



**BENEMÉRITA UNIVERSIDAD AUTÓNOMA
DE PUEBLA**

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

**“UNA APLICACIÓN PARA EVALUAR UN
CURSO DE ALGEBRA LINEAL UTILIZANDO
UNA RED NEURONAL GENERATIVA”**

TESIS

PARA OBTENER EL GRADO DE:

**LICENCIATURA EN INGENIERÍA EN CIENCIAS DE LA
COMPUTACIÓN**

PRESENTA:

LUIS DANIEL GONZALEZ JUSTO

DIRECTOR: RAFAEL LEMUZ LÓPEZ

ASESOR: RAFAEL LEMUZ LÓPEZ

Septiembre 2025

Agradecimientos

Quiero agradecer a Dios, mi constante guía y fortaleza en este proceso. Su luz ha guiado cada paso.

El amor y dedicación incondicional de mi mamá Consuelo y mamá Nicolasa han sido cruciales para esta tesis y mis logros. Gracias por mostrarme el valor del esfuerzo, la resiliencia y la fe. Su apoyo ha sido clave para mi avance.

Gracias, hermana María, por tu apoyo y aliento incondicional. Tu apoyo ha sido un equilibrio y motivación en este camino.

A María del Rosario, por ser el motor de cambio en mi vida.

Tu fuerza, armonía y nobleza inspiran lo mejor de mí, guiándome con tu humildad y siendo el pilar que sostiene mis sueños. En cada paso he sentido tu apoyo sereno, tu capacidad de escuchar sin juicio y tu entereza en los momentos de mayor incertidumbre. Agradezco tu claridad, compromiso y confianza en mi capacidad, pues han sido fundamentales para que este proyecto llegara a su término. Más que un resultado académico, este logro lleva tu huella y es también un reflejo del camino que hemos construido juntos. Ojalá el futuro nos siga ofreciendo motivos para caminar unidos hacia todo lo que aún nos espera.

A mi papá José Luis, quien influyó en mi formación. Gracias por su sacrificio, esfuerzo y por creer en mí.

Agradezco a todos los que contribuyeron al proyecto. Gracias a mis amigos, profesores y compañeros por su apoyo y consejos en momentos cruciales. Gracias a todos. Esta meta no se habría logrado sin su apoyo.

Contenido

Resumen	5
Introducción.....	6
1. Identificación del problema.....	8
1.1 Antecedentes	10
1.2 Formulación de objetivos	13
1.2.1 Objetivo General	14
1.2.2 Objetivos Específicos	14
1.3 Alcance y limitaciones.....	17
1.3.1 Alcance.....	17
1.3.2 Limitaciones	19
2. Marco Teórico.....	20
2.1 Introducción a las redes neuronales generativas.....	20
2.2 Electron como entorno de desarrollo	29
2.3 Álgebra lineal aplicada al aprendizaje automático	31
3. Desarrollo	33
3.1 Diseño de la aplicación	33
3.1.1 Arquitectura del sistema	33
3.1.2 Investigación y selección de modelos	35
3.1.3 Desarrollo del back-end:	35
3.1.4 Desarrollo del front-end:.....	37
3.1.5 Seguridad y optimización	40
3.2 Desarrollo de la aplicación.....	44
3.2.1 Introducción a Electron.....	44
3.2.2 Instalación y configuración inicial	45

3.2.3 Creando el punto de entrada (main.js).....	48
3.2.4 Script de precarga (preload.js).....	53
3.2.5 Creando la interfaz principal (index.html).....	55
3.2.6 Lógica del renderizado (renderer.js).....	58
3.2.7 Comunicación entre procesos.....	61
3.2.8 interfaz de chat.....	62
3.2.9 Interfaz de configuración.....	70
3.2.10 Funcionalidades avanzadas.....	72
3.3 Empaquetado y distribución.....	75
4. Resultados.....	78
5. Conclusiones.....	107
6. Bibliografía.....	109
7. Recursos y Herramientas.....	115

Resumen

La idea de este proyecto es crear una aplicación para computadora usando Electron que ayude a que la enseñanza del álgebra lineal en la universidad sea más fácil. Esta app está pensada para que los profesores tengan una herramienta que les permita crear y organizar ejercicios matemáticos, aprovechando las capacidades de redes neuronales generativas.

Con esta plataforma, los docentes pueden asignar tareas, armar exámenes y organizar el curso de forma más práctica. Los modelos que incluye están entrenados para generar problemas relacionados con temas importantes como matrices y cómo resolver sistemas de ecuaciones.

Una de las cosas que más busca esta herramienta es ayudar a los profesores a reducir el tiempo que gastan en tareas repetitivas y administrativas, para que puedan dedicar más atención a apoyar a cada estudiante.

Los primeros resultados muestran que la app puede resolver algunos ejercicios, lo que la hace una ayuda útil para las clases. No se trata de reemplazar a los profesores, sino de darles un soporte extra que haga la enseñanza mejor en un contexto educativo que cambia todo el tiempo.

Introducción

La inteligencia artificial ha cambiado muchísimo en los últimos años. Lo que empezó como una idea en los años cincuenta, ahora es algo que se usa en tareas muy distintas, muchas de las cuales ni se imaginaban en ese momento. Dentro de todo esto, está lo que se conoce como inteligencia artificial generativa. Son modelos que pueden crear contenido nuevo sin que nadie los programe paso a paso para eso (Bandi, Adapa y Kuchi, 2023). Algunos son bastante simples, pero otros, como las redes neuronales más avanzadas, pueden hacer cosas que se parecen a cómo pensamos los humanos. Hoy en día se usan en muchos campos, como medicina, transporte, finanzas, industria y, en este caso, en la educación.

Lo que hace especial a esta tecnología es que aprende con los datos que se le dan. A partir de eso, empieza a reconocer patrones y con el tiempo mejora en las tareas que realiza. Como cada vez vivimos en un mundo más digital, la inteligencia artificial se está usando más seguido. Y cuanto más se usa, más aprende. Esa es una de las razones por las que sigue evolucionando todo el tiempo.

La inteligencia artificial está empezando a meterse en el mundo de la educación. Se nota en cómo los estudiantes aprenden y también en la forma en que enseñan los profesores. Muchas tareas que antes tomaban mucho tiempo, como corregir exámenes o preparar materiales, ahora se pueden hacer con ayuda de estas herramientas. Eso libera un poco el trabajo y permite enfocarse más en lo creativo o en estar con los alumnos. Okaiyeto, Bai y Xiao (2023) comentan que todo esto trae ideas nuevas, pero también hay dudas sobre cómo usar esta tecnología correctamente. Por eso, muchas escuelas y universidades están empezando a pensar en cómo adaptarse.

Uno de los sistemas más conocidos es ChatGPT, que fue creado por OpenAI. Es un modelo de lenguaje que funciona con redes neuronales y puede entender preguntas y generar respuestas bastante buenas. Su uso en la educación puede ser útil, ya que responde dudas y también se adapta al tipo de usuario. Esto lo hace interesante para usar con alumnos y profesores. Aun así, todavía tiene problemas.

Por ejemplo, a veces da respuestas con errores o puede tener sesgos. También preocupa que se llegue a depender demasiado de este tipo de herramientas (Kasneci, 2023).

El álgebra lineal es una parte importante de las matemáticas, y por eso puede servir como buen ejemplo para ver cómo se puede aplicar la inteligencia artificial en la enseñanza. Esta materia es fundamental para entender cosas como ciencia de datos, programación o incluso temas de inteligencia artificial. El problema es que no hay muchas herramientas interactivas que ayuden a ver mejor los conceptos o a seguir el progreso de los alumnos, y eso hace que aprenderla sea más difícil. Chen (2023) menciona que el uso de inteligencia artificial en esta área podría ayudar mucho, sobre todo si se trabaja con situaciones reales y tecnología que motive más a los estudiantes.

Para enfrentar esto, la propuesta de esta tesis es crear una aplicación con Electron, que es una plataforma que permite hacer programas de escritorio usando tecnologías web. En la app, los usuarios van a poder elegir entre distintos modelos de lenguaje generativo según lo que necesiten. Estos modelos actuarían como asistentes virtuales para ayudar en el curso de álgebra lineal. Con este sistema, los profesores van a poder poner tareas, corregir exámenes y manejar otras cosas del curso de manera automática. Así, tendrían más tiempo para enfocarse en enseñar y acompañar a sus estudiantes.

Además de ser útil para resolver tareas, esta tecnología también busca mejorar cómo se relacionan los estudiantes con los profesores y con el contenido del curso. Cuando se usa un modelo generativo como si fuera un tutor, puede explicar temas difíciles, ayudar a resolver ejercicios y dar respuestas más claras. Esto no solo hace que aprender sea un poco más fácil, sino que también anima a los estudiantes a participar más. Por su parte, los docentes podrían tener más tiempo para hablar con los alumnos, notar errores comunes y ajustar la forma en que enseñan según lo que necesita cada uno.

Este proyecto no se queda solo en el álgebra lineal. También busca aportar ideas al debate general sobre cómo usar la inteligencia artificial en la educación. Hay varias cosas que hay que tener en cuenta, como si es accesible, si se puede aplicar bien a largo plazo, o si su uso es justo para todos. Si no se usa de forma responsable, puede terminar afectando el aprendizaje, sobre todo si se empieza a depender más de las respuestas rápidas que del diálogo entre estudiantes y profesores. Lim y sus colegas (2023) explican que este tipo de tecnología, aunque tiene muchas ventajas, también puede generar problemas, como la dependencia o los límites de lo que puede hacer en el aula.

1. Identificación del problema

Las tecnologías digitales, como la inteligencia artificial, han crecido rápido en los últimos años. Esto ha hecho que muchos sectores tengan que cambiar la forma en que funcionan, incluyendo la educación. Gracias a estos avances, ahora existen herramientas más interactivas y dinámicas que pueden ayudar a mejorar cómo se enseña y cómo se aprende. Aun así, no todo es tan simple. En materias como álgebra lineal, todavía hay varios problemas que hacen difícil aprovechar todo el potencial de estas tecnologías en el aula.

Según Katsamakos, Pavlov y Saklad (2024), la inteligencia artificial podría servir como apoyo para los docentes. Por ejemplo, puede ayudar a armar cursos, crear tareas nuevas, o incluso a enseñar de una manera que motive más a los estudiantes. También puede usarse para evaluar de formas distintas, que se enfoquen más en lo que realmente aprendió cada persona, no solo en repetir contenido.

El álgebra lineal es una rama importante de las matemáticas, aunque para muchos estudiantes puede ser un tema complejo. Esto sucede porque trabaja con ideas abstractas como espacios vectoriales, matrices o transformaciones, que no siempre se entienden fácil ni se aplican rápido. Para aprender bien, no basta solo con la teoría; es necesario practicar y ver ejemplos concretos.

Además, la enseñanza tradicional no suele adaptarse a cada estudiante. Generalmente, todos ven el mismo contenido y se evalúan de la misma forma, sin importar que algunos aprendan más rápido o de otra manera. Por eso, muchos se desconectan y solo se enfocan en lo que les resulta más simple, lo que provoca que no entiendan todo y pierdan interés o se sientan frustrados con las matemáticas.

Según Ávila Guama (2024), el álgebra lineal se percibe como una materia complicada y difícil de aplicar. Por eso, el autor sugiere que se usen herramientas tecnológicas que ayuden a que los estudiantes entiendan mejor y puedan relacionar los conceptos con casos prácticos.

La verdad es que enseñar pensando en cada alumno no es nada fácil. Hay muchos niveles diferentes en el aula y preparar algo que le sirva a todos lleva un montón de tiempo y recursos que a veces simplemente no se tienen. Además, la forma en que se evalúa muchas veces se queda en repetir cosas de memoria, como fórmulas o pasos, y eso no ayuda mucho. Lo que en serio importa en matemáticas es que los estudiantes puedan pensar bien, resolver cosas y aplicar lo aprendido a situaciones reales. Por eso estaría bueno cambiar el tipo de evaluaciones que se hacen. Las típicas pruebas ya no alcanzan. Según Barberá-Gregori y Suárez-Guerrero (2021), meter tecnología para evaluar no es solo usar computadoras o apps; también hay que ver qué impacto tiene eso en cómo se enseña, y eso complica todavía más el trabajo de los profes.

Usar redes neuronales generativas en clases de álgebra lineal puede cambiar bastante cómo aprendemos y cómo nos evalúan. Grassini (2023) menciona que la inteligencia artificial no solo está mejorando procesos administrativos, sino que también está cambiando la forma en que se enseña y se aprende en distintos niveles educativos.

Lo bueno de estas redes es que se adaptan al ritmo y estilo de cada persona. Pueden detectar en qué temas alguien tiene más dificultad y cambiar los ejercicios para ayudar con esas partes. También dan explicaciones claras, ofrecen prácticas hechas a medida y responden rápido, lo que facilita que uno pueda aprender de

manera más autónoma y efectiva. Para los profesores, es una herramienta que ayuda a evaluar mejor y a seguir el avance de cada estudiante.

Además, estas tecnologías hacen que el álgebra lineal sea más fácil de entender porque usan ejemplos visuales y situaciones concretas. Eso ayuda a que los temas que parecen muy abstractos sean más claros y que aprender sea más entretenido.

Pero, claro, no todo es tan simple. Hay que asegurarse de que todos los estudiantes puedan usar estas herramientas, incluso los que no tienen acceso a mucha tecnología o recursos económicos.

También es necesario establecer reglas claras para proteger los datos, mantener la transparencia en las evaluaciones automáticas y reconocer que la interacción humana sigue siendo esencial en la enseñanza. McDonald y su equipo (2024) mencionan que, aunque varias universidades ya están empezando a usar inteligencia artificial generativa, todavía falta entender bien cómo afectará a largo plazo y qué recursos serán necesarios para usarla de forma ética y sostenible.

Este estudio busca analizar cómo estas tecnologías pueden complementar la enseñanza tradicional del álgebra lineal, sin reemplazar a los profesores, sino más bien enriqueciéndola y adaptándola a los nuevos retos educativos que enfrentamos hoy.

1.1 Antecedentes

El álgebra lineal es una materia fundamental dentro de la formación en matemáticas y ciencias, siendo muy importante en carreras como ingeniería, economía y otras áreas aplicadas. Según Berman y Okubo (2015), tener un buen manejo de esta materia es crucial para el desarrollo profesional en estas disciplinas. Sin embargo, muchos estudiantes encuentran esta asignatura difícil, principalmente porque sus conceptos como las matrices y los espacios vectoriales son bastante abstractos. Por lo general, la enseñanza del álgebra lineal se enfoca mucho en la teoría, dejando poco espacio para que los estudiantes comprendan cómo aplicar realmente los conceptos. Esto hace que, en muchos casos, los alumnos simplemente memoricen los pasos o procedimientos, sin entender para

qué sirven en la práctica. Dorier (2003) ya había señalado que esta materia presenta un desafío especial precisamente por esa dificultad. Uno de los retos más grandes para los docentes es lograr que los estudiantes conecten las ideas teóricas con situaciones concretas. Muchas veces, los estudiantes se dedican a resolver ejercicios de forma mecánica o a repetir demostraciones, sin darse cuenta de que estos conocimientos tienen aplicaciones reales, por ejemplo, en el análisis de datos, la edición de imágenes o la creación de programas. Dorier y Sierpinska (2001) explican que, por la falta de un contexto claro y el alto nivel de abstracción, muchos estudiantes sienten que el álgebra lineal es inaccesible o demasiado complicada.

Desde el punto de vista de los docentes, también hay obstáculos importantes. Enseñar a grupos grandes dificulta personalizar las clases según las necesidades de cada estudiante. Lo más común es un método de enseñanza basado en la explicación teórica y el uso de fórmulas, que aunque cubre aspectos básicos, no garantiza que los alumnos entiendan profundamente o puedan aplicar lo aprendido en otros contextos. Larson y Zandieh (2013) indican que la enseñanza en grandes grupos limita la posibilidad de dar retroalimentación individual y de fomentar debates que ayuden a un aprendizaje más significativo, lo que resulta en un conocimiento superficial.

El método tradicional para enseñar álgebra lineal, con clases teóricas y ejercicios escritos, ha mostrado ser poco adaptable a las nuevas exigencias del ámbito educativo. En varios casos, corregir y evaluar consume tanto tiempo que los profesores no pueden diseñar actividades más creativas o innovadoras. Hillel (2000) advierte que cuando el foco está demasiado puesto en manipular símbolos y aplicar algoritmos sin relacionarlos con casos prácticos, el álgebra termina viéndose solo como un conjunto de reglas sin utilidad real.

Los cambios que trajo la pandemia han modificado tanto las herramientas disponibles como lo que esperan estudiantes y docentes. La enseñanza online y la necesidad de soluciones digitales dejaron claro que era urgente adaptar las formas de enseñar a esta nueva realidad. Durante y después de la pandemia,

quedó claro que el uso de plataformas digitales en la educación dejó de ser algo opcional. El contexto educativo vivió una transformación profunda a raíz de la pandemia, que obligó a replantear las formas en que se enseñaba y se evaluaba. Durante la pandemia, muchas universidades y colegios tuvieron que dejar de lado las evaluaciones tradicionales supervisadas en persona. En su lugar, adoptaron plataformas digitales que permitían mayor flexibilidad, no solo por razones educativas, sino también por cuestiones relacionadas con el trabajo y la ética, según Bond y Bedenlier (2023). Este cambio, aunque abrupto, se volvió imprescindible para continuar con el proceso formativo.

A partir de ahí, la tecnología dejó de ser un simple complemento para convertirse en una parte fundamental del aprendizaje. La inteligencia artificial, en particular, ya no es algo lejano o exclusivo de laboratorios; está presente en las aulas y tiene un impacto real en cómo se enseña y evalúa a los estudiantes. Mao, Chen y Liu (2024) señalan que los modelos generativos están alterando no solo cómo se evalúa en las aulas, sino también la forma en que profesores y estudiantes se relacionan con el conocimiento. Estas tecnologías infantiles ya no son meros recursos tecnológicos: están invitando a todos a replantearse el “qué” y el “cómo” de aprender.

La verdad, es fácil caer en la trampa. Para muchos estudiantes, la inteligencia artificial termina siendo ese camino corto para salir del paso con las tareas. El problema aparece cuando ese uso se vuelve costumbre y nadie se detiene a entender lo que realmente se está aprendiendo, o peor, lo que no se está aprendiendo. Por eso, el rol del docente sigue siendo fundamental: sin una guía que oriente cómo usar estas herramientas con sentido, el riesgo es que se vuelvan solo una forma más de pasar la materia sin entender nada. Y no es lo único: varias instituciones carecen del equipo, del soporte técnico o del personal preparado para integrar estas herramientas con calidad. En Bangladesh, por ejemplo, Naher et al. (2023) descubrieron que aunque hay entusiasmo respecto a la utilidad de la IA, también hay inquietud por su fiabilidad y por la idea de depender demasiado de ella.

Además, lo ético está ahí, fuerte y claro, aunque muchos prefieran ignorarlo. ¿Cómo se manejan los datos de cada estudiante? ¿Quién entiende exactamente el algoritmo que decide eso? ¿Qué es algo que no sea una pregunta primaria?

Algoritmos como ChatGPT son, como advierten Cotton, Cotton y Shipway (2024), un desafío para los métodos tradicionales de evaluación. Tenemos que reconsiderar en qué pensábamos que podíamos confiar para cuantificar el aprendizaje.

El resultado es que surgen hipótesis exactamente en el punto donde la IA podría desarrollarse hasta convertirse en realidad y presencia sin, sin embargo, hacer que el maestro se vuelva servil ante ella o eliminar la calidez humana. En este punto, se hace una sugerencia concreta: la inteligencia artificial no debería reemplazar a los maestros; más bien, se convierte en una herramienta que ahorra tiempo.

Muchas de las labores repetitivas que consumen horas, como calificar trabajos o registrar calificaciones, se automatizan y se crea una oportunidad para una interacción más rica centrada en la creatividad, el pensamiento crítico y la comprensión profunda.

Koh y Doroudi (2023) enfatizan que, si esta tecnología se utiliza de manera reflexiva e integrada en un plan educativo sólido, entonces realmente se convertirá en una fuerza sin precedentes en la transformación de la educación, y no solo en verbalizar el discurso.

1.2 Formulación de objetivos

En una nueva aplicación que inventaron, que hace que la enseñanza del álgebra lineal sea un recurso, su propósito es en un sistema simulado, este instrumento ayuda a los instructores con la asignación, corrección y calificación de pruebas y trabajos de clase. Su objetivo es proporcionar retroalimentación dinámica a los estudiantes a su propio ritmo.

Esta propuesta llevará la enseñanza al dato y el aprendizaje a la era de las redes neuronales generativas. Según Fauzi et al., ChatGPT puede aumentar la productividad de los estudiantes universitarios al proporcionar información, adquirir habilidades lingüísticas y promover la colaboración.

Una dificultad real que enfrentará esta herramienta es cómo enseñar álgebra lineal, que requiere su propio conjunto de medidas para comprender. Desplegado a gran escala, ofrece a los estudiantes un entorno personalizado que explica conceptos complejos como espacios vectoriales y matrices. Los instructores tendrían un instrumento que es definitivamente a prueba de auditorías, permitiendo así la reinversión del tiempo de gestión de la propiedad en la pedagogía.

Según Yu (2024), el hecho de que tecnologías como ChatGPT estén creciendo en número ilustra cómo el papel de los maestros en la educación no puede ser reemplazado.

En consecuencia, los objetivos de este trabajo se desglosan de la siguiente manera:

1.2.1 Objetivo General

Aplicación versátil basada en inteligencia artificial para la enseñanza del álgebra lineal

Nuestro objetivo es diseñar una aplicación accesible y sencilla para aprender álgebra lineal.

La intención aquí es aplicar redes neuronales generativas y notificación de datos a gran escala para, bajo las condiciones de la sociedad actual basada en el conocimiento, modernizar la enseñanza y la evaluación.

Su y Yang (2023) enfatizan este punto: ChatGPT es bueno para experiencias de aprendizaje personalizadas y ayudas didácticas.

Este intento está diseñado para proporcionar a las personas ayuda en línea útil e información general sobre álgebra lineal, superando la abstracción o naturaleza técnica que los estudiantes a menudo encuentran tan difícil.

Establece un contexto dinámico donde aquellos que estudian espacios vectoriales y matrices realmente lo entienden, y el efecto de esto puede medirse fácilmente en un mejor rendimiento académico.

Los maestros tendrán una herramienta automatizada para calificar el trabajo de los niños, liberándolos para participar en la interacción pedagógica y la investigación.

Sin embargo, Tlili et al. nos recuerdan encarecidamente que en cuanto a las implicaciones éticas de (popularizar) los modelos de lenguaje generativos en 2023 deben ser con moderación.

1.2.2 Objetivos Específicos

Se han definido objetivos específicos para desarrollar una innovadora aplicación de redes neuronales en álgebra lineal. Se centran en la enseñanza, el aprendizaje y la atención personalizada.

La red neuronal para generar y verificar contenido de álgebra lineal.

Crear un módulo para estudiar las respuestas de los estudiantes utilizando algoritmos de redes neuronales. Con APIs actualizadas, eliminar todos los errores en la calificación agregando la supervisión del profesor.

¿Por qué hizo eso? Hall (2024) examina cómo la inteligencia artificial generativa influye en la educación y sus consecuencias para la ética y la sociedad.

Se proporciona retroalimentación continua para corregir errores conceptuales.

La combinación de los aspectos conflictivos de cada definición nos permitirá ver cuándo las respuestas de los estudiantes son correctas.

Asignar un trabajo repulsivo a los estudiantes bajo supervisión. Déjalos a su aire, y una de las rondas habrá terminado. Detalles como que todo viene a aprenderse como un atajo habitual a menudo se omiten en el aprendizaje.

La detección de fallos está aumentando a medida que la lógica de aprendizaje crece. La red neuronal detectó un error general porque ya estaba presente en el patrón de escritura.

Añadir áreas problemáticas a esa clase de ejemplos en Forma Incomprensible ahora, dentro del rango de OCR imagológico. Cada estudiante recibe su propia instrucción adaptada de manera individual.

Pham, Nguyen, Hong, Ha y Hong Nguyen (2023) discuten cómo la IA puede personalizar el aprendizaje y promover la educación mediante la ingeniería robótica.

La interfaz de gestión de actividades de aprendizaje es fácil de usar para los maestros.

La adopción de nuevas tecnologías depende de la experiencia del usuario, pero tienen la capacidad de mejorar la eficiencia.

La interfaz necesita funcionar en todo, desde equipos de bajo rendimiento económico hasta estaciones de trabajo personales de alto rendimiento. Una aplicación compleja puede ocasionar errores y resistencia del usuario. Perder muchos recursos y el rendimiento del hardware mostrará las señales en su cara.

Es necesario crear una herramienta que sea fácil y eficiente de operar. Con las nuevas tecnologías disponibles, también podemos observar cómo disminuye la carga de trabajo del profesor y aumenta el rendimiento del estudiante.

Será posible juzgar a partir de mediciones reales si la aplicación es exitosa en ajustar el tiempo de evaluación y reducir la gestión redundante en el aula.

Las tareas se optimizan y se mejora el aporte educativo.

El efecto en la eficiencia que esta herramienta tendría al evaluar el rendimiento se da como su objetivo: Than et al., Comité de Ciencia y Tecnología del Pentágono de Austron Taiwan (2023) examina cómo la inteligencia artificial generativa puede realizar tareas complejas en la evaluación educativa tan bien o mejor que los humanos.

