UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

Ingeniero de Sistemas

AUTOR:

Tarazona Rodriguez, Jaime Alexander (orcid.org/0000-0003-1764-9749)

ASESOR:

Dr. Hilario Falcon, Francisco Manuel (orcid.org/0000-0003-3153-9343)

LÍNEA DE INVESTIGACIÓN:

Sistema de Información y Comunicaciones

LÍNEA DE RESPONSABILIDAD SOCIAL UNIVERSITARIA:

Desarrollo económico, empleo y emprendimiento

LIMA – PERÚ

2024

UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Declaratoria de Autenticidad del Asesor

Yo, HILARIO FALCON FRANCISCO MANUEL, docente de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA DE SISTEMAS de la UNIVERSIDAD CÉSAR VALLEJO SAC - LIMA NORTE, asesor de Tesis titulada: "Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología", cuyo autor es TARAZONA RODRIGUEZ JAIME ALEXANDER, constato que la investigación tiene un índice de similitud de 17%, verificable en el reporte de originalidad del programa Turnitin, el cual ha sido realizado sin filtros, ni exclusiones.

He revisado dicho reporte y concluyo que cada una de las coincidencias detectadas no constituyen plagio. A mi leal saber y entender la Tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad César Vallejo.

En tal sentido, asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

LIMA, 05 de Julio del 2024

Apellidos y Nombres del Asesor:	**Firma**
HILARIO FALCON FRANCISCO MANUEL **DNI:** 10132075 **ORCID:** 0000-0003-3153-9343	Firmado electrónicamente por: FHILARIOF el 05-07-2024 22:40:39

Código documento Trilce: TRI - 0797832

Declaratoria de originalidad del autor

UNIVERSIDAD CÉSAR VALLEJO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Declaratoria de Originalidad del Autor

Yo, TARAZONA RODRIGUEZ JAIME ALEXANDER estudiante de la de la escuela profesional de INGENIERÍA DE SISTEMAS de la UNIVERSIDAD CÉSAR VALLEJO SAC - LIMA NORTE, declaro bajo juramento que todos los datos e información que acompañan la Tesis titulada: "Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología", es de mi autoría, por lo tanto, declaro que la Tesis:

1. No ha sido plagiada ni total, ni parcialmente.
2. He mencionado todas las fuentes empleadas, identificando correctamente toda cita textual o de paráfrasis proveniente de otras fuentes.
3. No ha sido publicada, ni presentada anteriormente para la obtención de otro grado académico o título profesional.
4. Los datos presentados en los resultados no han sido falseados, ni duplicados, ni copiados.

En tal sentido asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de la información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

Nombres y Apellidos	**Firma**
TARAZONA RODRIGUEZ JAIME ALEXANDER **DNI:** 70479656 **ORCID:** 0000-0003-1764-9749	Firmado electrónicamente por: JTARAZONAR1 el 06-07-2024 11:42:21

Código documento Trilce: INV - 1786987

Dedicatoria

Este proyecto de investigación está dedicada a mi amada familia, en especial a mis queridos padres, cuyo inquebrantable apoyo ha sido la fuerza impulsora detrás de cada paso de este viaje académico. Su dedicación, paciencia y aliento constante han sido la brújula que me ha guiado a lo largo de esta travesía. Agradezco profundamente su sacrificio y amor, que han sido el cimiento sobre el cual se construye este logro. Este trabajo es un testimonio de su influencia positiva en mi vida, y les dedico con gratitud cada página, descubrimiento y aprendizaje que este proyecto encierra.

Agradecimiento

Este proyecto es tanto un logro académico como un tributo a mi amada familia y mis respetados docentes. Agradezco a mis queridos padres y hermanos por ser mi constante apoyo y motivación. A mis docentes, mi gratitud eterna por cada lección y orientación que ha dejado una huella indeleble en mi formación. Este proyecto lleva la esencia de su influencia y es un testimonio de su valioso impacto en mi vida académica y en mi formación profesional.

Índice de contenidos

Índice de tablas

Índice de figuras

Resumen

El presente estudio se enmarca dentro del Objetivo de Desarrollo Sostenible 3: Salud y Bienestar. Tuvo como objetivo determinar en qué medida el asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología. La metodología empleada tuvo un enfoque cuantitativo, fue de tipo aplicada y con un diseño pre experimental, empleando un pretest y postest. Con una muestra de 42 pacientes. La técnica de recolección de datos utilizada fue la encuesta, y el instrumento empleado fue un cuestionario. Los resultados mostraron una mejora significativa en la satisfacción del paciente tras la implementación del asistente virtual. Antes de la intervención, el 43% de los pacientes estaban insatisfechos, mientras que después, el 76% reportaron estar muy satisfechos. Se observaron mejoras notables en todas las dimensiones evaluadas: facilidad de uso, claridad de la información, tiempo de espera y eficiencia, y atención y comunicación. El análisis estadístico confirmó la efectividad del asistente virtual con una significancia asintótica bilateral de 0.000 ($p < 0.05$). Se concluye que el asistente virtual optimiza significativamente la satisfacción del paciente, mejorando la experiencia general en el instituto y demostrando el potencial de las tecnologías innovadoras en la atención médica.

Palabras clave: inteligencia artificial, satisfacción del paciente, asistente virtual, atención médica, tecnología de la información

Abstract

This study is framed within Sustainable Development Goal 3: Health and Wellbeing. Its objective was to determine to what extent the virtual assistant with artificial intelligence optimizes patient satisfaction at the National Institute of Ophthalmology. The methodology used had a quantitative approach, was applied and had a pre-experimental design, using a pretest and posttest. With a sample of 42 patients. The data collection technique used was the survey, and the instrument used was a questionnaire. The results showed a significant improvement in patient satisfaction after the implementation of the virtual assistant. Before the intervention, 43% of patients were dissatisfied, while afterward, 76% reported being very satisfied. Notable improvements were observed in all dimensions evaluated: ease of use, clarity of information, waiting time and efficiency, and attention and communication. The statistical analysis confirmed the effectiveness of the virtual assistant with a two-sided asymptotic significance of 0.000 ($p < 0.05$). It is concluded that the virtual assistant significantly optimizes patient satisfaction, improving the overall experience at the institute and demonstrating the potential of innovative technologies in healthcare.

Keywords: artificial intelligence, patient satisfaction, virtual assistant, healthcare, information technology

I. INTRODUCCIÓN

En efecto, el sector salud ha experimentado una considerable transformación en las últimas décadas, propiciada por los progresos tecnológicos y la progresiva adopción de soluciones digitales a nivel mundial. De acuerdo con informes de la Organización Mundial de la Salud (2021), más del 60% de los países miembros reportaron contar con estrategias nacionales de salud digital para el año 2020, lo que evidencia el compromiso de las naciones por aprovechar los beneficios de la digitalización en el ámbito sanitario, especialmente mediante la introducción de sistemas de información y el uso de inteligencia artificial para mejorar la atención médica.

En América Latina, la telesalud se convirtió en una solución vital para superar barreras geográficas y mejorar la accesibilidad a la atención médica. La inteligencia artificial desempeñó un papel fundamental al optimizar la programación de citas, facilitar la comunicación entre pacientes y médicos, y proporcionar respuestas inmediatas a preguntas frecuentes (Zia, 2024).

En Perú, el panorama de la salud digital presentó desafíos y oportunidades. El Ministerio de Salud (2023) reportó que solo el 30% de las instalaciones de atención médica pública tenían sistemas de información integrados para el año 2022. Esta situación evidenció la necesidad de promover la adopción de soluciones y fortalecer la infraestructura tecnológica en el sector salud peruano.

En este contexto, el Instituto Nacional de Oftalmología (INO) enfrentó retos significativos en la gestión de pacientes. La creciente demanda de servicios oftalmológicos, combinada con recursos limitados y procesos administrativos ineficientes, generó largas listas de espera y dificultades en la programación de citas. Se evidenció la falta de canales de atención al cliente accesibles. Esta deficiencia se tornó crítica dado que la mayoría de los pacientes de este instituto sufren de problemas visuales graves, lo que les impide utilizar efectivamente los métodos tradicionales de atención basados en la comunicación visual, como pantallas con números de turno. Asimismo, se observó que la gestión de citas y consultas se realizaba de manera presencial, lo que implicaba largas esperas y una experiencia de atención deficiente para los pacientes con discapacidades visuales. Un estudio realizado por Ramírez et al. (2020) en el INO reveló que el tiempo promedio de espera

para una consulta oftalmológica especializada era de 45 días, lo que superaba los estándares internacionales recomendados.

Ante esta situación, surgió la necesidad de realizar soluciones tecnológicas transformadoras que permitieran optimizar la gestión de pacientes en el INO. La inteligencia artificial emergió como una herramienta prometedora para abordar estos desafíos. Una meta-análisis realizada por García y López (2023) demostró que la implementación de asistentes virtuales basados en IA en entornos hospitalarios podía reducir los tiempos de espera en un 30% y mejorar la satisfacción del paciente en un 25%.

En consonancia con el avance hacia la digitalización en el Instituto Nacional de Oftalmología en el campo de la salud y la inteligencia artificial, este proyecto de investigación tiene como objetivo contribuir al Objetivo de Desarrollo Sostenible 3 salud y bienestar. Se presta especial atención a mejorar la igualdad de acceso a servicios de salud de calidad, incluyendo la atención oftalmológica, a través de soluciones tecnológicas innovadoras. Al reducir las listas de espera y optimizar la gestión de citas mediante un asistente virtual basado en IA, el proyecto aspira a fortalecer la eficiencia operativa del INO y mejorar la experiencia del paciente, particularmente para aquellos con discapacidades visuales, alineándose así con los principios de inclusión y accesibilidad.

En consecuencia, **el problema de investigación es el siguiente**: ¿De qué manera el asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología? Este planteamiento se desglosó en las siguientes **preguntas específicas**: **a)** ¿Cuáles son los niveles de satisfacción en cuanto a sus dimensiones facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología antes de la implementación del asistente virtual con inteligencia artificial?; **b)** ¿Cómo se puede diseñar y aplicar el asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología?; **c)** ¿Cuáles son los niveles de satisfacción en cuanto a sus dimensiones facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del pacientes del Instituto Nacional de Oftalmología después de la implementación del asistente virtual con inteligencia

artificial?; **d)** ¿Qué diferencias se observan en los puntajes de satisfacción del paciente entre el pre test y el post test después de la implementación del asistente virtual?

De manera similar, se exponen las razones fundamentales que respaldan este estudio, con el fin de proporcionar **justificaciones** exhaustivas que sustenten la necesidad de llevar a cabo la indagación.

La justificación teórica de este estudio se basó en la afirmación de Arana, et al. (2021), quienes destacaron la importancia de comprender integralmente la utilidad de los asistentes virtuales como una tendencia actual en la atención a clientes empresariales. Este enfoque buscó establecer los fundamentos para investigaciones posteriores, Ayudando de esta manera a elevar la calidad de los datos relacionados con el avance de los asistentes virtuales. Este conocimiento enriquecido pudo fomentar una implementación más efectiva y un rendimiento óptimo en diversas tareas. Además, según lo señalado por Charrois (2019), mejorar la atención al paciente al integrar un ayudante virtual con comprensión artificial resultó en una notable mejora en los procedimientos de atención.

La justificación metodológica, basada en el enfoque de Manjares y Echeverri (2020), implicó la adopción de enfoques o tácticas innovadoras para producir datos confiables. Se creó e implementó un plan de gestión de asistente virtual para mejorar la efectividad y robustez del sistema, ofreciendo orientación valiosa para abordar los problemas detectados en la investigación.

La justificación práctica, respaldada por las ideas de Arango (2019), se fundamentó en la premisa de que los Asistentes Virtuales facilitaron la interacción con clientes, ofreciendo soporte para diversas operaciones o servicios. Esta estrategia aseguró que los clientes lograran sus metas de forma eficaz y mejoró la calidad del servicio al automatizar procesos, optimizar los tiempos de atención y desarrollar la eficacia en la asistencia de servicios.

De modo que, se estableció como **objetivo general** lo siguiente**:** Determinar en qué medida el asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología. Esto, a su vez, proporcionó el marco para definir los distintos **objetivos específicos**: **a)** identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de

espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología antes de la aplicación del asistente virtual; **b)** diseñar y aplicar el asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología; **c)** identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología después de la aplicación del asistente virtual; **d)** analizar los puntajes obtenidos del pre y post test de la satisfacción del paciente para verificar la efectividad del asistente virtual.

Por otro lado, la presente investigación tuvo como **antecedentes de nivel internacional** los siguientes: en su estudio, García (2021) implementó un colaborador virtual para la atención al ciudadano sobre trámites. El objetivo fue desarrollar un sistema de inteligencia artificial conversacional para optimizar los procesos de información. La metodología incluyó el análisis de requerimientos, diseño, programación y evaluación del chatbot. Los instrumentos utilizados fueron entrevistas, observación y registros de interacción. Antes de la implementación, el 50% de los ciudadanos expresaron insatisfacción con los tiempos de respuesta. Después de implementar el chatbot, este porcentaje se redujo al 15%, con un 70% de consultas resueltas correctamente. Se concluye que los asistentes virtuales son una solución segura para perfeccionar la atención al ciudadano y optimizar los procesos de información.

Campos (2022) examinó la satisfacción del usuario en una clínica ambulatoria de un médico general con el objetivo de evaluar la calidad de la atención percibida por los pacientes. La metodología empleada consistió en encuestar a una muestra de 200 pacientes utilizando cuestionarios de satisfacción como instrumentos de medición. Según los resultados, el 60% de los pacientes reportaron estar satisfechos con la atención recibida, mientras que el 40% restante expresó sentir insatisfacción. Como conclusión, se señaló la presencia de áreas de mejora en la atención al beneficiario, como la disminución de los tiempos de espera y la mejora de la comunicación entre médicos y pacientes.

Caballero (2021) desarrolló un chatbot basado en inteligencia artificial con el objetivo de optimizar y perfeccionar los procesos de atención al usuario en un centro de soporte, buscando así brindar una experiencia más satisfactoria a los usuarios. La

metodología incluyó el diseño, programación y evaluación del chatbot utilizando técnicas de procesamiento de lenguaje natural. Los instrumentos utilizados fueron registros de interacción y encuestas de satisfacción. Antes de la implementación, el 55% de los clientes expresaron insatisfacción con los tiempos de espera. Después de implementar el chatbot, este porcentaje se redujo al 30%, con un 65% de consultas resueltas correctamente. Se concluye que los chatbots son un instrumento seguro para optimizar la complacencia del usuario y mejorar los métodos de soporte.

Aguilar y Hernández (2020) propusieron un asistente virtual utilizando software independiente para optimizar la comunicación en una institución. El objetivo fue desarrollar un chatbot que facilite la interacción entre la institución y sus usuarios. La metodología incluyó el análisis de requerimientos, diseño, implementación y pruebas del sistema. Los instrumentos utilizados fueron entrevistas, observación y registros de interacción. Antes de la implementación, el 60% de los usuarios expresaron insatisfacción con los tiempos de respuesta. Después de implementar el chatbot, este porcentaje se redujo al 25%, con un 75% de consultas resueltas correctamente. Se concluye que los asistentes virtuales son una alternativa viable y rentable para mejorar el servicio al usuario.

Manjarrés y Echeverri (2020) crearon un asistente virtual académico que utiliza tecnología de procesamiento del lenguaje natural para ayudar a los estudiantes con sus tareas escolares. Parte del proyecto fue el diseño, implementación y evaluación de un chatbot asentado en conocimiento artificial. Se utilizaron encuestas y registros de interacción como herramientas de medición. Los resultados expusieron que, antes de la ejecución del asistente virtual, los estudiantes dedicaban un promedio de 2 horas diarias a tareas administrativas, tiempo que se redujo en un 30% tras la introducción del chatbot, logrando un 85% de satisfacción entre los estudiantes. El estudio concluyó que los asistentes virtuales son herramientas efectivas para mejorar la experiencia estudiantil y optimizar los procesos académicos.

