UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA DE COMPUTACIÓN

DESARROLLO DE UN PROTOTIPO DE GENERACIÓN AUTOMÁTICA DE VIDEOS UTILIZANDO UNA API DE INTELIGENCIA ARTIFICIAL EN UN ENTORNO DE GOOGLE CLOUD PLATFORM Y DOCKER

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Ciencias de la Computación

AUTOR: ERICK BRYAN PILLIZA QUISHPE

TUTOR: GUSTAVO ERNESTO NAVAS RUILOVA

Quito - Ecuador
2025

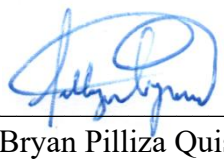
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Erick Bryan Pilliza Quishpe con documento de identificación N° 1719252353 manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total parcial el presente trabajo de titulación.

Quito, 5 de agosto de 2025

Atentamente,



Erick Bryan Pilliza Quishpe
1719252353

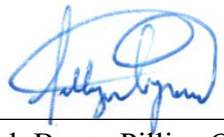
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Erick Bryan Pilliza Quishpe con documento de identificación N° 1719252353, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Artículo Académico: "Desarrollo de un prototipo de generación automática de videos utilizando una API de inteligencia artificial en un entorno de Google Cloud Platform y Docker", el cual ha sido desarrollado para optar por el título de: Ingeniero en Ciencias de la Computación, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 5 de agosto de 2025

Atentamente,



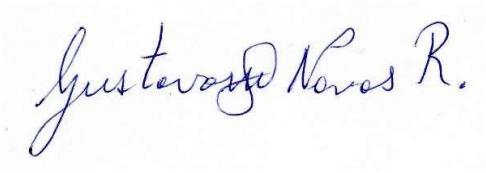
Erick Bryan Pilliza Quishpe
1719252353

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Gustavo Ernesto Navas Ruilova con documento de identificación N° 1705675625, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UN PROTOTIPO DE GENERACIÓN AUTOMÁTICA DE VIDEOS UTILIZANDO UNA API DE INTELIGENCIA ARTIFICIAL EN UN ENTORNO DE GOOGLE CLOUD PLATFORM Y DOCKER, realizado por Erick Bryan Pilliza Quishpe con documento de identificación N° 1719252353, obteniendo como resultado final el trabajo de titulación bajo la opción de Artículo Académico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 5 de agosto de 2025

Atentamente,



_____

Ing. Gustavo Ernesto Navas Ruilova MSc.
1705675625

DEDICATORIA

Dedico este trabajo a mis padres y a mi hermana, quienes, con su amor incondicional y su fortaleza, han sido mi mayor fuente de inspiración y apoyo. Cada día, su confianza en mí me impulsó a seguir adelante, incluso en los momentos más desafiantes. Gracias a su paciencia, palabras de aliento y ejemplo de perseverancia, pude superar las dificultades que se presentaron en el camino y convertirlas en aprendizajes valiosos.

A través de este artículo, quiero demostrar que nunca es tarde para perseguir un sueño o completar aquello que se ha comenzado. Lo importante es atreverse a dar el primer paso, porque cada esfuerzo, por pequeño que parezca, nos acerca un poco más a nuestras metas.

A ellos, con todo mi amor y gratitud, les dedico este logro.

Erick Bryan Pilliza Quishpe

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento a mi tutor de tesis, el Ing. Gustavo Navas, por su paciencia, apoyo y valiosa guía a lo largo de este artículo. Su compromiso constante, su atención a cada detalle en la evolución del trabajo y su disposición para ofrecer los recursos disponibles fueron fundamentales para la realización de este trabajo.

Un agradecimiento especial también al grupo de investigación IDEAGEOCA, cuyo respaldo en aspectos técnicos enriqueció significativamente este artículo. Asimismo, extiendo mi gratitud a la Universidad Politécnica Salesiana, institución que ha sido un pilar fundamental en mi formación personal y académica, brindándome los conocimientos necesarios para crecer profesionalmente.

Finalmente, quiero agradecer de corazón a mi familia, quienes han sido mi fortaleza y mi mayor fuente de motivación en esta hermosa carrera que he elegido. Sin su apoyo incondicional, este logro no habría sido posible.

A todos ustedes, mi más profundo agradecimiento.