Este objetivo muestra que las redes neuronales generativas cambian la educación. El uso de la IA en la educación inclusiva y efectiva y la enseñanza de álgebra lineal sienta un precedente (Dai, Liu y Lim, 2023). Estos tres autores mantienen que ChatGPT tenderá a desarrollar la educación.

Este método ayuda a estudiantes y profesores a utilizar una herramienta que hace una mejor comprensión y crea cierto nivel de interés en una disciplina compleja. Peres et al. en 2023, se observó que la inteligencia artificial está cambiando cómo se realiza la investigación y la educación.

1.3 Alcance y limitaciones

El proyecto crea una aplicación para enseñar álgebra lineal. Dado que esta rama de las matemáticas es perfectamente adecuada para su implementación, es lo mismo que un patrón repetitivo que es bueno para el entrenamiento de inteligencia artificial. Esta situación es clásica para redes neuronales generativas.

El objetivo es la enseñanza y la protección de evaluación para los estudiantes. Las tecnologías modernas incluyen Electron y redes neuronales.

Chocarro, Cortiñas y Marcos-Matás (2021), al observar a los asistentes educativos desde la perspectiva del personal docente, examinan los factores sociales y el lenguaje de los bots. Sus hallazgos destacan la necesidad de adaptar tecnologías a las características del usuario para mejorar la pedagogía.

En virtud de lo anterior, se define el alcance del proyecto en los siguientes puntos clave:

1.3.1 Alcance

Desarrollo de funcionalidades centrales

La aplicación ofrece funciones clave para necesidades académicas.

Evaluación automatizada: el sistema usa redes neuronales para calificar respuestas en álgebra lineal. Esta herramienta se adapta a preguntas cerradas, de desarrollo y abiertas.

Retroalimentación personalizada: registra a cada estudiante y brinda recomendaciones según su desempeño. Facilita el monitoreo continuo para identificar mejoras y seguir el progreso académico.

Aplicación que crea ejercicios personalizados sobre transformaciones lineales, matrices y autovalores. Se apoya en fortalecer debilidades al identificar patrones de resolución.

Interfaz clara que facilita el acceso y distribución para docentes y usuarios. La plataforma mejora la gestión de cursos y el seguimiento de estudiantes.

Optimización del entorno académico

Se propone usar esta herramienta en álgebra lineal para aliviar la carga administrativa de los docentes. Al automatizar tareas, se libera tiempo para una retroalimentación más significativa.

Zhu, Liu y Lee (2020) hallaron que la retroalimentación automatizada mejora el aprendizaje, especialmente en la escritura argumentativa en evaluaciones formativas. La tecnología no reemplaza al docente, sino que le permite centrarse en apoyar el aprendizaje.

En su estudio de 2020, Zhu, Liu y Lee descubrieron que la retroalimentación automatizada mejora el aprendizaje, especialmente en la escritura argumentativa en evaluaciones formativas. La tecnología no reemplaza al maestro, pero le permite enfocarse en apoyar el aprendizaje.

Escalabilidad y adaptabilidad

Originalmente diseñado para álgebra lineal, la herramienta es adaptable a cálculo, probabilidad y física; a través de esta flexibilidad, puede ampliarse sin cambiar de plataforma.

Montenegro-Rueda et al. (2023) El impacto de ChatGPT fue investigado en educación por primera vez, resaltando la necesidad de soluciones escalables en diferentes contextos y cursos.

Innovación tecnológica

Desarrollada con Electron, esta aplicación es compatible con Linux y Windows, incluso en dispositivos de bajos recursos. El uso de modelos generativos automatiza tareas y ofrece una experiencia de usuario personalizada. En estos términos, esta elección tecnológica garantiza que los educadores serán plenamente responsables de sus selecciones cuando aparezcan nuevos consejos de práctica de IA, sin comprometer los niveles de rendimiento requeridos para la educación.

1.3.2 Limitaciones

Bandi, Adapa y Kuchi (2023) señalan que, a pesar de los avances en inteligencia artificial generativa, persisten limitaciones que dificultan su uso práctico. Los autores mencionan obstáculos como altos costos, necesidad de muchos datos, alto consumo de recursos, inconsistencia en resultados y falta de métricas claras.

Estas limitaciones se pueden agrupar en distintas categorías. En primer lugar, están los **factores técnicos**. La calidad de la aplicación depende directamente del rendimiento de los modelos de lenguaje y redes neuronales generativas, así como de los datos con los que se entrenan. Si un modelo deja de estar disponible o actualizado, parte de la funcionalidad podría verse afectada.

También hay **limitaciones económicas**. El uso de ciertos servicios o APIs tiene un costo mensual que puede ser un obstáculo, sobre todo en contextos con presupuestos limitados, lo que puede afectar tanto el acceso como el mantenimiento a largo plazo de la herramienta.

Por otro lado, están las **barreras educativas**. La implementación de tecnologías como esta puede generar resistencia, tanto en docentes como en estudiantes, especialmente si están acostumbrados a métodos más tradicionales. En algunos casos, esto podría traducirse en desinterés o abandono del uso de la herramienta.

Finalmente, hay **factores contextuales** a considerar. No todas las instituciones tienen la infraestructura tecnológica necesaria, y algunas pueden tener objetivos pedagógicos muy específicos que no se alinean con este tipo de solución. Por lo

tanto, su efectividad puede variar mucho según el entorno educativo en el que se intente implementar.

2. Marco Teórico

2.1 Introducción a las redes neuronales generativas

Las Redes Neuronales Generativas (GANs) son una arquitectura en inteligencia artificial que ha ganado protagonismo en los últimos años. Se han vuelto importantes en muchos campos, ya que estos modelos pueden producir tanto salidas visuales como de texto a partir de alguna entrada.

Sin embargo, las redes neuronales tradicionales se utilizan para clasificar datos y tomar decisiones basadas en entradas predefinidas en un programa. Las redes generativas aprenden la expresión de los números en los datos para que sean capaces de producir nuevas cantidades con transiciones suaves a partir de los datos antiguos, pero manteniendo cierto contexto.

En el análisis de Bond-Taylor et al. (2021), las GANs difieren fundamentalmente en su diseño. Esto les permite no solo aprender distribuciones de datos, sino también crear nuevas muestras que representan la realidad de manera más realista a partir de espacios complejos y variados.

Por lo tanto, la distinción subyacente entre las redes neuronales tradicionales y las redes generativas (otro término para redes generativas adversariales) radica en que las primeras solo pueden diferenciar y categorizar información ya vista, mientras que las segundas pueden generar nuevas instancias de datos.

Esta versatilidad en la producción de tipos de contenido como imágenes, texto o síntesis de voz expande el ámbito de uso en muchas disciplinas sin sacrificar la capacidad inherente de las redes neuronales de evaluar patrones por su relevancia coincidente; incluso, según Liu et al. (2020), las GANs son un método excelente para crear escenas fotográficas o de video. Las imágenes creadas de esta manera

son tanto realistas como de alta resolución: algo difícil o imposible de lograr con técnicas estándar.

Actualmente, existen varios tipos de redes neuronales generativas, entre las que se encuentran las Redes Generativas Adversariales (GANs), así como modelos autorregresivos que han contribuido significativamente al campo, como aquellos basados en la arquitectura Transformer.

Por todas esas vías diferentes, tenemos un objetivo: generar nuevos datos.

1. Generative Adversarial Networks (GANs):

Estas redes constan de dos componentes neurales que compiten entre sí: un generador y un discriminador. El generador produce datos sintéticos que imitan los reales, mientras que el discriminador evalúa su autenticidad. Mediante este proceso adversarial, ambas redes se entrenan mutuamente, mejorando progresivamente hasta que los datos generados resultan indistinguibles de los reales. Goodfellow et al. (2014) introdujeron las GANs con un enfoque radicalmente distinto, superando en realismo a modelos generativos previos como los autoencoders variacionales que se basaban en el modelado explícito de distribuciones.

2. Modelos autorregresivos (Transformers):

Los modelos autorregresivos, como GPT-3 y GPT-4 (Transformador Generativo Preentrenado), son especialmente efectivos en la generación de texto. Su característica principal es la capacidad de predecir el siguiente elemento en una secuencia en función de los pasos anteriores. Así, pueden dar sentido a largas secuencias de dígitos o completar una oración en contexto. Esto se logra modelando interdependencias. Por lo tanto, están bien adaptados para la tarea de explicar conceptos matemáticos o simular al profesor para ejercicios de álgebra lineal.

Vaswani y su equipo (2017) dieron una charla en The Neural Architecture Workshop sobre The Transformer, habilitando arquitecturas que eliminan la necesidad tanto de recurrencias como de convoluciones. En su lugar, utiliza mecanismos de atención para capturar dependencias globales. Al mismo tiempo, esta innovación no solo ha

significado un gran aumento en la velocidad de entrenamiento, sino que también ha establecido nuevos estándares como los de la traducción automática.

3. Wasserstein Generative Adversarial Networks (WGAN):

Los WGANs, una versión modificada de GANs, abordan el colapso de la moda y la inestabilidad del entrenamiento. La innovación crucial es que, en lugar de usar probabilidades logarítmicas, emplea la distancia de Wasserstein para medir las discrepancias entre distribuciones. Esta métrica proporciona una medida más constante para evaluar si el modelo ha convergido.

Introducidos por Arjovsky et al. en 2017, los WGANs mejoraron los GANs tradicionales en la forma de una señal de aprendizaje más estable, así como un contenido más coherente tanto en estructura como en realidad.

Las redes neuronales generativas realizan aprendizaje no supervisado o semisupervisado dependiendo de sus datos de entrenamiento. Encuentran patrones en los datos sin usar etiquetas. Según Wang (2001), el aprendizaje no supervisado explora patrones en información no etiquetada, permitiendo modelar la distribución de los datos y generar ejemplos típicos para nosotros utilizando redes generativas.

El entrenamiento de modelos generativos generalmente puede dividirse en varias etapas de gran importancia:

Entrenamiento: La red neuronal generativa toma grandes datos de álgebra lineal y matemáticas como su alimento. Al comparar la entrada con la salida esperada, esta red mejora su precisión. Karras et al. (2017) proponen un entrenamiento progresivo para GANs, con la resolución del generador y discriminador aumentando gradualmente. Este método no solo lleva a un proceso de entrenamiento más estable, sino también a una generación de imágenes y textos de mayor calidad y variedad.

Generación: Después de que el modelo ha sido entrenado, produce respuestas relevantes y coherentes; esto es útil para generar nuevo material de recursos. Es capaz de escribir ejercicios de álgebra lineal o explicar métodos de resolución.

Kang et al. StudioGAN es una biblioteca de 2022 equipada con herramientas y taxonomía para GANs para generar imágenes y texto, y simplifica enormemente el proceso de generación y la comparación de arquitecturas generativas.

Evaluación: Para verificar la adecuación de la respuesta generada, debe someterse a una comparación con instancias previas. Mediante contraste y desambiguación, el autor modifica su exactitud para no volverse obsoleto. Heusel et al. utilizan el TTUR para entrenar GANs, logrando una convergencia más estable. Incorporan la distancia de Inception de Fréchet (FID) para evaluar la calidad de las imágenes y textos generados.

En resumen, el proceso de entrenamiento de modelos generativos se puede dividir en varias fases, cada una de gran importancia.

Es evidente que las redes neuronales tienen un gran impacto en la educación, especialmente en álgebra lineal. Esta tecnología puede mejorar la calidad de la enseñanza de diversas maneras:

1. Generación de contenido educativo:

Los beneficios de estas redes incluyen su función de crear nuevo contenido. Permiten a cada estudiante crear ejercicios de matemáticas con diferentes niveles de dificultad y temáticas. Los ejercicios son innumerables: una variación del mismo problema en niveles infinitesimalmente más pequeños que proporcionan práctica, pero sin material inicial.

Anantrasirichai y Bull (2022) indican que la inteligencia artificial y el aprendizaje automático, sin reemplazar la creatividad humana, aumentan la eficiencia en diversas industrias.

2. Evaluación automática y retroalimentación personalizada:

Este tipo de sistema puede analizar las respuestas de los estudiantes una a una. El modelo aprende de los ejercicios y exámenes de los estudiantes, se adapta a sus necesidades y ofrece retroalimentación instantánea. Sin embargo, Nguyen Thanh et al. argumentan que este tipo de herramientas es difícil de usar en el análisis crítico; proponen rediseñar las evaluaciones subiendo en la jerarquía de competencias.

3. Creación de explicaciones detalladas:

El contenido dinámico puede proporcionar razonamientos lúcidos para cuestiones matemáticas. Luego, al clarificar preguntas específicas y conceptos abstractos a través de explicaciones que un estudiante pueda entender, el conocimiento sirve a los maestros en su necesidad de formas estructuradas: prever las ideas explicadas en su objetivo en diamantes.

Se ajusta a las necesidades del estudiante y ofrece a los maestros una manera más sencilla de explicar.

Sin embargo, tales herramientas, por esenciales que sean, no están exentas de problemas. Kiryakova y Angelova (2023) revelan que en la Universidad de Trakia los profesores de matemáticas consideran útil ahorrar tiempo y generar interés en los estudiantes, pero les preocupa el uso ético, especialmente en términos del origen de los textos y cuál es su efecto potencial sobre el aprendizaje que los estudiantes aportan o retiran de ellos.

Como las más cruciales en la discusión subsiguiente se enumeran las siguientes dificultades reconocidas:

1. **Complejidad de implementación:** La infraestructura necesaria para estos sistemas requiere tecnología avanzada. Dada la naturaleza evolutiva de las redes neuronales generativas, la implementación exige actualizaciones constantes que pueden no estar optimizadas al momento de la publicación de este trabajo.
2. **Requerimiento de datos:** Para un funcionamiento óptimo, estos modelos requieren volúmenes inmensos de datos. Esto es necesario para que el

entrenamiento reconozca correctamente los ejercicios matemáticos junto con sus soluciones detalladas. Modelos con un entrenamiento matemático insuficiente pueden generar soluciones incorrectas.

3. Autenticidad de las fuentes: todos los datos obtenidos de estas herramientas deben ser verificados de forma independiente. Aunque basados en la investigación, la veracidad de cualquier información o texto generado no puede garantizarse automáticamente. Por lo tanto, no deben tomarse como hechos antes de ser probados de lo contrario. Sin embargo, su capacidad para resolver problemas matemáticos no parece presentar ninguna contradicción significativa al respecto.

4. Aceptación por parte de profesores y estudiantes: algunos profesores podrían ser escépticos ante la incorporación de tales tecnologías en sus aulas. Existe el temor de que estos modelos puedan intervenir en las interacciones humanas y reducir la participación activa de los estudiantes.

Consideraciones sobre los Modelos de Lenguaje

Zhou (2025) menciona la creciente popularidad de los modelos de lenguaje en los últimos años, lo que ha impulsado el desarrollo de varias arquitecturas.

Se puede observar que los modelos más populares en 2020 están basados en transformadores, lo cual resalta una analogía con el entrenamiento específico y las pruebas. Debido a esto, cada modelo responde de manera diferente a la misma pregunta.

No solo genera de manera coherente y precisa grandes cantidades de texto al azar (excelente para el entrenamiento), sino que también produce un trabajo de mantenimiento mínimo que vale la pena para un método neural de este nivel generalmente implica. Estos modelos forman parte del campo de la inteligencia artificial y el procesamiento del lenguaje natural. Dado que cada uno cuenta con propiedades internas diferentes, depender de un único modelo limita las posibilidades del usuario, quien podría beneficiarse de la competencia existente en este campo.

Yenduri et al. (2023) describen que los modelos de lenguaje natural, en especial los de tipo transformer, se constituyen de varios elementos fundamentales que permiten su capacidad de comprensión y generación de lenguaje. Entre estos elementos destacan: el mecanismo de atención, que evalúa la importancia relativa de cada palabra en una secuencia; las capas codificadoras y decodificadoras, que permiten recibir y generar texto; el embedding posicional, que proporciona información sobre el orden de las palabras; y las fases de preentrenamiento y ajuste fino, que adaptan el modelo a tareas específicas.

El parámetro de temperatura en la generación de texto

El parámetro de temperatura es clave para controlar la aleatoriedad de la generación. A medida que los valores comienzan a variar de 0 a 1 en este parámetro, se puede influir en la forma en que las palabras se unen. A temperaturas más altas, las respuestas se vuelven más variadas e impredecibles. Esto puede afectar su coherencia y exactitud, provocando incluso consecuencias no intencionadas. Tal comportamiento no es apropiado para ecuaciones algebraicas.

Por otro lado, al reducir la temperatura de respuesta, obtenemos resultados más conservadores que son predecibles y congruentes con métodos ordenados. Sin embargo, esto puede ser menos flexible en estilo, y las soluciones no son amigables para los estudiantes que no han dominado suficientemente la estructura de las ecuaciones algebraicas.

Según Peeperkorn et al. En 2024, la temperatura aparentemente hace que un modelo de aprendizaje automático sea más avanzado en creatividad. Cuando la temperatura es baja, un modelo es más preciso y fiable para tareas de vital importancia, como los ejercicios. Pero con una alta temperatura, enfrenta mejor la gran cantidad de tipos diversos de preguntas que pueden surgir. Es mejor conseguir un equilibrio aquí.

Este parámetro no siempre se controla cuando el modelo está en uso, por lo que diferentes personas deberían diversificar su uso de acuerdo con sus tareas futuras si queremos realizar el valor completo de este enfoque.

Si bien la temperatura es un aspecto clave algo similar al costado en los neumáticos, hay otros componentes que influyen en la efectividad de los modelos en álgebra lineal. Por ejemplo, Lialin et al. (2023). Para maximizar el rendimiento del modelo, los parámetros de cada matriz deben ajustarse meticulosamente. Los escritores en este documento proponen técnicas de ajuste fino flexibles que cambian solo unos pocos elementos relevantes entre muchos miles para ordenar y predisponer tareas específicas. Se presta especial atención a los siguientes puntos:

El tamaño del modelo indica la cantidad de tokens que puede manejar en una consulta. Esta métrica, fijada por los desarrolladores, define la cantidad de texto que el modelo puede manejar. Útil para ejercicios cortos, pero poco efectiva para textos largos. Las empresas actualizan constantemente esta capacidad.

La longitud del contexto es otro factor principal. Con modelos subsecuentes, converge para generar resultados coherentes a partir de muy poca información, conociendo el contexto. La viabilidad difiere para cada estilo de modelo y cláusula.

Con el ajuste fino o fine-tuning, incluso especializar cómo funciona un producto. Esto puede incluir mapear el rodaje de un modelo preentrenado con un pequeño conjunto de datos para cumplir una función única. Es útil en áreas como el álgebra lineal, pero no tanto en otros lugares. Por lo tanto, los modelos de caza de amplio espectro seguramente seguirán apareciendo.

Según Ballout et al. en 2023, varios modelos de diferentes enfoques utilizando arquitectura Transformer han tenido éxito en tareas de procesamiento de lenguaje. Los más destacados son:

GPT es uno de los modelos más famosos y grandes. Basado en la arquitectura Transformer, comprende relaciones complicadas entre palabras en contexto. Su versatilidad significa que, aunque su entrenamiento fue extenso, un error en cualquier tarea podría corregirse sin dificultad: también permite ajustar la temperatura (la temperatura es un parámetro que afecta tanto el compromiso como la eficiencia) para ajustar finamente cómo se escriben las respuestas.

BERT es de la estructura Transformer creada por Google AI. A diferencia de GPT, que es generativo, se enfoca en comprender y clasificar textos. Como tal, BERT puede usarse para interpretar sentimientos y preguntas. Esto le da una posición superior para manejar información semántica en múltiples capas.

T5 es un modelo que trata todas las tareas de PNL como texto a texto, por ejemplo, convierte la clasificación en un problema de traducción. Similar a BERT, fue creado por Google Research y además introduce dos decodificadores para entender el contexto de manera más flexible.

La mayoría de los modelos de lenguaje avanzados son respaldados por la arquitectura Transformer. Esta estructura permite al modelo centrarse en diferentes partes del texto a través de la atención, dando énfasis al contexto al responder. A diferencia de los métodos secuenciales anteriores donde los Transformers procesan texto palabra por palabra, pueden trabajar en paralelo. Los fragmentos clave de un documento son detectados al mismo tiempo: esto hace que el entrenamiento sea más rápido y la comprensión más robusta.

Tokens, en ambos modelos, actúan como unidades de manejo de texto. Cuanto mayor sea la sofisticación del modelo, más tokens podrá soportar sin perder coherencia. Demasiados tokens a la vez podrían perder contexto y el modelo producirá soluciones de sentido común significativas parece imposible dada una cantidad excesiva de ellos. Por lo tanto, recomendamos que limite cuántos tokens se utilizan en una sola consulta.

El “token” es la cantidad silábica en la cual se puede descomponer una palabra u oración para determinar su peso atómico en cada sentencia, así pues a mayor complejidad del modelo mayor la cantidad de tokens que puede recibir y entregar.

2.2 Electron como entorno de desarrollo

Electron es un marco de código abierto que puede ayudarte a construir aplicaciones de escritorio multiplataforma. Utiliza HTML para su estructura, CSS para embellecer las cosas y JavaScript para las acciones.

Al unir las tecnologías web a las funciones nativas en aplicaciones, la principal ventaja de Electron es su flexibilidad. Además, es una buena plataforma para un programa que maximiza los modelos subyacentes con una interfaz amigable al usuario y tiene buen rendimiento.

Además, es adaptable y se puede usar en Windows, macOS o incluso en sistemas Linux.

Electron se utiliza para lograr una integración fluida de la interfaz gráfica con la lógica de negocio. Al mismo tiempo, se beneficia del desarrollo multiplataforma de aplicaciones interactivas.

Electron es la herramienta apropiada para el proyecto porque proporciona una interfaz disponible e intuitiva para educadores en herramientas reales (una tarjeta, impresora) disponibles en todas partes, así como robustez y alta personalización.

A continuación se presentan características y beneficios de Electron para este proyecto:

Multiplataforma

Una ventaja de Electron es que puede crear aplicaciones ejecutables para muchos tipos de sistemas operativos. Usando HTML, CSS y JavaScript, los desarrolladores solo tienen que escribir un conjunto de código que se llevará a varias plataformas diferentes con solo pequeñas modificaciones. Esta característica es importante para el proyecto porque permite la creación de una aplicación que puede ser abordada al mínimo por muchos tipos de usuario y al mismo tiempo tiene poco efecto en la eficiencia. Esto aplica a todos los sistemas de usuarios.

Integración de tecnologías web

Electron aprovecha las tecnologías web estándar para la construcción de aplicaciones de escritorio. Esto involucra a una gran comunidad de desarrolladores con muchas variaciones de herramientas. Hace posible crear interfaces modernas usando HTML5, CSS3 y JavaScript.

Acceso a funciones nativas del sistema

Además, Electron tiene acceso a funciones nativas del sistema como la gestión de archivos e interacción con el sistema de archivos, soporte de bases de datos y conexiones con otras aplicaciones de escritorio. Te permite crear una aplicación que gestione exámenes, ofrezca retroalimentación y material educativo personalizado para estudiantes y cursos.

Actualización y mantenimiento sencillos

La arquitectura centralizada de Electron facilita actualizaciones y mantenimiento. Las mejoras en la aplicación son fáciles sin reestructurar el entorno ni comprometer datos. Esencial para aplicaciones que necesitan actualizaciones frecuentes, como nuevos modelos o correcciones de errores, sin afectar la información de los docentes.

Soport para node.js

Electron usa Node.js como entorno de ejecución para JavaScript en servidor. Permite usar módulos y funciones backend en la aplicación de escritorio, aprovechando Node.js para almacenamiento local, algoritmos complejos y llamadas a APIs sin perder rendimiento.

No obstante, Electron presenta algunas limitaciones que deben considerarse. La principal es que sus aplicaciones suelen ser más pesadas en comparación con las nativas, ya que incorporan el motor de renderizado Chromium junto con la lógica de la aplicación, lo que incrementa el tamaño del archivo de instalación. Además, su eficiencia general suele ser menor en comparación con las aplicaciones completamente nativas.

Sin embargo, para el caso específico de una aplicación destinada a evaluar un curso de álgebra lineal, donde la mayor parte de la lógica es de negocio y generación de contenido, estas limitaciones de rendimiento pueden gestionarse de manera apropiada sin que se conviertan en un impedimento para el desarrollo.

2.3 Álgebra lineal aplicada al aprendizaje automático

Dado que el álgebra lineal constituye una rama muy extensa de las matemáticas, para los fines de esta investigación se abordarán sus conceptos fundamentales, así como su aplicación en diversas áreas y su impacto en la tecnología moderna.

El álgebra lineal es fundamental en el aprendizaje automático, ya que proporciona el marco matemático para manejar datos. Se logra mediante matrices que optimizan modelos de aprendizaje profundo. Las operaciones en el desarrollo de la inteligencia artificial dependen del álgebra lineal. Singh (2023) señala que manipula datos de gran dimensión por medio del álgebra lineal. Métodos como el PCA (Análisis de Componentes Principales) los cuales son esenciales para reducir la dimensionalidad, extraer características y mejorar el rendimiento de modelos.

Esta área surge de las operaciones sobre vectores y matrices. Los vectores son objetos matemáticos que tienen tanto magnitud como dirección, y las matrices son estructuras bidimensionales para organizar datos, resolver ecuaciones, etc.