Lanzagorta et al. (2022) realizaron una investigación sobre el estado actual y las perspectivas futuras de la IA en medicina. El objetivo fue investigar cómo la IA podría afectar varios campos de la salud, como el diagnóstico, el tratamiento y la gestión de pacientes. La metodología residió en una exploración absoluta de la literatura científica. Los instrumentos utilizados fueron bases de datos especializadas

y artículos científicos. Los resultados mostraron que la IA ha mejorado la precisión diagnóstica en un 20-30% en comparación con métodos tradicionales, ha optimizado los procesos de gestión hospitalaria reduciendo tiempos de espera en un 15%, y ha permitido una atención más personalizada a los pacientes. Se concluye que la IA tiene un enorme potencial para transformar la práctica médica, mejorando la atención médica y los resultados en salud.

Guijarro (2020) desarrolló un asistente virtual para un sistema de información. El objetivo fue optimizar la práctica del usuario en la interacción con el sistema. Utilizó un enfoque de ingeniería de software y desarrolló un prototipo de asistente virtual. Los datos indicaron que el asistente virtual acrecentó la complacencia de los usuarios en un 65 % y disminuyó el tiempo de respuesta en un 52 %. Concluyó que los asistentes virtuales son una solución efectiva para optimizar la interacción entre los sistemas de información y los usuarios.

Varela et al. (2022) diseñaron un asistente virtual interactivo basado en inteligencia artificial conversacional. El objetivo fue optimizar la interacción y la práctica del usuario. Utilizaron una investigación exploratoria y descriptiva, así como el modelo incremental para el desarrollo web. Se utilizaron entrevistas, encuestas a compradores en línea y empleados de Juls Store para recopilar datos. Los resultados indicaron que el 90% de los compradores consideran transcendental el asistente virtual para recibir atención 24/7, y el 100% del personal de Juls Store se mostró satisfecho con el desarrollo del aplicativo. Concluyeron que los asistentes virtuales con IA conversacional son una solución efectiva para optimizar la intercomunicación entre los usufructuarios y los sistemas de información.

A nivel **nacional**, Flores y Lozano (2023) desarrollaron un asistente virtual con inteligencia artificial destinado a mejorar el cuidado del paciente. El propósito del estudio fue evaluar el impacto de los asistentes virtuales en la satisfacción de los pacientes. El estudio utilizó un enfoque cuantitativo y un diseño puramente experimental, con una muestra de 28 pacientes. Los resultados mostraron que el promedio de satisfacción del paciente antes de la implementación del asistente virtual (pretest) fue del 31,07%, y el promedio de satisfacción del paciente después de la implementación y postest fue del 43,68%. Esto representa un aumento en la satisfacción de aproximadamente el 12%. Los investigadores concluyeron que el

asistente virtual es un instrumento eficaz para optimizar la disposición de la atención, oprimir los tiempos de espera y, en consecuencia, aumentar la satisfacción del paciente.

Rojas et al. (2023) realizaron una revisión sistemática sobre el impacto de la inteligencia artificial en el progreso de la atención al usuario. El objetivo fue analizar cómo la IA puede optimizar la experiencia del cliente. Revisaron 40 estudios y encontraron que la implementación de asistentes virtuales con IA mejora la satisfacción del cliente en un 68% y reduce los tiempos de respuesta en un 55%. Concluyeron que la IA es una tecnología clave para optimar la eficacia de la atención al usuario en las organizaciones.

Ramírez (2021) desarrolló un asistente virtual para mejorar la atención al cliente en la consultoría de derecho laboral de la empresa Abogados Romero. Los registros de satisfacción y servicio al cliente recopilados durante un período de 45 días se analizaron utilizando métodos de investigación aplicada con métodos cuantitativos y diseño experimental. Los resultados mostraron que la implementación del asistente virtual redujo significativamente el tiempo de atención al cliente en un 40 % y aumentó la satisfacción del cliente en un 35 %. Estos hallazgos indican que el asistente virtual no solo mejoró la eficiencia operativa al acelerar el proceso de atención, sino que también incrementó la percepción positiva de los clientes hacia el servicio ofrecido por Abogados Romero, subrayando así su eficacia como herramienta para optimizar la atención al cliente en el ámbito legal laboral.

Artica (2020) implementó un sistema de asistente virtual en Electrocentro S.A. de Huancayo para mejorar la eficiencia y la satisfacción en la atención al cliente. Se empleó un enfoque de desarrollo de software y se desarrolló un modelo de asistente virtual. Según los hallazgos, el asistente virtual aumentó la satisfacción del cliente en un 71% y redujo los tiempos de respuesta en un 59%. La investigación concluyó que los asistentes virtuales representan una solución efectiva para optimizar la atención al cliente en empresas de servicios.

Mantilla y Gamarra (2023) desarrollaron un modelo de Asistente Personal Virtual empleando Inteligencia Artificial, implementado en las oficinas de TI del Tramo Estatal Peruano con el fin de mejorar la eficacia en la atención al ciudadano. La metodología aplicada incluyó una indagación aplicada con enfoque cuantitativo y

diseño experimental, utilizando registros de atención ciudadana durante un periodo de 30 días como muestra. Se utilizaron fichas de registro de atención y satisfacción ciudadana como instrumentos de medición. Los resultados demostraron que el uso del Asistente Virtual redujo el tiempo de atención a los ciudadanos en un 35% y aumentó la satisfacción en un 25%. Se concluye que este prototipo de asistente personal virtual basado en IA es una herramienta útil para optimizar los procesos de atención al ciudadano en las oficinas informáticas del sector público peruano.

Casazola et al. (2021) llevaron a cabo una revisión de la literatura enfocada en cómo los chatbots son útiles para los clientes dentro de las organizaciones. El objetivo fue examinar el impacto de la usabilidad de los chatbots en la satisfacción del cliente. El estudio adoptó un enfoque cualitativo y revisó 30 investigaciones. Los resultados demostraron que los chatbots que ofrecen una alta usabilidad percibida pueden incrementar la satisfacción del cliente hasta en un 72%. Los investigadores concluyeron que los chatbots representan un instrumento seguro para optimizar la atención al usuario, siempre y cuando sean diseñados con un enfoque centrado en la usabilidad.

De la Cruz (2022) creó un Asistente Virtual basado en la inteligencia artificial con el fin de facilitar el estudio para los alumnos de cuarto grado de la escuela de ingeniería con el objetivo de proveer asistencia y guía durante la elaboración de la tesis. El procedimiento utilizado fue un enfoque de cantidad y un diseño experimental, utilizando una muestra de cincuenta alumnos de Ingeniería. Se utilizaron interrogantes de satisfacción y pruebas de desempeño en el ámbito académico como herramientas para recolectar información. Las conclusiones evidenciaron que la utilización del Asistente virtual tuvo como consecuencia un incremento del 30% en la satisfacción de los alumnos y una mejoría del 20% en la elaboración de sus tesis. Se consideró que el Asistente Virtual basado en tecnología inteligente es una ayuda valiosa para los alumnos de primer año de la carrera de Ingeniería y que representa una ayuda para los investigadores.

Por otro lado, la presente investigación tuvo como **teorías y marco conceptual** lo siguiente:

Asistente virtual:

Adarsh (2022) en su artículo científico menciona que los asistentes virtuales son programas de software diseñados para llevar a cabo tareas o servicios en función de las órdenes o preguntas que los usuarios les proporcionan. Su objetivo principal es proporcionar una interfaz de lenguaje natural que facilite la comunicación entre seres humanos y computadoras. Dentro de esta categoría, los asistentes virtuales inteligentes (IVA) y los asistentes personales inteligentes (IPA) se destacan por su capacidad de interpretar el habla humana y responder utilizando voces sintetizadas. Estos asistentes pueden desempeñar una variedad de funciones, como responder preguntas, controlar dispositivos de automatización del hogar, gestionar el correo electrónico y el calendario, entre otros. Los usuarios pueden acceder a estos asistentes virtuales a través de diversas plataformas, incluyendo programas de chat en línea. Algunos asistentes virtuales están diseñados principalmente para el entretenimiento, mientras que otros se han desarrollado con propósitos prácticos y utilitarios.

Asimismo, Uikey (2022) señala que un asistente virtual para comandos de Internet y PC es un programa de aplicación que opera mediante el reconocimiento de voz y realiza tareas en nombre del usuario. Su principal objetivo es simplificar la interacción con una computadora, particularmente para aquellos sin experiencia técnica. Este asistente se enfoca en la ejecución de comandos tanto en el entorno de escritorio como en la web, y puede ser instalado como una aplicación de software en el ordenador del usuario, aprovechando tecnologías de inteligencia artificial y Python, con un enfoque especial en el reconocimiento de voz. Entre las ventajas que ofrece este asistente virtual a los usuarios de PC se encuentra la eliminación de la necesidad de escribir comandos manualmente, permitiendo a los usuarios dar órdenes por voz.

Además, proporciona una interfaz de usuario amigable, especialmente útil para aquellos que no están familiarizados con términos técnicos como el CMD. Esto no solo facilita la interacción, sino que también mejora la productividad al ejecutar comandos y completar tareas de manera eficiente en nombre del usuario. En conclusión, el progreso de un ayudante virtual para comandos de Internet y PC ofrece a los usuarios una forma conveniente y efectiva de interactuar con sus computadoras. Aprovechando la inteligencia artificial y el reconocimiento de voz, este asistente

simplifica la emisión de comandos y la ejecución de tareas, especialmente para aquellos sin conocimientos técnicos. Elimina la necesidad de entrada manual y proporciona una interfaz amigable, lo que contribuye a una mayor productividad y facilidad de uso.

Inteligencia Artificial:

Este es un campo multidisciplinario enfocado en la capacidad de las máquinas para copiar la inteligencia humana, permitiéndoles pensar y actuar de manera similar a los seres humanos. Xiumei (2022), en su artículo "Artificial Intelligence," indica que aunque la popularidad de la IA ha aumentado recientemente, su definición y comprensión no están completamente claras. En su esencia, la IA se fundamenta en la emulación de acciones humanas, pero abarca además actividades inteligentes que incluyen el procesamiento de datos, computación en la nube, internet de los objetos (IoT) y la robótica. La IA puede ser categorizada en varios niveles y capacidades.

Niveles de inteligencia artificial

Manish y Bilal (2022) describen tres niveles de inteligencia artificial: la IA limitada, que se centra en tareas específicas; la IA general, que busca aprender y razonar de manera similar a los humanos; y la superinteligencia artificial, que podría superar las capacidades cognitivas humanas. Además, la IA puede ser clasificada según su capacidad de memoria, desde robots reactivos hasta sistemas que utilizan la teoría de la mente para comprender necesidades y emociones humanas. El objetivo último es alcanzar la inteligencia artificial autoconsciente, que poseería conciencia, emociones, deseos y metas propias.

Aplicaciones y categorías de la inteligencia artificial

Chen (2023) enfatiza el uso de algoritmos y técnicas de aprendizaje automático por parte de la inteligencia artificial para analizar grandes cantidades de datos, aprender de ellos y tomar decisiones informadas. La utilización de la tecnología de la inteligencia artificial se extiende a diversas áreas como la sanidad, las financias, el transporte y la educación, aumentando la capacidad y precisión de la toma de decisiones. La IA se divide en IA restringida, diseñada para tareas específicas, e IA general, que intenta replicar la inteligencia humana en varios dominios.

OpenIA

La organización líder en investigación de inteligencia artificial (IA) que se dedica a desarrollar y promover IA amigable para garantizar que los beneficios de esta tecnología sean compartidos por toda la humanidad. Fundada en diciembre de 2015, OpenAI ha sido pionera en el campo de la IA, enfocándose en crear modelos avanzados de aprendizaje automático y redes neuronales, así como en fomentar un enfoque ético y responsable en el desarrollo de la IA. De acuerdo con Brockman (2023), uno de los objetivos fundamentales de OpenAI es "avanzar en la inteligencia digital de manera que sea más segura y útil, promoviendo la cooperación entre las personas y las máquinas" (p. 1). Este enfoque no solo subraya la importancia de la seguridad en el desarrollo de IA, sino que también destaca la intención de OpenAI de asegurar que la IA se utilice de manera ética y equitativa. OpenAI ha desarrollado una serie de modelos influyentes, siendo uno de los más notables GPT (Generative Pre-trained Transformer), que ha revolucionado la forma en que se entiende y utiliza el procesamiento del lenguaje natural. Los modelos de GPT han demostrado capacidades impresionantes en tareas como la generación de texto, la traducción automática y la respuesta a preguntas, lo que ha llevado a aplicaciones prácticas en diversos campos, incluyendo la atención médica, la educación y la investigación científica. La contribución de OpenAI al campo de la inteligencia artificial no solo radica en sus innovaciones tecnológicas, sino también en su compromiso con la ética y la transparencia. Al publicar sus investigaciones y colaborar con otras instituciones, OpenAI busca crear un entorno en el que la inteligencia artificial pueda desarrollarse de manera abierta y accesible, fomentando una cultura de cooperación y responsabilidad en el uso de esta poderosa tecnología.

Falcón LLM

Teoría del Falcon LLM: El Falcon LLM se fundamenta en la arquitectura de transformadores, que provocó una revolución en el procesamiento del lenguaje natural. Las relaciones complejas entre las palabras en secuencias de texto son capturadas por los mecanismos de atención utilizados en esta arquitectura, permitiendo un procesamiento más eficiente y efectivo del lenguaje. "Los transformadores han demostrado ser particularmente efectivos para modelar

dependencias a largo plazo en datos secuenciales, superando a arquitecturas previas en una variedad de tareas de NLP" (Vaswani et al., 2017).

Definición del Falcon LLM: El Falcon LLM es un modelo de lenguaje de gran escala desarrollado por el Technology Innovation Institute, diseñado para comprender y generar lenguaje natural en diversas aplicaciones. Destaca por su habilidad para procesar y producir texto de forma coherente y relevante en contexto, aprovechando extensos conjuntos de datos para su entrenamiento, demostrando capacidades comparables a modelos mucho más grandes en diversos benchmarks de NLP" (Technology Innovation Institute, 2023)

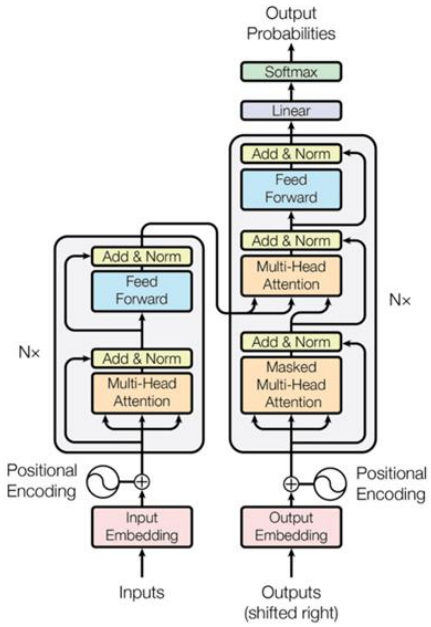


Figura 1. *Arquitectura de decodificador (Falcón LLM)*

Fuente*:* Cornell University

PLN (Procesamiento de Lenguaje Natural)

El proceso de la expresión natural (PLN) es un campo que combina lingüística, informática e comprensión artificial para computarizar la comprensión y producción del lenguaje humano. Según Németh y Koltai (2023), la aplicación del PLN en sociología, especialmente en el análisis cuantitativo de textos, presenta diversos retos y posibilidades. Uno de los desafíos principales es la falta de establecimiento del análisis de textos digitales como práctica institucionalizada en las ciencias sociales.

Sin embargo, la colaboración entre científicos informáticos y sociólogos ha dado lugar a iniciativas exitosas y ampliamente citadas, lo que demuestra el potencial del enfoque interdisciplinario.

Chandrasekaran (2023) destaca que el PLN, junto con el aprendizaje automático, facilita la extracción de información crítica de artículos epidemiológicos, mejorando la entrada de datos y asignando puntuaciones relevantes. La evolución del PLN se ha desarrollado en tres fases: empirismo, racionalismo y aprendizaje profundo. El uso de modelos como BERT ha mostrado una alta precisión en tareas como el análisis de sentimientos y la clasificación de secuencias, incluso detectando ciberacoso en Twitter con una mejora del 5% en la precisión.