Erick Bryan Pilliza Quishpe

DESARROLLO DE UN PROTOTIPO DE GENERACIÓN AUTOMÁTICA DE VIDEOS UTILIZANDO UNA API DE INTELIGENCIA ARTIFICIAL EN UN ENTORNO DE GOOGLE CLOUD PLATFORM Y DOCKER

DEVELOPMENT OF A PROTOTYPE OF AUTOMATIC VIDEO GENERATION USING AN ARTIFICIAL INTELLIGENCE API IN A GOOGLE CLOUD PLATFORM AND DOCKER ENVIRONMENT

Erick Pilliza-Quishpe[1], Gustavo Navas-Ruilova[2]

Resumen

El presente trabajo aborda el desarrollo de un prototipo para la generación automática de videos mediante la integración de una API de inteligencia artificial. Este prototipo fue diseñado y desarrollado utilizando Python como lenguaje principal, y se aloja en un entorno de Google Cloud Platform (GCP), aprovechando su escalabilidad y capacidad de gestión en la nube. Además, se empleó Docker para garantizar la portabilidad y consistencia del sistema entre diferentes entornos de ejecución. La arquitectura combina modelos avanzados de procesamiento de lenguaje natural, como ChatGPT, para la generación de texto, junto con la API de D-ID para la creación de videos animados. Este enfoque permite a los usuarios transformar texto personalizado o generado automáticamente en contenido multimedia de manera dinámica. Los resultados demuestran la eficacia de integrar tecnologías de IA en aplicaciones prácticas para la automatización creativa, facilitando procesos de creación de contenido innovadores y accesibles.

Palabras clave: Modularidad, Asincronía, Autenticación, Ecosistema en la nube, Desarrollo Web, APIs

Abstract

This work addresses the development of a prototype for the automatic generation of videos through the integration of an artificial intelligence API. The prototype was designed and developed using Python as the main programming language and is hosted in a Google Cloud Platform (GCP) environment, leveraging its scalability and cloud management capabilities. Additionally, Docker was utilized to ensure the system's portability and consistency across different execution environments. The architecture combines advanced natural language processing models, such as ChatGPT, for text generation, with the D-ID API for creating animated videos. This approach allows users to dynamically transform personalized or automatically generated text into multimedia content. The results demonstrate the effectiveness of integrating AI technologies into practical applications for creative automation, streamlining innovative and accessible content creation processes.

Keywords: Modularity, Asynchrony, Authentication, Cloud Ecosystem, Web Development, APIs

[1] Estudiante de la Carrera de Computación, Universidad Politécnica Salesiana.

[2] Docente de la Carrera de Computación, Universidad Politécnica Salesiana.

Autor para correspondencia: erickpilliza14@gmail.com

1. Introducción

En la vertiginosa evolución tecnológica, se observa un cambio disruptivo en la forma en que las máquinas interactúan con los humanos, logrando niveles de autonomía y creatividad impensables hace apenas una década. La inteligencia artificial (IA) se ha consolidado como una herramienta esencial para abordar problemas complejos en múltiples disciplinas, desde la medicina hasta el entretenimiento [1]. Entre estas aplicaciones, la generación automática de contenido audiovisual ha emergido como un área de investigación innovadora, con un impacto significativo en la personalización de experiencias digitales y la automatización de procesos creativos [2].

El desarrollo de sistemas para la creación de videos generados automáticamente mediante IA es un ejemplo destacado de cómo las tecnologías avanzadas están siendo integradas en plataformas prácticas. Este enfoque combina algoritmos de aprendizaje profundo, s y generación de contenido visual, permitiendo resultados más eficientes y adaptables [3]. Sin embargo, la implementación de estos sistemas plantea desafíos técnicos relacionados con la integración de múltiples herramientas y la necesidad de infraestructuras escalables y seguras.

En este contexto, el presente trabajo propone un prototipo para la generación automática de videos mediante el uso de una API de inteligencia artificial, implementado en un entorno que combina Google Cloud Platform (GCP) y Docker. Esta solución, diseñada y ejecutada con Python, explora el potencial de las tecnologías emergentes para ofrecer una herramienta versátil y de alto rendimiento. Además, aborda aspectos clave como la interoperabilidad entre sistemas, la portabilidad de aplicaciones y la eficiencia en el uso de recursos [4].

El análisis de esta propuesta se sustenta en una revisión exhaustiva de investigaciones recientes, lo que permite contextualizar sus contribuciones dentro del panorama actual del desarrollo tecnológico. Este prototipo no solo pretende demostrar la viabilidad técnica del enfoque, sino también destacar el valor de la IA como motor de innovación en la creación de contenido audiovisual personalizado y automatizado.

2. Materiales y Métodos

En esta sección, se da a conocer los materiales que se utilizaron para el desarrollo del sistema, así como los métodos empleados en la implementación del prototipo. El sistema de generación automática de videos se desarrolló utilizando el apoyo de APIs de inteligencia artificial, infraestructura en la nube y contenedores Docker para asegurar la portabilidad y escalabilidad del mismo.

2.1. Materiales

Los materiales empleados para el desarrollo del prototipo. Dado que el enfoque principal del proyecto es la integración y uso de tecnologías basadas en inteligencia artificial, se detalla el conjunto de herramientas y software utilizados para garantizar el correcto funcionamiento del sistema.