Los sistemas de ecuaciones lineales también son fundamentales en el álgebra lineal. Surgen de matrices y se resuelven a través de métodos como la eliminación de Gauss y la descomposición LU.

La base del análisis de sistemas dinámicos reside en valores propios, vectores propios y determinantes igualmente. Aydin (2017) escribe que los babilonios alrededor del año 1800 a.C. usaron métodos matriciales para resolver sistemas lineales de 2 ecuaciones. Cardano creó un método en 1550, cada para resolverlos.

Lagrange utilizó matrices para optimizar funciones en el siglo XVIII, y Gauss introdujo el determinante.

El caso del álgebra lineal en el siglo XIX fue un período de gran generación. Cauchy formuló teoremas sobre determinantes, valores propios, diagonalización y matrices semejantes. Grassmann definió el vector por productos y operaciones. Estableció una base fundamental para esta disciplina.

El término 'matriz' fue inventado por J. Sylvester y decía que se refería a todo lo que es una disposición rectangular de números. Representó un cambio masivo para la teoría del álgebra lineal tal como la conocemos hoy.

A. Cayley realizó un estudio de las transformaciones lineales y su conexión con las operaciones matriciales.

El álgebra lineal es la base para los algoritmos de aprendizaje automático y las redes neuronales de aprendizaje profundo. La Descomposición SVD permite manipular datos multidimensionales y optimizar sistemas en modo.

Las transformaciones lineales también pueden manipularse en imágenes 3D gracias a esta disciplina. Técnicas como el PCA utilizan operaciones matriciales para disminuir la dimensionalidad de grandes cantidades de datos, incluso para identificar tendencias y patrones.

El álgebra lineal ha tenido un impacto reciente en el desarrollo tecnológico. Tanto la criptografía como la computación cuántica utilizan este conocimiento en herramientas seguras y eficientes. La inteligencia artificial necesita procesamiento de lenguaje natural y representa Matrix para modelar relaciones entre datos.

La manipulación lógica de grandes datos ha producido un nuevo conjunto de algoritmos en el álgebra lineal que son rápidos de calcular y baratos. Los problemas de optimización y logística utilizan modelos lineales para reducir costos.

El algoritmo QR fue introducido por Kublanovskaya (1961) para el álgebra lineal numérica. Por supuesto, esta forma de calcular valores propios y vectores propios se vuelve mucho más rápida en la práctica.

El álgebra lineal es indispensable para una multitud de aplicaciones actuales. Es importante que los jóvenes comprendan plenamente estos principios desde el principio como fundamento; sentir en lenguajes capaces.

Convertir este curso en una aplicación diseñada para sus objetivos más deseables podría inspirar a los estudiantes a sumergirse en el campo.

3. Desarrollo

El desarrollo de la aplicación se realizó en varias fases:

3.1 Diseño de la aplicación

Esta aplicación utiliza tecnologías modernas para automatizar la evaluación de exámenes y ejercicios. El objetivo es proporcionar retroalimentación personalizada para liberar tiempo a los docentes y fomentar un aprendizaje más interactivo.

Este capítulo describe la arquitectura del sistema y sus componentes principales para crear una aplicación eficiente e intuitiva.

Las redes neuronales generativas se personalizan gracias al entrenamiento específico de cada usuario. Esta tecnología puede generar respuestas correctas pero diferentes según el contexto y la consulta. Las redes neuronales generativas permiten una producción de contenido infinita y personalizada. La aparición de modelos de lenguaje ha generado respuestas más sistemáticas o creativas, ampliando las posibilidades para los usuarios.

Restringir al usuario a una única aplicación sería contraproducente para la personalización de estas herramientas. Este enfoque permite al usuario ajustar su experiencia al modelo que mejor se adapte a sus preferencias.

Elegir Electron como plataforma se ajusta a este enfoque gracias a su naturaleza multiplataforma y su uso de tecnologías web estándar como HTML, CSS y JavaScript. Estas tecnologías facilitan la integración de modelos de lenguaje y ofrecen una experiencia personalizable al usuario.

3.1.1 Arquitectura del sistema

Esta aplicación opera bajo una estructura modular, la cual integra los siguientes componentes esenciales: la interfaz de usuario, la interfaz de configuración y la interfaz widget.

La interfaz de usuario (UI), desarrollada en Electron, se divide en tres secciones principales. La primera es la interfaz de chat, donde el usuario selecciona el modelo de lenguaje a utilizar mediante un menú desplegable que contiene todos los modelos con comunicación implementada. Esta interfaz también incluye un cuadro de diálogo para interactuar con el modelo y un historial de conversaciones registradas. Es importante mencionar que una misma conversación puede ser utilizada por más de un modelo, aunque no al mismo tiempo. Adicionalmente, cuenta con una opción para que el usuario ingrese sus propias claves de API, ya que la utilización de los modelos de lenguaje depende enteramente de las credenciales proporcionadas por el usuario.

La segunda sección corresponde a la interfaz de configuración. En esta, el usuario puede almacenar sus configuraciones preferidas y ajustar parámetros como la personalidad del modelo de lenguaje; por ejemplo, puede configurarlo para que funcione como un asistente especializado en álgebra con fines didácticos. Asimismo, permite la importación de nuevos modelos en formatos como Live2D o Spine, que actuarán como widgets gráficos interactivos. Otras opciones disponibles incluyen: modificar el número máximo de tokens permitidos por consulta para optimizar el uso de recursos; activar un modo silencioso que inhibe las notificaciones emergentes del widget; gestionar las claves de API de los modelos; y especificar una ruta local hacia una carpeta de música para permitir la reproducción de audio en segundo plano.

Finalmente, la interfaz widget carga gráficos animados para renderizar un personaje virtual mediante tecnologías como Live2D, Spine, Three.js o Pixi.js en un canvas. Su función principal es integrar tres botones interactivos que permiten acceder a las otras interfaces: uno abre la ventana de chat, otro la ventana de configuración, y un tercero controla la reproducción de música en segundo plano sin abrir una ventana

adicional. Este widget también es capaz de obtener información del sistema en tiempo real.

3.1.2 Investigación y selección de modelos

Se estudiaron diversos modelos de lenguaje generativo basados en redes neuronales generativas, seleccionando aquellos que han presentado avances significativos en los últimos años, junto con características como accesibilidad sencilla y actualización constante. Los modelos considerados son los siguientes:

GPT de OpenAI: se destaca como uno de los líderes en investigación de tecnologías de inteligencia artificial, con contribuciones notables que facilitan el desarrollo de asistentes virtuales.

R+ de Cohere: aunque menos difundido, ofrece capacidades intelectuales interesantes. Cohere proporciona una versión gratuita para pruebas que, si bien no alcanza el rendimiento de su modelo empresarial, cumple funciones de asistencia, aunque con cierto sacrificio en precisión.

Gemini de Google DeepMind: Detecta contenido generado por IA y transforma tu texto para que suene más humano con nuestro Detector de Contenido IA. Simplemente pega tu texto y obtén resultados precisos y naturales en cuestión de segundos.

DeepSeek v3 de DeepSeek: ha adquirido relevancia reciente debido a su versatilidad e innovación en el área de modelos de lenguaje. Se presenta como una alternativa robusta dirigida a usuarios promedio, con alta adaptabilidad incluso en dispositivos de capacidades moderadas, por lo que resulta adecuado para proyectos de pequeña escala.

3.1.3 Desarrollo del back-end:

Para el desarrollo del back-end se adoptó un modelo cliente-servidor. Este modelo permite emplear un servidor centralizado para gestionar todas las solicitudes del cliente. La comunicación con la interfaz de usuario y los modelos de inteligencia

artificial se lleva a cabo a través de arquitecturas de API REST, lo que llevó al diseño de un sistema con varias capas.

La **capa de presentación**, o front-end, que se desarrolló en Electron, actúa como el cliente. Su función principal es manejar la interacción entre los eventos de la interfaz HTML y el back-end. Esta capa se encarga de manipular la interfaz gráfica destinada a los docentes; además, se ocupa de enviar las solicitudes HTTP a la API y de recibir las respuestas en formato JSON para mostrar los datos en la aplicación. Asimismo, establece comunicación con la capa siguiente mediante peticiones AJAX. La configuración del sistema, obtenida mediante archivos JSON, permite personalizar el comportamiento del asistente y utilizar datos meteorológicos para generar un mensaje inicial.

La **capa de controladores** gestiona la interacción entre el cliente (Electron) y la lógica de negocio. En esta capa se reciben las peticiones HTTP desde el front-end y se valida la integridad de los datos. Una vez validados, los datos se envían a los servicios correspondientes; posteriormente, las respuestas recibidas se retornan al cliente. Esta capa importa funciones para interactuar con servicios de modelos de lenguaje largo, traducción, síntesis de audio y un módulo de obtención de datos meteorológicos. Para ello, emplea dependencias como markdown-it para renderizar mensajes en Markdown, además de los módulos fs y path de Node.js para la lectura de archivos de configuración.

La **capa de servicios** contiene la lógica de negocio y las conexiones con los modelos de lenguaje. Aquí se aplican las reglas de negocio de la aplicación y se procesa la lógica operativa. Esta capa evalúa las respuestas generadas por el modelo de lenguaje, gestiona los costos asociados a las consultas y devuelve la respuesta a la capa de controladores en formato JSON.

La **capa de persistencia** administra la base de datos local y el almacenamiento de modelos, en caso de no utilizarse una API externa. Su función es almacenar y recuperar datos. Las sesiones se guardan en objetos de tipo "chat", donde cada uno se identifica mediante un ID único. La sesión activa se controla mediante un método específico para prevenir problemas de concurrencia.

El flujo de una solicitud en el back-end sigue una secuencia específica: el usuario realiza una consulta académica a través de la interfaz en la capa de presentación. La solicitud es recibida por la capa de controladores, donde se valida y se dirige al servicio pertinente. En la capa de servicios, la solicitud se procesa según la lógica de negocio y se consulta al modelo de lenguaje seleccionado. La respuesta generada pasa por la capa de persistencia para su almacenamiento local antes de ser presentada al usuario.

Las tecnologías empleadas en el desarrollo del back-end son las siguientes:

- **Electron:** Framework para la creación de la aplicación de escritorio utilizando tecnologías web.
- **Node.js:** Entorno de ejecución para JavaScript en el lado del servidor.
- **HTML:** Lenguaje para estructurar los elementos de las interfaces.
- **CSS:** Lenguaje para definir los estilos de las interfaces.
- **JavaScript:** Lenguaje para controlar la lógica y los eventos de las interfaces.

3.1.4 Desarrollo del front-end:

El front-end de la aplicación facilita la interacción del usuario con el sistema. Esta interfaz fue hecha con Electron, un framework que permite crear aplicaciones de escritorio nativas con tecnologías web.

El front-end se basa en componentes reutilizables de HTML, CSS y JavaScript. Esta estructura tiene tres capas:

Capa de UI: Intuitiva y con componentes visuales interactivos. Permite crear y eliminar sesiones de chat con confirmación, generadas con parámetros como personalidad y clima. La interfaz permite una navegación dinámica entre sesiones. Cada sesión tiene su propio botón. Los elementos clave son: un contenedor para el chat con mensajes y avatares del usuario y asistente; una lista de sesiones con opción de eliminación; y un botón para regenerar la última respuesta del asistente.

Capa de Gestión de Estado: Aquí se gestiona el almacenamiento local de datos del cliente.

Capa de Comunicación con el Back-end: Envía y recibe mensajes de modelos de lenguaje mediante solicitudes API con AJAX. Gestiona eventos entre la interfaz de usuario y el proceso principal de Electron, controla el DOM y monitoriza el estado del sistema para generar mensajes contextuales de un asistente virtual.

Todos estos elementos se integran en una interfaz intuitiva que carga una pantalla tipo "widget". Este widget, basado en tecnologías como Live2D o Spine, supera las limitaciones de las imágenes estáticas al reconstruir un esqueleto y texturas (ya sean de diseño propio o de origen externo) para utilizarlo como interfaz principal. Se presenta en un canvas de tamaño ajustable y desplazable dentro del monitor. Este componente cuenta con articulaciones para realizar animaciones, ya sean predefinidas o activadas por la interacción con el cursor, y puede mostrar mensajes espontáneos e informativos sobre el sistema. Al ser presionado, despliega tres botones de acceso rápido con las siguientes funcionalidades:

El **primer botón** abre una ventana de diálogo para consultas. Incluye diversos elementos interactivos: un botón para crear nuevas sesiones de chat; una lista de botones con las sesiones actuales, cada una con un contexto independiente; y una cabecera con un menú desplegable para seleccionar el modelo de lenguaje, un botón para ocultar o mostrar la lista de sesiones, y un campo de texto para insertar la clave de acceso API junto a un botón para guardarla en el archivo de configuración. El lienzo de mensajes es un elemento de chat que se actualiza en tiempo real y se adapta al contenido automáticamente. En la parte inferior, el usuario dispone de un área de texto para enviar consultas, un botón de envío (o la tecla Enter), y un botón que convierte el contenido de archivos PDF a texto para enviarlo como consulta, necesario ya que no todos los modelos interpretan documentos PDF de forma nativa. Finalmente, incluye un botón para regenerar la última respuesta del modelo.

El **segundo botón** da acceso a una pantalla de configuración. Permite añadir un nuevo modelo visual (L2D o Spine) y guardar la configuración del usuario en una

lista de perfiles. Un área de texto permite redefinir la "personalidad" del asistente, un conjunto de parámetros que condicionan todas sus respuestas para lograr un enfoque personalizado. También incluye un campo para establecer el número máximo de tokens permitidos, función útil para controlar los costos asociados al uso de la API. Un checkbox activa o desactiva el "modo silencio", que suprime las alertas emergentes para una operación discreta. Asimismo, se incluyen apartados dedicados para insertar y guardar claves de acceso de forma individual en el archivo de configuración. Por último, un campo de texto permite especificar la ruta completa a una carpeta local con archivos de audio en formato MP3 para su reproducción.

El **tercer botón** controla un reproductor de música interno. Al presionarlo, se reproduce en segundo plano la música ubicada en la ruta preconfigurada; al volver a presionarlo, la reproducción se detiene.

Respecto al flujo de mensajes, estos se formatean antes de su envío: las consultas del usuario se transforman en "instrucciones" adaptadas al formato esperado por los modelos de lenguaje. El contenido se renderiza en Markdown para garantizar su legibilidad. Se realiza un cálculo estimado de tokens basado en la longitud del texto, definiendo límites según el modelo seleccionado y eliminando mensajes antiguos si se supera dicho límite para evitar errores. Las respuestas se gestionan inmediatamente, actualizando la interfaz en tiempo real, e incluye la funcionalidad de regenerar respuestas, que elimina la última interacción del asistente y consulta nuevamente al modelo.

Las tecnologías empleadas para la elaboración del front-end son las siguientes:

- **HTML:** Para el diseño de los elementos estáticos (botones, áreas de texto) y dinámicos (lienzo de mensajes, sesiones) de la interfaz.
- **CSS:** Para el diseño y estilización de los componentes, otorgando un formato comprensible y agradable para el usuario.
- **JavaScript:** Para implementar la lógica de la interfaz, manejar los eventos de interacción del usuario y realizar las solicitudes a la capa de servicios (back-end).

3.1.5 Seguridad y optimización

El manejo de las claves de API se realiza localmente y su validación ocurre durante el inicio de la aplicación. Por consiguiente, en ningún momento se comparten estos datos sensibles con el servidor ni se incluyen en respuestas a servicios de terceros.

La mayoría de las funciones incluyen bloques try/catch para atrapar posibles fallos en las llamadas a las APIs. Esta estrategia cuenta con un mecanismo de contingencia (fallback) que, si hay un error en un endpoint principal, redirige la solicitud a la API de OpenAI.

Los mensajes se truncaron de manera estratégica para mantener el límite de tokens acorde a los parámetros elegidos por el usuario. Asimismo, el contexto de la conversación se actualiza periódicamente en un intervalo de diez mensajes; este enfoque garantiza que la información más reciente permanezca relevante, previniendo ambigüedades en las respuestas.

La gestión de los mensajes se activa cuando presionas el botón de envío o la tecla "Enter". Si no hay un identificador (ID) para la conversación, este evento inicia una nueva sesión y, a través de eventos intercalados, agrega tanto el mensaje del usuario como el del asistente al chat. Hay una función central que revisa los mensajes para evitar duplicados en la misma conversación, creando así un entorno de consulta y respuesta. Los prompts se envían a diferentes destinos a través de canales específicos para el usuario y el asistente, lo que ayuda a prevenir la interceptación de mensajes.

Si bien la monitorización del sistema es una función valiosa para un asistente, las limitaciones de acceso a procesos en Windows, sistema de interés para este proyecto, imposibilitaron esta tarea. Las dificultades para acceder a user32.dll o a cmdlets de PowerShell limitaron la capacidad de obtener nombres de ventanas y proporcionar al asistente datos para mensajes personalizados. Esta limitación no se presenta en entornos Linux, donde herramientas como xdotool permiten interactuar con el sistema de ventanas X11; no obstante, esta alternativa solo está disponible en su versión para Linux.

Cabe destacar que la información del sistema a la que el asistente accede no es de carácter confidencial o personal. Se limita a datos generales como el sistema operativo, el uso de RAM y de CPU. En un contexto de uso, y solo en Linux mediante herramientas como xdotool y xprop (previa instalación por parte del usuario), podría acceder al nombre de las aplicaciones o ventanas abiertas. Adicionalmente, obtiene la hora, la fecha local y una ubicación aproximada exclusivamente para recuperar datos meteorológicos. Es imperativo mencionar que toda esta información se utiliza únicamente para mostrar mensajes de asistencia; no se recopila, almacena o transmite a terceros.

Desafíos y Soluciones

El desarrollo de esta aplicación enfrentó varios desafíos técnicos y de diseño:

- **Precisión en la evaluación automática:** Dado que se trata de un asistente de álgebra lineal, debe enviar y recibir expresiones algebraicas de forma correcta. Para abordar este problema, se planeó implementar técnicas de procesamiento matemático avanzado, como la integración de herramientas LaTeX para la interpretación de expresiones. Sin embargo, esta integración requiere una manipulación precisa del contenido de los mensajes para transformarlo en expresiones LaTeX, lo que incluye discernir entre contenido algebraico y no algebraico. La exhaustiva manipulación de todos los casos posibles constituye un desafío significativo.
- **Adaptabilidad del modelo generativo:** La ejecución local de modelos generativos de lenguaje depende del repositorio oficial del modelo. Para DeepSeek-V3, con 671 mil millones de parámetros (671B), los requisitos hardware son considerablemente altos. Se recomienda una GPU NVIDIA dedicada con un mínimo de 8 GB de VRAM para versiones destiladas de 7B, un procesador multinúcleo de al menos 4 núcleos, 8 GB de RAM para modelos de 1.5B a 7B, y 16 GB de RAM para modelos de 14B y 32B. El almacenamiento requerido puede alcanzar los 715 GB. Dada la elevada demanda de recursos, no se recomienda la implementación local para este proyecto.

- **Costos de API:** El uso de APIs conlleva un costo determinado por el volumen de tokens empleados, el cual puede variar. Si bien su implementación es más sencilla y los requisitos técnicos son más asequibles, el costo asociado recae completamente en el usuario, quien debe proporcionar su propia clave de acceso. Aunque existen APIs con niveles gratuitos, su disponibilidad depende enteramente de la decisión del usuario, por lo que no constituyen una alternativa garantizada.
- **Compatibilidad en asistencia completa:** La compatibilidad multiplataforma se ve comprometida. Aunque distintos sistemas operativos comparten similitudes conceptuales en la gestión de procesos, difieren en su implementación. Para Node.js, las funciones que permiten acceder a ciertos datos de los procesos, como sus nombres, no operan de manera idéntica en todos los sistemas debido a las librerías disponibles. Si bien la aplicación dispone de todas sus características personalizadas en Linux (su entorno nativo), su versión para Windows, en la que se documenta este proyecto, carece de la funcionalidad para enviar mensajes de asistencia basados en las ventanas del sistema abiertas.

Impacto y Futuro de la Aplicación

Esta aplicación integra tecnologías contemporáneas, lo que le confiere un potencial tangible para transformar la enseñanza del álgebra lineal al ofrecer procesos automatizados de evaluación basados en inteligencia artificial que facilitan el aprendizaje. En el futuro, el rápido avance de los modelos de lenguaje va a permitir mejorar su aplicación, aumentando tanto la precisión como las capacidades, y ampliando su alcance a otras áreas de las matemáticas o de la enseñanza en general, a través de tutorías inteligentes adaptadas a nuevos contenidos.

El uso de redes neuronales generativas en la educación es un avance muy prometedor. Este proyecto pretende constituir un paso importante hacia la creación de entornos de aprendizaje más dinámicos e interactivos para estudiantes y docentes.

Evaluación Automática y Retroalimentación

El diseño implementado satisface la necesidad de ofrecer asistencia interactiva en una aplicación con alta capacidad de personalización, que optimiza los recursos para proveer una solución moderna y con un desempeño óptimo. Su capacidad para evaluar el contenido de documentos PDF y ofrecer retroalimentación instantánea al docente es una de sus principales virtudes.

Procesamiento de Respuestas:

- Dada la naturaleza de los modelos de lenguaje, el sistema posee una alta capacidad para la identificación y corrección automática de errores, así como para el análisis de sus procedimientos.
- Las respuestas de los estudiantes se comparan con soluciones generadas por modelos de lenguaje especializados en álgebra lineal, lo que permite una tutoría acertada en el área.

Generación de Retroalimentación:

- La tecnología empleada es capaz de generar explicaciones detalladas y personalizadas para cada ejercicio, ofreciendo una retroalimentación clara y concisa sobre aciertos y errores.
- Alimentando al modelo con los datos de un estudiante, se puede generar un historial que identifique patrones de aprendizaje y ofrezca asesoría sobre su desempeño.

Seguridad y Privacidad

- Esta aplicación maneja datos sensibles, como claves de acceso, información de estudiantes y el contenido de las conversaciones. La seguridad en el manejo de esta información es fundamental. Las solicitudes AJAX aseguran la integridad de los datos a través de cifrado, y las claves se almacenan únicamente de forma interna. La validación de una clave se lleva a cabo mediante una solicitud POST, lo que garantiza su seguridad durante la transmisión.

- Es importante señalar que, aunque el cifrado de las conversaciones las protege de terceros, no se puede descartar su posible uso como datos de entrenamiento por parte de los proveedores de APIs. El manejo de la información como fuente de entrenamiento depende completamente de las políticas de uso de cada API

Por último, es crucial considerar los límites éticos en el uso de estas tecnologías para resolver problemas. Aunque la automatización y la inteligencia artificial facilitan procedimientos básicos, el verdadero objetivo de esta aplicación es ofrecer asistencia, no reemplazar la interacción personal entre el docente y el alumno.

3.2 Desarrollo de la aplicación

3.2.1 Introducción a Electron

Electron es un framework de código abierto desarrollado por GitHub que permite crear aplicaciones de escritorio multiplataforma mediante tecnologías web como HTML, CSS y JavaScript. Gracias a su integración con el motor de renderizado Chromium y el entorno de ejecución Node.js, proporciona una estructura completa que simplifica el desarrollo de aplicaciones con interfaces web y acceso nativo al sistema operativo. Su popularidad ha llevado a la creación de herramientas muy reconocidas, como Visual Studio Code, Slack, Discord, Atom y Microsoft Teams.

Las ventajas de Electron lo hacen una opción muy viable para el desarrollo de software. Entre sus principales características destacan las siguientes:

Multiplataforma: a partir de una única base de código, es posible compilar aplicaciones para Windows, macOS y Linux sin complicaciones significativas.

Tecnologías familiares: utiliza HTML, CSS y JavaScript, lenguajes que son ampliamente conocidos en el mundo del desarrollo web y que cuentan con una gran cantidad de documentación y apoyo de la comunidad.

Acceso al sistema: gracias a Node.js, las aplicaciones pueden comunicarse directamente con el sistema de archivos y los procesos del equipo, lo que permite combinar una interfaz moderna con capacidades nativas.

Comunidad activa: hay un vasto ecosistema de paquetes y herramientas, respaldado por una comunidad de usuarios que es activa y colaborativa.

Actualizaciones automáticas: cuenta con soporte integrado para implementar actualizaciones automáticas, lo que hace que el mantenimiento de las aplicaciones sea mucho más sencillo.

Para iniciar el desarrollo de una aplicación en Electron, es necesario cumplir con los siguientes requisitos previos:

1. Tener instalado Node.js en su versión 12.x o superior.
2. Contar con npm, que se incluye por defecto junto con Node.js.
3. Utilizar un editor de código, como VS Code, para facilitar la programación.
4. Poseer conocimientos básicos en HTML, CSS, JavaScript y Node.js.

3.2.2 Instalación y configuración inicial

Inicialización del proyecto

El primer paso consiste en la creación de una carpeta que contendrá el proyecto principal. Una vez instalado Node.js en el sistema, debe abrirse una ventana de terminal y ejecutarse los siguientes comandos: utilizar mkdir para crear la carpeta

del proyecto y cd para ubicarse dentro de ella. Posteriormente, se ejecuta el comando npm init -y para generar un archivo package.json básico.