El PLN encuentra aplicaciones en diversos campos, como el reconocimiento de gestos en lenguaje de señas, la traducción de texto mediante procesamiento de imágenes y la predicción del habla. La alineación de textos, influenciada por factores como la procedencia y el tema, es un proceso fundamental en el PLN. En conjunto, el PLN ha evolucionado como un campo de investigación interdisciplinario, abarcando áreas como la interacción persona-computadora, la inteligencia artificial, la lingüística cuantitativa y la traducción automática, transformando la manera en que interactuamos con el lenguaje y los datos.

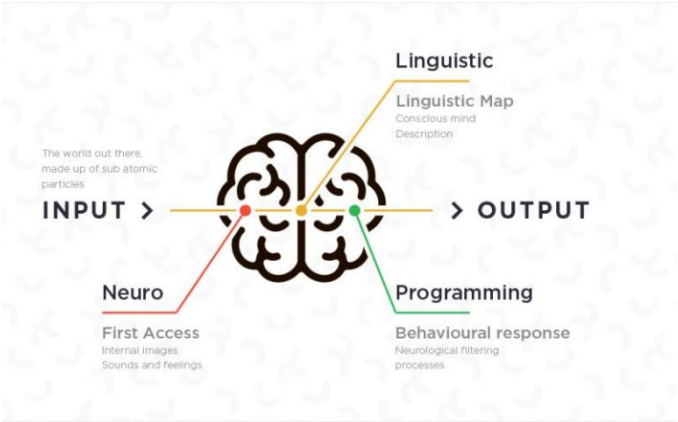


Figura 2: *Como funciona un PLN*

Fuente: NLP Academy

Arquitectura del Asistente virtual

Los Asistentes de Escritorio Virtual (VDA) están diseñados para ayudar a los usuarios en una variedad de actividades, como la navegación en sistemas informáticos, el acceso a archivos y aplicaciones, y la programación de reuniones. Según Patil (2022), estos asistentes ofrecen servicios avanzados como el aprendizaje automático y el procesamiento del lenguaje natural y la toma de decisiones inteligentes. Los VDA automatizan tareas rutinarias, realizándolas de manera más eficiente que los humanos, lo que ahorra tiempo y aumenta la productividad. Además, mejoran la precisión en tareas que requieren alta exactitud, reduciendo el riesgo de errores comunes.

Para mejorar la experiencia del usuario, los VDA proporcionan una interfaz conversacional utilizando el procesamiento del lenguaje natural, lo que hace que la interacción con las computadoras sea más intuitiva y sencilla. Hye (2020) complementa esta visión al describir los dispositivos asistentes virtuales como aquellos que incorporan una pantalla y procesadores para brindar una interfaz descriptiva de usuario (GUI). Esta GUI muestra datos relacionados con diversos dispositivos, así como etiquetas y controles de entrada, proporcionando información útil cuando el dispositivo no reconoce un comando verbal y ofreciendo instrucciones sobre cómo enseñar al dispositivo a ejecutar la acción deseada.

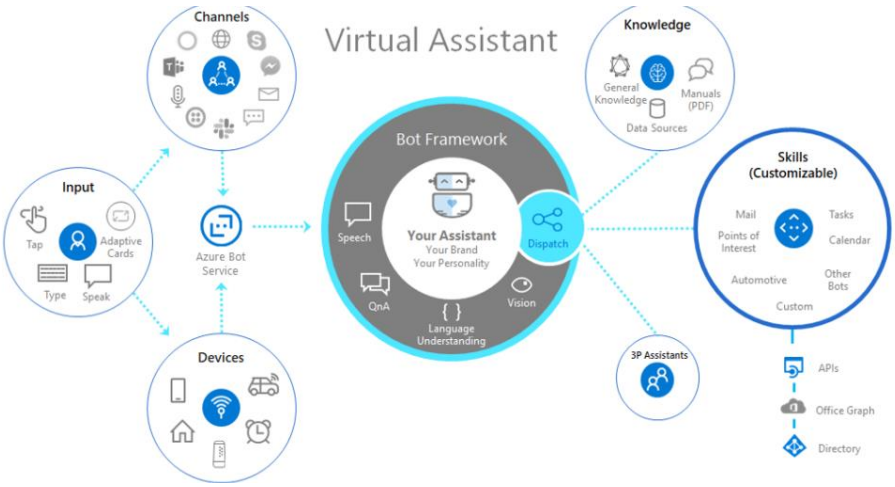


Figura 3. *Arquitectura de un Virtual Assistant*

Fuente: Microsoft Learning

Metodologías ágiles

Las metodologías de mejora de software ligero, según Sankhe (2022), tienen como objetivo adaptarse a los cambiantes requisitos de los clientes y asegurar su satisfacción en los proyectos de desarrollo. Scrum y Extreme Programming son ejemplos comunes de estas metodologías, utilizadas por diversas empresas en proyectos de diferentes escalas.

Un elemento clave de las metodologías ágiles es la importancia dada a la colaboración y retroalimentación continua de los usuarios a lo largo del proceso de progreso. Esta interacción constante proporciona flexibilidad y la capacidad de adaptarse a necesidades que pueden cambiar con el tiempo. Las metodologías ágiles colocan al usuario en el foco del proceso de desarrollo, garantizando que sus expectativas y requisitos sean factores cruciales en la exitosa creación e implementación de productos de software.

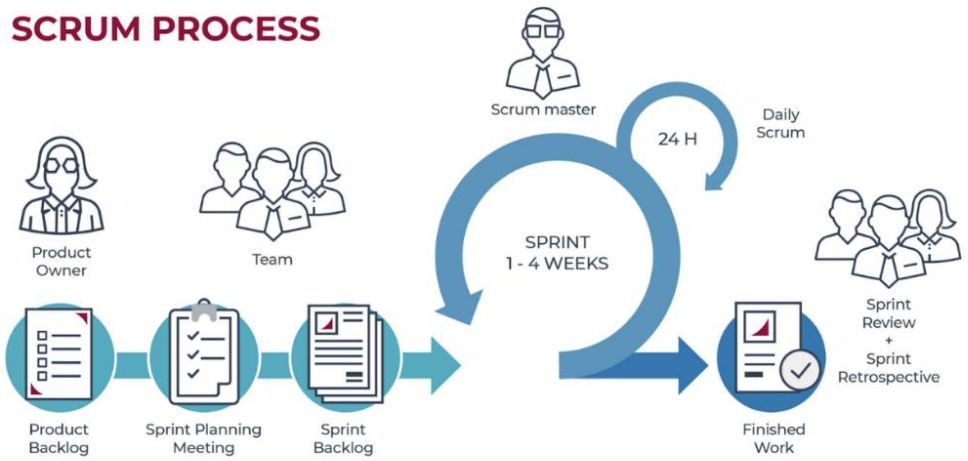


Figura 4. *Scrum Process*

Fuente: Agile System Development Structure

Satisfacción del paciente

La satisfacción del paciente se describe a la percepción general que tiene el paciente sobre su experiencia al programar y asistir a una consulta médica. Este concepto engloba varios aspectos, como la facilidad de uso del sistema de programación, la eficiencia y rapidez del proceso, la claridad de la información transmitida, la calidad de la atención brindada y la conversación admitida, además de la totalidad de personas que conforman el equipo. Es un parámetro fundamental para determinar la calidad de los servicios de salud desde la perspectiva del paciente y representa la capacidad del sistema para atender las necesidades y expectativas de los usuarios. De acuerdo a Batbaatar et al. (2017), la comodidad del paciente es de suma importancia como parámetro de la calidad de la asistencia médica y se reconoce cada vez más como un resultado válido del encuentro clínico, subrayando su importancia como indicador clave del rendimiento del sistema de salud.

Facilidad de Uso: se refiere a lo simple y accesible que resulta el proceso de programación de citas médicas para el paciente. Según Zhang et al. (2020), la facilidad de uso en los sistemas de programación de citas médicas es transcendental para optimizar la experiencia del usuario y acrecentar la eficiencia operativa en los entornos de atención médica.

Tiempo y Eficiencia: Engloba la rapidez del proceso de programación y la eficacia del sistema en manejar las solicitudes de los pacientes. "La eficiencia en la programación de citas médicas no solo reduce los tiempos de espera, sino que también mejora la satisfacción del paciente y optimiza la utilización de los recursos sanitarios" (Cayirli y Veral, 2019).

Claridad de la Información: Se refiere a la precisión y comprensibilidad de la información proporcionada al paciente sobre su cita médica. "La claridad en la comunicación de los detalles de la cita médica es crucial para reducir la ansiedad del paciente y mejorar la adherencia a las visitas programadas" (Gupta y Denton, 2018).

Atención y Comunicación: Se refiere a la aptitud de la interacción entre el usuario y el personal durante la programación de citas. Según Street et al. (2019), una comunicación empática y efectiva durante este proceso es transcendental para mejorar la satisfacción del usuario y instituir una correspondencia de confianza con el proveedor de salud.

Por otro lado en cuanto a la **hipótesis** general de la investigación se tiene la siguiente: Hi: El asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología; Ho: El asistente virtual con inteligencia artificial no optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología.

II. METODOLOGÍA

En cuanto al **tipo** de investigación se consideró de tipo aplicada, siguiendo la definición de Rus (2020), que se enfoca en el diseño de modelos para resolver problemas específicos o satisfacer necesidades de comunidades o entidades particulares. Este enfoque también se centra en abordar problemas prácticos y utilitarios que afectan a la sociedad.

En relación al **enfoque** de investigación fue cuantitativo. Según la definición de Ortega (2022) describió la investigación cuantitativa como un enfoque sistemático para recopilar y analizar datos de variadas fuentes. Para medir y cuantificar el problema de investigación, este método utiliza operaciones y estadísticas.

El **diseño** de investigación adoptado fue pre experimental, específicamente se empleó un diseño de pretest y postest, según la definición proporcionada por Velázquez (2022), Este método implica llevar a cabo una prueba inicial para evaluar la variable dependiente. Después, se implementa el tratamiento experimental y, finalmente, se lleva a cabo una evaluación posterior para evaluar el impacto del tratamiento en la variable dependiente. Además, Meza (2022) indica que estos diseños se llevan a cabo con un único grupo y, en algunos casos, pueden ser realizados incluso con un solo individuo, como en el argumento de estudios de caso con una sola medición, manteniendo su esencia como diseño pretest o postest.

Esquema del diseño preexperimental

G_E: O_1 - X - O_2

Dónde:

G_E = 42 pacientes del INO

O_1 = Pretest de la variable satisfacción del paciente

X = Asistente virtual con IA

O_2= Postest de la variable satisfacción del paciente

En cuanto a las **variables** de estudio, se tienen las siguientes:

Para la primera **variable independiente**, Según la investigación de Kiran (2023), un asistente virtual con inteligencia artificial (IA) es un programa que puede interactuar de forma autónoma con los usuarios utilizando tecnologías avanzadas como el procesamiento del lenguaje natural (PLN) para comprender comandos y preguntas. La IA facilita el aprendizaje automático, lo que permite que el asistente se adapte y mejore con el tiempo. Además de reconocer voz, puede realizar diversas funciones como enviar mensajes, buscar información en internet y controlar dispositivos inteligentes.

En cuanto a la **variable dependiente**, la satisfacción del paciente, según la definición de Unir (2024), se refiere a la medida en que la atención sanitaria ha cumplido con las expectativas del usuario, siendo fundamental para optimizar los servicios de salud y la calidad asistencial.

En relación a la **población** estudiada, esta estuvo compuesta por 5,212 pacientes atendidos mensualmente durante el año 2023, siguiendo con la definición de Laura (2021), se entiende como población a un conjunto de elementos, ya sea finito o infinito, que comparten una o varios atributos específicos determinados por la problemática y el propósito de la indagación.

El tamaño de la **muestra** se determinó seleccionando 42 pacientes del INO, siguiendo los criterios establecidos por Laura (2021), menciona que la muestra se define como un subgrupo limitado y representativo, seleccionado de manera calculada a partir de la población disponible. Para asegurar la fiabilidad de una muestra, es imperativo calcular su tamaño mediante métodos matemáticos y oprimir al pequeño margen de error.

$$T.M = (Z^2 * N) / (Z^2 + 4N (EE)^2)$$

Donde:

T.M = Tamaño de la muestra

Z = 1.96 (Nivel de confianza al 95%)

N = 5,212 (Población total del estudio)

EE = 0.15 (Error estimado al 15%)

Sustituyendo los valores:

T.M = (1.96^2 * 5,212) / (1.96^2 + 45,212$(0.15)^2$)

T.M = (3.8416 * 5,212) / (3.8416 + 45,2120.0225)

T.M = 20,022.3392 / (3.8416 + 469.08)

T.M = 20,022.3392 / 472.9216

T.M = 42.34

Redondeando al número entero más cercano: **T.M = 42**

Por otro lado, Se utilizó un **muestreo** probabilístico, tal como lo define Baena (2018) como un método en el que ciertos elementos de una población son selección para simbolizar la suma de la población. La principal ventaja del museo se basa en la posibilidad de obtener información sobre poblaciones extensas de manera económica, viable y rápida. Para este estudio se utilizará específicamente el muestreo aleatorio simple.

En cuanto a la **técnica** empleada en este estudio, se recurrió a la encuesta para valorar el nivel de satisfacción de los pacientes, siguiendo la definición proporcionada por Salas (2020). La encuesta consiste en administrar un cuestionario a un modelo de individuos para recabar información acerca de sus sentires, actitudes y procedimientos.

Para evaluar la satisfacción del paciente, se empleó un cuestionario, según la definición de Díaz (2023). Este **instrumento** de recolección de fundamentos consiste en una serie de preguntas diseñadas para recopilar información de los participantes en el estudio.

En cuanto la **validez** del instrumento fue evaluada mediante el juicio de especialistas, se contó con la colaboración de tres especialistas expertos en el área, quienes revisaron tanto la estructura del instrumento como la redacción de sus ítems, asegurándose de que fueran coherentes con los indicadores y dimensiones definidos por Parra (2020). La validación del instrumento de investigación implicó evaluar las preguntas de la encuesta para asegurar su confiabilidad.

En cuanto a la evaluación de la **confiabilidad** del instrumento de satisfacción del paciente en el marco de esta investigación, Se empleó un nivel de confianza del 95% en los análisis estadísticos. Según Ñaupas y colaboradores (2018), un instrumento se considera confiable cuando las mediciones no muestran cambios significativos al aplicarlo en diferentes momentos o a diferentes personas. En resumen, si una evaluación realizada hoy arroja resultados similares a los obtenidos un mes después, se concluye que el instrumento es confiable. La confiabilidad se evidencia en la consistencia de los resultados a lo largo del tiempo y en diversas situaciones de aplicación del instrumento.

En cuanto al **procesamiento de datos**, se realizó una investigación donde se implementó un asistente virtual y la muestra pudo interactuar con el asistente virtual. Se utilizó el programa SPSS 26 para realizar un análisis estadístico después de recopilar los datos en una hoja de cálculo de Excel. En el análisis de datos, se calcularon las frecuencias utilizando un método descriptivo, cuyos resultados se mostraron en tablas y figuras. Además, se realizó un estudio inferencial para verificar la hipótesis sugerida. Según Ortega (2022), el procesamiento de la información se adaptó a las necesidades específicas, tipos de datos recopilados, disponibilidad de tiempo y otros factores relevantes.

En cuanto al **análisis de los datos**, se ejecutó la valoración de normalidad de Shapiro-Wilk en las puntuaciones de la variable de satisfacción del paciente. Sin embargo, al utilizar la prueba de Wilcoxon, que es un método no paramétrico, se descubrió que los datos no siguen una distribución típica. Blanca (2023) afirma que la Prueba de Shapiro-Wilk es una herramienta crucial en estadística para determinar si un conjunto de datos es normal. Sin embargo, Deng y Qin (2022) explican que la prueba de Wilcoxon se utiliza para comparar el rango medio de dos muestras relacionadas y encontrar diferencias potenciales entre ellas.