2.1.1 Software

- **Flask**: Framework de Python utilizado para desarrollar la interfaz del usuario y manejar las solicitudes entre el cliente y el servidor.
- **Requests**: Librería de Python empleada para realizar solicitudes HTTP, crucial para la comunicación con las APIs.

- **OpenAI**: Librería oficial para interactuar con la API de OpenAI GPT, facilitando la generación de texto dinámico.
- **Postman**: Herramienta utilizada para probar y verificar las integraciones con las APIs antes de su implementación en el prototipo.
- **Docker**: Plataforma utilizada para la contenerización del prototipo, asegurando un entorno de desarrollo reproducible y escalable.

2.2. Métodos

Los métodos utilizados para llevar a cabo el desarrollo, integración y evaluación del prototipo propuesto. Estos métodos se centraron en garantizar la correcta comunicación entre las APIs y los componentes del sistema, así como en evaluar su funcionalidad dentro de un entorno web montado en Google Cloud Platform(GCP).

2.2.1 Integración de APIs de inteligencia artificial

El componente central del prototipo es la integración de APIs, ya que permiten la conexión eficiente y segura con servicios externos especializados en la generación de contenido. La integración de la API de ChatGPT facilita la creación de texto dinámico y coherente a partir de entradas del usuario, mientras que la API de D-ID transforma estos textos en videos interactivos, ofreciendo una experiencia multimedia completa. El uso de tecnologías de autenticación avanzadas, como el esquema Bearer Token, garantiza la seguridad en cada interacción con estas APIs, protegiendo tanto los datos del usuario como el flujo de trabajo del sistema. De esta forma, las APIs no solo optimizan el desarrollo y la funcionalidad del sistema, sino que también contribuyen a la escalabilidad, flexibilidad y seguridad del proyecto. Ambas integraciones trabajan de manera coordinada, logrando un flujo eficiente y seguro desde la generación textual hasta la creación de contenido multimedia, asegurando la calidad y seguridad del prototipo de principio a fin.

2.2.1.1 API de ChatGPT

La integración de la API de ChatGPT fue fundamental para la generación de texto a partir de las solicitudes de entrada del usuario. Este proceso empleó el esquema de autenticación *Bearer Token*, garantizando un acceso seguro y restringido al servicio. La comunicación con la API de ChatGPT se realizó mediante solicitudes HTTP utilizando el método POST, enviando datos en formato JSON, que incluían las instrucciones específicas y el contexto necesario para la generación de contenido, para comprobar y posteriormente implementar esto se utilizó Postman como se puede divisar en la figura 1. Para mejorar la seguridad, todas las interacciones con la API fueron protegidas mediante el protocolo HTTPS, evitando la exposición de datos sensibles durante el tránsito.

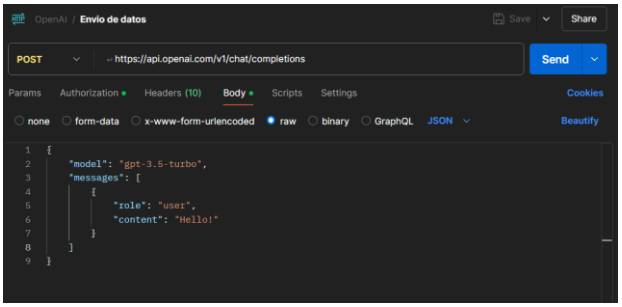


Figura1. Prueba de conexión hacia la API de OpenAI por el método POST en Postman.

2.2.1.2 API de D-ID

La integración de la API de D-ID fue esencial para la generación automática de videos a partir de texto proporcionado por el usuario. Este proceso se llevó a cabo utilizando el esquema de autenticación Basic, el cual garantiza un acceso restringido y seguro al servicio. La comunicación con la API de D-ID se realiza principalmente mediante solicitudes HTTP utilizando el método POST, enviando los datos en formato JSON. En estas solicitudes se incluyen varios parámetros, como el texto a ser convertido en un video, las configuraciones del proveedor de voz, como el idioma y el identificador de voz, así como la imagen que servirá de fondo para la animación. Esta

solicitud se muestra en la figura 2, que ilustra cómo se estructuran y envían los datos hacia la API para iniciar la creación del video.

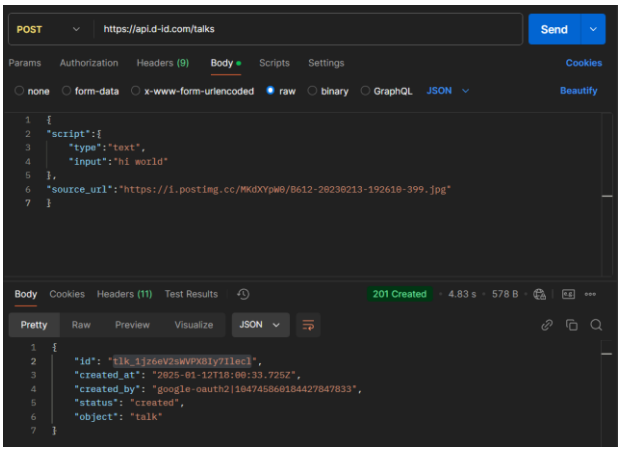


Figura 2. Prueba de conexión hacia la API de D-ID por el método POST en Postman.