A continuación, se debe instalar Electron como dependencia de desarrollo, junto con el resto de las dependencias planificadas para el proyecto. A continuación se

```
Node.js command prompt

D:\proyecto>mkdir asistente

D:\proyecto>cd asistente

D:\proyecto\asistente>npm init -y
Wrote to D:\proyecto\asistente\package.json:

{
  "name": "asistente",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

D:\proyecto\asistente>
```

Figura 1. Ejemplo de cómo iniciar un proyecto

detallan las dependencias utilizadas en esta aplicación.

```
D:\proyecto\asistente>npm install --save-dev electron
npm warn deprecated boolean@3.2.0: Package no longer supported.
e info.

added 70 packages, and audited 71 packages in 23s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Figura 2. Ejemplo de cómo instalar dependencias

Dependencias de Desarrollo (devDependencies)

Estas se emplean exclusivamente durante las fases de desarrollo y empaquetado de la aplicación:

- **electron (^26.4.0)**: Constituye el núcleo del framework, permitiendo la creación de aplicaciones de escritorio. Proporciona la base necesaria para ejecutar la aplicación en modo desarrollo mediante el comando `npm start`.
- **electron-builder (^24.6.4)**: Herramienta utilizada para generar instaladores y ejecutables mediante los comandos `npm run pack` y `npm run dist`.
- **electron-packager (^17.1.2)**: Alternativa para empaquetar la aplicación en sistemas Linux, ya que no se generan archivos `.exe` para esta plataforma. El comando `npm run package` crea una compilación manual para Linux x64.

Dependencias de producción

Estas dependencias se incluyen en la versión final de la aplicación y son esenciales para su funcionamiento. Se clasifican en los siguientes grupos:

Conexiones y APIs externas

- **axios (^1.5.0)**: Cliente HTTP para realizar peticiones a APIs de servicios web, en particular a modelos de lenguaje.
- **openai (^4.10.0)**: Esta es la librería oficial para interactuar con la API de OpenAI y sus modelos más recientes.
- **node_characterai (^1.2.1)**: Conexión con Character.AI, que permite dotar al chatbot de una personalidad configurable y adaptable.

Manipulación de archivos y datos

- **pdf-lib (^1.17.1)**: Permite crear, editar y manipular archivos PDF desde Node.js.
- **dotenv (^16.3.1)**: Gestiona variables de entorno, como las claves de acceso a las APIs, mediante un archivo `.env`.

Procesamiento de texto y markdown

- **markdown-it (^14.0.0) y markdown (^0.5.0)**: Renderizan texto en formato Markdown a HTML, permitiendo formatear respuestas para el usuario y mostrarlas en tiempo real.

Utilidades generales

- **lodash (^4.17.21)**: Facilita la manipulación de arrays y objetos mediante funciones como cloneDeep.
- **colors (^1.4.0)**: Añade colores a los registros de consola para una depuración visual.
- **node-cmd (^5.0.0)**: Ejecuta comandos del sistema directamente desde Node.js.
- **ps-list (^8.1.1)**: Permite listar procesos en ejecución, resultando útil para el monitoreo de aplicaciones.
- **readline-sync (^1.4.10)**: Lee entradas de usuario de forma síncrona en la consola.
- **abort-controller (^3.0.0)**: Proporciona mecanismos para cancelar peticiones HTTP, como las realizadas con axios, cuando sea necesario.

Tras la instalación de todas las dependencias, el archivo package.json quedará modificado, reflejando la configuración inicial que define la estructura del proyecto.

3.2.3 Creando el punto de entrada (main.js)

El archivo **main.js** constituye el punto de entrada de la aplicación en Electron, encargándose de la creación de la ventana principal y del manejo de los eventos del ciclo de vida, desde su inicio hasta su finalización. El código principal se estructura bajo una arquitectura modular, garantizando un manejo eficiente de las ventanas y un sistema de comunicación robusto entre los procesos internos.

Arquitectura general

El proceso principal está diseñado para administrar múltiples ventanas mediante un sistema organizado en tres capas fundamentales.

Capa de gestión principal (Proceso Main)

En esta capa reside el núcleo de la aplicación, responsable de coordinar los componentes durante la inicialización. Sus funciones incluyen la configuración de las ventanas, la carga de archivos según las preferencias del usuario almacenadas en config.json y la creación del menú de bandeja del sistema (Tray). Asimismo, emplea ipcMain como mediador para enviar y recibir mensajes entre los procesos de renderizado. Una vez cargada la configuración y inicializadas las ventanas, esta capa gestiona eventos como el cierre de la aplicación, la persistencia de configuraciones y la detección de interacciones del usuario mediante el movimiento del cursor. Determina también cuándo mostrar las distintas ventanas a través del widget, permitiendo una manipulación coordinada por parte del usuario y manteniendo un flujo de datos consistente.

Capa de ventanas (BrowserWindow – Procesos Renderer)

Esta capa corresponde a las interfaces gráficas y se encarga de mantener las ventanas disponibles para que la capa de gestión pueda acceder a ellas según las solicitudes del usuario. Las ventanas incluidas son las siguientes:

- **Ventana principal (Widget):** es la ventana inicial, que cubre todo el monitor mostrando una animación Live2D o Spine como elemento interactivo. Incorpora transparencia mediante el evento setIgnoreMouseEvents para ignorar clics en zonas no interactivas, y es ajustable en posición y tamaño según los requerimientos del usuario.
- **Ventana de comunicación:** permanece oculta inicialmente y contiene la interfaz para la comunicación con el modelo de lenguaje, incluyendo capacidades de depuración durante el desarrollo.
- **Ventana de música:** también oculta, incorpora un reproductor con funcionalidad shuffle controlado mediante IPC para reproducir la música del usuario desde una ruta especificada.

- **Ventana de configuración:** oculta por defecto, permite ajustar las preferencias del usuario, tales como la ruta de la música, las claves de acceso, el modo silencioso o el número máximo de tokens permitidos.

Capa de servicios (Funcionalidades auxiliares)

En esta capa se incluye el soporte para el funcionamiento óptimo de las anteriores. Abarca la persistencia de datos, mediante la lectura y escritura en `config.json` para almacenar las preferencias del usuario, así como en `model_position.json` para guardar la posición y restaurar el estado del widget en caso de fallos. También realiza la gestión de archivos mediante diálogos nativos para la selección de rutas de música, junto con su validación y manejo de errores. Por último, la lógica de interacción detecta colisiones del cursor con la ventana widget mediante la función `capturePage` y un análisis de píxeles; además, gestiona un temporizador que actualiza estados y verifica la posición del mouse cada 5 ms. Esta capa opera como un back-end ligero vinculado al front-end, encargándose de operaciones críticas como el guardado de configuraciones de manera eficiente y confiable.

Componentes principales

El archivo principal gestiona los eventos medulares de la aplicación mediante los siguientes componentes:

- **Sistema de ventanas:** las ventanas siguen un patrón de diseño Factory donde funciones especializadas como `createHiddenChatWindow` o `createHiddenMusicWindow` se encargan de instanciarlas con configuraciones predefinidas.
- **Comunicación interprocesos (IPC):** el sistema IPC de Electron se utiliza extensivamente mediante `ipcMain` e `ipcRenderer`, bajo un patrón Publicador-Suscriptor que define canales para distintas operaciones. Los canales principales son: `model-message` (para el modelo animado), `chat-message` (para el sistema de chats), `music-message` (para control de música) y `settings-message` (para ajustes de configuración). Esto permite una comunicación eficiente entre componentes.

- **Gestión de estado y configuración:** el sistema emplea persistencia mediante los archivos config.json y model_position.json, los cuales son leídos mediante la función readConfigFile bajo un patrón Singleton que garantiza coherencia en el acceso a las configuraciones en toda la aplicación.

Flujo de ejecución

El ciclo de vida de la aplicación sigue la siguiente secuencia:

1. **Inicialización:** se cargan las configuraciones, se crea la ventana principal widget y se configura el menú de bandeja.
2. **Arranque:** se inician las ventanas secundarias en modo oculto, se establecen los listeners IPC y se configura la detección del cursor.
3. **Bucle principal:** se manejan los eventos interactivos, se establece la comunicación entre componentes y se actualizan los estados.
4. **Finalización:** se persisten los estados y se liberan los recursos.

Componentes implementados

Se integraron diversos patrones de diseño en la clase principal para asegurar su eficiencia y mantenibilidad:

- **Observer:** implementado mediante el sistema IPC para notificaciones entre componentes.
- **Factory:** utilizado en las funciones de creación de ventanas.
- **Singleton:** para el acceso global a la configuración del sistema.
- **Decorator:** aplicado en la funcionalidad de transparencia interactiva.
- **Mediator:** representado por el proceso principal que media entre las ventanas.

La adopción de estos componentes permite que la clase principal cumpla con estándares adecuados de programación, manteniendo una estructura coherente que favorece el desempeño esperado de la aplicación.

```

1  const { app, BrowserWindow, ipcMain } = require('electron');
2  const path = require('path');
3
4  let mainWindow;
5
6  function createWindow() {
7    // Crear la ventana del navegador
8    mainWindow = new BrowserWindow({
9      width: 800,
10     height: 600,
11     webPreferences: {
12       nodeIntegration: false, // Desactivado por seguridad
13       contextIsolation: true, // Activado por seguridad
14       preload: path.join(__dirname, 'preload.js') // Usar un script de precarga
15     }
16   });
17
18   // Cargar el archivo HTML principal
19   mainWindow.loadFile(path.join(__dirname, 'src', 'index.html'));
20
21   // Abrir las herramientas de desarrollo (solo en desarrollo)
22   mainWindow.webContents.openDevTools();
23
24   // Evento cuando la ventana es cerrada
25   mainWindow.on('closed', () => {
26     mainWindow = null;
27   });
28 }
29
30 // Evento cuando Electron ha terminado de inicializarse
31 app.whenReady().then(createWindow);
32
33 // Salir cuando todas las ventanas estén cerradas
34 app.on('window-all-closed', () => {
35   if (process.platform !== 'darwin') {
36     app.quit();
37   }
38 });
39
40 app.on('activate', () => {
41   if (mainWindow === null) {
42     createWindow();
43   }
44 });
45
46 // Aquí puedes agregar manejadores IPC para comunicación entre procesos
47 ipcMain.handle('perform-action', (event, ...args) => {
48   // Lógica para manejar la acción
49   return 'resultado';
50 });

```

Figura 3. Ejemplo de como crear un código main

3.2.4 Script de precarga (preload.js)

Las aplicaciones en Electron incorporan un script de precarga, componente fundamental que actúa como interfaz entre el front-end y el back-end. Este script permite tener control sobre las funcionalidades de Node.js y Electron, integrando la lógica del front-end sin poner en riesgo información sensible a posibles vulnerabilidades. Por esta razón, es crucial implementar esta clase de manera adecuada. En el presente proyecto, el desarrollo se lleva a cabo conforme al siguiente estándar.

En primer lugar, se realiza la importación de módulos de Electron y Node.js necesarios para garantizar la funcionalidad del código. Los módulos utilizados incluyen:

- contextBridge, para exponer de manera segura los módulos principales en el front-end;
- ipcRenderer, que permite la comunicación entre procesos mediante mensajes;
- screen, que proporciona información sobre las pantallas del sistema;
- path, para el manejo de rutas de archivos;
- fs/promise, que habilita operaciones de lectura y escritura mediante promesas;
- node-cmd, para ejecutar comandos en el sistema operativo.

Después, se realiza la exposición de APIs al front-end utilizando la función contextBridge.exposeInMainWorld. Se presentan tres grupos de funcionalidades, diseñados con un enfoque en la seguridad y la gestión eficiente de recursos:

1. **Módulo electron:** Este módulo facilita la comunicación entre procesos y el acceso a la información de la pantalla. A través de los métodos ipcRenderer, se utilizan los eventos on y removeListener, lo que permite escuchar y eliminar listeners de eventos activados por el usuario. De este modo, el front-end puede comunicarse de forma segura, óptima y eficiente con el proceso principal.

2. **Módulo nodeModules:** Este módulo se encarga de la manipulación de archivos JSON y del módulo path. Sus funciones principales incluyen readJsonFile, que lee y analiza un archivo JSON, manejando situaciones en las que el archivo no existe o tiene un formato inválido; y updateJsonFile, que actualiza el archivo combinando datos existentes con nuevos, utilizando fs.writeFile para guardar los cambios de manera persistente. Adicionalmente, expone el módulo path en el front-end para manipular rutas de archivos.

3. **Módulo nodeCmd:** Expone la ejecución de comandos del sistema operativo mediante la función run, la cual ejecuta un comando usando node-cmd e invoca un callback con el resultado. Esto permite ejecutar scripts y herramientas del sistema de forma segura, evitando la acumulación de recursos en cada llamada.

El archivo preload.js juega un papel crucial al asegurar que los recursos del front-end estén disponibles para el usuario, al mismo tiempo que mitiga los riesgos de seguridad asociados con información sensible. Asimismo, el uso de bloques try-catch captura errores durante operaciones con archivos o comunicaciones, evitando el bloqueo de la aplicación. Finalmente, las comunicaciones implementadas con ipcRenderer no se muestran completamente durante el

```
1  const { contextBridge, ipcRenderer } = require('electron');
2
3  // Exponer código principal al proceso de renderizado de forma segura
4  contextBridge.exposeInMainWorld('electronAPI', {
5    performAction: (...args) => ipcRenderer.invoke('perform-action', ...args),
6    onUpdate: (callback) => ipcRenderer.on('update', callback)
7  });
```

Figura 4. Ejemplo de una función preload

renderizado, lo que previene posibles filtraciones de información.

3.2.5 Creando la interfaz principal (index.html)

Para la interfaz principal, que se ejecutará a través del archivo main.js, puedes usar cualquier entorno de trabajo frontend como React, Vue o Angular; sin embargo, en este proyecto decidimos optar por HTML plano. La interfaz que hemos diseñado cumple con los estándares básicos del desarrollo web e incluye elementos multimedia, animaciones y una interacción dinámica para el usuario.. En esta sección se describe la organización del código de la interfaz, haciendo énfasis en su estructura HTML, los estilos CSS y las dependencias de JavaScript.

Estructura jerárquica del documento HTML

El código se basa en la estructura estándar de HTML5, dividida en tres segmentos clave:

- **Encabezado (<head>):** El encabezado sigue las convenciones propias de HTML. Incluye metadatos básicos, como `charset="utf-8"` para el soporte de caracteres especiales, la etiqueta `viewport` para la adaptabilidad en dispositivos móviles, y estilos embebidos mediante la etiqueta `<style>`, definidos en CSS.
- **Cuerpo (<body>):** El cuerpo del documento gestiona las capas superpuestas que conforman la interfaz. Entre estos componentes se encuentran elementos ocultos, como las coordenadas del cursor, la ruta de archivos y la posición del widget. Asimismo, incluye un contenedor principal con un lienzo para manejar gráficos dinámicos y un elemento `<div>` con un reproductor multimedia para animaciones Spine o Live2D. También contiene botones de acción flotantes para interactuar con otras ventanas, un área de texto emergente que permite enviar mensajes personalizados con efectos visuales, y un lienzo oculto destinado a la captura de pantalla y al seguimiento del movimiento del usuario.
- **Scripts:** Al final del cuerpo se incorporan las dependencias de JavaScript necesarias para manipular eventos y garantizar una carga óptima del documento.

Sistema de diseño y maquetación (CSS)

El diseño de estilos se basa en el uso de flexbox y posicionamiento absoluto o fijo para lograr una interfaz adaptable. Se organiza en tres propiedades principales:

- **Diseño principal:** Con margen y relleno establecidos en cero, emplea un fondo transparente pensado para integración en aplicaciones embebidas, como WebView en Electron. Incluye un contenedor flexible que ocupa el 100% del viewport y centra los elementos mediante flexbox, permitiendo que la capa gráfica aproveche estas propiedades.
- **Capas gráficas:** El lienzo principal tiene posición fija y cubre toda la pantalla. Aprovecha las propiedades del diseño principal y se ajusta al contenido del contenedor flexible. El reproductor multimedia posee dimensiones predefinidas y permanece oculto inicialmente. Los botones flotantes se posicionan de manera absoluta, superponiéndose a otros elementos para facilitar la interacción.
- **Personalización de la barra de desplazamiento:** Mediante la pseudo-clase `webkit-scrollbar`, se aplica un diseño estético con bordes redondeados y colores cálidos.

Componentes interactivos

La interfaz integra componentes interactivos que brindan una experiencia moderna y dinámica sin comprometer la usabilidad. Estos incluyen:

- **Botones con imágenes:** Los botones para abrir el chat, acceder a la configuración y controlar la reproducción de audio presentan un diseño minimalista y permanecen ocultos hasta que el usuario interactúa con el widget, momento en el que se muestran mediante transiciones suaves.
- **Área de texto:** Diseñada con bordes suaves y un fondo semitranslúcido de tonalidad suave, cuenta con animaciones integradas para mostrar mensajes asociados al personaje virtual.

Admite desplazamiento y movimiento, lo que mejora su utilidad para el usuario.

- **Elementos ocultos:** Componentes como las herramientas de depuración, el almacenamiento de rutas y los ajustes de posición se mantienen ocultos para evitar saturación visual.

Animaciones y efectos visuales

Se implementaron efectos que contribuyen a una experiencia más atractiva y cómoda, tales como:

- Oscilación para simular vibración sutil.
- Rebote para generar movimiento horizontal ligero.
- Desplazamiento dinámico en el cuadro de texto.

Integración de bibliotecas externas

La aplicación depende de diversas tecnologías externas para su funcionamiento óptimo, las cuales se clasifican de la siguiente manera:

Renderizado gráfico:

- Three.js para modelado 3D.
- PixiJS para renderizado 2D acelerado por hardware.
- Live2D Cubism para animaciones de personajes.
- Spine para animaciones esqueléticas.

Utilidades:

- jQuery para manipulación del DOM.
- Lodash para funciones auxiliares.
- Html2canvas para captura de pantalla.

Estructura modular

Los scripts locales se organizan de forma modular para mejorar la mantenibilidad y escalabilidad:

- renderer.js gestiona la lógica principal.

- charData.js contiene los datos de personalidad del asistente.
- constructorchoice.js maneja la configuración inicial.

Además, se implementaron mecanismos de escalado dinámico para ajustar el lienzo y los contenedores al redimensionar la ventana, controles de visibilidad para mostrar u ocultar elementos según el contexto, y transiciones fluidas en botones y áreas de texto.

De este modo, el archivo index.html constituye una interfaz principal tipo widget que funciona como un asistente virtual animado, garantizando un rendimiento óptimo, responsividad y facilidad de mantenimiento.

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Mi Aplicación Electron</title>
7   <link rel="stylesheet" href="styles.css">
8 </head>
9 <body>
10  <div class="container">
11    <h1>¡Hola desde Electron!</h1>
12    <button id="action-btn">Realizar acción</button>
13    <div id="result"></div>
14  </div>
15
16  <script src="renderer.js"></script>
17 </body>
18 </html>
```

Figura 5. Ejemplo de un documento HTML

3.2.6 Lógica del renderizado (renderer.js)

Este archivo implementa la lógica subyacente a la interfaz de usuario y gestiona la interacción con el usuario. Mientras que el archivo index define la representación visual de la interfaz, este documento contiene el script con las instrucciones para gestionar eventos, manipular el DOM y comunicarse con el proceso principal mediante IPC (Comunicación entre Procesos), activados como respuesta a la interacción del usuario.

La estructura general del código se organiza para gestionar eventos y manipulaciones del DOM, manteniendo como requisito que toda acción sea disparada por la interacción del usuario. Asimismo, se administra la gestión de ventanas secundarias, siguiendo un flujo de ejecución predefinido que maneja las distintas secciones:

- **Inicialización:** Espera a que el DOM se cargue completamente antes de ejecutar cualquier operación.
- **Configuración de elementos:** Obtiene referencias a los elementos del DOM y asigna sus valores iniciales.
- **Manejo de eventos:** Define listeners para eventos como click o contextmenu.
- **Comunicación IPC:** Utiliza `electron.ipcRenderer` para establecer comunicación con el proceso principal.
- **Gestión de ventanas:** Controla la apertura y el cierre de ventanas secundarias.

Los componentes principales son los siguientes:

En la etapa de **elementos del DOM y la configuración inicial**, una vez que el DOM está cargado, se obtienen las claves únicas de los elementos, como los botones de chat, configuración y música, así como el área de texto para contenido interactivo y los contenedores. Estos elementos tienen un diseño minimalista y permanecen ocultos hasta que se cumplen ciertas condiciones activadas por el usuario

La **gestión de ventanas y botones** se lleva a cabo a través de un array llamado `openWindows`, que se encarga de rastrear las ventanas abiertas. La función `initializeMenuFunctionality` se encarga de manejar estos eventos: si no hay ventanas abiertas, se añade una nueva al array y se abre; si ya hay una abierta, se cierra y se vuelve a abrir. Simultáneamente, se ocultan los botones en el widget, permitiendo al usuario acceder a distintas ventanas desde la interfaz principal y manteniendo la claridad visual.

Hablando de la **comunicación IPC**, este archivo la utiliza para mostrar mensajes en el asistente que se encuentra en el textSpace. Los mensajes, que abordan temas de interés, se eligen de manera aleatoria. A través de los eventos chatmsg-show, setmsg-show y musimsg-show, se presentan contenidos relacionados con las ventanas de chat, configuración o música que están abiertas. Asimismo, las funciones chat-created, settings-created y music-created informan al proceso principal que se ha abierto una ventana secundaria.

La **función** typeText mejora la experiencia de usuario mediante preprocesamiento de texto, que incluye la eliminación de emojis y ajustes de formato. Además, se incluye una animación de escritura que divide el mensaje en palabras y las muestra una a una. La duración de la visualización se determina según la longitud del texto, con una velocidad máxima de 200 palabras por minuto. Asimismo, ajusta el diseño calculando el tamaño del contenido para evitar desbordamientos, y finalmente, el texto se desvanece y el cuadro se oculta tras un tiempo determinado.

En cuanto a la **apertura de ventanas secundarias**, al hacer clic en un botón específico (como chat, configuración, etc.), se abre una ventana particular que recibe la configuración asociada, y se notifica al proceso principal para el seguimiento correspondiente.

Se han incorporado **efectos visuales** para mejorar la comodidad del usuario: los botones presentan un efecto de rebote (bounce) al mostrarse, mientras que el texto exhibe un efecto de oscilación (wiggle) al aparecer y se desvanece con una animación suave al desaparecer.

Flujo de ejecución de renderer.js:

El flujo de ejecución de renderer.js se puede resumir de la siguiente manera:

1. **Inicio:** La interfaz muestra el área interactiva, manteniendo ocultos los botones.
2. **Interacción:** Cuando el usuario hace clic en el área interactiva, los botones se muestran o se ocultan. Al pulsar un botón, se abre una ventana secundaria y los botones desaparecen.

3. **Mensajes:** Después de un breve intervalo, el asistente envía mensajes animados que luego se desvanecen. Si el usuario hace clic sobre un mensaje, se abre la ventana de chat con el texto incluido.
4. **Cierre:** Las ventanas secundarias se gestionan mediante un array lo que impide que se abran múltiples instancias de una misma ventana.

De este modo, mediante la aplicación de buenas prácticas de programación, se consigue controlar los eventos en la interfaz widget de manera óptima, minimizando la posibilidad de errores.

```
document.getElementById('action-btn').addEventListener('click', async () => {  
  const result = await window.electronAPI.performAction('param1', 'param2');  
  document.getElementById('result').textContent = result;  
});
```

Figura 6. Ejemplo de un evento de control

3.2.7 Comunicación entre procesos

Con los archivos creados, es necesario explicar el funcionamiento de la comunicación entre estos procesos y la fiabilidad inherente a dicha comunicación, ya que el mismo método será replicado en los procesos subsiguientes. Este sistema, proporcionado por Electron para la interacción entre ventanas y los procesos que las gestionan, no se detallará en las próximas secciones; por lo tanto, su funcionalidad se esclarece en este apartado.

Electron maneja dos tipos de procesos: los procesos principales, que se ejecutan en Node.js y administran el ciclo de vida de la aplicación, y los procesos de renderizado, los cuales representan cada ventana de la aplicación con su propia interfaz web. La comunicación entre ellos se realiza mediante el módulo de Electron denominado Inter-Process Communication (IPC). Este módulo establece un puente de comunicación entre el front-end web y el back-end de Electron, permitiendo el envío de datos desde la interfaz de usuario al proceso principal, encargado de ejecutar la operación solicitada. De este modo, se garantiza la integridad y consistencia de los datos durante todo el proceso.