En este estudio, se han seguido estrictamente los **aspectos éticos** para preservar la privacidad de los participantes, y para referenciar adecuadamente a los autores según las pautas de las normas ISO 690. Además, no se han encontrado evidencias de plagio en los datos recopilados, por otro lado, los capítulos fueron redactados siguiendo las directrices de la guía de investigación de la institución. Según la definición de Ortega (2023), La ética en este contexto abarca los principios

y valores morales que guían la labor académica, garantizando que la indagación se lleve a cabo de modo responsable, íntegro y sin causar daño a los individuos ni a la comunidad en su conjunto.

III. RESULTADOS

Tabla 1. *Niveles de satisfacción según las dimensiones del paciente del Instituto Nacional de Oftalmología antes de la aplicación del asistente virtual.*

Nivel de logro	Facilidad de Uso		Claridad de la Información		Tiempo de espera y eficiencia		Atención y comunicación	
	f	%	f	%	f	%	f	%
Nada satisfecho	19	45%	20	48%	20	48%	26	62%
Satisfecho	17	40%	15	36%	19	45%	12	29%
Muy satisfecho	6	14%	7	17%	3	7%	4	10%
Total	42	100%	42	100%	42	100%	42	100%

Fuente: elaboración propia.

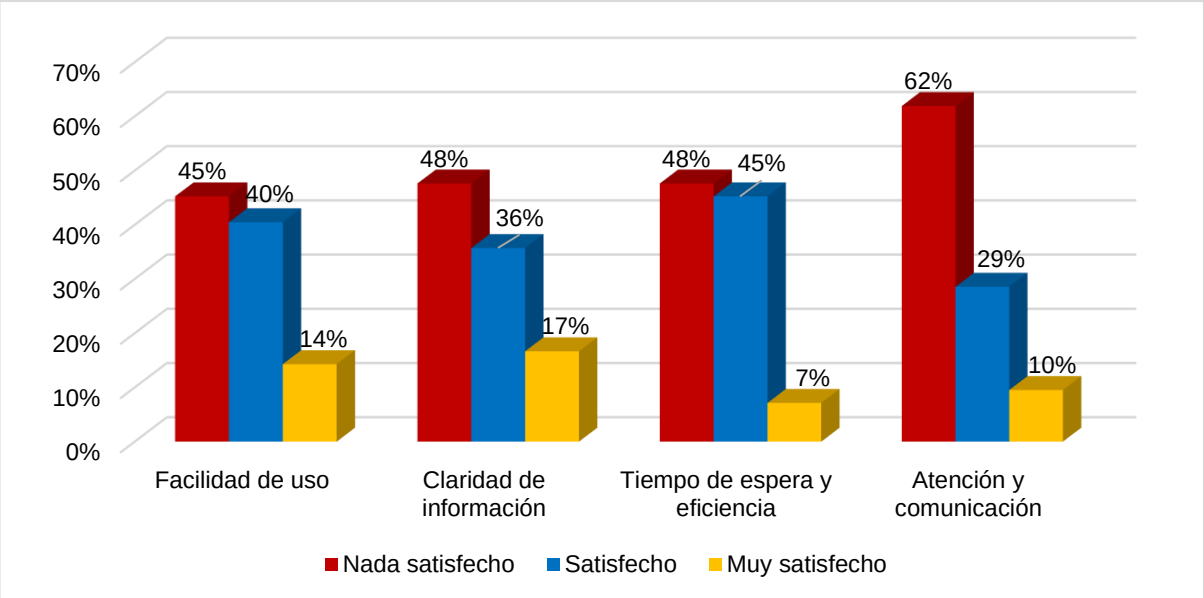


Figura 5. Niveles de satisfacción según las dimensiones del paciente del Instituto Nacional de Oftalmología antes de la aplicación del asistente virtual.

Fuente: elaboración propia.

Los datos presentados en la tabla 1 y la figura 5 muestran los niveles de las dimensiones del nivel de satisfacción de los pacientes del Instituto Nacional de Oftalmología antes de la aplicación del asistente virtual. En cuanto a la facilidad de uso, el 45% de la población está completamente insatisfecha, el 40% está satisfecha y el 14% está muy satisfecha. En cuanto a las dimensiones de claridad de la

información, el 48% de las personas dijo estar muy insatisfecha, el 36% dijo estar satisfecha y el 17% dijo estar muy satisfecha. Asimismo, en cuanto a tiempo de espera y eficiencia, el 48% se mostró completamente insatisfecho, el 45% satisfecho y el 7% muy satisfecho. Finalmente, para la dimensión atención y comunicación el 62% se mostró completamente insatisfecho, el 29% satisfecho y el 10% muy satisfecho. Esto sugiere que la mayoría de los pacientes experimentan lagunas de insatisfacción antes de utilizar un asistente virtual.

Tabla 2. *Niveles de satisfacción según las dimensiones del paciente del Instituto Nacional de Oftalmología después de la aplicación del asistente virtual.*

Nivel de logro	Facilidad de Uso		Claridad de la Información		Tiempo de espera y eficiencia		Atención y comunicación	
	f	%	f	%	f	%	f	%
Nada satisfecho	1	2%	2	5%	0	0%	0	0%
Satisfecho	4	10%	14	33%	24	57%	5	12%
Muy satisfecho	37	88%	26	62%	18	43%	37	88%
Total	42	100%	42	100%	42	100%	42	100%

Fuente: elaboración propia.

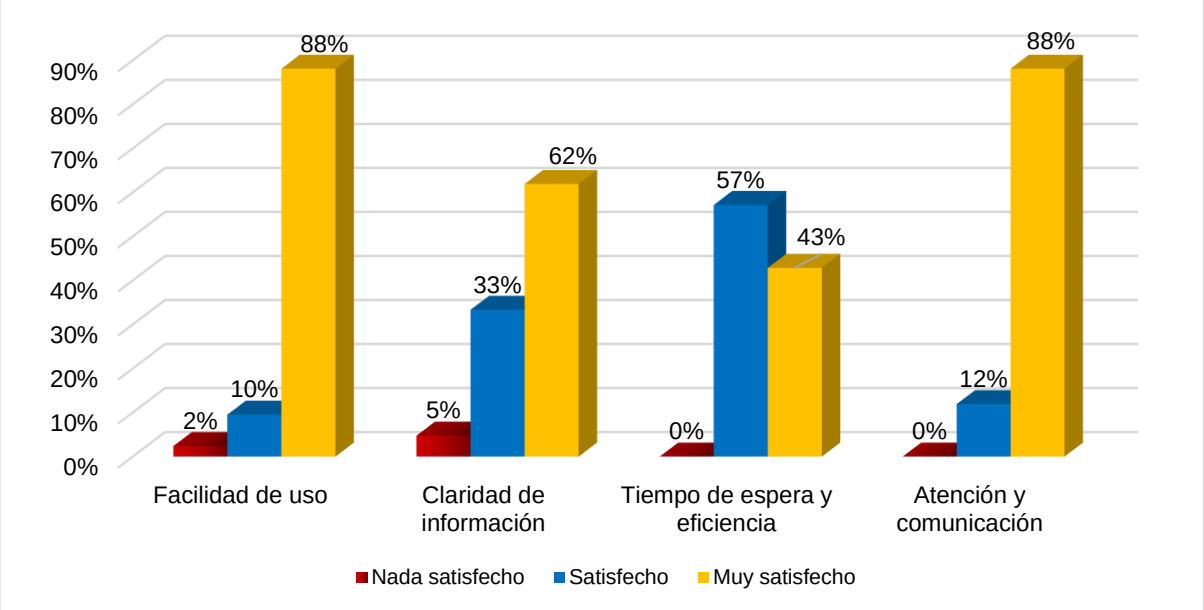


Figura 6. *Niveles de satisfacción según las dimensiones del paciente del Instituto Nacional de Oftalmología después de la aplicación del asistente virtual.*

Fuente: elaboración propia.

Los datos presentados en la tabla 2 y la figura 6 muestran los niveles de las dimensiones del nivel de satisfacción de los pacientes del Instituto Nacional de Oftalmología después de la aplicación del asistente virtual. En cuanto a la dimensión usabilidad, el 2% se mostró completamente insatisfecho, el 10% satisfecho y el 88% muy satisfecho. En cuanto a la claridad de la información, el 2% de las personas se consideran completamente insatisfechas, el 33% en un nivel satisfactorio y el 62% muy satisfechas. De manera similar, para las dimensiones de tiempo de espera y eficiencia, el 0% de las personas dijo estar completamente insatisfecha, el 57% dijo estar satisfecha y el 43% dijo estar muy satisfecha. Finalmente, en lo que respecta a las dimensiones de atención y comunicación, el 0% de las personas se encuentra en el nivel completamente insatisfecho, el 12% en el nivel satisfactorio y el 88% en el nivel muy satisfecho. Esto indica que la implementación del asistente virtual permite mejorar la satisfacción de los pacientes.

Tabla 3. *Niveles de satisfacción de los pacientes antes y después de la aplicación del asistente virtual*

Nivel de logro	Pre test		Post test	
	f	%	f	%
Nada satisfecho	18	43%	1	2%
Satisfecho	16	38%	9	21%
Muy satisfecho	8	19%	32	76%
Total	42	100%	42	100%

Fuente: elaboración propia.

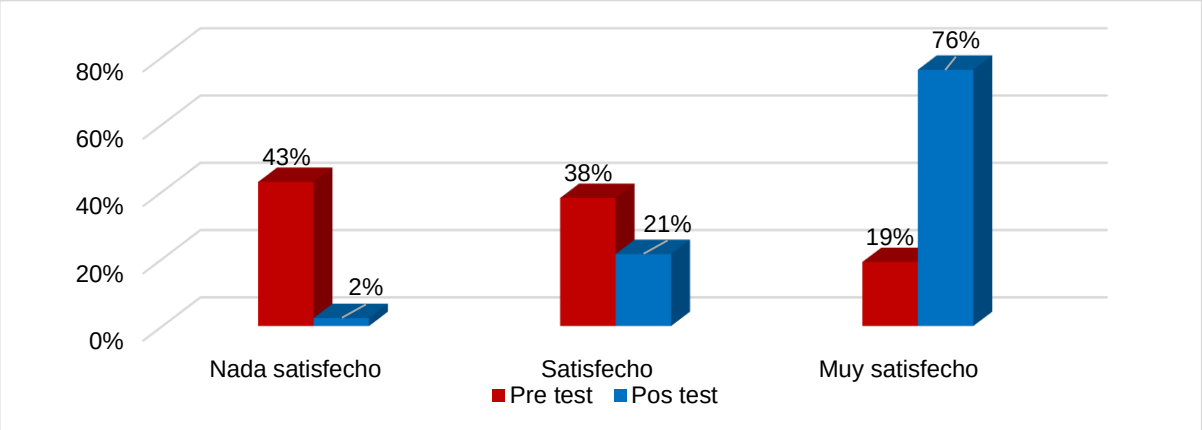


Figura 7. *Niveles de satisfacción de los pacientes antes y después de la aplicación del asistente virtual*

Los datos de la tabla 3 y la figura 7 muestran una mejora significativa en los niveles de satisfacción de los pacientes del Instituto Nacional de Oftalmología tras la implementación del asistente virtual. Antes de su aplicación, el 43% de los pacientes estaban "Nada satisfechos" y solo el 19% "Muy satisfechos". Después de la implementación, estos porcentajes cambiaron drásticamente: únicamente el 2% quedaron "Nada satisfechos", mientras que el 76% reportaron estar "Muy satisfechos". Esta transformación notable, con un aumento del 57% en la categoría de "Muy satisfechos", sugiere que el asistente virtual ha tenido un impacto positivo sustancial en la experiencia de los pacientes, mejorando significativamente la calidad percibida de los servicios ofrecidos por el instituto.

Prueba de hipótesis

Para contrastar la suposición general sobre la satisfacción del usuario antes y después de usar un asistente virtual, probamos si los datos mostraban patrones paramétricos o no paramétricos. Dado que el número de datos fue inferior a 50, se decidió utilizar la prueba de Shapiro-Wilk para el análisis de normalidad.

Regla:

Si $p_{valor} \leq 0.05$, los datos siguen una distribución no paramétrica.

Si $p_{valor} > 0.05$, los datos siguen una distribución paramétrica.

Tabla 4. *Prueba de normalidad*

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Satisfacción del paciente Pre test	,122	42	,124	,939	42	,027
Satisfacción del paciente Post test	,168	42	,005	,895	42	,001
a. Corrección de significación de Lilliefors						

Fuente: elaboración propia.

Los resultados de la Tabla 4 muestran que los valores de significancia (p) para la satisfacción del paciente tanto en el pretest como en el postest son inferiores a 0,05. Esto muestra que los datos no siguen una distribución normal. Por lo que se decidió utilizar la prueba no paramétrica de Wilcoxon.

Es así que en la prueba inferencial:

H_0: El asistente virtual con inteligencia artificial no optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología

H_1: El asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología

Para la confirmación que los análisis son correctos se aplicó el análisis mediante el P_{valor} de los resultados de la aplicación de la prueba de wilcoxon satisfacción de los pacientes.

Regla:

Si $p_{valor} \leq 0.05$, se rechaza la Hipótesis nula

Si $p_{valor} > 0.05$, se acepta la Hipótesis alterna

Tabla 5. *Análisis de la significancia de los resultados de la prueba de wilcoxon*

	Satisfacción del paciente Post test – Satisfacción del paciente Pre test
Z	-5,727[b]
Sig. asintótica(bilateral)	,000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Fuente: elaboración propia.

La tabla 5 revela que la significancia de la prueba de Wilcoxon, aplicada a la satisfacción del paciente antes y después del test, es de 0.000. Conforme a la regla de decisión, se rechaza la hipótesis nula y se acepta la hipótesis alterna. Esto demuestra que el asistente virtual con inteligencia artificial mejora la satisfacción del paciente en el Instituto Nacional de Oftalmología.

IV. DISCUSIÓN

En este capítulo, se discuten los resultados obtenidos a partir de la recolección de datos para cada uno de los objetivos planteados.

Los resultados del pre-test, antes de la implementación del asistente virtual, revelan bajos niveles de satisfacción en todas las dimensiones evaluadas. En la dimensión de facilidad de uso para acceder al sistema de gestión de citas y consultas, el 45% de los pacientes se encuentran nada satisfechos; en claridad de la información, el 48%; en tiempo de espera y eficiencia, el 48%; y en atención y comunicación, el 62%. Estos hallazgos son consistentes con los reportados por Campos (2022), quien encuentra que el 40% de los pacientes expresan insatisfacción en la consulta externa de medicina familiar, destacando la necesidad de mejorar aspectos como los tiempos de espera y la comunicación entre médicos y pacientes. De manera similar, Caballero (2021) reporta que antes de la implementación de un chatbot en un centro de soporte, el 55% de los clientes expresan insatisfacción con los tiempos de espera. Estos estudios refuerzan nuestros resultados, subrayando la importancia de abordar las deficiencias en la satisfacción del paciente en diversos aspectos de la atención médica, particularmente en áreas como la facilidad de uso de los servicios para acceder a una cita médica, la claridad de la información proporcionada, la eficiencia en los tiempos de espera y la calidad de la atención y comunicación

Los resultados post-implementación muestran una mejora sustancial en todas las dimensiones evaluadas. En la dimensión de facilidad de uso, el porcentaje de pacientes muy satisfechos aumenta del 14% al 88%; en claridad de la información, del 17% al 62%; en tiempo de espera y eficiencia, del 7% al 43%; y en atención y comunicación, del 10% al 88%. Estos resultados son comparables a los obtenidos por García (2021), quien reporta una reducción del 35% en la insatisfacción de los ciudadanos tras la implementación de un asistente virtual para trámites, con un 70% de consultas resueltas correctamente. De manera similar, Aguilar y Hernández (2020) observan que después de implementar un chatbot, el porcentaje de usuarios insatisfechos con los tiempos de respuesta se reduce del 60% al 25%, con un 75% de consultas resueltas correctamente. Estos estudios corroboran nuestros hallazgos sobre la efectividad de los asistentes virtuales para mejorar la satisfacción del usuario,

especialmente en aspectos como la facilidad de uso, la claridad de la información proporcionada, la eficiencia en los tiempos de espera y la calidad de la atención y comunicación. La mejora significativa en todas las dimensiones evaluadas sugiere que el asistente virtual implementado en el Instituto Nacional de Oftalmología es efectivo para abordar las deficiencias previamente identificadas en la satisfacción del paciente.