Una vez enviada la solicitud POST, la API devuelve un identificador único para el video generado. Este identificador se utiliza en una solicitud posterior con el método GET para verificar el estado del video y obtener el enlace final del resultado. La solicitud GET consulta continuamente el estado de la generación del video y, cuando el estado es "done", se obtiene la URL para acceder al video final. Esta interacción se muestra en la figura 3, donde se muestra cómo se obtienen los datos de la API para verificar la finalización del proceso y descargar el video.

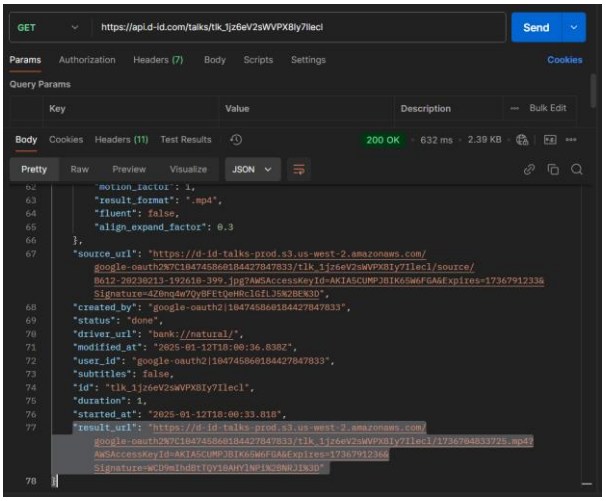


Figura 3. Prueba de conexión hacia la API de D-ID por el método GET en Postman.

2.2.2 Diseño del Prototipo

El diseño del prototipo para la generación automática de videos comenzó con el objetivo de crear una plataforma interactiva y funcional que permita a los usuarios generar contenido de video a partir de texto. La arquitectura del prototipo se basó en la arquitectura propuesta en el articulo presentado por Navas [5]. Para este propósito, se seleccionó Flask, un marco de trabajo ligero para el desarrollo de aplicaciones web en Python, debido a su facilidad de integración con diversos servicios y su capacidad para gestionar solicitudes HTTP de manera eficiente. El servidor web utilizado fue Flask, el cual despliega la página. Para garantizar el acceso desde una IP externa, se configuró el entorno en Google Cloud Platform (GCP). Posteriormente, se utilizaron las APIs, las cuales se comunican a través de los métodos GET y POST, permitiendo finalmente que el usuario reciba la información procesada, tal como se muestra en la Figura 4.

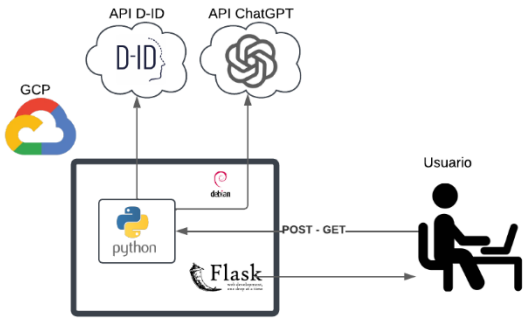


Figura 4. Arquitectura del prototipo

La interfaz del prototipo consta de dos formularios principales: uno para generar texto a partir de una solicitud del usuario mediante la API de ChatGPT, y otro para convertir dicho texto en un video utilizando la API de D-ID. El sistema está diseñado de manera que, en primer lugar, los usuarios pueden ingresar un texto o dejar que ChatGPT lo genere de manera automática, y luego, con un solo clic, ese texto es procesado para producir un video con una imagen y una voz configurada previamente.

Dando como resultado lo que se puede divisar en la figura 5.

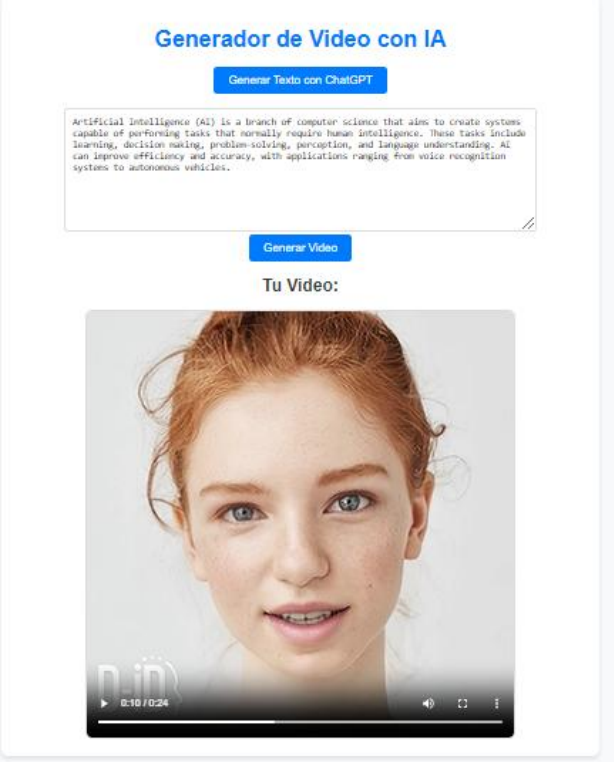


Figura 5. Interfaz del prototipo generado.