Desde la página web, se realiza una llamada segura, cuyo nombre se muestra en toda la interfaz. Para la transacción de datos, se utiliza la función `sendData` en el front-end. La información se envía a través del canal `send-data` y es recibida por el proceso principal. Este proceso escucha los mensajes, recibe tanto el evento como los datos transmitidos, y garantiza que la información esté protegida en todo momento.

```
// En preload.js
contextBridge.exposeInMainWorld('electronAPI', {
  onUpdate: (callback) => ipcRenderer.on('update', callback)
});

// En main.js
mainWindow.webContents.send('update', { message: 'Datos actualizados' });
```

Figura 8. Ejemplo de comunicación desde el proceso principal al renderizado

Una vez que se resuelve la solicitud, el proceso principal debe enviar la respuesta al proceso de renderizado. Para lograr esto, se establece un canal de escucha en el front end, creando un patrón de notificación unidireccional. La llamada se expone en el navegador y se registra una función de retorno (callback) para recibir las actualizaciones. Con la función IPC, se escuchan los mensajes en el canal de actualización y se ejecuta el callback correspondiente cuando estos llegan.

```
// En preload.js
contextBridge.exposeInMainWorld('electronAPI', {
  sendData: (data) => ipcRenderer.send('send-data', data)
});

// En main.js
ipcMain.on('send-data', (event, data) => {
  console.log('Datos recibidos:', data);
});
```

Figura 7. Ejemplo de comunicación desde el renderizado a el proceso principal

3.2.8 interfaz de chat

Esta interfaz corresponde a una ventana secundaria, encargada de gestionar la interacción conversacional con diversos modelos de lenguaje. Aunque comparte

una estructura similar con la interfaz de la ventana principal, se diferencia tanto en su diseño como en las tecnologías que la componen. Mientras la ventana principal utiliza PixiJS, Three.js y un Canvas para renderizar elementos gráficos, esta interfaz se especializa en establecer conexiones con modelos de lenguaje y manejar interacciones textuales con el usuario. Para ello, emplea tecnologías como PDF.js, MathJax y la administración de claves de acceso. Asimismo, a diferencia de la interfaz principal (centrada en elementos visuales estáticos), aquí predominan componentes interactivos como formularios, selectores de modelos, carga de archivos PDF y un diseño clásico de chat que incluye un encabezado, un área de mensajes y campos de entrada.

Estructura jerárquica html

La interfaz cuenta con un encabezado que utiliza MathJax para renderizar fórmulas matemáticas en los mensajes del chat, junto con controles CSS para los elementos visuales. El cuerpo se compone de los siguientes elementos: un panel lateral ubicado dentro de un contenedor, el cual incluye una lista de sesiones y un botón para crear nuevas. Dicho contenedor puede ocultarse o mostrarse mediante un toggle.

El contenedor principal consta de tres secciones clave:

- **Encabezado (Header):** incluye un selector de modelos de lenguaje (como GPT, Gemini, DeepSeek), un campo para ingresar la clave de acceso que se conecta directamente con el back-end y el botón para alternar la visibilidad del panel lateral.
- **Área de chat:** espacio dedicado a mostrar el historial de la conversación.

- **Pie de página (Footer):** comprende un campo de texto autoajutable en altura según el contenido, acompañado de botones con funcionalidades específicas: enviar mensajes, cargar archivos PDF y regenerar respuestas.

```

<div id="app-container">
  <div id="header" class="header-container">
    <div>
      <label for="model-select">Select Model:</label>
      <select id="model-select">
        <option value="gpt-3.5-turbo-0125">GPT-3.5 Turbo</option>
        <option value="gpt-4o">GPT 4</option>
        <option value="gemini">Gemini</option>
        <option value="open-source">OpenSource</option>
        <option value="cohere">Cohere</option>
        <option value="deepseek">DeepSeek</option>
      </select>
    </div>
    <div class="toggle-button">
      <button id="toggle-chat" class="emoji-button" onClick="toggleContainer('side-container', 'toggle-chat')">&#128172;</button>
    </div>
    <div class="header-right">
      <div class="centered-element">
        <label for="api-key-input">API Key:</label>
        <input id="api-key-input" type="text" placeholder="Endpoint name" />
        <button id="api-key-submit" class="button-green">Set Endpoint</button>
      </div>
    </div>
  </div>

  <div id="chat-container">
  </div>
  <div id="footer">
    <div class="input-container">
      <textarea id="user-input" rows="3" placeholder="Type your message..."></textarea>
      
      <button id="upload-pdf-button" class="button-green">Upload PDF</button>
      <input type="file" id="file-input" accept=".pdf" style="display: none;" />
      <button id="regenerate-button">Regenerate Last Answer</button>
    </div>
  </div>
</div>

```

Figura 9. Body del chat

De este modo, el flujo de interacción para el usuario implica: seleccionar un modelo entre los disponibles actualizando su configuración correspondiente, gestionar sesiones de chat (crear nuevas o eliminar existentes), cargar archivos PDF para su lectura y utilizar un campo de entrada dinámico.

Funcionalidad independiente: renderer.js

El manejo de operaciones en la interfaz de chat sigue un enfoque modular, diseñado para distribuir responsabilidades entre tres subsistemas principales: la interfaz de usuario y sus eventos, la lógica de negocio asociada al chat y la personalización mediante prompts. Estos tres procesos convergen en el archivo renderer.js, el cual centraliza la funcionalidad de un asistente basado en redes neuronales generativas.

A diferencia del archivo homónimo de la ventana principal, aquí solo se invocan las dependencias estrictamente necesarias. La comunicación entre procesos se realiza

```
const { initUI } = require('./ui');
const { initChat } = require('./chat');
const { initPrompts } = require('./prompt');

async function main() {
  initUI();
  initChat();
  initPrompts();
}
main();
```

Figura 10. Archivo `renderer.js` en la ventana secundaria de chat

implícitamente mediante IPC (llamadas a procedimientos entre procesos), y cada subsistema tiene las siguientes responsabilidades:

Chat.js

Este archivo constituye el núcleo del subsistema de chat. Es utilizado por `renderer.js` y se encarga de la gestión de conversaciones, la integración con modelos de lenguaje y la persistencia de datos. Incluye módulos especializados para interactuar con APIs de diversos modelos como GPT-3, GPT-4, Gemini, Cohere y DeepSeek, así como funciones de traducción y obtención de datos meteorológicos. Además, hace uso de librerías de Node.js como `path` y `fs` para gestionar archivos, y `markdown-it` para renderizar mensajes.

Los datos principales que maneja son los chats y un objeto que almacena el contenido de cada conversación indexado por ID, mientras que la función `currentChatId` mantiene el estado de la conversación activa. El procesamiento de mensajes sigue un pipeline estructurado:

- **Recepción:** los mensajes se reciben mediante `window.addEventListener('message')` y se envían a través de eventos de botón o la tecla Enter.

- **Transformación:** se aplican instrucciones mediante `createInstruction()`, con patrones específicos para cada modelo, y se procesa el markup con sanitización.
- **Renderizado:** se integran avatares con imágenes (`user.png` y `assistant.png`) y se da soporte a ecuaciones matemáticas mediante `MathJax`.

El contexto de la conversación se actualiza constantemente: cada diez mensajes se reescribe utilizando el historial reciente. Al iniciar, se añaden datos meteorológicos como contexto inicial. Las conversaciones se guardan en el almacenamiento local y se sobrescriben los archivos `config.json` y `model_position.json`.

Cada modelo satisface necesidades distintas: GPT-3.5 y GPT-4 soportan grandes cantidades de tokens (16k y 128k, respectivamente); Gemini incluye traducción automática; Cohere está optimizado para respuestas concisas; y DeepSeek admite contextos extensos. La función `createChatCompletion` adapta los mensajes de forma óptima para el modelo seleccionado.

Adicionalmente, el archivo incorpora mecanismos de seguridad para el control de tokens, un sistema de failover automático que recurre a la API de OpenAI en caso de error, preservación del mensaje original durante regeneraciones, y reconstrucción precisa del historial tras fallos. También permite la creación y eliminación de chats con confirmación, y mantiene la persistencia entre sesiones mediante `localStorage`.

Ui.js

Este archivo actúa como capa intermedia entre la interfaz y la lógica de negocio del chat. Gestiona la interacción del usuario, la monitorización y la coordinación de eventos. Su arquitectura es modular y se comunica mediante IPC; integra seis funciones de `chat.js` y accede a datos meteorológicos. Además, ejecuta comandos y monitoriza recursos del sistema operativo.

El flujo central de mensajes implementa un mecanismo de throttling mediante la variable `isSendingMessage` para evitar solapamientos en los canales model-

message (respuestas generadas) y chat-message (interacciones conversacionales).

La monitorización de ventanas emplea las herramientas xdotool (para rastrear ventanas visibles) y xprop (para extraer metadatos como clase y título), filtrando aplicaciones según su estado previo. Esta funcionalidad solo está disponible en Linux y puede generar comentarios espontáneos sobre ventanas abiertas con un 30% de probabilidad.

La mensajería automatizada se basa en un sistema de temporizadores: al iniciar la aplicación, se envía un mensaje de saludo con fecha y hora mediante `sendGreetingsMessage`; cada cinco horas, se envía un mensaje meteorológico con temperatura y estado actual (`sendWeatherMessage`); existe también una probabilidad aleatoria de que cada 5 a 25 minutos se envíe un mensaje acorde con la personalidad del asistente (`sendAssistantMessage`); y cada 33 minutos, se envía un mensaje con información de monitorización del sistema (uso de RAM y CPU) mediante `sendSpecMessage`. Estas funciones están disponibles en todos los sistemas operativos.

El control de la interfaz gráfica se realiza mediante un patrón observador que responde a eventos de clic o teclado, validando entradas vacías. Se incluye una biblioteca de más de ochenta prompts predefinidos, organizados por categorías, que permiten interacciones dinámicas y evitan la repetición. Esta personalización busca generar una experiencia más orgánica y similar a la de un asistente humano.

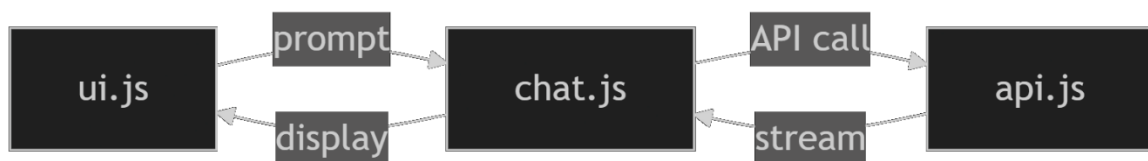


Figura 11. Diagrama de relación entre las 3 clases principales

Api.js

Este archivo gestiona las capacidades de chat con servicios externos, funcionando como puente entre la lógica de negocio y los proveedores de modelos de lenguaje. Soporta múltiples proveedores: OpenAI, Google Gemini, Cohere, DeepSeek y modelos de código abierto ejecutados localmente, así como servicios auxiliares como DeepL para traducción y síntesis de voz local en Linux.

La gestión multi-modelo se especializa en la generación de respuestas mediante siete funciones de tipo streaming (`createChatCompletion` para respuestas en tiempo real) y siete de tipo blocking (`createChatCompletionSimple` para respuestas completas). También incluye una función de traducción (`translateText`) usando DeepL, y una función de síntesis de voz (`synthesizeTextAndGetAudio`) mediante un servicio local en el puerto 5000 (exclusivo para Linux).

Los mensajes contextuales se transforman a un formato compatible con cada proveedor, normalizando heterogéneamente las salidas mediante un patrón push con callbacks. Para los streams, se utiliza un mecanismo de buffer: OpenAI emplea JSON delimited por `\n\n`, Gemini usa Server-Sent Events delimited por `\n`, y Cohere utiliza secuencias JSON delimited por `\n`. En caso de error en el stream, se notifica por consola y se ejecuta un callback.

La gestión de credenciales solicita en cada consulta la última clave ingresada por el usuario desde el archivo de configuración. Estas credenciales solo se emplean durante la llamada al servicio, sin comprometer su seguridad en tránsito. Los parámetros del modelo se establecen con una temperatura de 0.9, lo que favorece respuestas precisas pero no exclusivamente lógicas. El contenido se filtra para evitar respuestas ofensivas. Al finalizar el envío y respuesta, se utiliza caching para reutilizar tokens de acceso y un buffer para chunks incompletos; se estima el uso de tokens y se ejecuta un failover automático en caso de error durante la transacción.

Así, el ciclo se completa: la interfaz envía el mensaje, el chat recibe la petición, transforma el mensaje para la API correspondiente, recibe el stream de respuesta, lo envía al chat para su lectura y finalmente lo muestra al usuario.

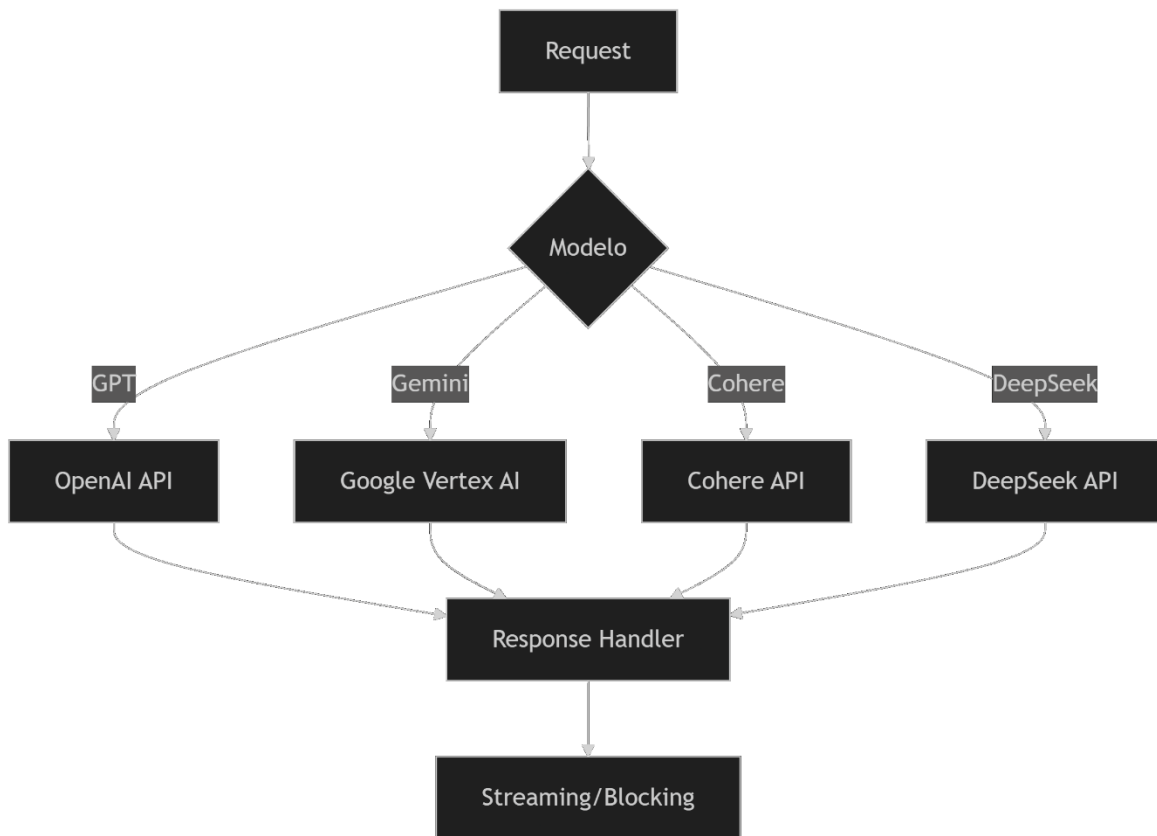


Figura 12. Diagrama de flujo sobre como distribuir las solicitudes

Prompt.js

Este componente completa el ecosistema de la ventana secundaria de chat mediante plantillas predefinidas para la interacción con los modelos de lenguaje. Incluye prompts basados en tres perfiles especializados y permite la gestión personalizada mediante operaciones CRUD: `addCustomPrompt` añade nuevos prompts, `submitCustomPrompt` procesa formularios, y `createPromptElement` los renderiza en la interfaz.

Estos prompts no inciden directamente en las conversaciones con el usuario, sino que actúan como temas sobre los cuales se pueden enviar mensajes desde widgets temáticos. Son independientes del modelo seleccionado y no interactúan con las APIs del usuario; su función principal es enriquecer la interacción mediante

comentarios aleatorios y variados, contribuyendo a una experiencia más dinámica y atractiva.

3.2.9 Interfaz de configuración

El documento de configuración es una interfaz gráfica creada en HTML que muestra al usuario todas las opciones disponibles. Esta interfaz permite gestionar modelos visuales, ya sean de tipo Spine o Live2D, con funcionalidades para agregar nuevos elementos y marcarlos como favoritos. Asimismo, incluye la configuración de la personalidad del asistente, el ajuste de parámetros técnicos, tales como el número máximo de tokens, la activación del modo silencioso, la administración de claves API para servicios como Cohere, OpenAI o DeepSeek, y la definición de una ruta para archivos de música.

En lo que respecta a los estilos visuales, se implementaron utilizando CSS para diseñar las tarjetas de las secciones del menú, aplicar efectos de desplazamiento (hover) en los botones, organizar de manera flexible los modelos favoritos y ajustar las dimensiones de las áreas de texto. Respecto a la lógica funcional, esta se limita a la interacción con el sistema de archivos para sobrescribir el archivo config.json y actualizar la configuración de modelos favoritos, así como para leer y visualizar dichos modelos durante la carga inicial. Toda comunicación con procesos externos se realiza mediante IPC (Comunicación entre Procesos) con el proceso principal de Electron, lo que elimina la necesidad de incorporar dependencias adicionales.

```

<div class="menu-item">
  <label for="addModel">Añadir un nuevo modelo:</label>
  <input type="file" id="addModelInput" style="display: none;" accept=".model3.json, .skel"
  <button id="addModel">Añadir</button>
  <button id="addToFavorites" style="display: block;">Añadir a favoritos</button>
</div>

<div class="menu-item">
  <h2>Modelos favoritos</h2>
  <div id="favoriteModels">
  </div>
</div>

<div class="menu-item">
  <h2>Cambiar personalidad</h2>
  <label for="personalityInput">Nueva personalidad:</label>
  <textarea id="personalityInput" rows="4" spellcheck="false" placeholder="Escribe tu nuev
  <button id="savepersonality">Guardar</button>
</div>
<div class="menu-item">
  <h2>Cambiar Máximo de Tokens</h2>
  <label for="maxTokensInput">Nuevo Máximo de Tokens:</label>
  <input type="number" id="maxTokensInput" placeholder="Ingresa el nuevo máximo de tokens.
  <button id="saveMaxTokens">Guardar</button>
</div>
<div class="menu-item">
  <h2>Modo silencioso</h2>
  <label for="silentModeCheckbox">Activar modo silencioso:</label>
  <input type="checkbox" id="silentModeCheckbox">
</div>
<div class="menu-item">
  <h2>Llave API de Cohere</h2>
  <label for="coherenceApiKeyInput">Ingresa tu llave API de Cohere:</label>
  <input type="text" id="coherenceApiKeyInput" placeholder="Ingresa tu llave de Cohere..."
  <button id="saveCoherenceApiKey">Guardar</button>
</div>
<div class="menu-item">
  <h2>Llave API de OpenAI</h2>
  <label for="openaiApiKeyInput">Ingresa tu llave API de OpenAI:</label>
  <input type="text" id="openaiApiKeyInput" placeholder="Ingresa tu llave de OpenAI..."
  <button id="saveOpenaiApiKey">Guardar</button>
</div>
<div class="menu-item">
  <h2>Llave API de DeepSeek</h2>
  <label for="deepseekApiKeyInput">Ingresa tu Llave API de DeepSeek:</label>
  <input type="text" id="deepseekApiKeyInput" placeholder="Ingresa tu llave API de DeepSeek
  <button id="saveDeepseekApiKey">Guardar</button>

```

Figura 13. Formato html para el menú de configuración

3.2.10 Funcionalidades avanzadas

La ventaja de Electron radica en sus funciones versátiles, las cuales permiten una gran variedad de operaciones. A continuación, se destacan las más relevantes:

Acceso al sistema de archivos

El acceso al sistema de archivos se lleva a cabo mediante un manejador seguro que opera la lectura y escritura de archivos. Gracias al uso de los patrones `async` y `await` durante estas operaciones, se obtiene un mecanismo robusto para el manejo de errores, lo que permite que el proceso de renderizado acceda eficientemente a dichas funciones.

Para ello, se emplea el módulo `fs` de Node.js para interactuar con el sistema de archivos, junto con el módulo `path` para el manejo de rutas. La comunicación entre procesos se realiza mediante IPC (Inter-Process Communication), utilizando un manejador que puede ser invocado desde el renderizador. Basta con un evento detonante y la ruta del archivo que se desea leer o escribir para que la función `fs` ejecute una promesa que lleva a cabo la transacción correspondiente. La ventaja de Electron radica en sus funciones versátiles, las cuales permiten una gran variedad de operaciones. A continuación, se destacan las más relevantes:

```
const fs = require('fs');
const path = require('path');

ipcMain.handle('read-file', async (event, filePath) => {
  try {
    const data = await fs.promises.readFile(filePath, 'utf-8');
    return { success: true, data };
  } catch (error) {
    return { success: false, error: error.message };
  }
});
```

Figura 14. Ejemplo de la función que permite el acceso a archivos

Notificaciones del sistema

La capacidad de mostrar mensajes emergentes constituye una cualidad de gran utilidad en aplicaciones tipo asistente. Electron incorpora un módulo específico de notificaciones; al importarlo en el proyecto, es posible definir una función que se encargue de crearlas y mostrarlas. Estas notificaciones poseen una estructura sencilla, compuesta por un título, un cuerpo de mensaje, la instancia de la notificación y el método responsable de exhibirla en pantalla.

```
const { Notification } = require('electron');

function showNotification(title, body) {
  new Notification({
    title,
    body
  }).show();
}
```

Figura 15. Ejemplo de cómo implementar notificaciones

Tray (icono en la bandeja del sistema)

Otra función de notable utilidad es el sistema Tray común en aplicaciones que operan en segundo plano. Se trata de un icono ubicado cerca del reloj en la bandeja del sistema de Windows, acompañado de un menú contextual con opciones y un tooltip informativo. Su propósito principal es indicar que la aplicación se está ejecutando y proporcionar acceso rápido a diversas funcionalidades.

Para implementarlo, primero se importa la clase Tray y se crea una instancia para gestionar el icono y su menú contextual, especificando la ruta de la imagen en formatos PNG, JPG o ICO que servirá como icono. La inicialización del Tray debe realizarse únicamente una vez que la aplicación Electron esté completamente lista, mediante la función `whenReady`.

Posteriormente, se construye el menú con las opciones deseadas y sus correspondientes acciones, y se asigna el texto del tooltip. De este modo, la aplicación contará con un Tray personalizado que, mediante comunicación IPC, puede controlar la posición del widget, cerrar la aplicación o recargarla en caso de

ser necesario. Estos eventos fueron diseñados con el fin de asistir al usuario y monitorear el estado de la aplicación, incluso cuando se ejecuta en segundo plano, además de permitir guardar la posición del widget según sus preferencias.

```
const { Tray, Menu, nativeImage } = require('electron');
const path = require('path');

let tray = null;

app.whenReady().then(() => {
  // Cargar imagen para diferentes resoluciones
  const iconPath = path.join(__dirname, 'assets', 'icon.png');
  const image = nativeImage.createFromPath(iconPath);
  image.setTemplateImage(true); // Para macOS

  tray = new Tray(image);

  // Menú contextual
  const contextMenu = Menu.buildFromTemplate([
    {
      label: 'Abrir',
      click: () => {
        if (mainWindow.isDestroyed()) {
          createWindow(); // Recrear ventana si fue cerrada
        } else {
          mainWindow.show();
        }
      }
    },
    { type: 'separator' },
    {
      label: 'Modo Silencioso',
      type: 'checkbox',
      checked: false,
      click: (item) => toggleSilentMode(item.checked)
    },
    { type: 'separator' },
    { label: 'Salir', click: () => app.quit() }
  ]);

  // Configuraciones
  tray.setToolTip(`${app.name} v${app.getVersion()}`);
  tray.setContextMenu(contextMenu);

  // Evento de doble clic (Windows/Linux)
  tray.on('double-click', () => mainWindow.show());
});
```

Figura 16. Ejemplo de cómo crear un tray

3.3 Empaquetado y distribución

Una vez concluida la aplicación, es necesario seguir un proceso de empaquetado para distribuirla en diversos dispositivos. No obstante, los distintos sistemas operativos requieren métodos de distribución específicos para llegar a los usuarios finales. Afortunadamente, Electron cuenta con diversas herramientas que facilitan el empaquetado para múltiples plataformas, como `electron-packager` y `electron-builder`.

El empaquetado consiste en convertir el código fuente de la aplicación en un archivo ejecutable, listo para ser distribuido en sistemas como Windows, macOS o Linux. Este proceso incluye el runtime de Electron, por lo que no requiere una instalación externa. Además, realiza una compresión adecuada de los archivos que componen la aplicación, tales como HTML, CSS, JavaScript o imágenes, con el fin de optimizar el uso de recursos.

Mediante esta compresión, es posible proteger el código fuente para evitar modificaciones no autorizadas; si bien se trata de una funcionalidad opcional, permite generar ejecutables específicos para cada sistema operativo y arquitectura, como `.exe`, `.dmg` o `.AppImage`, entre otros.

Electron-packager

Una de las herramientas disponibles en Electron es `electron-packager`. Este método te permite empaquetar tu aplicación en un formato binario que funciona en diferentes plataformas, todo a través de la línea de comandos de Node.js. Desde la raíz de tu proyecto, simplemente ejecutas el comando `npx`, que te deja correr paquetes de npm sin tener que instalarlos globalmente, seguido del paquete `electron-packager`.

A continuación, se indica el directorio actual donde se encuentra la aplicación, el nombre de la aplicación resultante y la plataforma objetivo mediante el argumento `--platform`, cuyos valores pueden ser `win32` (Windows), `darwin` (macOS) o `linux`. Finalmente, se especifica la arquitectura con el argumento `--arch=x64`.