El análisis de los puntajes obtenidos del pre y post-test revela un cambio dramático en los niveles de satisfacción general. Antes de la implementación del asistente virtual, el 43% de los pacientes están nada satisfechos y solo el 19% muy satisfechos. Después de la implementación, estos porcentajes se invierten, con solo el 2% nada satisfechos y el 76% muy satisfechos. Esta mejora significativa se alinea con los hallazgos de Flores y Lozano (2023), quienes observan un incremento del 12% en el índice de satisfacción de los pacientes tras la implementación de un asistente virtual en el Instituto Nacional de Salud del Niño. En su estudio, la media de satisfacción de los pacientes aumenta del 31.07% antes de la implementación al 43.68% después de la implementación. Asimismo, nuestros resultados son consistentes con los de Artica (2020), quien reporta que la implementación de un asistente virtual en una empresa de Electrocentro mejora la satisfacción del cliente en un 71% y reduce los tiempos de respuesta en un 59%. La magnitud de la mejora observada en nuestro estudio, que supera incluso a la reportada en estos estudios previos, sugiere que el asistente virtual implementado en el Instituto Nacional de Oftalmología es particularmente efectivo en abordar las necesidades específicas de los pacientes y mejorar su experiencia general con los servicios ofrecidos.

La prueba de hipótesis realizada mediante el test de Wilcoxon aplicada a la satisfacción del paciente en el pre-test y post-test arroja un valor de 0.000. De acuerdo con la regla de decisión, se rechaza la hipótesis nula y se acepta la hipótesis alternativa, demostrando que el asistente virtual con inteligencia artificial mejora la satisfacción del paciente en el Instituto Nacional de Oftalmología. Además, nuestros hallazgos son coherentes con los resultados de Corales y Lozano (2023), quienes encontraron que el índice de satisfacción es menor a 0.05, lo que les permitió aceptar la hipótesis alternativa y rechazar la nula. Concluyeron que el desarrollo de un asistente virtual con inteligencia artificial ayuda a mejorar la satisfacción del paciente en el Instituto Nacional de Salud del Niño.

V. CONCLUSIONES

En la investigación propuesta, se llegaron a las siguientes conclusiones:

Primero, antes de la implementación del asistente virtual en el Instituto Nacional de Oftalmología, los niveles de satisfacción de los pacientes eran notablemente bajos en todas las áreas evaluadas. La mayoría expresó insatisfacción con la facilidad de uso del sistema de citas, la claridad de la información proporcionada, los tiempos de espera y la calidad de la atención recibida, con porcentajes significativos de insatisfacción que alcanzaron hasta el 62%.

Segundo, tras la implementación del asistente virtual, se observó una transformación significativa. Los pacientes reportaron niveles mucho más altos de satisfacción, con mejoras sustanciales en la facilidad de uso del sistema, la claridad de la información, la eficiencia en los tiempos de espera y la calidad de la atención y comunicación recibida, con porcentajes de satisfacción que alcanzaron hasta el 88%.

Tercero, el análisis comparativo entre los puntajes del pre y post-test reveló un cambio drástico en la percepción de los pacientes. La implementación del asistente virtual no solo redujo considerablemente la insatisfacción inicial, sino que también elevó significativamente los niveles de satisfacción general, reflejando una mejora notable en la experiencia del paciente.

En cuarto lugar, los hallazgos demuestran claramente que el asistente virtual con inteligencia artificial ha sido exitoso en mejorar la satisfacción de los pacientes en el Instituto Nacional de Oftalmología. Este trabajo destaca el efecto beneficioso de las tecnologías innovadoras en la optimización de la calidad de la atención médica y la experiencia general del paciente.

VI. RECOMENDACIONES

Primero, es fundamental instaurar un método de valoración continua del desempeño del asistente virtual en el Instituto Nacional de Oftalmología. Se sugiere realizar evaluaciones periódicas que abarquen la satisfacción del paciente, la eficacia operativa y la calidad del servicio proporcionado. Estos análisis no solo ayudarán a identificar áreas de mejora, sino que también permitirán ajustar el diseño y la funcionalidad del asistente para optimizar su impacto y eficacia a largo plazo.

Segundo, para futuras investigaciones en este campo, es importante incorporar indicadores específicos de satisfacción del paciente relacionados con el uso del asistente virtual. Se recomienda desarrollar estudios que profundicen en la facilidad de uso percibida por los usuarios, la claridad de la información proporcionada y la percepción general de la calidad del servicio. Un enfoque mixto que combine métodos cuantitativos, como encuestas estructuradas, con métodos cualitativos, como entrevistas en profundidad, permitirá obtener una comprensión más completa y matizada del impacto del asistente virtual en la experiencia del paciente.

Tercero, es esencial establecer un programa de mantenimiento y actualización continua del sistema del asistente virtual en el Instituto Nacional de Oftalmología. Esto implica no solo mantener la tecnología actualizada y adaptable a las necesidades cambiantes del entorno médico, sino también integrar el asistente con otros sistemas hospitalarios, como registros médicos electrónicos, para mejorar la coordinación y eficiencia del cuidado. Además, se deben implementar políticas sólidas de seguridad y privacidad que cumplan con normativas internacionales y locales, como GDPR en la Unión Europea, para proteger la información confidencial de los pacientes y garantizar la confianza en el uso del asistente virtual. Esto incluye la realización de auditorías periódicas y la monitorización de accesos para identificar posibles brechas de seguridad y asegurar un manejo ético y legal de los datos de salud.

Cuarto, Se sugiere continuar desarrollando y mejorando el asistente virtual, basándose en los resultados positivos actuales que reflejan un notable aumento en la satisfacción de los pacientes. El asistente virtual ha demostrado ser efectivo para mejorar la atención en citas médicas y consultas, reduciendo las largas colas y facilitando el acceso a fechas disponibles, proporcionando una atención más eficiente y satisfactoria para los usuarios.

REFERENCIAS

ABURTO, A., GARCÍA, E. y VELÁSQUEZ, L., 2021. Asistente virtual para el personal docente en la Universidad César Vallejo de la Ciudad de Trujillo [en línea]. Tesis de licenciatura. Trujillo: Universidad César Vallejo. [Consulta: 4 julio 2024]. Disponible en: https://repositorio.ucv.edu.pe/handle/20.500.12692/52422

AGUILAR, A. y HERNÁNDEZ, D., 2020. Propuesta de un Asistente Virtual utilizando herramientas de software libre para mejorar el proceso de comunicación [en línea]. Tesis de licenciatura. Universidad Nacional. [Consulta: 4 julio 2024]. Disponible en: http://hdl.handle.net/11056/21883

ARANA, J., COLLANTES, R y MAMANI, R, 2021. Modelo de chatbot basado en inteligencia artificial para incrementar la satisfacción del cliente en empresas de venta de alimentos, Callao 2021 [en línea]. Callao: Universidad Nacional del Callao. [Consulta: 5 julio 2024]. Disponible en: https://repositorio.unac.edu.pe/handle/20.500.12952/6087

ARTICA, E., 2020. Implementación de un sistema de virtual para atención al cliente en Electrocentro S.A. de Huancayo [en línea]. Tesis de maestría. Huancayo: Universidad Continental. [Consulta: 4 julio 2024]. Disponible en: https://hdl.handle.net/20.500.12394/8251

BLANCA, T., 2023. Prueba de Shapiro-Wilk: qué es, cómo funciona y un ejemplo real [en línea]. Trabajo Final. [Consulta: 4 julio 2024]. Disponible en: https://trabajofinal.es/prueba-shapiro-normalidad-ejemplo/

CABALLERO, R., 2021. Chatbot: una propuesta viable para la atención al cliente en el centro de soporte de la UCI. Revista Cubana de Ciencias Informáticas [en línea]. 15, 216-232. [Consulta: 4 julio 2024]. Disponible en: https://rcci.uci.cu/?journal=rcci&page=article&op=view&path%5B%5D=2289&path%5B%5D=0

CAMPOS, L., 2022. Nivel de satisfacción del usuario en la atención recibida en la consulta externa de medicina familiar de la UMF No 47 del Instituto Mexicano del Seguro Social San Luis Potosí [en línea]. Tesis de licenciatura. San Luis Potosí: Universidad Autónoma de San Luis Potosí. [Consulta: 4 julio 2024]. Disponible en: https://repositorioinstitucional.uaslp.mx/xmlui/handle/i/7600

CASAZOLA, O., et al., 2021. La usabilidad percibida de los chatbots sobre la atención al cliente en las organizaciones: una revisión de la literatura. Interfases [en línea]. 14(014), 184-204. [Consulta: 4 julio 2024]. Disponible en: https://doi.org/10.26439/interfases2021.n014.5401

CORALES, S. y LOZANO, K., 2023. Desarrollo de asistente virtual con inteligencia artificial para la atención del paciente del instituto nacional de Salud Niño [en línea]. Tesis de licenciatura. Lima: Universidad César Vallejo. [Consulta: 4 julio 2024]. Disponible en: https://repositorio.ucv.edu.pe/handle/20.500.12692/123752

DE LA CRUZ, D., 2022. Asistente virtual basado en Inteligencia Artificial como herramienta de tesis para estudiantes universitarios de la carrera de Ingeniería [en línea]. Artículo científico de licenciatura. Universidad Privada del Norte. [Consulta: 4 julio 2024]. Disponible en: https://hdl.handle.net/11537/33369

DENG, S. y QIN, J., 2022. Graduation planning analysis using WILCOXON rank sum test. Academic Journal of Humanities & Social Sciences [en línea]. 5(8), 84-89. [Consulta: 4 julio 2024]. Disponible en: https://doi.org/10.25236/AJHSS.2022.050814

DÍAZ, M., 2023. Para qué sirve la observación [en línea]. [Consulta: 4 julio 2024]. Disponible en: https://www.codimg.com/education/blog/es/para-que-sirve-la

GARCÍA, A. y LÓPEZ, M., 2023. Accesibilidad y Calidad de Atención en Servicios Oftalmológicos del Perú. Revista Peruana de Medicina Experimental y Salud Pública. 40(2), 289-301.

GARCÍA, N., 2021. Implementación de un sistema virtual (asistente virtual) para la atención al cliente sobre la información de trámites al ciudadano por medio de

interacciones conversacionales, para la Secretaría de Educación de la Ciudad de Girardot en el año 2021 [en línea]. Tesis de licenciatura. Girardot: Universidad Piloto de Colombia. [Consulta: 4 julio 2024]. Disponible en: https://repository.unipiloto.edu.co/handle/20.500.12277/11613

GARIBAY, F., 2020. Diseño e implementación de un asistente virtual (chatbot) para ofrecer atención a los clientes de una aerolínea mexicana por medio de sus canales conversacionales [en línea]. Tesis de maestría. Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación. [Consulta: 4 julio 2024]. Disponible en: https://infotec.repositorioinstitucional.mx/jspui/handle/1027/402

GUIJARRO, P., 2020. Asistente virtual para un sistema de información [en línea]. Tesis de licenciatura. Alicante: Universidad de Alicante. [Consulta: 4 julio 2024]. Disponible en: http://hdl.handle.net/10045/101934

LANZAGORTA, D., CARRILLO, D. y CARRILLO, R., 2022. Inteligencia artificial en medicina: presente y futuro. Gaceta Médica de México [en línea]. 158(Supl. 1), 17-21. [Consulta: 4 julio 2024]. Disponible en: https://doi.org/10.24875/gmm.m22000688

LAURA, J., 2021. Plan E-commerce para la mejora de la calidad de atención al cliente en una empresa editorial, Lima 2021 [en línea]. Tesis de licenciatura. Lima: Universidad Norbert Wiener. [Consulta: 4 julio 2024]. Disponible en: https://hdl.handle.net/20.500.13053/4953

MANJARRÉS, R. y ECHEVERRI, M., 2020. Asistente virtual académico utilizando tecnologías cognitivas de procesamiento de lenguaje natural. Revista Politécnica [en línea]. 16(31), 85-96. [Consulta: 4 julio 2024]. Disponible en: https://doi.org/10.33571/rpolitec.v16n31a7

MANTILLA, R., 2023. Prototipo de Asistente Personal Virtual basado en Inteligencia Artificial aplicado en las oficinas de TI del Sector Público Peruano. Revista Politécnica [en línea]. 19(34), 1-14. [Consulta: 4 julio 2024]. Disponible en: https://revistas.unitru.edu.pe/index.php/PGM/article/view/5781/5828

MEZA, F., 2022. Diseños de Investigación Cuasi experimental y Preexperimental en Psicología [en línea]. Psicología Online. [Consulta: 4 julio 2024]. Disponible en: https://franmt7.com/2022/06/24/disenos-de-investigacioncuasiexperimental-y-preexperimental-en-psicologia/

MINISTERIO DE SALUD DEL PERÚ, 2023. Informe Anual sobre Servicios Oftalmológicos en el Perú. Lima: Minsa.

ORGANIZACIÓN MUNDIAL DE LA SALUD (OMS), 2024. Global Report on Vision. Geneva: World Health Organization.

ORTEGA, C., 2022. Procesamiento de datos de investigación [en línea]. QuestionPro. [Consulta: 4 julio 2024]. Disponible en: https://www.questionpro.com/blog/es/procesamiento-de-datos

ORTEGA, C., 2023. Qué es la investigación explicativa [en línea]. QuestionPro. [Consulta: 4 julio 2024]. Disponible en: https://www.questionpro.com/blog/es/investigacion-explicativa/

PARRA, A., 2020. Pasos para validar un instrumento de investigación [en línea]. QuestionPro. [Consulta: 4 julio 2024]. Disponible en: https://www.questionpro.com/blog/es/pasos-para-validar-un-instrumento

RAMÍREZ, L., CHEN, S. y KUMAR, A., 2024. Satisfacción del paciente y adherencia al tratamiento en oftalmología: una perspectiva global. Journal of Global Health. 14(1), 010407.