Para el desarrollo, se optó por una estructura modular, separando el código en componentes específicos para manejar la generación de texto, la integración con D-ID, y la gestión de la interfaz de usuario. Este enfoque no solo facilita el mantenimiento del sistema, sino que también permite realizar mejoras de manera independiente en cada uno de los componentes, sin afectar al resto del prototipo.

2.2.3 Manejo de Errores en la Integración de APIs.

El manejo adecuado de errores es una parte integral del desarrollo de sistemas que interactúan con servicios externos, como es el caso de las APIs utilizadas en este prototipo: ChatGPT para la generación de texto y D-ID para la creación de videos a partir del texto generado. En el contexto de este proyecto, se ha implementado un sistema robusto para la gestión de errores, lo que permite mejorar tanto la estabilidad como la experiencia del usuario. A continuación, se detalla cómo se gestionan los errores en el código específico de la integración de ambas APIs..

2.2.3.1 Gestión de errores de red

Dado que el sistema se comunica con las APIs a través de solicitudes HTTP, se consideró la posibilidad de interrupciones en la conexión a Internet o tiempo de espera prolongado en las respuestas. Para ello se establecieron tiempos límite *(timeouts)* en todas las solicitudes realizadas mediante la librería *requests* de Python. Esto permite que el sistema no quede bloqueado en caso de que una solicitud no reciba respuesta en el tiempo esperad y también se implementó reintentos automáticos con un número limitado de intentos. Si una solicitud fallaba debido a un error temporal, como una caída en la red, el sistema intentará realizar la solicitud nuevamente tras un breve intervalo.

2.2.3.2 Validación de respuestas

Para poder asegurar que las respuestas de las APIs sean válidas y cumplan con el formato esperado, se verificó que las respuestas HTTP incluyeran códigos de estado *(status codes)* exitosos, como 200 tal como lo observamos en figura 6. En caso contrario, el sistema generaba mensajes de error personalizados para facilitar la depuración, también se implementaron validaciones en el contenido de las respuestas JSON para garantizar que incluyeran los datos necesarios antes de procesarlos. Si faltaban campos críticos, el sistema informaba del problema al usuario.

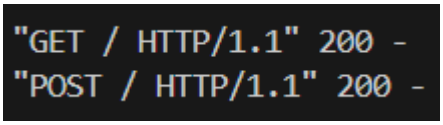


Figura 6. Validación de una respuesta correcta.

2.2.3.3 Errores de autenticación

La autenticación es un proceso fundamental para garantizar la seguridad y el acceso autorizado a las APIs de OpenAI y D-ID. En el prototipo desarrollado, la autenticación se gestiona

utilizando tokens Bearer, un método estándar para la autorización de solicitudes HTTP. Sin embargo, pueden surgir varios tipos de errores relacionados con este proceso: Si el token Bearer utilizado en la solicitud es incorrecto, ha expirado o no está incluido en el encabezado de autorización, las APIs pueden responder con códigos de error como *401 Unauthorized* un ejemplo de esta respuesta la podemos ver en la figura 7 o *403 Forbidden*.

Figura 7. Mensaje de respuesta cuando el token no está autorizado.

Para manejar estos errores, el sistema analiza el código de respuesta y genera mensajes claros para el usuario en la terminal, indicando la necesidad de actualizar o incluir el token adecuado, los encabezados HTTP mal configurados, como un formato incorrecto en el campo *Authorization*, pueden provocar errores de autenticación. Para evitar esto, se verifican los parámetros de configuración durante las pruebas y se registran los errores en caso de fallos inesperados y por último las APIs limitan el número de solicitudes fallidas en un período determinado como medida de protección contra ataques de fuerza bruta. En caso de alcanzar este límite, la API puede devolver un error temporal, como *429 Too Many Requests*. Para manejar esta situación, el sistema implementa tiempos de espera antes de reintentar la solicitud y notifica al usuario sobre el estado del sistema.