Como resultado, se generan tres carpetas, una por cada plataforma, con la versión empaquetada correspondiente en arquitectura de 64 bits. Esta herramienta ofrece una forma sencilla y rápida de crear el ejecutable, aunque no genera instaladores, solo el archivo ejecutable capaz de iniciar la aplicación. Por ello, resulta adecuada para aplicaciones de escritorio que no requieren librerías adicionales, pero es menos recomendable en proyectos complejos que dependen de otras dependencias. En este proyecto se utilizó electron-packager, ya que no existen planes de distribución a gran escala a corto plazo.

Electron-builder

Electron-builder es una herramienta más avanzada, diseñada para aplicaciones complejas que requieren múltiples dependencias. Permite crear tanto ejecutables como instaladores mediante el comando `npx`, sin necesidad de instalar el paquete globalmente.

El proceso está automatizado y simplifica la distribución; basta con especificar las plataformas objetivo mediante argumentos como `--win`, `--mac` o `--linux`. Es especialmente útil para aplicaciones destinadas a su publicación en tiendas, ya que admite generadores de instaladores como NSIS, DMG o DEB, entre otros.

Otra ventaja significativa es su integración con electron-updater, que permite actualizaciones automáticas y publicación mediante GitHub o Snap Store. No obstante, su configuración resulta más compleja en comparación con otras opciones.

Electron-forge

Otra alternativa es electron-forge, un framework completo tipo «todo-en-uno» que simplifica tanto el desarrollo como el empaquetado. Utiliza tecnologías como Webpack y TypeScript, aunque carece del mismo nivel de personalización que electron-builder en lo referente a dependencias.

Para utilizarlo, debe ejecutarse el comando de creación con la plantilla deseada, por ejemplo, `--template=webpack`, seguido del comando `electron-forge make``. A pesar de sus limitaciones en personalización, ambas herramientas

electron-builder y electron-forge son opciones viables para la distribución de aplicaciones.

Encriptación

Electron permite además proteger el código fuente mediante el formato ASAR, que empaqueta el código en un archivo con extensión `.asar`. Este proceso se realiza mediante el comando `npx asar pack ./asistente ./asistente.asar`.

Sin embargo, este método no se considera totalmente seguro, ya que es posible extraer el contenido con herramientas como `asar extract`. Para una mayor protección, se recomienda emplear soluciones como `bytenode` o herramientas comerciales como `nexe`. Cabe señalar que, en este proyecto, no se implementó ningún método de encriptación, aunque es importante mencionar su existencia dentro del ecosistema de Electron.

Distribución

Una vez empaquetada, la aplicación puede distribuirse en diversos formatos según la plataforma. Para Windows, pueden generarse archivos `.exe`, `.msi` o `.appx` usando `electron-builder` con instaladores NSIS; para macOS, formatos como `.dmg`, `.pkg` o `.zip` mediante instaladores DMG; y para Linux, `.deb`, `.rpm` o `.AppImage`.

La distribución puede realizarse mediante descarga directa desde un servidor web, a través de GitHub Releases o en tiendas de aplicaciones como Microsoft Store, Mac App Store o Snap Store. En este proyecto, sin embargo, no se contempla una distribución beyond el ámbito demostrativo, ya que el empaquetado se realizó con

```
"scripts": {
  "pack": "electron-builder --dir",
  "dist": "electron-builder"
},
"build": {
  "appId": "com.ejemplo.miapp",
  "win": {
    "target": "nsis"
  },
  "mac": {
    "category": "public.app-category.utilities"
  },
  "linux": {
    "target": ["AppImage", "deb"]
  }
}
```

Figura 17. Ejemplo de un script con empaquetado y distribución

fines ilustrativos y no de difusión pública.

4. Resultados

Una vez empaquetada y distribuida la aplicación, al ejecutar el archivo .exe se inicia un asistente virtual animado basado en un modelo Live2D (L2D) precargado. Este asistente conserva todas las capacidades motoras para las cuales fue diseñado originalmente. El motor interno de la aplicación se encarga de cargar tanto el modelo como sus animaciones, mientras que el acceso al modelo de lenguaje permite generar respuestas de manera autónoma.

La interfaz del asistente está implementada como un widget con capacidad de movimiento y seguimiento del cursor. Puede superponerse sobre otras aplicaciones y desplazarse libremente por el monitor principal, además de permitir el ajuste de



Figura 18. Widget en funcionamiento con un mensaje de bienvenida

su tamaño. Al hacer clic sobre él, se despliegan tres botones con las funcionalidades predefinidas: el primero abre la interfaz de chat, el segundo accede a la configuración y el tercero reproduce música desde una ruta preestablecida. Cabe destacar que, aunque tanto la configuración como la ruta musical se establecen con antelación, la aplicación permite modificar todos los parámetros de configuración disponibles.

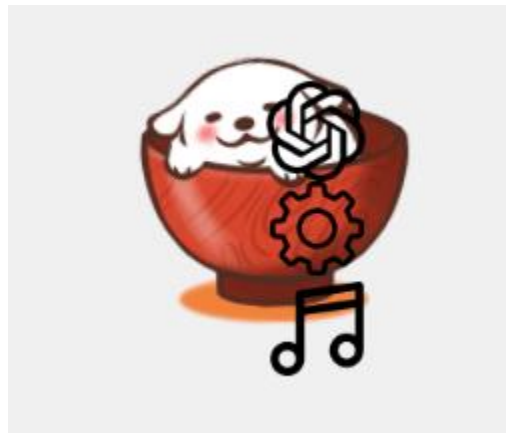


Figura 19. Widget con los botones presentes

La interfaz de chat consta de un panel lateral destinado a gestionar las sesiones de conversación, incluyendo un botón para eliminarlas y otro para crear nuevas. Todas las sesiones utilizan por defecto el modelo de lenguaje de Cohere, aunque es posible seleccionar otro modelo. Asimismo, la barra lateral puede ocultarse y cuenta con un campo para ingresar nuevas claves de acceso. En la sección principal, se muestra una interfaz de chat en tiempo real con el modelo de lenguaje, mientras

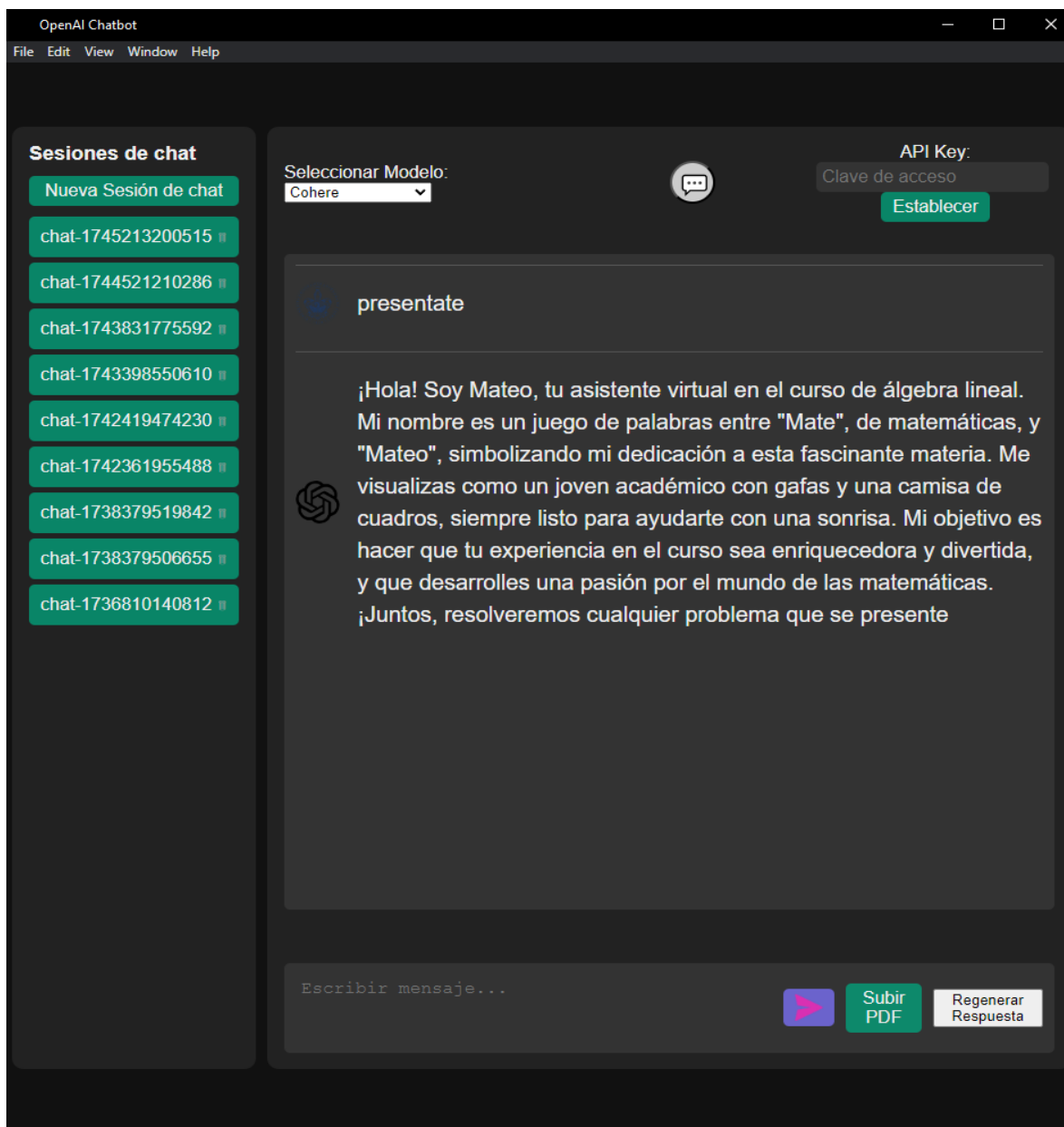


Figura 20. Interfaz de chat funcional

que en la parte inferior se encuentra un área para redactar mensajes, cargar archivos PDF o regenerar respuestas.

Al hacer clic en el botón de configuración, se abre una interfaz que muestra varios campos que puedes modificar. Cada uno de estos campos afecta directamente a la sección correspondiente en el archivo config.json. Por otro lado, el botón de reproducción musical no abre una interfaz gráfica; en su lugar, reproduce el audio en segundo plano.

El resultado es una interfaz de usuario que es tanto intuitiva como funcional. La división de menús por funcionalidades facilita la accesibilidad y el manejo de la aplicación sin contratiempos, lo cual evidencia una ejecución correcta que cumple con el objetivo de proporcionar una herramienta de naturaleza intuitiva. Además, se logra un uso eficiente de los recursos del sistema, ya que en las pruebas realizadas el consumo resultó significativamente inferior en comparación con los requisitos de hardware actuales. Esto garantiza un funcionamiento estable e ininterrumpido en la mayoría de los equipos disponibles en el mercado, convirtiendo así a la aplicación en una alternativa viable para el ámbito educativo.







Nombre	Estado	CPU	Memoria	Disco	Red	GPU
 live2d-assistant		0%	245.2 MB	0 MB/s	0 Mbps	0%
 live2d-assistant		0%	5.9 MB	0 MB/s	0 Mbps	0%
 live2d-assistant		0%	36.4 MB	0 MB/s	0 Mbps	0%
 live2d-assistant		0.8%	42.0 MB	0 MB/s	0 Mbps	0%
 live2d-assistant		1.1%	42.5 MB	0 MB/s	0 Mbps	6.1%
 live2d-assistant		0.1%	28.3 MB	0 MB/s	0 Mbps	0%

Figura 21. Consumo de recursos de la aplicación en ejecución

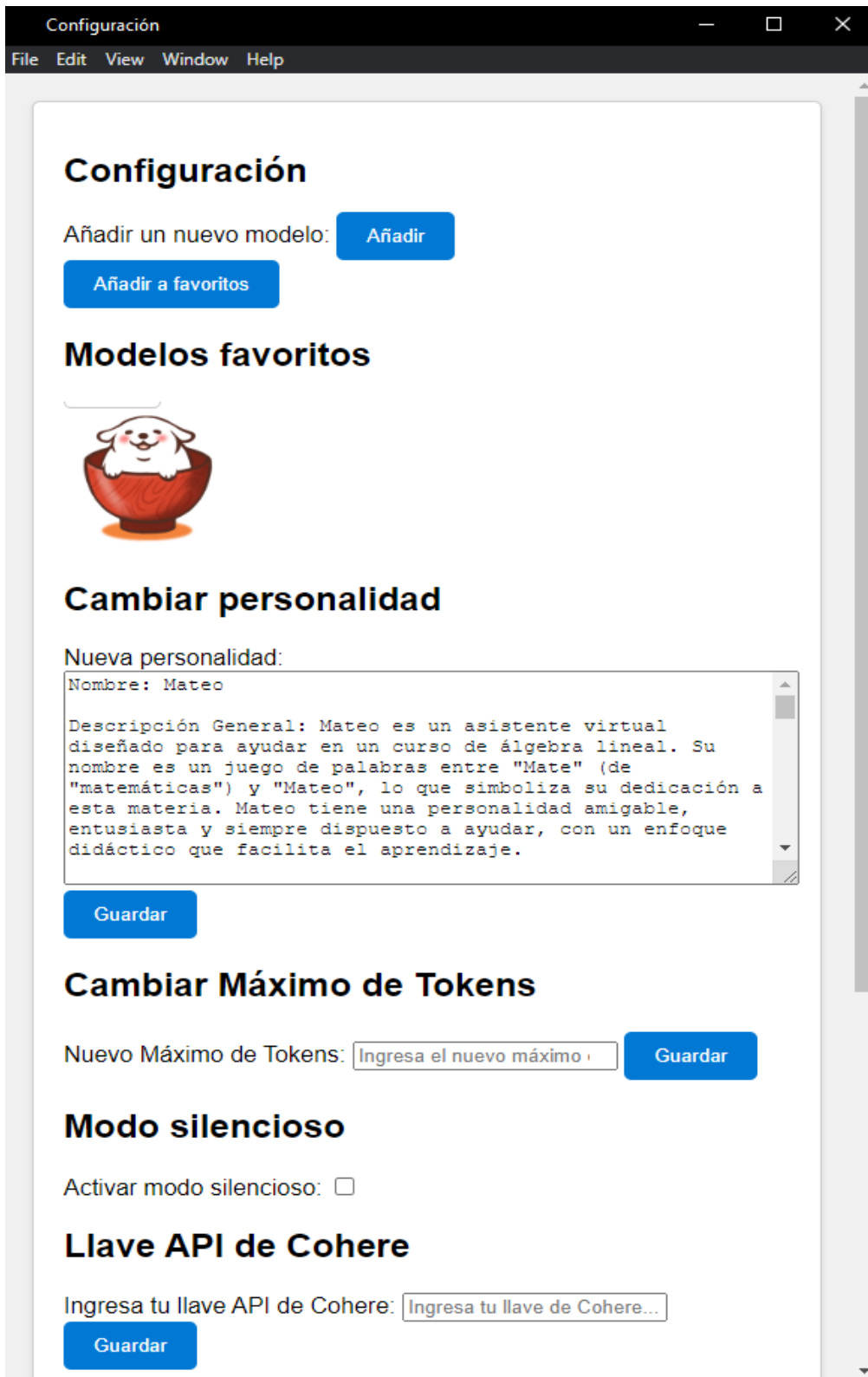


Figura 22. Interfaz de configuración

Prueba de asistencia: comparativa de respuesta a diversos modelos

Una vez comprobada la carga inicial de la configuración propuesta, es necesario verificar que el asistente posee la capacidad efectiva de brindar asesoría. Para ello, se seleccionó un ejercicio de álgebra lineal con el objetivo de resolverlo mediante distintos modelos de lenguaje; ello permite observar sus ventajas y desventajas en diversas propuestas. El ejercicio, resuelto de manera incorrecta de forma

1. (2 Puntos) Calcular el vector $2A-3B$.)

$$\mathbf{A} = (\pi, 0, -\pi)$$

$$\mathbf{B} = (2\pi, \pi, \pi)$$

$$-2\mathbf{A}-3\mathbf{B} = (-2\pi, 0\pi, 2\pi) - (6\pi, 3\pi, 3\pi) = (-8\pi, -3\pi, -\pi)$$

Figura 23. Ejercicio de suma de vectores

intencional, busca obtener retroalimentación de los modelos.

Cohere se presenta como una alternativa que ofrece dos tipos de modelo: una versión empresarial, que emplea sus últimas tecnologías, y un modelo gratuito con capacidades y procesamiento limitados. Sin embargo, para considerarlo una opción

Calcular el vector $2A - 3B$

$$\mathbf{A} = (\pi, 0, -\pi) \quad \mathbf{B} = (2\pi, \pi, \pi)$$

Para calcular $2A$, multiplicamos cada componente del vector A por 2: $2A = (2\pi, 0, -2\pi)$

Ahora, para calcular $-3B$, multiplicamos cada componente del vector B por -3 : $-3B = (-6\pi, -3\pi, -3\pi)$

A continuación, sumamos los dos vectores componente a componente: $2A - 3B = (2\pi, 0, -2\pi) + (-6\pi, -3\pi, -3\pi) = (-4\pi, -3\pi, -5\pi)$

Así, la respuesta final es: $2A - 3B = (-4\pi, -3\pi, -5\pi)$

Figura 24. Calculo propuesto por el modelo

viable y económica, se incluyó en la comparación. La respuesta que se dio al ejercicio 2A-3B, que se resolvió incorrectamente en el ejemplo, fue la correcta. En la figura 25, se muestra la retroalimentación propuesta por el modelo. Además, la retroalimentación logró identificar con precisión el error en el procedimiento original, que intentaba resolver $-2A-3B$ en lugar del ejercicio que se proponía. Se puede notar que el modelo entiende el error y mantiene un enfoque amigable hacia el usuario.

Recuerda prestar atención a los signos y componentes correctos al realizar operaciones con vectores. ¡Es fácil cometer un error de escritura o cálculo! Pero estoy seguro de que, con un poco de práctica, mejorarás en estos ejercicios.

Figura 25. Retroalimentación propuesta por el modelo

GPT-4o, uno de los modelos de lenguaje más conocidos, requiere el uso de API de pago, lo que puede resultar en una opción costosa. En las pruebas realizadas con

¡Vamos a revisar juntos!

Primero vamos a multiplicar cada vector por sus respectivos escalares:

Para el vector $A = (\pi, 0, -\pi)$, multiplicamos por 2: $2A = 2 * (\pi, 0, -\pi) = (2\pi, 0, -2\pi)$

Para el vector $B = (2\pi, \pi, \pi)$, multiplicamos por 3: $3B = 3 * (2\pi, \pi, \pi) = (6\pi, 3\pi, 3\pi)$

Ahora restamos los dos resultados: $2A - 3B = (2\pi, 0, -2\pi) - (6\pi, 3\pi, 3\pi)$

Haciendo la resta componente a componente:

- Para la primera componente: $2\pi - 6\pi = -4\pi$
- Para la segunda componente: $0 - 3\pi = -3\pi$
- Para la tercera componente: $-2\pi - 3\pi = -5\pi$

Figura 26. Ejercicio resuelto con modelo de lenguaje GPT-4o

el mismo ejercicio, aunque se obtuvo el resultado correcto, se notó una diferencia significativa en el procedimiento.

Este modelo asume un papel más orientado a la asistencia, ofreciendo una explicación detallada paso a paso en comparación con el modelo anterior. Tal característica lo posiciona como una opción con mayor capacidad de respuesta como asistente, al mantener una disposición constante para ayudar y mejorar.

Veamos dónde pudo estar el error. En tu procedimiento, al calcular " $-2A - 3B$ ", parece que hubo una confusión con los signos y los coeficientes escalares. Vamos a compararlo con el cálculo correcto que mencioné antes:

1. Multiplicación de los vectores:

- $2A$ para $A = (\pi, 0, -\pi)$: Tú escribiste $(-2\pi, 0, 2\pi)$ como resultado, pero se debería haber multiplicado 2 por cada componente de A : $(2\pi, 0, -2\pi)$.
- $3B$ para $B = (2\pi, \pi, \pi)$: Tú escribiste $(6\pi, 3\pi, 3\pi)$, lo cual es correcto para 3 veces el vector B .

2. Resta de los vectores:

- En lugar de calcular " $-2A - 3B$ ", estás tratando de restar directamente algunas componentes, pero los signos, sobre todo en el vector resultante que encontraste, parecen estar mezclados.

Figura 27. Explicación detallada del error con modelo GPT-4o

Además, el modelo proporciona retroalimentación sutil sobre los errores, con un tono más humano que enfatiza el aprendizaje a partir de los fallos; un enfoque que está integrado en la personalidad artificial del sistema y que tiende a ofrecer gestos de este tipo durante la interacción con el usuario.

Sin embargo, la operación inicial era $2A - 3B$, al cambiar el signo y con la consecuente lógica de $(-2A$ y $-3B)$ parece que fuiste guiado a un cálculo diferente. Comprobemos los signos en esas operaciones para evitar más problemas.

Espero que esto te ayude a ver el error. Si necesitas más aclaraciones, ¡estoy aquí para ayudarte!

Figura 28. El asistente permite ayudar con el seguimiento de los problemas en GPT-4o

Por otra parte, DeepSeek surge como un modelo de lenguaje reciente con altas capacidades de razonamiento, cuyo uso mediante API también tiene un costo. Este modelo fue igualmente capaz de llegar a la respuesta correcta, aunque mediante un método propio y metódico. Asimismo, reconoció el error y abordó ambos casos sin

Paso 1: Multiplicar el vector A por 2

Multiplicamos cada componente de **A** por 2:
[$2A = 2 \times (\pi, 0, -\pi) = (2\pi, 0, -2\pi)$]

Paso 2: Multiplicar el vector B por 3

Multiplicamos cada componente de **B** por 3:
[$3B = 3 \times (2\pi, \pi, \pi) = (6\pi, 3\pi, 3\pi)$]

Paso 3: Restar 3B de 2A

Ahora restamos componente a componente:
[$2A - 3B = (2\pi, 0, -2\pi) - (6\pi, 3\pi, 3\pi)$]
[$= (2\pi - 6\pi, 0 - 3\pi, -2\pi - 3\pi)$]
[$= (-4\pi, -3\pi, -5\pi)$]

Figura 29. Problema resuelto con el modelo Deepseek v3

desanimar al usuario; además, propuso temas para reforzar los conocimientos de

manera personalizada, manteniendo un margen discreto como asistente sin insistir en los errores.

El modelo reconoce el error e intenta abordar ambos casos sin desanimar al usuario, además de proponer temas para concretar los conocimientos de un modo muy personal sin desanimar al usuario, mantiene un margen discreto como asistente sin

Si quisiste calcular $-2A - 3B$ (en lugar de $2A - 3B$), el procedimiento sería:

1. $-2A = (-2\pi, 0, 2\pi)$
2. $-3B = (-6\pi, -3\pi, -3\pi)$
3. $-2A - 3B = (-2\pi - 6\pi, 0 - 3\pi, 2\pi - 3\pi) = (-8\pi, -3\pi, -\pi)$

¡Tu resultado es correcto si el ejercicio era $-2A - 3B$! Pero si era $2A - 3B$, la respuesta correcta es $(-4\pi, -3\pi, -5\pi)$.

¿En qué parte necesitas más ayuda?

- ¿Quieres repasar cómo se multiplica un vector por un escalar?
- ¿O cómo se restan vectores?

¡Estoy aquí para lo que necesites! 😊

"Recuerda: Los vectores son como flechitas, y las operaciones son solo sumar o restar sus pasos en cada dirección." 🚀

Figura 30. Seguimiento al problema usando deepseek-v3

hostigar con errores.

Si bien los ejercicios planteados involucran operaciones de escalar y aritmética básica, para demostrar la utilidad real del asistente es necesario elevar la complejidad de las solicitudes. Esto permitirá evaluar si los modelos son capaces

de auxiliar en elementos más técnicos del álgebra lineal, razón por la cual se proponen más ejercicios como guía para futuras pruebas.

2.-(3 Puntos) Dados los vectores $\mathbf{v}_1=(-1,0,-3,1)$ y $\mathbf{v}_2=(-1,2,-1,0)$ Calcular:

$$\text{La norma } \|\mathbf{v}_1\| = \sqrt{(-1)^2 + (0)^2 + (-3)^2 + (1)^2} = \sqrt{1 + 0 + 9 + 1} = \sqrt{11}$$

$$\text{La norma } \|\mathbf{v}_2\| = \sqrt{(-1)^2 + (2)^2 + (-1)^2 + (0)^2} = \sqrt{1 + 4 + 1 + 0} = \sqrt{6}$$

$$\text{La proy}_{\mathbf{v}_2} \mathbf{v}_1 = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_2\|} \mathbf{v}_2 = \frac{4}{6} (-1, 2, -1, 0)$$

Figura 31. Ejercicios de magnitud y proyección resuelto

Los siguientes ejercicios involucran la magnitud y la proyección de vectores, operaciones que requieren un mayor nivel de complejidad en su resolución debido al uso de raíces y conceptos propios del álgebra lineal. Las respuestas de los

¡Vamos a revisar tus cálculos!

Para encontrar la norma de un vector, también conocida como su longitud o magnitud, sumamos los cuadrados de cada componente del vector y luego calculamos la raíz cuadrada del resultado.