RAMÍREZ, W., 2021. Asistente virtual para apoyar la atención al cliente en consultas legales de ámbito laboral en la empresa abogados Romero [en línea]. [Consulta: 4 julio 2024]. Disponible en: https://orcid.org/0000-0003-0283-9080

ROJAS, K., LÓPEZ, V. y MENDOZA, A., 2023. El impacto de la Inteligencia Artificial en la mejora de la atención al cliente: Una revisión sistémica. Innovación Y Software [en línea]. 4(2), 201-222. [Consulta: 4 julio 2024]. Disponible en: https://doi.org/10.48168/innosoft.s12.a90

VARELA, E., et al., 2022. Inteligencia Artificial Conversacional para el Diseño de un Asistente Virtual Interactivo. En: Memorias de la Décima Segunda Conferencia Iberoamericana de Complejidad, Informática y Cibernética: CICIC 2022 [en línea]. International Institute of Informatics and Cybernetics, pp. 237-241. [Consulta: 4 julio 2024]. Disponible en: https://doi.org/10.54808/CICIC2022.01.237

VELÁZQUEZ, A., 2022. Cuál es la diferencia entre población y muestra [en línea]. QuestionPro. [Consulta: 4 julio 2024]. Disponible en: https://www.questionpro.com/blog/es/diferencia-entre-poblacion-y-muestra/

ANEXOS

Anexo 3. Matriz de consistencia

<table>
<tr><td>TITULO</td><td colspan="7">Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología.</td></tr>
<tr><td rowspan="2">PROBLEMA</td><td rowspan="2">OBJETIVOS</td><td rowspan="2">HIPÓTESIS</td><td colspan="5">VARIABLES E INDICADORES</td></tr>
<tr><td colspan="4">VARIABLE 1: Asistente virtual con inteligencia artificial</td><td rowspan="2">METODOLOGÍA</td></tr>
<tr><td>DIMENSIONES</td><td>INDICADORES</td><td>INSTRUMENTO</td><td>ESCALA DE MEDICIÓN</td></tr>
<tr><td>GENERAL
¿De qué manera el asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología?</td><td>GENERAL
Determinar en qué medida el asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología</td><td rowspan="5">Hi: El asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología,
Ho: El asistente virtual con inteligencia artificial no optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología.</td><td rowspan="3">Arquitectura y tecnologías</td><td>Selección adecuada de tecnologías de IA</td><td rowspan="5"></td><td rowspan="5"></td><td rowspan="5">TIPO DE INVESTIGACIÓN: Aplicada
ENFOQUE DE INVESTIGACIÓN: Cuantitativo
DISEÑO DE INVESTIGACIÓ: Pre experimental
POBLACIÓN: 5,212 pacientes atendidos mensualmente durante el año 2023
MUESTRA: 42 pacientes del INO
MUESTREO: Probabilístico, aleatorio simple</td></tr>
<tr><td>ESECIFICOS
a) ¿Cuáles son los niveles de satisfacción en cuanto a sus dimensiones facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología antes de la implementación del asistente virtual con inteligencia artificial.</td><td>ESECIFICOS
a) identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología antes de la aplicación del asistente virtual</td><td>Diseño de la arquitectura de software</td></tr>
<tr><td>Integración de componentes y servicios</td></tr>
<tr><td rowspan="2">Funcionalidades y servicios</td><td>Definición de funciones y servicios clave</td></tr>
<tr><td>Diseño de flujos de interacción</td></tr>
<tr><td>b) ¿Cómo se puede diseñar y aplicar el</td><td>b) diseñar y aplicar el asistente virtual con inteligencia artificial para optimizar la satisfacción del</td><td colspan="5">VARIABLE 2: Satisfacción del paciente</td></tr>
<tr><td rowspan="3">Facilidad de Uso</td><td>Facilidad de acceso a los servicios</td><td rowspan="5">Cuestionario</td><td rowspan="5">Escala Likert (1-5)</td></tr>
<tr><td>Sencillez en los procesos y trámites</td></tr>
<tr><td>Comodidad y confort en la atención</td></tr>
<tr><td rowspan="3">Claridad de la Información</td><td>Comprensión de la información recibida</td></tr>
<tr><td>Claridad en las explicaciones del personal</td></tr>
<tr><td>Satisfacción con la información</td></tr>
<tr><td></td><td>Tiempo de espera para la atención</td></tr>
</table>

asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología?	paciente del Instituto Nacional de Oftalmología		Tiempo de espera y eficiencia	Agilidad y oportunidad en los servicios			
	c) identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología después de la aplicación del asistente virtual			Percepción de eficiencia en la atención			
c) ¿Cuáles son los niveles de satisfacción en cuanto a sus dimensiones facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del pacientes del Instituto Nacional de Oftalmología después de la implementación del asistente virtual con inteligencia artificial?			Facilidad de Uso	Trato amable y respetuoso del personal			
				Capacidad de escucha y empatía			
	d) analizar los puntajes obtenidos del pre y post test de la satisfacción del paciente para verificar la efectividad del asistente virtual.			Confianza y seguridad transmitida			
d) ¿Qué diferencias se observan en los puntajes de satisfacción del paciente entre el pre test y el post test después de la implementación del asistente virtual?							

Anexo 2. Tabla de operacionalización de variables

Variables de estudio	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Escala de medición
Asistente virtual con inteligencia artificial	Es una aplicación de software que utiliza tecnologías de inteligencia artificial para interactuar con los usuarios de manera natural, proporcionando información, servicios y asistencia de manera autónoma y adaptativa (Santillán y Gaitán, 2019)	Se elaboró un software de Asistente virtual con inteligencia artificial.	Arquitectura y tecnologías	Selección adecuada de tecnologías de IA	
				Diseño de la arquitectura de software	
				Integración de componentes y servicios	
			Funcionalidades y servicios	Definición de funciones y servicios clave	
				Diseño de flujos de interacción	
Satisfacción del Paciente	La satisfacción del paciente se refiere al grado en que el paciente percibe que sus necesidades y expectativas han sido cumplidas por los servicios de salud recibidos (Álvarez, 2022)	La satisfacción del paciente se operacionaliza a través de las siguientes dimensiones: 1) Facilidad de uso, 2) Claridad de la información, 3) Tiempo de espera y eficiencia, y 4) Atención y comunicación. Estas dimensiones se miden a través de indicadores específicos.	Facilidad de Uso	Facilidad de acceso a los servicios	Escala Likert (1-5)
				Sencillez en los procesos y trámites	
				Comodidad y confort en la atención	
			Claridad de la Información	Comprensión de la información recibida	
				Claridad en las explicaciones del personal	
				Satisfacción con la información	
			Tiempo de espera y eficiencia	Tiempo de espera para la atención	
				Agilidad y oportunidad en los servicios	
				Percepción de eficiencia en la atención	
			Atención y comunicación	Trato amable y respetuoso del personal	
				Capacidad de escucha y empatía	
				Confianza y seguridad transmitida	

Anexo 3. Instrumentos de recolección de datos

FICHA DE VALIDACIÓN DE CONTENIDO PARA UN INSTRUMENTO

INSTRUCCIÓN: A continuación, se le hace llegar el instrumento de recolección de datos guía de entrevista que permitirá recoger la información en la presente investigación: **Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología.** Por lo que se le solicita que tenga a bien evaluar el instrumento, haciendo, de ser caso, las sugerencias para realizar las correcciones pertinentes. Los criterios de validación de contenido son:

CRITERIOS	DETALLE	CALIFICACIÓN
Suficiencia	El/la ítem/pregunta pertenece a la dimensión/subcategoría y basta para obtener la medición de esta.	1: de acuerdo 0: en desacuerdo
Claridad	El/la ítem/pregunta se comprende fácilmente, es decir, su sintáctica y semántica son adecuadas.	1: de acuerdo 0: en desacuerdo
Coherencia	El/la ítem/pregunta tiene relación lógica con el indicador que está midiendo.	1: de acuerdo 0: en desacuerdo
Relevancia	El/la ítem/pregunta es esencial o importante, es decir, debe ser incluido.	1: de acuerdo 0: en desacuerdo

Nota. Criterios adaptados de la propuesta de Escobar y Cuervo (2008)

MATRIZ DE VALIDACIÓN DEL CUESTIONARIO SATISFACCIÓN DEL PACIENTE

Satisfacción del paciente: La satisfacción del paciente se refiere al grado en que el paciente percibe que sus necesidades y expectativas han sido cumplidas por los servicios de salud recibidos (Álvarez, 2022)

DIMENSIÓN	INDICADOR	ÍTEM	SUFICIENCIA	CLARIDAD	COHEERENCIA	RELEVANCIA	OBSERVACIÓN
Facilidad de Uso	-Facilidad de acceso a los servicios -Sencillez en los procesos y trámites -Comodidad y confort en la atención	¿Qué tan fácil le resulta acceder al sistema de gestión de citas y consultas?	1	1	1	1	
		¿Cómo calificaría la sencillez del proceso para solicitar o modificar una cita?	1	1	1	1	
		¿Qué tan cómodo se siente con el método de gestión de citas y consultas?	1	1	1	1	
Claridad de la Información	-Comprensión de la información recibida -Claridad en las explicaciones del personal -Satisfacción con la información proporcionada	¿Qué tan comprensible es la información que recibe sobre sus citas y consultas?	1	1	1	1	
		¿Cómo calificaría la claridad de las explicaciones sobre los procesos de atención?	1	1	1	1	
		¿Qué tan satisfecho está con la información proporcionada sobre los servicios del instituto?	1	1	1	1	

Tiempo de espera y eficiencia	-Tiempo de espera para la atención -Agilidad y oportunidad en los servicios -Percepción de eficiencia en la atención	¿Cómo calificaría el tiempo de espera para obtener o modificar una cita?	1	1	1	1	
		¿Qué tan ágil y oportuna considera la atención en el proceso de gestión de citas y consultas?	1	1	1	1	
		¿Cómo percibe la eficiencia general del sistema de gestión de citas y consultas?	1	1	1	1	
Atención y comunicación	-Trato amable y respetuoso del personal -Capacidad de escucha y empatía -Confianza y seguridad transmitida	¿Cómo calificaría el trato recibido durante el proceso de gestión de citas?	1	1	1	1	
		¿Qué tan bien siente que se escuchan y entienden sus necesidades específicas como paciente con discapacidad visual?	1	1	1	1	
		¿Qué nivel de confianza y seguridad le transmite el sistema de gestión de citas y consultas?	1	1	1	1	

UNIVERSIDAD CÉSAR VALLEJO

Cuestionario de Satisfacción del Paciente

Título: Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología.

Objetivo General: Determinar en qué medida el asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología

OE1: Identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología antes de la aplicación del asistente virtual

OE3: Identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología después de la aplicación del asistente virtual

Tipo de aplicación: [ ] Pre-test [ ] Post-test

Instrucciones: Por favor, responda a las siguientes preguntas utilizando la escala del 1 al 5: 1 = Muy insatisfecho / Muy malo, 2 = Insatisfecho / Malo, 3 = Neutral / Regular, 4 = Satisfecho / Bueno, 5 = Muy satisfecho / Muy bueno

Dimensión: Facilidad de Uso

1. ¿Qué tan fácil le resulta acceder al sistema de gestión de citas y consultas?

Respuesta: [ ]

2. ¿Cómo calificaría la sencillez del proceso para solicitar o modificar una cita?

Respuesta: [ ]

3. ¿Qué tan cómodo se siente con el método de gestión de citas y consultas?

Respuesta: [ ]

Dimensión: Claridad de la Información

4. ¿Qué tan comprensible es la información que recibe sobre sus citas y consultas?

Respuesta: [ ]

5. ¿Cómo calificaría la claridad de las explicaciones sobre los procesos de atención?

Respuesta: [ ]

6. ¿Qué tan satisfecho está con la información proporcionada sobre los servicios del instituto?

Respuesta: [ ]

Dimensión: Tiempo de espera y eficiencia

7. ¿Cómo calificaría el tiempo de espera para obtener o modificar una cita?

Respuesta: [ ]

8. ¿Qué tan ágil y oportuna considera la atención en el proceso de gestión de citas y consultas?

Respuesta: [ ]

9. ¿Cómo percibe la eficiencia general del sistema de gestión de citas y consultas?

Respuesta: [ ]

Dimensión: Atención y comunicación

10. ¿Cómo calificaría el trato recibido durante el proceso de gestión de citas?

Respuesta: [ ]

11. ¿Qué tan bien siente que se escuchan y entienden sus necesidades específicas como paciente con discapacidad visual?

Respuesta: [ ]

12. ¿Qué nivel de confianza y seguridad le transmite el sistema de gestión de citas y consultas?

Respuesta: [ ]

FICHA DE VALIDACIÓN DE JUICIO DE EXPERTO

Nombre del instrumento	Cuestionario
Nombres y apellidos del experto	Carlos Alberto Vargas Cardenas
Documento de identidad	42556608
Años de experiencia en el área	16
Máximo grado académico	Maestro en Edumática y Docencia Universitaria
Nacionalidad	Peruana
Institución	Ministerio del Interior
Cargo	Coordinador de Sistemas
Número telefónico	997032292
Correo electrónico	cvargas@mininter.gob.pe cvargasca17@ucvvirtual.edu.pe
Firma	
Fecha	04/07/2024

FICHA DE VALIDACIÓN DE CONTENIDO PARA UN INSTRUMENTO

INSTRUCCIÓN: A continuación, se le hace llegar el instrumento de recolección de datos guía de entrevista que permitirá recoger la información en la presente investigación: **Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología.** Por lo que se le solicita que tenga a bien evaluar el instrumento, haciendo, de ser caso, las sugerencias para realizar las correcciones pertinentes. Los criterios de validación de contenido son:

CRITERIOS	DETALLE	CALIFICACIÓN
Suficiencia	El/la ítem/pregunta pertenece a la dimensión/subcategoría y basta para obtener la medición de esta.	1: de acuerdo 0: en desacuerdo
Claridad	El/la ítem/pregunta se comprende fácilmente, es decir, su sintáctica y semántica son adecuadas.	1: de acuerdo 0: en desacuerdo
Coherencia	El/la ítem/pregunta tiene relación lógica con el indicador que está midiendo.	1: de acuerdo 0: en desacuerdo
Relevancia	El/la ítem/pregunta es esencial o importante, es decir, debe ser incluido.	1: de acuerdo 0: en desacuerdo

Nota. Criterios adaptados de la propuesta de Escobar y Cuervo (2008)

MATRIZ DE VALIDACIÓN DEL CUESTIONARIO SATISFACCIÓN DEL PACIENTE

Satisfacción del paciente: La satisfacción del paciente se refiere al grado en que el paciente percibe que sus necesidades y expectativas han sido cumplidas por los servicios de salud recibidos (Álvarez, 2022)

DIMENSIÓN	INDICADOR	ÍTEM	SUFICIENCIA	CLARIDAD	COHEERENCIA	RELEVANCIA	OBSERVACIÓN
Facilidad de Uso	-Facilidad de acceso a los servicios -Sencillez en los procesos y trámites -Comodidad y confort en la atención	¿Qué tan fácil le resulta acceder al sistema de gestión de citas y consultas?	1	1	1	1	
		¿Cómo calificaría la sencillez del proceso para solicitar o modificar una cita?	1	1	1	1	
		¿Qué tan cómodo se siente con el método de gestión de citas y consultas?	1	1	1	1	
Claridad de la Información	-Comprensión de la información recibida -Claridad en las explicaciones del personal -Satisfacción con la información proporcionada	¿Qué tan comprensible es la información que recibe sobre sus citas y consultas?	1	1	1	1	
		¿Cómo calificaría la claridad de las explicaciones sobre los procesos de atención?	1	1	1	1	
		¿Qué tan satisfecho está con la información proporcionada sobre los servicios del instituto?	1	1	1	1	

Tiempo de espera y eficiencia	-Tiempo de espera para la atención -Agilidad y oportunidad en los servicios -Percepción de eficiencia en la atención	¿Cómo calificaría el tiempo de espera para obtener o modificar una cita?	1	1	1	1	
		¿Qué tan ágil y oportuna considera la atención en el proceso de gestión de citas y consultas?	1	1	1	1	
		¿Cómo percibe la eficiencia general del sistema de gestión de citas y consultas?	1	1	1	1	
Atención y comunicación	-Trato amable y respetuoso del personal -Capacidad de escucha y empatía -Confianza y seguridad transmitida	¿Cómo calificaría el trato recibido durante el proceso de gestión de citas?	1	1	1	1	
		¿Qué tan bien siente que se escuchan y entienden sus necesidades específicas como paciente con discapacidad visual?	1	1	1	1	
		¿Qué nivel de confianza y seguridad le transmite el sistema de gestión de citas y consultas?	1	1	1	1	

UNIVERSIDAD CÉSAR VALLEJO

Cuestionario de Satisfacción del Paciente

Título: Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología.

Objetivo General: Determinar en qué medida el asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología

OE1: Identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología antes de la aplicación del asistente virtual

OE3: Identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología después de la aplicación del asistente virtual

Tipo de aplicación: [ ] Pre-test [ ] Post-test

Instrucciones: Por favor, responda a las siguientes preguntas utilizando la escala del 1 al 5: 1 = Muy insatisfecho / Muy malo, 2 = Insatisfecho / Malo, 3 = Neutral / Regular, 4 = Satisfecho / Bueno, 5 = Muy satisfecho / Muy bueno

Dimensión: Facilidad de Uso

1. ¿Qué tan fácil le resulta acceder al sistema de gestión de citas y consultas?

Respuesta: [ ]

2. ¿Cómo calificaría la sencillez del proceso para solicitar o modificar una cita?