2.2.4 Configuración del Entorno

Para implementar el prototipo de generación automática de videos, se utilizó una instancia virtual en Google Cloud Platform (GCP). La creación de la máquina virtual (VM) fue un paso clave en la implementación del sistema, ya que permitió alojar el entorno de ejecución de la aplicación Flask y las dependencias necesarias para interactuar con las APIs de OpenAI y D-ID. A continuación, se describen los pasos seguidos para configurar correctamente el entorno y asegurar que la aplicación sea accesible desde una IP externa**.**

2.2.4.1 Creación de la instancia en GCP

Se creó una instancia de máquina virtual en GCP utilizando el sistema operativo Debian. Esta instancia fue configurada con los recursos adecuados para ejecutar el servidor Flask y las herramientas necesarias para la integración de las APIs. Durante la configuración de la VM, se habilitó una IP externa para acceder a la aplicación desde fuera de la red interna de GCP como podremos observar en la figura 8 esta toma la dirección externa de 34.59.87.216 en la cual se alojará el prototipo.

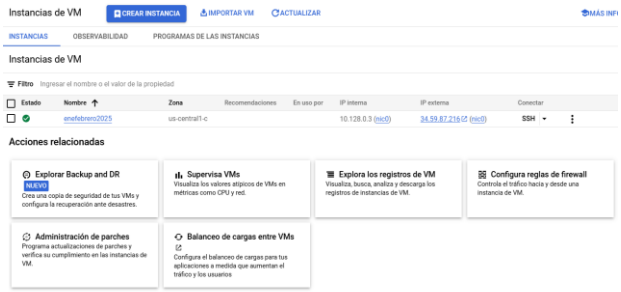


Figura 8. Detalles de la instancia creada.

2.2.4.2 Instalación del entorno de desarrollo

Tras acceder a la instancia de la VM mediante SSH, se realizó la instalación de los componentes básicos necesarios para ejecutar la aplicación Flask y las APIs requeridas.

Se ejecutó el comando *"sudo apt update && sudo apt upgrade -y"* para asegurarse de que los paquetes del sistema estuvieran actualizados como lo observaremos en la figura 9.

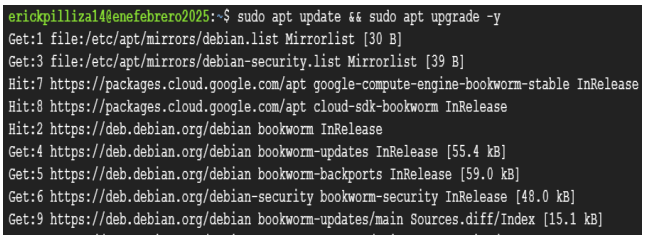


Figura 9. Actualización del sistema Debian.

Python y pip son esenciales para instalar y administrar las dependencias de la aplicación. Se instalaron por medio del comando *"sudo apt install python3 python3-pip -y"*, para verificar que todo este correcta nos deberá salir los mensajes que podremos observar en la figura 10.

```
erickpilliza14@enefebrero2025:~$ sudo apt install python3 python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.11.2-1+b1).
python3-pip is already the newest version (23.0.1+dfsg-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Figura 10. Instalación de Python y PIP.

Una vez instalados Python y pip, se procedió a instalar las librerías necesarias para la aplicación. Las principales dependencias fueron Flask y las bibliotecas para la interacción con las APIs. Se utilizó el siguiente comando para instalarlas *"sudo pip3 install flask openai==0.28 requests --break-system-packages"* y para verificar la correcta instalación se podrá observar un texto similar al de la figura 11.

```
erickpilliza14@enefebrero2025:~$ sudo pip3 install flask openai==0.28 requests --break-system-packages
Requirement already satisfied: flask in /usr/lib/python3/dist-packages (2.2.2)
Requirement already satisfied: openai==0.28 in /usr/local/lib/python3.11/dist-packages (0.28.0)
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (2.28.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from openai==0.28) (4.67.1)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from openai==0.28) (3.11.11)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->
openai==0.28) (2.4.4)
```

Figura 11. Instalación de librerías.

2.2.4.1 Configuración de Flask para la accesibilidad externa

En esta sección se detalla el proceso llevado a cabo para permitir que el prototipo basado en Flask sea accesible desde dispositivos externos, considerando tanto los ajustes realizados en el código como la configuración de la red.

El servidor del prototipo fue implementado para escuchar en todas las interfaces de red disponibles en la máquina virtual (0.0.0.0), garantizando que acepte solicitudes desde cualquier dirección IP externa. Se estableció el puerto 80 como predeterminado para la comunicación y se lo hizo por medio de las líneas de código que se pueden ver en la figura 12.