Primero, para el vector $\mathbf{v}_1 = (-1, 0, -3, 1)$:

$$\|\mathbf{v}_1\| = \sqrt{(-1)^2 + (0)^2 + (-3)^2 + (1)^2} = \sqrt{1 + 0 + 9 + 1} = \sqrt{11}$$

$$\text{Así que } \|\mathbf{v}_1\| = \sqrt{11}$$

Ahora, para el vector $\mathbf{v}_2 = (-1, 2, -1, 0)$:

$$\|\mathbf{v}_2\| = \sqrt{(-1)^2 + (2)^2 + (-1)^2 + (0)^2} = \sqrt{1 + 4 + 1 + 0} = \sqrt{6}$$

Por lo tanto, $\|\mathbf{v}_2\| = \sqrt{6}$

Figura 32. Magnitudes de vectores resueltos por modelo cohere

ejercicios en esta ocasión son correctas, por lo que se espera una retroalimentación positiva por parte de los modelos. Se inicia con el modelo de Cohere.

Se observa que los ejercicios de magnitud pueden resolverse de manera adecuada mediante este modelo, junto con una breve descripción del método utilizado.

A continuación, para encontrar el producto punto (proyección) de v_1 sobre v_2 , usas la fórmula:

$$v_1 \cdot v_2 = \|v_1\| \cdot \|v_2\| \cdot \cos(\theta)$$

Donde θ es el ángulo entre los vectores. Pero como tienes los vectores y sus normas, puedes calcular directamente el producto punto:

$$v_1 \cdot v_2 = (-1) \cdot (-1) + (0) \cdot (2) + (-3) \cdot (-1) + (1) \cdot (0) = 1 + 0 + 3 + 0 = 4$$

Así que $v_1 \cdot v_2 = 4$

Y finalmente, para encontrar la proyección de v_1 sobre v_2 , dividimos el producto punto por la norma al cuadrado de v_2 :

$$\text{proy}_{v_2} v_1 = v_1 \cdot v_2 / \|v_2\|^2 = 4 / (\sqrt{6})^2 = 4 / 6$$

Figura 33. Ejercicio de proyección resuelto con modelo Cohere

Asimismo, es posible obtener la proyección vectorial con el resultado correcto. No obstante, la retroalimentación no es óptima, ya que se limita a presentar los resultados numéricos. Esto podría considerarse una ventaja si el objetivo es únicamente la obtención de la respuesta; sin embargo, si se busca proporcionar asesoría educativa, este enfoque resulta insuficiente. Para determinar la utilidad de

este modelo en comparación con los demás, es necesario contrastar sus salidas con las de otras herramientas.

¡Vamos a revisar tus cálculos juntos!

1. Cálculo de las normas:

Para $v_1=(-1,0,-3,1)$:

La norma de un vector (v_1) es calculada como:

$$\|v_1\| = \sqrt{(-1)^2 + (0)^2 + (-3)^2 + (1)^2}$$
$$= \sqrt{1 + 0 + 9 + 1}$$
$$= \sqrt{11}$$

Tu cálculo para $\|v_1\|$ es correcto. ¡Bien hecho!

Figura 34. Ejercicio resuelto para la magnitud v_1

Para $v_2=(-1,2,-1,0)$:

La norma de un vector (v_2) es calculada como:

$$\|v_2\| = \sqrt{(-1)^2 + (2)^2 + (-1)^2 + (0)^2}$$
$$= \sqrt{1 + 4 + 1 + 0}$$
$$= \sqrt{6}$$

Tu cálculo para $\|v_2\|$ también es correcto. ¡Muy bien!

Figura 35. Ejercicio resuelto para la magnitud v_2

Al realizar la misma consulta en el modelo de GPT, se obtiene una respuesta que destaca por su enfoque más natural en la retroalimentación al abordar las

La formula para la proyección de v_1 sobre v_2 es:

$$\text{proy}_{v_2} v_1 = (v_1 \cdot v_2 / \|v_2\|^2) * v_2$$

Primero, vamos a calcular el producto punto ($v_1 \cdot v_2$):

$$v_1 \cdot v_2 = (-1)(-1) + (0)(2) + (-3)(-1) + (1)(0)$$

$$= 1 + 0 + 3 + 0$$

$$= 4$$

El producto punto $v_1 \cdot v_2$ es 4, lo cual es correcto.

Ahora, usando la fórmula de proyección:

$$\text{proy}_{v_2} v_1 = (4 / (\sqrt{6})^2) * (-1, 2, -1, 0)$$

$$= (4 / 6) * (-1, 2, -1, 0)$$

$$= (2 / 3) * (-1, 2, -1, 0)$$

$$= (-2/3, 4/3, -2/3, 0)$$

Por lo tanto, la proyección final es $((-2/3, 4/3, -2/3, 0))$.

¡Excelente trabajo revisando estos cálculos! Espero que te ayude a entender mejor el proceso. ¿Hay algo más en lo que pueda asistirte?

Figura 36. Ejercicio de proyección resuelto con modelo GPT

magnitudes, lo cual es esencial para un asistente educativo, especialmente en comparación con el modelo anterior. La retroalimentación que ofrece es excelente y se siente más personalizada, enfocándose en revisar los ejercicios de una manera didáctica.

Esta diferencia se debe, en parte, a la mayor capacidad de tokens tanto en la entrada como en la salida, lo que permite respuestas más extensas y detalladas,

1. Norma de $v_1 = (-1, 0, -3, 1)$

Tu cálculo:

$$\|v_1\| = \sqrt{(-1)^2 + (0)^2 + (-3)^2 + (1)^2} = \sqrt{1 + 0 + 9 + 1} = \sqrt{11}$$

✓ **Correcto.**

2. Norma de $v_2 = (-1, 2, -1, 0)$

Tu cálculo:

$$\|v_2\| = \sqrt{(-1)^2 + (2)^2 + (-1)^2 + (0)^2} = \sqrt{1 + 4 + 1 + 0} = \sqrt{6}$$

✓ **Correcto.**

Figura 37. Ejercicio de magnitud resuelto por modelo DeepSeek-v3

sin comprometer la exactitud de los resultados.

Para finalizar la comparación, se emplea el modelo DeepSeek-v3 con la misma consulta utilizada en los modelos anteriores. Su respuesta se caracteriza por ser precisa y concisa, yendo directo al grano mientras mantiene la coherencia en los resultados. Este aspecto es crucial, ya que, más allá de la conexión personal en la interacción, la corrección objetiva de los ejercicios es vital en un contexto académico donde las calificaciones dependen directamente de estos resultados. En otras

palabras, los modelos deben mostrar concordancia entre sus respuestas para ser considerados soluciones viables a la problemática planteada en este trabajo.

3. Proyección de v_1 sobre v_2 (proy_{v2} v_1)

Fórmula:

$$\text{proy}_{v_2} v_1 = [(v_1 \cdot v_2) / (\|v_2\|^2)] * v_2$$

Paso 1: Producto punto ($v_1 \cdot v_2$)

$$v_1 \cdot v_2 = (-1)(-1) + (0)(2) + (-3)(-1) + (1)(0) = 1 + 0 + 3 + 0 = 4$$

✅ **Correcto.**

Paso 2: Dividir entre $\|v_2\|^2$

$$\|v_2\|^2 = (\sqrt{6})^2 = 6$$

$$\text{Entonces: } (v_1 \cdot v_2) / \|v_2\|^2 = 4 / 6 = 2/3$$

Paso 3: Multiplicar por v_2

$$\text{proy}_{v_2} v_1 = (2/3) * v_2 = (2/3)(-1, 2, -1, 0) = (-2/3, 4/3, -2/3, 0)$$

◆ **Observación:**

Tú escribiste:

$$\text{"proy}_{v_2} v_1 = (4/6) v_2"$$

- **4/6 simplificado es 2/3**, así que en esencia está bien.
- Pero la proyección debe expresarse como el vector resultante **(-2/3, 4/3, -2/3, 0)**.

Figura 38. Ejercicio de proyección resuelto por DeepSeek-v3

Al analizar los resultados de la proyección de vectores, se nota una mejora en la asistencia y la retroalimentación en comparación con otros modelos. DeepSeek-v3 se enfoca especialmente en corregir detalles que, aunque no son errores, son

importantes para el aprendizaje y pueden ser señalados por el docente al estudiante.

3.- (5 puntos) Dadas las siguientes matrices calcular:

$$A = \begin{pmatrix} 2 & 3 \\ -1 & 1 \end{pmatrix} \text{ y } B = \begin{pmatrix} -1 & 1 \\ 2 & -3 \end{pmatrix}$$

$$A^2 - 2B = \begin{pmatrix} 1 & 9 \\ -3 & -2 \end{pmatrix} - 2 \begin{pmatrix} -1 & 1 \\ 2 & -3 \end{pmatrix} = \begin{pmatrix} 1+2 & 9-2 \\ -3-4 & -2+6 \end{pmatrix} = \begin{pmatrix} 3 & 7 \\ -7 & 4 \end{pmatrix}$$

Figura 39. Ejercicio de suma se matrices

Se sugiere un ejercicio de dificultad moderada que incluye la multiplicación de matrices, seguida de una multiplicación escalar y una resta de matrices. Esta operación compuesta implica diversos pasos y dado que el ejercicio fue resuelto

Veamos, calcularemos el cuadrado de la matriz A y luego restaremos el doble de la matriz B:

Primero, calculamos A al cuadrado:

$$A^2 = A * A =$$

$$\begin{bmatrix} 2 & 3 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ -1 & 1 \end{bmatrix}$$

Ahora, multiplicamos las matrices:

$$\begin{bmatrix} 4 + 3*(-1) & 23 + 31 \\ (-1)2 + 32 & (-1)3 + 11 \end{bmatrix}$$

Simplificando:

$$\begin{bmatrix} 1 & 9 \\ -2 & -2 \end{bmatrix}$$

Figura 40. Ejercicio de matrices resuelto por modelo Cohere incorrectamente

correctamente, se espera que los distintos modelos evaluados alcancen la misma respuesta y proporcionen la retroalimentación adecuada.

Al probar el ejercicio con el modelo de Cohere, se notó que no pudo calcular correctamente el cuadrado de la matriz. Por lo tanto, la solución del ejercicio es incorrecta desde el principio. La versión gratuita de Cohere no tiene la capacidad de entender contextos complejos al mismo nivel que otros modelos de pago. Esto hizo que no interpretara de manera precisa la estructura de las matrices, así que no se recomienda su uso para evaluar contenidos matemáticos en un curso de álgebra lineal. Sin embargo, sigue siendo una opción viable para generar contenido o resolver consultas más abstractas, como recomendar ejercicios, diseñar temas de estudio o crear material educativo. Es importante mencionar que la principal ventaja de este modelo es que es gratuito, lo que lo convierte en una opción económica para generar nuevo contenido educativo. A pesar de esto, su limitación para interpretar contextos complejos lo hace poco adecuado para evaluar contenido lógico en exámenes o ejercicios más elaborados. Al final, la decisión sobre su uso queda en manos del usuario.

Por otro lado, al hacer la misma consulta al modelo GPT-4o, se obtuvo una respuesta que muestra su habilidad para resolver correctamente el cálculo matricial. Este modelo se presenta como una opción más robusta en comparación con el anterior. Es fundamental recordar que la multiplicación de matrices sigue un procedimiento específico con reglas bien definidas, que son diferentes de las de la multiplicación por un escalar. Así, el modelo necesita entender el contexto del ejercicio para ofrecer una respuesta clara. En el caso de GPT-4o, no solo llega al resultado correcto de la multiplicación, sino que además ofrece los pasos intermedios del proceso, manteniendo una asistencia constante durante la resolución. Esta característica resulta especialmente valiosa si se desea brindar retroalimentación detallada al estudiante en caso de error. No obstante, debe señalarse que la notación matricial utilizada podría no ser la forma más óptima de presentar la información.

¡Claro! Vamos a revisar el ejercicio paso a paso. Quieres encontrar el resultado de la operación A al cuadrado menos dos veces B.

$$\text{Datos: } A = \begin{pmatrix} 2 & 3 \\ -1 & 1 \end{pmatrix}$$

$$B = \begin{pmatrix} -1 & 1 \\ 2 & -3 \end{pmatrix}$$

Primero, calculemos A al cuadrado (A multiplicado por A).

$$A * A = \begin{pmatrix} 2 & 3 \\ -1 & 1 \end{pmatrix} \times \begin{pmatrix} 2 & 3 \\ -1 & 1 \end{pmatrix}$$

Para multiplicar matrices, calculamos cada elemento de la matriz resultado tomando filas de la primera matriz y columnas de la segunda matriz:

$$\text{[Primer elemento: } (2 \cdot 2 + 3 \cdot (-1)) = 4 - 3 = 1]$$

$$\text{[Segundo elemento: } (2 \cdot 3 + 3 \cdot 1) = 6 + 3 = 9]$$

$$\text{[Tercer elemento: } (-1 \cdot 2 + 1 \cdot (-1)) = -2 - 1 = -3]$$

$$\text{[Cuarto elemento: } (-1 \cdot 3 + 1 \cdot 1) = -3 + 1 = -2]$$

Así que A al cuadrado es:

$$\begin{pmatrix} 1 & 9 \\ -3 & -2 \end{pmatrix}$$

Figura 41. Cálculo de A cuadrada por el modelo GPT-4o

De esta manera, el modelo logra encontrar la respuesta correcta y brindar una asistencia adecuada para el ejercicio que se le presenta.

Luego, calculemos 2 veces B. Multiplicamos cada elemento de B por 2:

$$2B = \begin{bmatrix} 2 \cdot (-1) & 2 \cdot 1 \\ 2 \cdot 2 & 2 \cdot (-3) \end{bmatrix}$$

Esto nos da:

$$\begin{bmatrix} -2 & 2 \\ 4 & -6 \end{bmatrix}$$

Finalmente, restamos 2B de A al cuadrado, simplemente restando cada elemento correspondiente:

$$\begin{pmatrix} 1 & 9 \\ -3 & -2 \end{pmatrix} - \begin{pmatrix} -2 & 2 \\ 4 & -6 \end{pmatrix} = \begin{pmatrix} 1+2 & 9-2 \\ -3-4 & -2+6 \end{pmatrix}$$

Esto nos da la matriz final:

$$\begin{bmatrix} 3 & 7 \\ -7 & 4 \end{bmatrix}$$

Eso coincide exactamente con lo que encontraste. ¡Buen trabajo!
Si tienes más dudas, estoy aquí para ayudarte

Figura 42. Solución al ejercicio y retroalimentación correcta del mismo

Su capacidad para entender el contexto lo convierte en una herramienta valiosa para evaluar contenido educativo, ya sea en exámenes o ejercicios, y también brilla en la creación de material académico. Sin embargo, es fundamental tener en cuenta que el uso de esta tecnología implica un costo monetario variable, por lo que es responsabilidad del usuario sopesar los beneficios en relación con los gastos que conlleva. A pesar de esto, se establece como una opción integral de apoyo en la mayoría de los ámbitos académicos.

Por último, DeepSeek-v3 se presenta como una tecnología emergente en el momento de redactar este documento. Se posiciona como una alternativa muy

competitiva frente a los mejores modelos de lenguaje disponibles, manteniendo un costo razonable para los usuarios.

Paso 1: Calcular A^2 (A multiplicada por A)

Para multiplicar $A \times A$, seguimos la regla de multiplicación de matrices (filas \times columnas):

$$A^2 = A \times A =$$

$$\left(\begin{array}{cc} (2)(2) + (3)(-1) & (2)(3) + (3)(1) \end{array} \right)$$

$$\left(\begin{array}{cc} (-1)(2) + (1)(-1) & (-1)(3) + (1)(1) \end{array} \right)$$

Calculamos cada posición:

1. Primera fila, primera columna (arriba a la izquierda):

$$(2)(2) + (3)(-1) = 4 - 3 = 1$$

2. Primera fila, segunda columna (arriba a la derecha):

$$(2)(3) + (3)(1) = 6 + 3 = 9$$

3. Segunda fila, primera columna (abajo a la izquierda):

$$(-1)(2) + (1)(-1) = -2 - 1 = -3$$

4. Segunda fila, segunda columna (abajo a la derecha):

$$(-1)(3) + (1)(1) = -3 + 1 = -2$$

Resultado de A^2 :

$$\left(\begin{array}{cc} 1 & 9 \end{array} \right)$$

$$\left(\begin{array}{cc} -3 & -2 \end{array} \right)$$

Figura 43. Ejercicio de matrices resuelto por DeepSeek-v3

Es evidente que su habilidad para crear contenido y su impresionante capacidad de razonamiento le permiten no solo resolver ejercicios, sino también resaltar los

puntos clave durante la revisión. Esto incluye desglosar los pasos, verificar las respuestas y estructurar correctamente matrices de 2x2.

Paso 2: Calcular 2B (B multiplicada por 2)

Multiplicamos cada elemento de **B** por 2:

2B =

$$(2(-1) \ 2(1))$$

$$(2(2) \ 2(-3))$$

Resultado de 2B:

$$(-2 \ 2)$$

$$(4 \ -6)$$

Figura 44. Obteniendo el valor final de B con modelo DeepSeek-v3

Se nota una metodología bien organizada para abordar los problemas, lo que

Paso 3: Restar $A^2 - 2B$

Ahora restamos **2B** de A^2 (elemento por elemento):

$$(1 \ 9) - (-2 \ 2) = (1 - (-2) \ 9 - 2)$$

$$(-3 \ -2) (4 \ -6) (-3 - 4 \ -2 - (-6))$$

Simplificamos:

1. Primera fila, primera columna: $1 - (-2) = 3$

2. Primera fila, segunda columna: $9 - 2 = 7$

3. Segunda fila, primera columna: $-3 - 4 = -7$

4. Segunda fila, segunda columna: $-2 - (-6) = 4$

Resultado final de $A^2 - 2B$:

$$(3 \ 7)$$

$$(-7 \ 4)$$

Figura 45. Resultado final obtenido con el modelo DeepSeek-v3

representa una mejora significativa en la asistencia para resolver de ejercicios.

Este modelo obtiene la solución de manera óptima y sin errores, demostrando una gran comprensión del contexto y habilidades avanzadas de generación de contenido. Gracias a su amplio manejo de tokens, puede estructurar respuestas adecuadas sin escatimar en recursos.

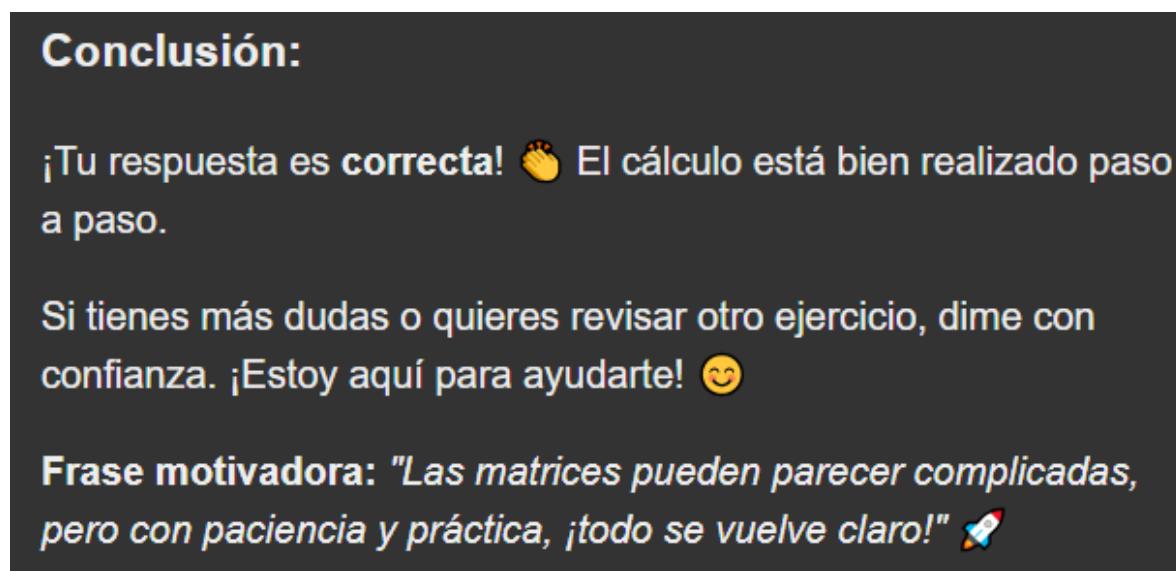


Figura 46. Retroalimentación al ejercicio por parte de DeepSeek-v3

Además, su retroalimentación se distingue por un tono amigable y positivo, lo que convierte a DeepSeek-v3 en una opción válida y versátil como asistente para un curso de álgebra lineal. Aunque su uso tiene un costo, se presenta como una solución innovadora en el panorama actual. Se ofrecen alternativas al usuario, quien tiene la última palabra sobre qué modelo utilizar y cómo hacerlo.

Pruebas rigurosas

Después de llevar a cabo con éxito las pruebas en entornos controlados con problemas predefinidos, se creó un nuevo conjunto de pruebas rigurosas, diseñadas por diferentes usuarios. El propósito fue validar el desempeño de la aplicación basada en redes neuronales generativas mediante la simulación de escenarios reales de uso en un entorno académico universitario. Estas pruebas se centraron en cuatro objetivos específicos: la generación y evaluación automática de contenido, la retroalimentación personalizada, la usabilidad para docentes y la reducción de carga administrativa. Con estos objetivos, se busca evaluar de manera integral la efectividad de la solución desde distintos enfoques.

Diseño del estudio

Se seleccionó una muestra diversa de usuarios, conformada por estudiantes de carreras afines a la ingeniería, docentes con diferentes niveles de experiencia en herramientas digitales y usuarios sin experiencia previa en dichas herramientas o en ingeniería. Esta selección busca representar una realidad educativa heterogénea, incorporando distintos niveles de familiaridad con los contenidos y las tecnologías.

Las pruebas aplicadas consistieron en sesiones prácticas simuladas en las que se monitoreó el rendimiento del sistema, el comportamiento del modelo de lenguaje, los tiempos de respuesta y la interacción usuario-sistema para estimar el impacto educativo. Se emplearon bases de datos existentes con ejercicios resueltos para las pruebas de evaluación, y se modificaron algunas respuestas para probar el mecanismo de retroalimentación.

Se ejecutaron un total de 48 casos de prueba, logrando una tasa de éxito del 85,4%. Para cada subobjetivo, se aplicaron métricas rigurosas que incluyeron indicadores cuantitativos y observaciones cualitativas obtenidas de los comentarios y la retroalimentación directa de los usuarios.

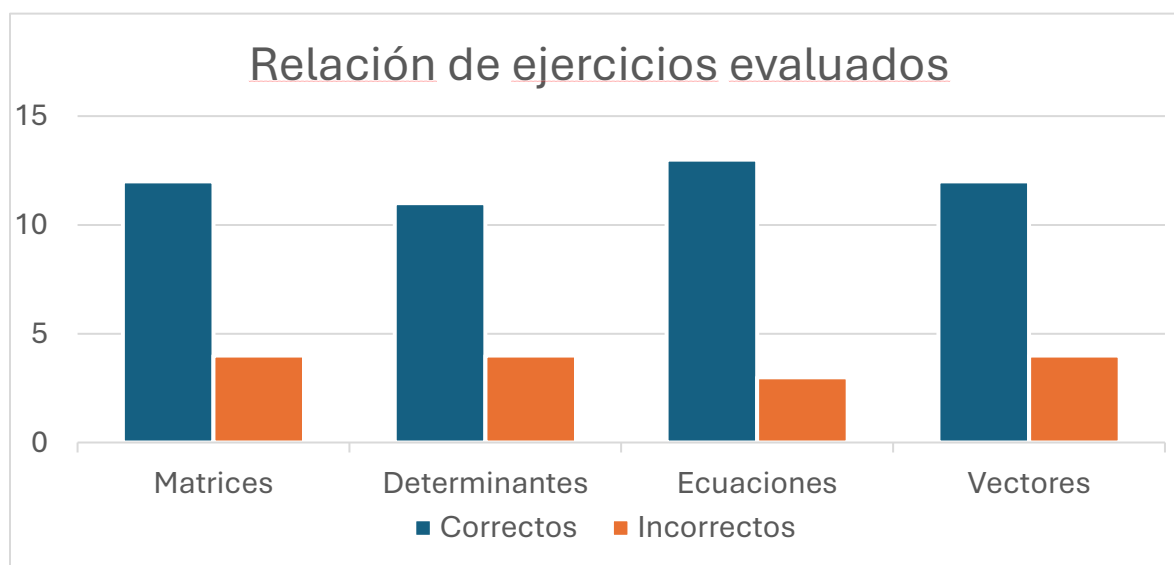
Subobjetivo 1: Generar y evaluar contenido de álgebra lineal

Este primer objetivo tuvo como finalidad validar la capacidad de la aplicación para generar ejercicios de álgebra lineal de manera autónoma y evaluar

Tipo de ejercicio	Generados	Válidos	Evaluación correcta
Matrices	16	14	12
Determinantes	11	9	7
Sistemas de ecuaciones	13	12	10
Vectores	12	10	8
Total	52	45	37

automáticamente las respuestas proporcionadas. Se generaron 52 ejercicios en total, distribuidos cuidadosamente entre los temas de matrices, determinantes, sistemas de ecuaciones y vectores. De estos, solo 45 se consideraron válidos para fines educativos; siete presentaban errores, como matrices inversas con determinantes cero, productos de vectores que no eran dimensionalmente compatibles y problemas de autovalores con propiedades indefinidas.

El sistema logró evaluar correctamente 37 ejercicios, lo que representa un 84,6% de los ejercicios válidos. Esta métrica mostró que había una mayor dificultad para generar contenido completamente válido con notación matemática precisa, especialmente en lo que respecta a determinantes y vectores. Sin embargo, la tasa general de aciertos destaca la notable capacidad de los modelos de lenguaje para interpretar y evaluar contenido matemático, sobre todo cuando se utilizan plantillas bien estructuradas.