Respuesta: [ ]

3. ¿Qué tan cómodo se siente con el método de gestión de citas y consultas?

Respuesta: [ ]

Dimensión: Claridad de la Información

4. ¿Qué tan comprensible es la información que recibe sobre sus citas y consultas?

Respuesta: [ ]

5. ¿Cómo calificaría la claridad de las explicaciones sobre los procesos de atención?

Respuesta: [ ]

6. ¿Qué tan satisfecho está con la información proporcionada sobre los servicios del instituto?

Respuesta: [ ]

Dimensión: Tiempo de espera y eficiencia

7. ¿Cómo calificaría el tiempo de espera para obtener o modificar una cita?

Respuesta: [ ]

8. ¿Qué tan ágil y oportuna considera la atención en el proceso de gestión de citas y consultas?

Respuesta: [ ]

9. ¿Cómo percibe la eficiencia general del sistema de gestión de citas y consultas?

Respuesta: [ ]

Dimensión: Atención y comunicación

10. ¿Cómo calificaría el trato recibido durante el proceso de gestión de citas?

Respuesta: [ ]

11. ¿Qué tan bien siente que se escuchan y entienden sus necesidades específicas como paciente con discapacidad visual?

Respuesta: [ ]

12. ¿Qué nivel de confianza y seguridad le transmite el sistema de gestión de citas y consultas?

Respuesta: [ ]

FICHA DE VALIDACIÓN DE JUICIO DE EXPERTO

Nombre del instrumento	Cuestionario
Nombres y apellidos del experto	Dino Michael Quinteros Navarro
Documento de identidad	41567782
Años de experiencia en el área	17
Máximo grado académico	Maestría Dirección de Tecnología de la Información
Nacionalidad	Peruana
Institución	Universidad César Vallejo
Cargo	Docente Universitario
Número telefónico	981306306
Correo electrónico	dquinterosna@ucv.edu.pe
Firma	
Fecha	01 / 06 / 2024

FICHA DE VALIDACIÓN DE CONTENIDO PARA UN INSTRUMENTO

INSTRUCCIÓN: A continuación, se le hace llegar el instrumento de recolección de datos guía de entrevista que permitirá recoger la información en la presente investigación: **Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología.** Por lo que se le solicita que tenga a bien evaluar el instrumento, haciendo, de ser caso, las sugerencias para realizar las correcciones pertinentes. Los criterios de validación de contenido son:

CRITERIOS	DETALLE	CALIFICACIÓN
Suficiencia	El/la ítem/pregunta pertenece a la dimensión/subcategoría y basta para obtener la medición de esta.	1: de acuerdo 0: en desacuerdo
Claridad	El/la ítem/pregunta se comprende fácilmente, es decir, su sintáctica y semántica son adecuadas.	1: de acuerdo 0: en desacuerdo
Coherencia	El/la ítem/pregunta tiene relación lógica con el indicador que está midiendo.	1: de acuerdo 0: en desacuerdo
Relevancia	El/la ítem/pregunta es esencial o importante, es decir, debe ser incluido.	1: de acuerdo 0: en desacuerdo

Nota. Criterios adaptados de la propuesta de Escobar y Cuervo (2008)

MATRIZ DE VALIDACIÓN DEL CUESTIONARIO SATISFACCIÓN DEL PACIENTE

Satisfacción del paciente: La satisfacción del paciente se refiere al grado en que el paciente percibe que sus necesidades y expectativas han sido cumplidas por los servicios de salud recibidos (Álvarez, 2022)

DIMENSIÓN	INDICADOR	ÍTEM	SUFICIENCIA	CLARIDAD	COHEERENCIA	RELEVANCIA	OBSERVACIÓN
Facilidad de Uso	-Facilidad de acceso a los servicios -Sencillez en los procesos y trámites -Comodidad y confort en la atención	¿Qué tan fácil le resulta acceder al sistema de gestión de citas y consultas?	1	1	1	1	
		¿Cómo calificaría la sencillez del proceso para solicitar o modificar una cita?	1	1	1	1	
		¿Qué tan cómodo se siente con el método de gestión de citas y consultas?	1	1	1	1	
Claridad de la Información	-Comprensión de la información recibida -Claridad en las explicaciones del personal -Satisfacción con la información proporcionada	¿Qué tan comprensible es la información que recibe sobre sus citas y consultas?	1	1	1	1	
		¿Cómo calificaría la claridad de las explicaciones sobre los procesos de atención?	1	1	1	1	
		¿Qué tan satisfecho está con la información proporcionada sobre los servicios del instituto?	1	1	1	1	

Tiempo de espera y eficiencia	-Tiempo de espera para la atención -Agilidad y oportunidad en los servicios -Percepción de eficiencia en la atención	¿Cómo calificaría el tiempo de espera para obtener o modificar una cita?	1	1	1	1	
		¿Qué tan ágil y oportuna considera la atención en el proceso de gestión de citas y consultas?	1	1	1	1	
		¿Cómo percibe la eficiencia general del sistema de gestión de citas y consultas?	1	1	1	1	
Atención y comunicación	-Trato amable y respetuoso del personal -Capacidad de escucha y empatía -Confianza y seguridad transmitida	¿Cómo calificaría el trato recibido durante el proceso de gestión de citas?	1	1	1	1	
		¿Qué tan bien siente que se escuchan y entienden sus necesidades específicas como paciente con discapacidad visual?	1	1	1	1	
		¿Qué nivel de confianza y seguridad le transmite el sistema de gestión de citas y consultas?	1	1	1	1	

UNIVERSIDAD CÉSAR VALLEJO

Cuestionario de Satisfacción del Paciente

Título: Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología.

Objetivo General: Determinar en qué medida el asistente virtual con inteligencia artificial optimiza la satisfacción del paciente del Instituto Nacional de Oftalmología

OE1: Identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología antes de la aplicación del asistente virtual

OE3: Identificar los niveles de satisfacción en su dimensión facilidad de uso, claridad de la información, tiempo de espera y eficiencia, atención y comunicación del paciente del Instituto Nacional de Oftalmología después de la aplicación del asistente virtual

Tipo de aplicación: [ ] Pre-test [ ] Post-test

Instrucciones: Por favor, responda a las siguientes preguntas utilizando la escala del 1 al 5: 1 = Muy insatisfecho / Muy malo, 2 = Insatisfecho / Malo, 3 = Neutral / Regular, 4 = Satisfecho / Bueno, 5 = Muy satisfecho / Muy bueno

Dimensión: Facilidad de Uso

1. ¿Qué tan fácil le resulta acceder al sistema de gestión de citas y consultas?

Respuesta: [ ]

2. ¿Cómo calificaría la sencillez del proceso para solicitar o modificar una cita?

Respuesta: [ ]

3. ¿Qué tan cómodo se siente con el método de gestión de citas y consultas?

Respuesta: [ ]

Dimensión: Claridad de la Información

4. ¿Qué tan comprensible es la información que recibe sobre sus citas y consultas?

Respuesta: [ ]

5. ¿Cómo calificaría la claridad de las explicaciones sobre los procesos de atención?

Respuesta: [ ]

6. ¿Qué tan satisfecho está con la información proporcionada sobre los servicios del instituto?

Respuesta: [ ]

Dimensión: Tiempo de espera y eficiencia

7. ¿Cómo calificaría el tiempo de espera para obtener o modificar una cita?

Respuesta: [ ]

8. ¿Qué tan ágil y oportuna considera la atención en el proceso de gestión de citas y consultas?

Respuesta: [ ]

9. ¿Cómo percibe la eficiencia general del sistema de gestión de citas y consultas?

Respuesta: [ ]

Dimensión: Atención y comunicación

10. ¿Cómo calificaría el trato recibido durante el proceso de gestión de citas?

Respuesta: [ ]

11. ¿Qué tan bien siente que se escuchan y entienden sus necesidades específicas como paciente con discapacidad visual?

Respuesta: [ ]

12. ¿Qué nivel de confianza y seguridad le transmite el sistema de gestión de citas y consultas?

Respuesta: [ ]

FICHA DE VALIDACIÓN DE JUICIO DE EXPERTO

Nombre del instrumento	Cuestionario
Nombres y apellidos del experto	Jim Oscar Mendoza Espinoza
Documento de identidad	77090070
Años de experiencia en el área	5 AÑOS
Máximo grado académico	Ingeniero de Sistemas
Nacionalidad	Peruano
Institución	Municipalidad Distrital de La Punta
Cargo	Desarrollador de Sistemas
Número telefónico	984187817
Correo electrónico	jimoscarmendoza @gmail.com
Firma	
Fecha	04-06-2024

Anexo 4. Evidencias de la ejecución de la propuesta

1. **Descripción del Proyecto**

Este proyecto utiliza tecnologías web modernas para proporcionar una plataforma fácil de usar que genera texto impulsado por IA. Las tecnologías y herramientas implementadas en este proyecto aseguran una gestión eficiente, una experiencia de usuario mejorada y una interfaz de usuario estéticamente agradable y flexible. A continuación, se detallan los aspectos que hacen especial a nuestro proyecto:

- **Next.js**: Utiliza Next.js con una estructura de enrutador de aplicaciones para una gestión y escalabilidad eficiente de las páginas.
- **Tailwind CSS + DaisyUI**: Ofrece un sistema de diseño personalizable y responsivo, haciendo que la interfaz de usuario sea tanto estéticamente agradable como funcionalmente flexible.
- **Framer Motion**: Integra animaciones para mejorar las interacciones del usuario y proporcionar una experiencia dinámica.
- **Heroicons**: Utiliza Heroicons para crear íconos vectoriales limpios y escalables, mejorando el atractivo visual y la navegación del usuario.

2. **Configuración del Proyecto**

Paso 1: Clonar el Repositorio

Primero, clona el repositorio del proyecto en tu máquina local. Puedes hacerlo utilizando el siguiente comando:

```
git clone https://github.com/tu-usuario/tu-proyecto.git
cd tu-proyecto
```

Paso 2: Instalar Dependencias

Navega al directorio del proyecto y ejecuta el siguiente comando para instalar todas las dependencias necesarias:

```
cd your-project-directory
npm install
```

Paso 3: Ejecutar el Proyecto

Una vez finalizada la instalación de las dependencias, ejecuta el siguiente comando para iniciar el servidor de desarrollo:

```
npm run dev
```

Paso 4: Listo para Empezar

Una vez que el servidor de desarrollo esté en funcionamiento, tendrás una plantilla completa lista para usar. Podrás comenzar a crear y personalizar tu proyecto de inmediato.

3. **Componentes del Proyecto**

- **Next.js**
  Next.js proporciona una estructura robusta para el desarrollo de aplicaciones web, permitiendo una gestión eficiente de las páginas y facilitando la escalabilidad del proyecto.

- **Tailwind CSS + DaisyUI**
  Tailwind CSS junto con DaisyUI ofrece una amplia gama de opciones de personalización para el diseño de la interfaz de usuario, asegurando que sea responsiva y visualmente atractiva.

- **Framer Motion**
  Framer Motion se utiliza para implementar animaciones fluidas y dinámicas, mejorando las interacciones del usuario y la experiencia general en la plataforma.

- **Heroicons**
  Heroicons proporciona íconos vectoriales que son limpios y escalables, mejorando la navegación y el atractivo visual de la aplicación.

4. **Estructura del Código**
   La plantilla de CodingNutella está organizada de una manera que promueve la reutilización y la facilidad de mantenimiento. Al seguir un enfoque basado en características, garantizamos que nuestro código sea fácil de navegar y que los componentes sean fáciles de reutilizar en proyectos similares. A continuación, se muestra una descripción detallada de la estructura del código:

5. **Enrutamiento de Aplicaciones**
   Las páginas y la navegación de la aplicación se gestionan a través del enrutador basado en archivos de Next.js. Al crear archivos en el directorio pages, configuramos automáticamente rutas que coinciden con los nombres de los archivos. Esto simplifica el proceso de agregar nuevas páginas y vincular las distintas secciones de la aplicación.

Estructura del Directorio

```
src/
├── app/                    # Directory for app-wide setti
│   ├── page.tsx            # Main app component
│   ├── layout.tsx          # Common layout component used
│   └── globals.css         # Global styles for the entire
├── pages/                  # Directory for individual pag
│   ├── about/              # About page directory
│   │   ├── page.tsx        # Specific 'about' page compon
│   │   └── style.css       # Specific CSS for the about p
│   └── contact/            # Contact page directory
│       ├── page.tsx        # Specific 'contact' page comp
│       └── style.css       # Specific CSS for the contact
├── api/                    # Directory for API integratio
│   ├── user.js             # API interactions for user da
│   └── products.js         # API interactions for product
├── context/                # Context API for global state
│   ├── UserContext.tsx     # Context for user state
│   └── ThemeContext.tsx    # Context for theme state
├── components/             # Reusable UI components and s
│   ├── ui/                 # UI-specific components like
│   │   ├── Button.tsx      # Button component
│   │   ├── InputField.tsx # Input field component
│   │   └── Modal.tsx       # Modal component
│   └── sections/           # Components specific to certa
│       ├── HeroSection.tsx # Hero section component with
│       └── FooterSection.tsx # Footer component with lin
├── hooks/                  # Custom React hooks
│   └── useFetch.ts         # Example hook for fetching da
├── utils/                  # Utility functions
│   └── formatter.ts        # Example utility for formatti
└── public/                 # Static files like images and
    ├── assets/
    │   ├── logo.svg        # Logo used across the app
    │   └── favicon.ico     # Favicon for the website
```

6. Componentes Reutilizables

Los componentes reutilizables se almacenan en el directorio src/components/ui. Estos componentes están diseñados para ser utilizados en distintas partes de la aplicación, promoviendo la consistencia y reduciendo la duplicación de código. A continuación, se muestra un ejemplo de la estructura del directorio y una breve descripción de algunos de los componentes incluidos.

Estructura del Directorio

```
src/
├── components/
│   ├── ui/
│   │   ├── Button.tsx
│   │   ├── InputField.tsx
│   │   └── Modal.tsx
```

Ejemplos de Componentes Reutilizables

6.1. Botón (Button.tsx)

El componente Button es un botón reutilizable que puede ser utilizado en varias partes de la aplicación. Este componente puede recibir diferentes propiedades para personalizar su apariencia y comportamiento.

Ejemplo de uso:

```tsx
import React from 'react';

interface ButtonProps {
  onClick: () => void;
  className?: string;
  children: React.ReactNode;
}

const Button: React.FC<ButtonProps> = ({ onClick, className, children }) =>
  return (
    <button onClick={onClick} className={`btn ${className}`}>
      {children}
    </button>
  );
};

export default Button;
```

6.2. Campo de Entrada (InputField.tsx)

El componente InputField es un campo de entrada reutilizable que puede ser utilizado para recibir texto del usuario. Este componente puede recibir propiedades para personalizar su apariencia y validación.

Ejemplo de uso:

```tsx
import React from 'react';

interface InputFieldProps {
  label: string;
  type: string;
  value: string;
  onChange: (e: React.ChangeEvent<HTMLInputElement>) => void;
  className?: string;
}

const InputField: React.FC<InputFieldProps> = ({ label, type, value, onC
  return (
    <div className={`input-field ${className}`}>
      <label>{label}</label>
      <input type={type} value={value} onChange={onChange} />
    </div>
  );
};

export default InputField;
```

6.3. Modal (Modal.tsx)

El componente Modal es un modal reutilizable que puede ser utilizado para mostrar información adicional o solicitar acciones del usuario sin salir de la página actual.

```tsx
import React from 'react';

interface ModalProps {
  title: string;
  children: React.ReactNode;
  onClose: () => void;
}

const Modal: React.FC<ModalProps> = ({ title, children, onClose }) => {
  return (
    <div className="modal">
      <div className="modal-content">
        <div className="modal-header">
          <h2>{title}</h2>
          <button onClick={onClose}>Close</button>
        </div>
        <div className="modal-body">
          {children}
        </div>
      </div>
    </div>
  );
};

export default Modal;
```

6.4. Beneficios de los Componentes Reutilizables

- **Consistencia:** Al usar componentes reutilizables, garantizamos una apariencia y comportamiento consistentes en toda la aplicación.
- **Mantenimiento:** Facilita el mantenimiento del código, ya que cualquier cambio realizado en un componente reutilizable se refleja en todas las partes de la aplicación donde se usa.
- **Productividad:** Aumenta la productividad al reducir la necesidad de escribir código repetitivo y permite a los desarrolladores centrarse en la lógica de negocio.