```
184  if __name__ == "__main__":
185      app.run(host="0.0.0.0", port=80, debug=True)
```

Figura 12. Código del método que inicia el servidor web embebido de Flask.

3. Resultados y Discusión

3.1 Resultados

3.1.1 Beneficios y Limitaciones de la Integración de la API de D-ID

Uno de los beneficios principales de integrar la API de D-ID en una aplicación web es su capacidad para transformar texto en videos de manera eficiente y realista. Esta funcionalidad permite la creación de contenido audiovisual personalizado, útil en sectores como educación, marketing y comunicación corporativa [6]. Además, la API ofrece una integración modular, lo que facilita su combinación con otros servicios como OpenAI GPT, ampliando el alcance del sistema [7].

Entre las limitaciones, se encuentra la dependencia de una conexión estable a internet para interactuar con la API, lo que puede generar problemas en entornos con conectividad limitada. Además, las restricciones en cuanto a cuotas de uso y capacidades de procesamiento impactan el rendimiento en aplicaciones que requieren un alto volumen de contenido generado [8].

3.1.2 Desafíos Técnicos y Soluciones Durante la Integración

Durante el proceso de integración, surgieron varios desafíos técnicos. Uno de ellos fue la configuración del entorno de desarrollo para asegurar la accesibilidad externa al servidor web Flask. Esto implicó configurar el servidor para escuchar en todas las interfaces de red (0.0.0.0) y en el puerto 80, permitiendo así que las

solicitudes externas fueran aceptadas. Se realizaron ajustes en el firewall del sistema y en las reglas de red de la máquina virtual de Google Cloud Platform (GCP), asegurando la accesibilidad adecuada sin comprometer la seguridad [9].

Otro desafío fue el manejo de autenticación mediante tokens Bearer, donde se implementó un sistema seguro para gestionar estos tokens y minimizar riesgos de exposición. Además, se optimizó la latencia en las solicitudes y la calidad de los videos generados mediante la biblioteca requests, complementada con técnicas de manejo de excepciones para garantizar la integridad del sistema [10].

3.1.3 Impacto en el Rendimiento General

El impacto de la integración de la API en el rendimiento general fue significativo. Aunque la API genera resultados rápidos, la latencia de las solicitudes incrementa el tiempo total de procesamiento. En promedio, la creación de un video de 30 segundos tomó entre 10 y 15 segundos, dependiendo de factores como el tráfico en los servidores y la complejidad del texto [11].

Para mitigar este impacto, se implementaron técnicas como asincronía en las solicitudes HTTP y almacenamiento en caché de respuestas para reducir tiempos de espera perceptibles. Estas estrategias mejoraron notablemente la experiencia del usuario [12].

3.2 Discusión

3.2.1 Beneficios y Oportunidades

La integración de la API de D-ID demuestra el potencial de las tecnologías basadas en IA para automatizar la creación de contenido audiovisual. Su combinación con OpenAI GPT ofrece una solución innovadora que podría transformar sectores como la educación y el marketing [13].

El diseño modular del sistema es una ventaja significativa, ya que permite escalar el prototipo para incorporar funcionalidades adicionales, como voces más naturales o efectos visuales avanzados [14].

3.2.2 Limitaciones y Áreas de Mejora

Aunque el prototipo cumple con su objetivo, las limitaciones detectadas evidencian la necesidad de optimización. La dependencia de APIs externas lo hace vulnerable a latencia, fallos del servicio o restricciones de uso. Una solución sería implementar un enfoque híbrido que combine generación local de contenido con integración de APIs externas, reduciendo la dependencia y mejorando la resiliencia del sistema [15].

4. Conclusiones

El desarrollo e integración de la API de D-ID en una aplicación web para la generación automática de contenido audiovisual ha demostrado ser una solución innovadora y eficiente, con aplicaciones significativas en sectores como la educación, el marketing y la comunicación corporativa. La capacidad de transformar texto en videos de manera realista y personalizada representa un avance importante en la automatización de la creación de contenido mediante tecnologías basadas en inteligencia artificial.

Entre los principales beneficios identificados, destaca la facilidad de integración modular que ofrece la API de D-ID, lo que permite su combinación con otros servicios como OpenAI GPT, maximizando las posibilidades del sistema. Además, la implementación de técnicas como la asincronía en las solicitudes HTTP y el almacenamiento en caché de respuestas mejoraron significativamente la experiencia del usuario, mitigando el impacto de la latencia inherente a las solicitudes a servidores externos.

Sin embargo, se identificaron limitaciones que afectan el desempeño y la resiliencia del

