txt
  - GDPrompt10.txt
  - GDPrompt11.txt
  - GDPrompt12.txt
  - GDPrompt13.txt
  - GDPrompt14.txt
  - GDRespuesta9.txt
  - GDRespuesta10.txt
  - GDRespuesta11.txt
  - GDRespuesta12.txt
  - GDRespuesta13.txt
  - GDRespuesta14.txt

Directorio: estructura-h5p

URL: https://github.com/francazorlaa/Repositorio-TFG/tree/master/estructura-h5p

Este directorio contiene la estructura completa de una plantilla base para un paquete H5P de tipo "Vídeo Interactivo". Incluye todas las librerías de contenido, las librerías del editor y los ficheros de configuración necesarios para que el paquete sea funcional. Esta estructura es la que se replica en memoria durante el proceso de generación de paquetes descrito en el capítulo 6.

- **Directorios de librerías y dependencias:**
  - FontAwesome-4.5
  - H5P.AdvancedText-1.1
  - H5P.Audio-1.5
  - H5P.Blanks-1.14
  - H5P.CKEditor-1.0

- H5P.DragNBar-1.5
- H5P.DragNDrop-1.1
- H5P.DragNResize-1.2
- H5P.DragQuestion-1.14
- H5P.DragText-1.10
- H5P.FontIcons-1.0
- H5P.FreeTextQuestion-1.0
- H5P.GoToQuestion-1.3
- H5P.GuidedTour-1.0
- H5P.IVHotspot-1.2
- H5P.Image-1.1
- H5P.InteractiveVideo-1.26
- H5P.JoubelUI-1.3
- H5P.Link-1.3
- H5P.MarkTheWords-1.11
- H5P.MaterialDesignIcons-1.0
- H5P.MultiChoice-1.16
- H5P.MultiMediaChoice-0.3
- H5P.Nil-1.0
- H5P.OpenEndedQuestion-1.0
- H5P.Question-1.5
- H5P.Questionnaire-1.3
- H5P.SimpleMultiChoice-1.1
- H5P.SingleChoiceSet-1.11
- H5P.Summary-1.10
- H5P.Table-1.1
- H5P.Text-1.1
- H5P.TextUtilities-1.3
- H5P.Transition-1.0
- H5P.TrueFalse-1.8
- H5P.Video-1.6
- H5PEditor.ColorSelector-1.3
- H5PEditor.DragQuestion-1.10
- H5PEditor.Duration-1.1
- H5PEditor.InteractiveVideo-1.25
- H5PEditor.RadioGroup-1.1
- H5PEditor.RangeList-1.0

    - H5PEditor.SelectToggleFields-1.1
    - H5PEditor.ShowWhen-1.0
    - H5PEditor.SingleChoiceSetTextualEditor-1.0
    - H5PEditor.SummaryTextualEditor-1.1
    - H5PEditor.TableList-1.0
    - H5PEditor.Timecode-1.2
    - H5PEditor.UrlField-1.2
    - H5PEditor.VerticalTabs-1.3
    - H5PEditor.Wizard-1.2
    - Shepherd-1.0
    - Tether-1.0
    - jQuery.ui-1.10
- **Directorio de contenido:**
    - content
- **Fichero de manifiesto:**
    - h5p.json

Directorio: codigo-final

URL: https://github.com/francazorlaa/Repositorio-TFG/tree/master/codigo-final

Este directorio contiene los ficheros HTML finales y consolidados de la herramienta, listos para ser utilizados. Cada fichero integra el código HTML, CSS y JavaScript necesario para el funcionamiento de un módulo específico.

- CodigoFinalH5P.html: Contiene el código completo del generador de paquetes H5P.
- CodigoFinalPagina.html: Contiene el código de la herramienta de anotación para ser insertado en un recurso de tipo "Página" de Moodle.
- CodigoFinalTarea.html: Contiene la versión de la herramienta de anotación adaptada para un recurso de tipo "Tarea" de Moodle, incluyendo la funcionalidad de entrega.

La documentación de esta estructura de directorios y ficheros proporciona una visión completa de todos los elementos generados y organizados durante el desarrollo del proyecto, asegurando la replicabilidad del trabajo realizado.