Subobjetivo 2: Retroalimentación personalizada

En este apartado se midió la precisión del sistema para identificar errores en las respuestas de los estudiantes y proporcionar una retroalimentación útil. Sobre la base de datos de ejercicios, se simuló errores comunes, como fallos de procedimiento, errores conceptuales, operaciones algebraicas mal ejecutadas y uso incorrecto de la notación vectorial. De 26 errores insertados, el sistema consiguió

Figura 49. Tabla con la distribución de los tipos de ejercicio por errores

detectar correctamente 17, lo que representa un 73% de precisión, con un tiempo promedio de respuesta de 2,25 segundos.

Error simulado	Detectado correctamente	Detectado incorrectamente	Tiempo de respuesta (seg)
Procedimiento de matrices	5	3	2.0
Concepto de determinantes	4	3	2.4
Paso algebraico	5	1	2.1
Notación vectorial	3	2	2.5
Promedio general	73%		2.25 seg

La reducción en la precisión se atribuye a la mayor complejidad de los errores simulados, muchos de los cuales requerían un análisis profundo del proceso de resolución y no solo de la respuesta final. A pesar de esto, el sistema logró emitir mensajes comprensibles y pertinentes en la mayoría de los casos.

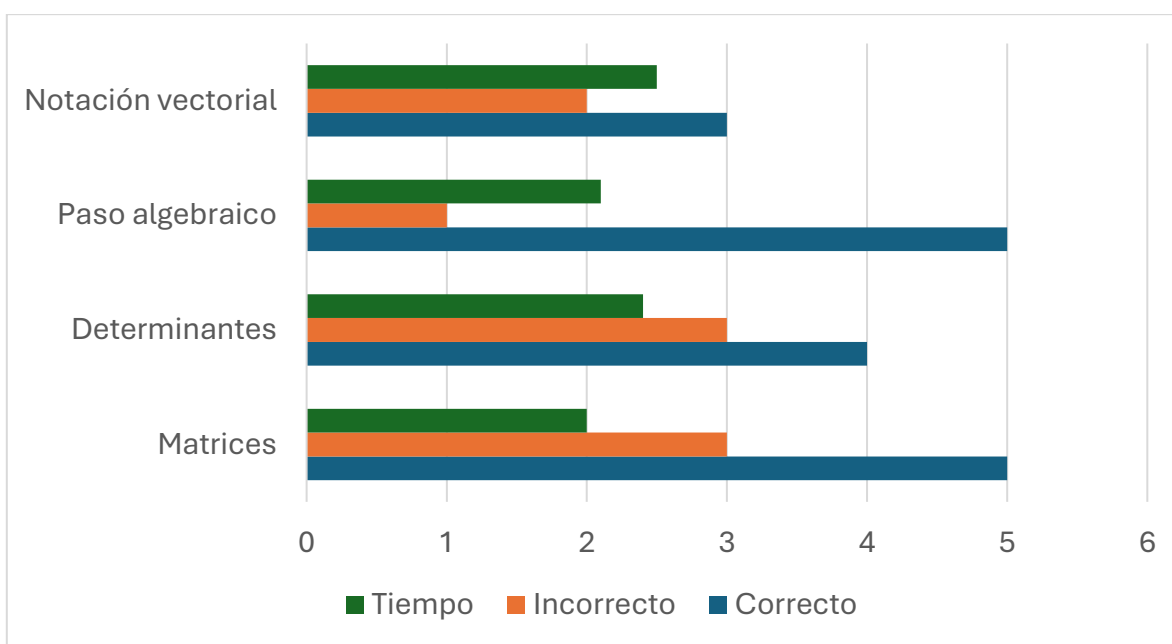


Figura 50. Gráfico de la complejidad en la retroalimentación con tiempo para cada ejercicio

Subobjetivo 3: Interfaz amigable para docentes

Para evaluar la experiencia de uso desde la perspectiva del docente, se aplicó la escala System Usability Scale (SUS), obteniendo un puntaje final de 77,3 sobre 100, valor que se interpreta como una buena usabilidad general. Las tareas evaluadas incluyeron la creación de actividades, la revisión del progreso del curso, la resolución de ejercicios y la asignación de retroalimentación. La aplicación presentó

una curva de aprendizaje breve, con tiempos de ejecución bajos y un número reducido de errores de navegación.

Tarea	Tiempo promedio (min)	Errores cometidos	Clics promedio
Crear nueva actividad	1.5	1 (inserción incompleta)	7
Revisar progreso del curso	2.8	1 (confusión menú y filtros)	9
Resolver ejercicio	1.7	0	6
Asignar retroalimentación	1.3	0	5

Figura 51. Tabla comparativa entre los tiempos promedio por actividad y los clics promedio

Los comentarios cualitativos de los usuarios sugirieron que la aplicación es “funcional” y “práctica”, aunque algunos experimentaron confusión con ciertos menús o botones. La mayoría coincidió en que la plataforma reduce pasos innecesarios y mejora la gestión del curso.

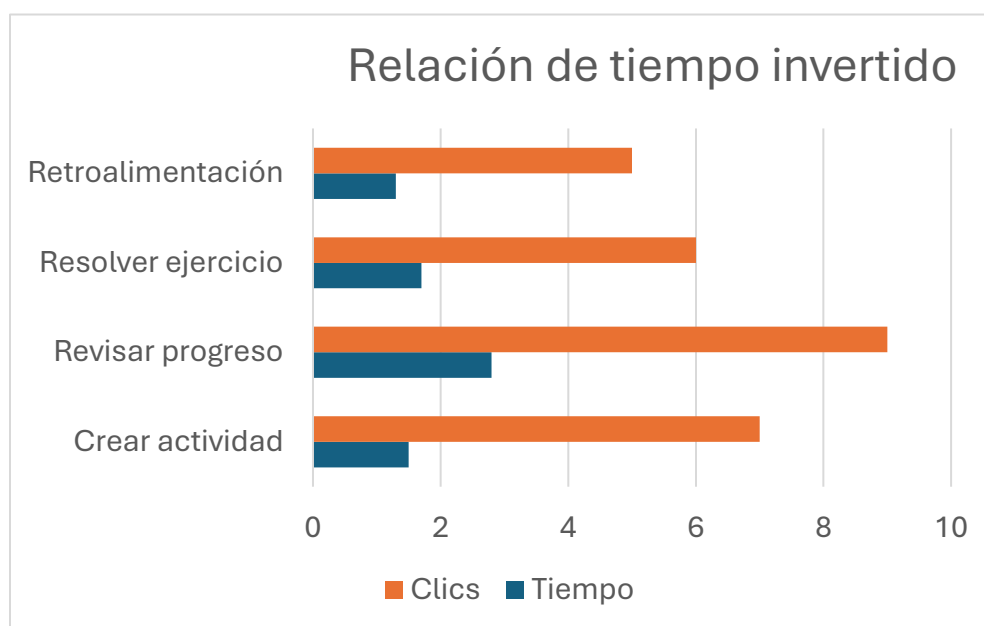


Figura 52. Gráfico de tiempo empleado en actividades contra clics

Subobjetivo 4: Reducción de carga administrativa

Finalmente, se estudió el beneficio de la aplicación en la disminución de tareas repetitivas y su impacto potencial en el aprendizaje. No obstante, medir este objetivo de forma cuantitativa requiere un curso completo para evaluar y monitorear el uso de la aplicación; a falta de este, solo es posible realizar una estimación basada en

los objetivos anteriores. Cabe señalar que la cantidad de alumnos y los métodos de evaluación varían, por lo que esta investigación adopta un punto de partida especulativo.

Asumiendo un tiempo considerable para las actividades de creación, evaluación y retroalimentación, la medición de los tiempos anteriores permite especular una reducción estimada del 72,9% en comparación con el tiempo que un docente invierte semanalmente en estas tareas. Así, de una duración pronosticada de 3,5 horas en promedio, sería posible reducir el tiempo a 55 minutos con el uso de la aplicación.

De igual modo, medir con precisión el desempeño académico estudiantil requeriría numerosos grupos de control. Asumir las capacidades de aprendizaje individual no es tan determinable como medir el tiempo del docente, por lo que este aspecto no fue considerado en las presentes pruebas.

Actividad	Tiempo sin app	Tiempo con app	Ahorro estimado
Crear ejercicios	2.2 horas	35 minutos	73.5%
Evaluar exámenes	3.5 horas	55 minutos	73.8%
Redactar retroalimentación	1.7 horas	30 minutos	70.6%
Total	7.4 horas	2.0 horas	72.9%

Figura 53. Tabla con tiempos especulativos en las actividades de un docente

Mediante la realización de diversas pruebas y combinaciones de ejercicios con distintos modelos, se demuestra que la aplicación puede ofrecer asesoría y retroalimentación personalizada e inmediata para los ejercicios propuestos, tal como se pretendía en los objetivos. La aplicación utiliza redes neuronales generativas para evaluar y crear contenido, lo que acelera notablemente el proceso de evaluación y se alinea con el objetivo de aliviar la carga administrativa.

Los resultados muestran que se ha desarrollado una herramienta que implementa de manera intuitiva y sencilla tecnologías de redes neuronales generativas para ayudar a los docentes, especialmente en un curso de álgebra lineal. Hasta la fecha

de publicación, la aplicación incorpora las mejores opciones tecnológicas disponibles en su funcionalidad.

Estos hallazgos sugieren que la integración de redes neuronales generativas en el ámbito educativo tiene un gran potencial para enriquecer la enseñanza y el aprendizaje de materias complejas como el álgebra lineal. A pesar de las limitaciones inherentes a una versión inicial, como errores de interpretación, ajustes en la usabilidad o fallos en la implementación, las ventajas de esta solución superan con creces los desafíos encontrados. El enfoque de evaluación utilizado en este estudio demuestra que, a través de pruebas variadas, realistas y rigurosas, es posible crear soluciones tecnológicas efectivas que se centren tanto en el contenido académico como en la experiencia del usuario.

5. Conclusiones

La enseñanza tradicional ha sido, a lo largo de la historia, un método confiable para transmitir conocimiento. Sin embargo, con los avances tecnológicos, como la introducción de computadoras en las aulas y, más recientemente, las adaptaciones que surgieron a raíz de la pandemia, su papel ha cambiado, abriendo nuevas oportunidades en el ámbito educativo. En este contexto, la inteligencia artificial se presenta como una herramienta con un potencial significativo, siempre que se utilice como apoyo y no como sustituto del trabajo docente.

La aplicación desarrollada en este proyecto constituye una aportación al campo de la educación, específicamente en la enseñanza del álgebra lineal a nivel universitario. Su propósito es complementar los métodos tradicionales mediante el uso de tecnologías emergentes, como la inteligencia artificial y las redes neuronales generativas, en combinación con el desarrollo de aplicaciones multiplataforma a través de Electron.

El sistema está diseñado para crear contenido educativo que se ajuste a las necesidades tanto de docentes como de estudiantes. Gracias a modelos de lenguaje avanzados, facilita la creación y resolución de ejercicios de álgebra lineal

en diferentes niveles de dificultad y temas variados, como operaciones con matrices, determinantes, sistemas de ecuaciones lineales, espacios vectoriales, transformaciones lineales y operaciones con vectores, entre otros. Esto brinda al docente herramientas para automatizar la elaboración de material didáctico dinámico y variado, al mismo tiempo que optimiza el tiempo de preparación y evaluación, favoreciendo un seguimiento más individualizado de los estudiantes.

Desde el punto de vista técnico, la aplicación presenta una estructura sólida, segura y escalable, con una interfaz diseñada para una experiencia de uso clara y accesible. Incorpora la capacidad de interpretar información a partir de texto o documentos PDF, procesándola con el modelo de lenguaje, además de un módulo para representar operaciones aritméticas con notación matemática adecuada, lo que facilita la comprensión de los contenidos.

Este trabajo demuestra que las redes neuronales generativas pueden ser una gran ayuda en el aprendizaje, sirviendo como un recurso para personalizar la enseñanza y como una herramienta para automatizar procesos relacionados con la labor docente. Además, muestra que la inteligencia artificial no viene a reemplazar el componente humano, sino que puede fortalecer la enseñanza tradicional y abrir nuevas oportunidades de investigación en otras disciplinas.

La aplicación que se ha desarrollado tiene el potencial de convertirse en un recurso valioso para los docentes que buscan fomentar una educación más interactiva y adaptada a las necesidades de sus estudiantes. Además, establece una base para futuras mejoras, como la integración con otras plataformas educativas o la expansión hacia más áreas de las matemáticas. Dado que la inteligencia artificial es un campo en constante evolución, los desarrollos de este tipo deberán incorporar los avances que surjan, garantizando su actualización y manteniendo la pertinencia de sus aplicaciones en el ámbito educativo.

6. Bibliografía

- 1 Bandi, A., Adapa, P. V. S. R., & Kuchi, Y. E. V. P. K. (2023). The power of generative AI: A review of requirements, models, input–output formats, evaluation metrics, and challenges. *Future Internet*, 15(8), 260. <https://doi.org/10.3390/fi15080260>
- 2 Okaiyeto, S. A., Bai, J., & Xiao, H. (2023). Generative AI in education: To embrace it or not. *International Journal of Agricultural and Biological Engineering*, 16(3), 285–286. <https://doi.org/10.25165/j.ijabe.20231603.8486>
- 3 Kasneci, E. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 102274. <https://doi.org/10.1016/j.lindif.2023.102274>
- 4 Chen, K. (2023). The teaching exploration of linear algebra under the background of artificial intelligence. *Curriculum and Teaching Methodology* 6, 29–37. <https://doi.org/10.23977/curtm.2023.062005>
- 5 Lim, W. M., Gunasekara, A., Pallant, J. L., Pallant, J. I., & Pechenkina, E. (2023). Generative AI and the future of education: Ragnarok or reformation? A paradoxical perspective from management educators. *International Journal of Management Education*, 21(2), Article 100790. <https://doi.org/10.1016/j.ijme.2023.100790>
- 6 Katsamakas, E., Pavlov, O. V., & Saklad, R. (2024). Artificial intelligence and the transformation of higher education institutions. *arXiv*. <https://doi.org/10.48550/arXiv.2402.08143>
- 7 Ávila Guamán, L. O. ., Briones Calvache , J. O. ., Hidalgo Cajo , D. P. ., & Calderón Gutiérrez, J. P. . (2024). Innovación en la enseñanza de álgebra lineal en la educación superior: integración de tecnologías interactivas y enfoques didácticos. *Reincisol.*, 3(6),4971–4988. [https://doi.org/10.59282/reincisol.V3\(6\)4971-4988](https://doi.org/10.59282/reincisol.V3(6)4971-4988)

- 8 Barberá-Gregori, E., & Suárez-Guerrero, C. (2021). Evaluación de la educación digital y digitalización de la evaluación. *RIED-Revista Iberoamericana De Educación a Distancia*, 24(2), 33–40. <https://doi.org/10.5944/ried.24.2.30289>
- 9 Grassini, S. (2023). Shaping the future of education: Exploring the potential and consequences of AI and ChatGPT in educational settings. *Education Sciences*, 13(7), 692. <https://doi.org/10.3390/educsci13070692>
- 10 McDonald, N., Johri, A., Ali, A., & Hingle, A. (2024). Generative artificial intelligence in higher education: Evidence from an analysis of institutional policies and guidelines. *arXiv*. <https://doi.org/10.48550/arXiv.2402.01659>
- 11 Berman, A., & Okubo, K. (2015). Issues surrounding teaching linear algebra. En S. Cho (Ed.), *The proceedings of the 12th International Congress on Mathematical Education* (págs. 65-72). Springer. https://doi.org/10.1007/978-3-319-12688-3_65
- 12 Dorier, J.-L. (2003). Teaching linear algebra at university. *arXiv preprint*. <https://doi.org/10.48550/arXiv.math/0305018>
- 13 Dorier, J.-L., & Sierpinska, A. (2001). Research into the teaching and learning of linear algebra. In D. Holton (Ed.), *The teaching and learning of mathematics at university level* (pp. 255–273). Springer. https://doi.org/10.1007/0-306-47231-7_19
- 14 Larson, J. E., & Zandieh, M. (2013). Challenges in teaching linear algebra: A study of instructors' perspectives. *International Journal of Mathematical Education in Science and Technology* 44(7), 961–973. <https://doi.org/10.1007/978-3-319-66811-6>
- 15 Hillel, J. (2000). Modes of description and the problem of representation in linear algebra. *Research in Collegiate Mathematics Education*, 4(1), 191–207. https://doi.org/10.1007/0-306-47224-4_7

- 16 Bond, M., & Bedenlier, S. (2023). Beyond emergency remote teaching: Did the pandemic lead to lasting changes in higher education? *International Journal of Educational Technology in Higher Education*, 20(1), 1–14. <https://doi.org/10.1186/s41239-023-00428-z>
- 17 Mao, J., Chen, B., & Liu, J. C. (2024). Generative artificial intelligence in education and its implications for assessment. *TechTrends*, 68(1), 58–66. <https://doi.org/10.1007/s11528-023-00911-4>
- 18 Naher, J., Maruf, M. H., Bakht, S. T., & Sadaf, S. S. (2023). A survey to understand the experience of ChatGPT usage among engineering university students in Bangladesh. *Technium: Romanian Journal of Applied Sciences and Technology* 15, 1–8. <https://doi.org/10.47577/technium.v15i.9660>
- 19 Cotton, D. R. E., Cotton, P. A., & Shipway J. R. (2024). Chatting and cheating: Ensuring academic integrity in the era of ChatGPT. *Innovations in Education and Teaching International*, 61(2), 228–239. <https://doi.org/10.1080/14703297.2023.2190148>
- 20 Koh, E., & Doroudi, S. (2023). Learning, teaching, and assessment with generative artificial intelligence: Towards a plateau of productivity. *Learning: Research and Practice*, 9(2), 109–116. <https://doi.org/10.1080/23735082.2023.2264086>
- 21 Dai, Y., Liu, A., & Lim, C. P. (2023). Reconceptualizing ChatGPT and generative AI as a student-driven innovation in higher education. *Procedia CIRP*, 119, 84–90. <https://doi.org/10.1016/j.procir.2023.05.002>
- 22 Peres, R., Schreier, M., Schweidel, D., & Sorescu, A. (2023). On ChatGPT and beyond: How generative artificial intelligence may affect research, teaching, and practice. *International Journal of Research in Marketing*, 40(2), 269–275. <https://doi.org/10.1016/j.ijresmar.2023.03.001>
- 23 Su, J., & Yang, W. (2023). Unlocking the power of ChatGPT: A framework for applying generative AI in education. *ECNU Review of Education*, 6(3), 355–366. <https://doi.org/10.1177/20965311231168423>

- 24 Tlili, A., Shehata, B., Adarkwah, M. A., Bozkurt, A., Hickey D. T., Huang, R., & Agyemang, B. (2023). What if the devil is my guardian angel: ChatGPT as a case study of using chatbots in education. *Smart Learning Environments*, 10(1), Article 15. <https://doi.org/10.1186/s40561-023-00237-x>
- 25 Hall, R. (2024). Generative AI and re-weaving a pedagogical horizon of social possibility. *International Journal of Educational Technology in Higher Education*, 21(1), 12. <https://doi.org/10.1186/s41239-024-00445-6>
- 26 Pham, T., Nguyen, T. B., Ha, S., & Ngoc, N. T. N. (2023). Digital transformation in engineering education: Exploring the potential of AI-assisted learning. *Australasian Journal of Educational Technology* 39(5), 1–19. <https://doi.org/10.14742/ajet.8825>
- 27 Thanh, B. N., Vo, D. T. H., Nhat, M. N., Pham, T. T. T., Trung, H. T., & Ha Xuan, S. (2023). Race with the machines: Assessing the capability of generative AI in solving authentic assessments. *Australasian Journal of Educational Technology* 39(5), 59–81. <https://doi.org/10.14742/ajet.8902>
- 28 Chocarro, R., Cortiñas, M., & Marcos-Matás, G. (2021). Teachers' attitudes towards chatbots in education: A technology acceptance model approach considering the effect of social language, bot proactiveness, and users' characteristics. *Educational Studies*, 47(6), 650–667. <https://doi.org/10.1080/03055698.2020.1850426>
- 29 Zhu, M., Liu, O. L., & Lee, H.-S. (2020). The effect of automated feedback on revision behavior and learning gains in formative assessment of scientific argument writing. *Computers & Education*, 143, 103668. <https://doi.org/10.1016/j.compedu.2019.103668>
- 30 Montenegro-Rueda, M., Fernández-Cerero, J., Fernández-Batanero, J. M., & López-Meneses, E. (2023). Impact of the implementation of ChatGPT in education: A systematic review. *Computers*, 12(8), 153. <https://doi.org/10.3390/computers12080153>

- 31 Bandi, A., Adapa, P. V. S. R., & Kuchi, Y. E. V. P. K. (2023). The power of generative AI: A review of requirements, models, input–output formats, evaluation metrics, and challenges. *Future Internet*, 15(8), 260. <https://doi.org/10.3390/fi15080260>
- 32 Bond-Taylor, S., Leach, A., Long, Y., & Willcocks, C. G. (2021). Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. arXiv. <https://doi.org/10.48550/arXiv.2103.04922>
- 33 Liu, M.-Y., Huang, X., Yu, J., Wang, T.-C., & Mallya, A. (2020). Generative Adversarial Networks for Image and Video Synthesis: Algorithms and Applications. arXiv. <https://doi.org/10.48550/arXiv.2008.02793>
- 34 Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. arXiv. <https://doi.org/10.48550/arXiv.1406.2661>
- 35 Arjovsky M., Chintala, S., & Bottou, L. (2017). Wasserstein Generative Adversarial Networks. Proceedings of the 34th International Conference on Machine Learning, PMLR 70:214–223. <https://doi.org/10.48550/arXiv.1701.07875>
- 36 Wang, D. (2001). Unsupervised Learning: Foundations of Neural Computation. *AI Magazine*, 22(2), 101. <https://doi.org/10.1609/aimag.v22i2.1565>
- 37 Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive Growing of GANs for Improved Quality Stability and Variation. arXiv. <https://doi.org/10.48550/arXiv.1710.10196>
- 38 Kang, M., Shin, J., & Park, J. (2022). StudioGAN: A Taxonomy and Benchmark of GANs for Image Synthesis. arXiv. <https://doi.org/10.48550/arXiv.2206.09479>

- 39 Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. arXiv. <https://doi.org/10.48550/arXiv.1706.08500>
- 40 Anantrasirichai, N., & Bull, D. (2022). Artificial intelligence in the creative industries: a review. *Artificial Intelligence Review*, 55, 589–656. <https://doi.org/10.1007/s10462-021-10039-7>
- 41 Nguyen Thanh, B., Vo, D. T. H., Nguyen Nhat, M., Pham, T. T. T., Thai Trung, H., & Ha Xuan, S. (2023). Race with the machines: Assessing the capability of generative AI in solving authentic assessments. *Australasian Journal of Educational Technology* 39(5), 59–81. <https://doi.org/10.14742/ajet.8902>
- 42 Nguyen Thanh, B., Vo, D. T. H., Nguyen Nhat, M., Pham, T. T. T., Thai Trung, H., & Ha Xuan, S. (2023). Race with the machines: Assessing the capability of generative AI in solving authentic assessments. *Australasian Journal of Educational Technology* 39(5), 59–81. <https://doi.org/10.14742/ajet.8902>
- 43 Kiryakova, G., & Angelova, N. (2023). ChatGPT A challenging tool for the university professors in their teaching practice. *Education Sciences*, 13(10), 1056. <https://doi.org/10.3390/educsci13101056>
- 44 Zhou, K. (2025). A Survey of Large Language Models. arXiv. <https://doi.org/10.48550/arXiv.2303.18223>
- 45 Yenduri, G., Ramalingam, M., Selvi, C. G., Supriya, Y., Srivastava, G., Maddikunta, P. K. R., Raj, D. G., Jhaveri, R. H., Wang, W., Vasilakos, A. V., & Gadekallu, T. R. (2023). Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions. arXiv. <https://doi.org/10.48550/arXiv.2305.10435>
- 46 Peeperkorn, M., Kouwenhoven, T., Brown, D., & Jordanous, A. (2024). Is Temperature the Creativity Parameter of Large Language Models? arXiv. <https://doi.org/10.48550/arXiv.2405.00492>

- 47 Lialin, V., Deshpande, V., Yao, X., & Rumshisky A. (2023). Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. arXiv. <https://doi.org/10.48550/arXiv.2303.15647>
- 48 Singh, K. (2023). The role of mathematics in artificial intelligence and machine learning. International Journal for Research Publication and Seminar, 14(5), 186–197. <https://doi.org/10.36676/jrps.v14.i5.1434>
- 49 Aydin, S. (2017). On the history of some Linear Algebra Concepts: From Babylon to PreTechnology. The Online Journal of Science and Technology- January 7(1).
- 50 Kublanovskaya, V. N. (1961). On some algorithms for the solution of the complete eigenvalue problem. USSR Computational Mathematics and Mathematical Physics, 1(3), 637–657. [https://doi.org/10.1016/0041-5553\(63\)90168-X](https://doi.org/10.1016/0041-5553(63)90168-X)

7. Recursos y Herramientas

https://web.pdx.edu/~erdman/LINALG/Linalg_pdf.pdf

https://jhefferow3.uvm.edu/linearalgebra/#current_version