sistema, tales como la dependencia de una conexión estable a internet, las restricciones de uso de las APIs externas y la latencia en el procesamiento de solicitudes. Estas áreas representan oportunidades claras para futuras mejoras, incluyendo el diseño de sistemas híbridos que combinen la generación local de contenido con el uso de APIs externas para reducir vulnerabilidades y optimizar el rendimiento.

Por último, la experiencia adquirida durante este proyecto resalta la importancia de considerar tanto los beneficios como los desafíos técnicos en la integración de APIs en sistemas web. A medida que las tecnologías basadas en inteligencia artificial continúan evolucionando, se espera que soluciones como esta puedan ampliarse para abordar necesidades más complejas y diversificadas, consolidando su valor en la creación de contenido dinámico y adaptable.

Referencias

[1] S. J. Russell y P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Pearson, 2019.

[2] A. Radford et al., “Language models are few-shot learners,” Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020. https://dl.acm.org/doi/abs/10.5555/3495724.3495883

[3] C. Zhang y P. Yu, “Deep learning-based visual content generation: A review,” IEEE Transactions on Multimedia, vol. 21, no. 12, pp. 2952–2968, Dec. 2019.

https://dl.acm.org/doi/10.1145/3279952

[4] D. Kumar y R. Singh, “Containerization with Docker: An efficient approach to scalable systems,” IEEE Cloud Computing, vol. 6, no. 3, pp. 25–33, June 2019. https://ieeexplore.ieee.org/document/8710346

[5] G. Navas, J. Proaño-Orellana, R. Orizondo, and A. Terreros, "Exams as a Service: Synergies between ChatGPT, and Cloud Computing for Education", 4th International Conference on Smart Technology SmartTech-IC 2024. [Aceptada].

[6] T. E. Onyejelem y E. M. Aondover, "Digital Generative Multimedia Tool Theory (DGMTT): A Theoretical Postulation in the Era of Artificial Intelligence," Advances in Machine Learning and Artificial Intelligence, vol. 5, no. 2, pp. 01-09, 2024. https://www.researchgate.net/profile/Aondover-Eric-Msughter-2/publication/381847209_Digital_Generative_Multimedia_Tool_Theory_DGMTT_A_Theoretical_Postulation_in_the_Era_of_Artificial_Intelligence/links/6681b85d2aa57f3b8264188d/Digital-Generative-Multimedia-Tool-Theory-DGMTT-A-Theoretical-Postulation-in-the-Era-of-Artificial-Intelligence.pdf

[7] D. V. Kornienko, S. V. Mishina, S. V. Shcherbatykh, y M. O. Melnikov, "Principles of securing RESTful API web services developed with Python frameworks," en Journal of Physics: Conference Series, vol. 2094, no. 3, art. 032016, IOP Publishing, nov. 2021. https://iopscience.iop.org/article/10.1088/1742-6596/2094/3/032016

[8] R. Koçi, "A data-driven approach to prescribe web API evolution," 2024. https://upcommons.upc.edu/handle/2117/406401

[9] A. F. Nugraha, H. Kabetta, I. K. S. Buana, y R. B. Hadiprakoso, "Performance and security comparison of JSON Web Tokens (JWT) and Platform Agnostic Security Tokens (PASETO) on RESTful APIs," en 2023 IEEE International Conference on Cryptography, Informatics, and

Cybersecurity (ICoCICs), pp. 15-22, ago. 2023.https://ieeexplore.ieee.org/abstract/document/10277377

[10] I. Stupar y D. Huljenic, "Model-based cloud service deployment optimisation method for minimisation of application service operational cost," Journal of Cloud Computing, vol. 12, no. 1, art. 23, 2023.https://link.springer.com/article/10.1186/s13677-023-00389-8

[11] H. Hosseini, B. Xiao, A. Clark y R. Poovendran, "Attacking Automatic Video Analysis Algorithms: A Case Study of Google Cloud Video Intelligence API," en *Proceedings of the 2017 on Multimedia Privacy and Security (MPS '17)*, Association for Computing Machinery, New York, NY, EE. UU., pp. 21–32,2017. https://doi.org/10.1145/3137616.3137618

[12] M. Fan, X. Yang, T. Yu, Q. V. Liao, y J. Zhao, "Human-AI collaboration for UX evaluation: Effects of explanation and synchronization," Proceedings of the ACM on Human-Computer Interaction, vol. 6, no. CSCW1, pp. 1-32, 2022. https://dl.acm.org/doi/abs/10.1145/3512943

[13] Y. Dai, A. Liu, y C. P. Lim, "Reconceptualizing ChatGPT and generative AI as a student-driven innovation in higher education," Procedia CIRP, vol. 119, pp. 84-90, 2023. https://www.sciencedirect.com/science/article/pii/S2212827123004407

[14] K. Sullivan, W. G. Griswold, H. Rajan, Y. Song, Y. Cai, M. Shonle, y N. Tewari, "Modular aspect-oriented design with XPIs," ACM Transactions on Software Engineering and Methodology, vol. 20, no. 2, art. 5, 42 págs., ago. 2010. https://doi.org/10.1145/1824760.1824762.

[15] H. A. Nguyen, H. D. Phan, S. S. Khairunnesa, S. Nguyen, A. Yadavally, S. Wang, y T. Nguyen, "A hybrid approach for inference between behavioral exception API documentation and implementations, and its applications," en Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, pp. 1-13, oct. 2022. https://dl.acm.org/doi/abs/10.1145/3551349.3560434