7. Animaciones

Las animaciones que mejoran la experiencia del usuario al brindar comentarios interactivos se almacenan en el directorio src/components/animations. Estas animaciones están diseñadas para ser reutilizadas en distintos componentes, manteniendo la coherencia visual y reduciendo la duplicación de código. A continuación, se describe cómo están organizadas y cómo se pueden utilizar estas animaciones en la aplicación.

7.1. Estructura:

las animaciones se implementan como contenedores versátiles que se pueden integrar o eliminar fácilmente de su proyecto. Nuestros componentes de animación, como FadeInTransition, mejoran la experiencia del usuario al brindar transiciones visuales fluidas sin alterar la estructura subyacente del componente.

7.2. Código de componente

Este enfoque garantiza que las animaciones sean autónomas y reutilizables, lo que las hace perfectas para mejorar la estética de cualquier proyecto sin el riesgo de causar interrupciones si se eliminan.

```tsx
import React from 'react';
import { motion } from 'framer-motion';
import { useInView } from 'react-intersection-observer';


type FadeInTransitionProps = {
  children: React.ReactNode;
  delay?: number;
  className?: string;
};

const FadeInTransition: React.FC<FadeInTransitionProps> =
  const { ref, inView } = useInView({
    triggerOnce: true,
    threshold: 0.1,
  });

  const fadeInVariant = {
    hidden: { opacity: 0 },
    visible: {
      opacity: 1,
      transition: { delay, duration: 0.5, ease: "easeInOu
    },
  };

  return (
    <motion.div
      ref={ref}
      initial="hidden"
      animate={inView ? "visible" : "hidden"}
      variants={fadeInVariant}
      className={className}
    >
      {children}
    </motion.div>
  );
};

export default FadeInTransition;
```

7.3. Pautas de Estilo

Nuestra configuración de estilo le permite ajustar fácilmente cualquier parte de un componente mediante Tailwind CSS. En el archivo globals.css, encontrará clases adicionales para elementos como espaciado, degradados y sombras, lo que le brinda aún más formas de modificar la apariencia de su aplicación.

7.4. Tailwind CSS

Tailwind CSS es una poderosa herramienta de utilidad que permite la personalización rápida y eficiente de los estilos. A continuación se muestra un ejemplo de cómo puede estructurar su archivo globals.css y utilizar Tailwind CSS en su proyecto.

Ejemplo de globals.css:

```
/* globals.css */
@tailwind base;
@tailwind components;
@tailwind utilities;

/* Clases personalizadas adicionales */
.spacing-lg {
  margin: 2rem;
  padding: 2rem;
}

.gradient-bg {
  background: linear-gradient(90deg, #ff7e5f, #feb47b);
}

.shadow-xl {
  box-shadow: 0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0,
}
```

7.5. Temas de DaisyUI

DaisyUI incluye una variedad de temas predefinidos, como “claro”, “oscuro” y “magdalena”. Estos temas están listos para usar y personalizan todos los elementos de DaisyUI con sus conjuntos de colores específicos. Para usar uno de estos temas, debe agregarlo a su archivo de configuración de Tailwind y activarlo configurando el atributo data-theme en su etiqueta HTML.

7.5.1. Instalación de DaisyUI:

Primero, asegúrese de tener DaisyUI instalado en su proyecto. Puede hacerlo mediante npm o yarn:

```
npm install daisyui
# o
yarn add daisyui
```

7.5.2. Configuración de Tailwind CSS:

Agregue DaisyUI a su archivo de configuración de Tailwind (tailwind.config.js):

```
// tailwind.config.js
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [require('daisyui')],
  daisyui: {
    themes: ["light", "dark", "cupcake"], // Agregar temas aquí
  },
}
```

7.5.3. Activación de un Tema en HTML:

Configure el atributo data-theme en la etiqueta HTML para activar un tema específico:

```
<!DOCTYPE html>
<html lang="en" data-theme="light"> <!-- Cambie "light" a cualquier t
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.
  <title>My Application</title>
  <link href="/path/to/globals.css" rel="stylesheet">
</head>
<body>
  <!-- Su contenido -->
</body>
</html>
```

7.5.4. Creación de Temas Personalizados

Para obtener detalles sobre la creación de temas personalizados, puede visitar el Generador de Temas de DaisyUI. Esto le permitirá crear y ajustar temas específicos según sus necesidades.

- **Acceso al Generador de Temas:**
  Visite la Documentación de los Temas de DaisyUI para obtener más información sobre cómo utilizar los temas predefinidos y personalizarlos.

- **Uso del Generador de Temas:**
  Utilice el Generador de Temas de DaisyUI para crear sus propios temas personalizados. Este generador le permitirá ajustar los colores y estilos de su tema y generar el código necesario para incorporarlo en su proyecto.

8. API

8.1. /api/chattogpt/route.ts

Este archivo de ruta de API de Next.js (chattogpt.ts) maneja solicitudes POST para una API específica. Define una ruta que espera la entrada del usuario y la carga opcional de archivos. El proceso implica la extracción del contenido de texto de los archivos cargados (si los hay) junto con la entrada del usuario. Estos datos se pasan a una función llamada chatToGPT para su posterior procesamiento.

8.1.1. Ganchos y Dependencias:

Este código se enfoca en la lógica del lado del servidor dentro de una ruta de API de Next.js y no utiliza ganchos de React. Dependencias clave incluyen:

- Next.js: para manejar solicitudes y respuestas HTTP (NextResponse y NextRequest).
- Módulos Node.js como fs para operaciones de sistema de archivos y path para el manejo de rutas.
- fsPromises del módulo fs para operaciones asíncronas de sistema de archivos.
- Funciones personalizadas como encodeImageToBase64, extractTextFromDoc, extractTextFromPdf, y chatToGPT desde el directorio @/components/utils.

8.1.2. Funciones Auxiliares:

- encodeImageToBase64: Convierte archivos de imagen a formato base64.
- extractTextFromDoc: Extrae texto de documentos de Word.
- extractTextFromPdf: Extrae texto de archivos PDF.
- chatToGPT: Interactúa con el modelo GPT de OpenAI, generando respuestas basadas en la entrada del usuario y el contexto de los archivos.

8.1.3. Funcionalidad:

- La ruta desactiva el análisis automático del cuerpo en Next.js, manejando manualmente la carga útil del formulario.
- Al recibir una solicitud POST, extrae datos de entrada y carga de archivo opcional del formulario.
- Procesa el tipo de archivo cargado:
    - Codifica imágenes en base64.
    - Extrae texto de documentos de Word y PDF.
    - Lee contenido de otros tipos de archivo en formato UTF-8.
- Limpia el archivo temporal utilizado para el procesamiento después de cada operación.

- Envía la entrada del usuario y el contenido del archivo a chatToGPT para generar una respuesta.
- Retorna una respuesta JSON con el resultado del chat y estado 200 en caso de éxito, o un mensaje de error y estado 500 en caso de fallo durante el procesamiento.

```
// pages/api/chattogpt.ts

import { NextResponse, NextRequest } from "next/server";
import fs from "fs";
import path from "path";
import { promises as fsPromises } from "fs";
import { encodeImageToBase64 } from "@/components/utils/e
import { extractTextFromDoc } from "@/components/utils/ex
import { extractTextFromPdf } from "@/components/utils/ex
import { chatToGPT } from "@/components/utils/OpenAiv1";

// Disable the default body parsing in Next.js for this r
 const config = {
  api: {
    bodyParser: false,
  },
};

export async function POST(req: NextRequest) {
  try {
    const formData = await req.formData();
    const userInput = formData.get("userInput") as string
    const file = formData.get("file") as File;

    let fileContent = "";
    let fileType = "";

    if (file) {
      const tempDir = path.join(process.cwd(), "tmp");
      if (!fs.existsSync(tempDir)) {
        fs.mkdirSync(tempDir);
      }

      const tempFilePath = path.join(tempDir, `${Date.now
      const arrayBuffer = await file.arrayBuffer();
      await fsPromises.writeFile(tempFilePath, Buffer.fro

      fileType = file.type;
```

8.2. Archivo ruta.ts 1

Esta función del lado del servidor está implementada en una aplicación Next.js para manejar solicitudes POST. Aquí se detalla su funcionalidad:

- Es una función asíncrona que espera un objeto NextRequest como parámetro.
- Deshabilita el análisis automático del cuerpo en Next.js para esta ruta específica.
- Dentro de la función:
    - Utiliza req.formData() para obtener los datos del formulario de la solicitud.
    - Extrae el valor del campo "userInput" de los datos del formulario.
    - Llama a la función imageGeneration desde la utilidad OpenAiv1, pasando la entrada del usuario como contenido para generar una imagen.
    - Formatea la imagen resultante en una estructura específica.
    - Registra el resultado en la consola.
    - Devuelve una respuesta JSON usando NextResponse.json, con el resultado formateado, código de estado 200 y tipo de contenido "application/json" en los encabezados de respuesta.
    - Maneja cualquier error detectado, lo registra en la consola y devuelve una respuesta JSON indicando un error con código de estado 500.

8.2.1. Dependencias Externas:

El código depende de:

- Objetos NextResponse y NextRequest para manejar solicitudes y respuestas del servidor en Next.js.
- La función imageGeneration desde @/components/utils/OpenAiv1 para generar una imagen basada en la entrada del usuario.

8.2.2. Uso:

Para utilizar esta función:

- Debe asociarse con una ruta específica en la aplicación Next.js para manejar solicitudes POST.
- Al recibir una solicitud POST en esa ruta, la función procesa la solicitud, genera una imagen basada en la entrada del usuario, la formatea en una estructura específica y responde con la URL de la imagen en formato JSON.
- En caso de errores durante este proceso, se gestionan adecuadamente devolviendo una respuesta de error.

```
import { imageGeneration } from "@/components/utils/OpenAi
import { NextResponse, NextRequest } from "next/server";


// Disable the default body parsing in Next.js for this rc


export async function POST(req: NextRequest) {
  try {
    const formData = await req.formData();
    const userInput = formData.get("userInput") as string;

    const result = await imageGeneration({ content: userIr
    console.log(result)
const formatResult = {
  message:{
    role:'assistant',
    content:result[0].url,
    type:'image'
  }
}
    return NextResponse.json(formatResult, {
      status: 200,
      headers: {
        "Content-Type": "application/json",
      },
    });
  } catch (error) {
    console.error("Error processing request:", error);
    return NextResponse.json(
      { error: "Error processing request" },
      {
        status: 500,
        headers: {
          "Content-Type": "application/json",
        },
      }
    );
  }
}
```

8.3. Archivo ruta.ts 2

Este componente no es un componente típico de React para interfaces de usuario, sino una ruta de API en una aplicación Next.js. Su función principal es manejar solicitudes POST para interactuar con el modelo GPT (Transformador preentrenado generativo), generando respuestas de texto basadas en la entrada del usuario.

8.3.1. Ganchos Utilizados:

No se utilizan ganchos de React, ya que este componente actúa como un controlador de ruta API en Next.js, manejado principalmente por los objetos NextRequest y NextResponse.

8.3.2. Funciones Auxiliares:

- talkToGpt: Importada desde el directorio utils, se utiliza para comunicarse con el modelo GPT de OpenAI y generar respuestas basadas en la entrada del usuario recibida.

8.3.3. Uso:

- Cuando se envía una solicitud POST al endpoint de API /api/talktogpt, se recibe la entrada del usuario desde el cuerpo de la solicitud.
- Esta entrada se pasa a la función talkToGpt para generar una respuesta utilizando el modelo GPT.
- La respuesta del modelo GPT se devuelve como una respuesta JSON con un estado 200 (OK).

8.3.4. Dependencias:

El componente depende de:

- Objetos del lado del servidor Next.js (NextRequest y NextResponse) para manejar solicitudes y respuestas adecuadamente.
- La función utilitaria externa talkToGpt del módulo OpenAiv1 ubicado en el directorio utils, utilizada para interactuar con el modelo GPT.

8.3.5. Tipificaciones de TypeScript:

- interface userInputParams: Define la estructura de los parámetros de entrada, esperando específicamente un campo userInput de tipo cadena.

```
// pages/api/talktogpt.ts

 // Import the ChatToGPT function from the utils director
import { talkToGpt } from "@/components/utils/OpenAiv1";
import { NextRequest, NextResponse } from "next/server";

// Define the structure of the subscription parameters
interface userInputParams {
  userInput: string;
}

// Handle POST requests to this API endpoint
export async function POST(request: NextRequest) {
  // Extract user input from the request body
  const { userInput }: userInputParams = await request.js

  // Send the user input to the ChatToGPT function and ge
  const chatgptResponse = await talkToGpt({ content: user
console.log(chatgptResponse)
  // Return the response as JSON with a status of 200 (OK
  return new NextResponse(JSON.stringify( chatgptResponse
    status: 200,
    headers: {
      "Content-Type": "application/json", // Set the cont
    },
  });
}
```

8.4. Archivo ruta.ts

Este fragmento de código representa una función del lado del servidor en una aplicación Next.js diseñada para manejar solicitudes POST. A continuación se detallan sus aspectos principales:

8.4.1. Ganchos y Dependencias Externas:

- Este código no implica componentes React, por lo que no utiliza ganchos React. Se centra exclusivamente en la lógica del lado del servidor dentro del entorno Next.js.
- Las dependencias externas utilizadas incluyen:
  - next/server para importar NextResponse y NextRequest.
  - Módulos estándar de Node.js como fs (Sistema de archivos) y path, utilizados para operaciones de archivos y gestión de rutas.
  - Funciones utilitarias como transcribeAudio, talkToGpt, imageGeneration, y chatToGPT desde @/components/utils/OpenAiv1 para tareas específicas como transcripción de audio, generación de imágenes basadas en texto y comunicación con modelos GPT.

8.4.2. Función POST:

- Esta función es asíncrona y maneja las solicitudes POST recibidas por el servidor.
- Espera un objeto NextRequest como parámetro.
- Analiza los datos del formulario de la solicitud, buscando un archivo de audio y un campo mode que especifica el tipo de procesamiento a realizar.
- Realiza diferentes acciones según el valor de mode:
  - Para el modo audio, transcribe el archivo de audio y consulta a GPT para generar una respuesta.
  - Para el modo image, genera una imagen basada en la transcripción.
  - Por defecto (o en el modo text), utiliza la transcripción para chatear con el modelo GPT.
- Escribe temporalmente el archivo de audio en formato WAV de forma sincrónica, lo procesa según el modo especificado y luego elimina el archivo temporal.
- Finalmente, devuelve una respuesta JSON con el resultado procesado o una respuesta de error si ocurre alguna excepción durante el proceso.

8.4.3. Manejo de Errores:

- En caso de errores durante el procesamiento del archivo o llamadas a la API, la función detecta y registra el error. Luego, devuelve una respuesta JSON de error con un código de estado apropiado.

```
import { NextResponse, NextRequest } from 'next/server';
import fs from 'fs';
import path from 'path';
import { transcribeAudio, talkToGpt, imageGeneration, cha
 // Adjust the path to your helper

export async function POST(req: NextRequest) {
  try {
    // Parse form data from the request
    const formData = await req.formData();
    const file = formData.get('audio');
    const mode = formData.get('mode');

    if (!file || !(file instanceof Blob)) {
      console.error('No audio file received or file is no
      return NextResponse.json({ error: 'No audio file re
    }

    // Convert the file to a buffer
    const buffer = Buffer.from(await file.arrayBuffer());

    // Define the file path for storing the temporary WAV
    const tempDir = path.join(process.cwd(), 'tmp');
    if (!fs.existsSync(tempDir)) {
      fs.mkdirSync(tempDir);
    }
    const filePath = path.join(tempDir, 'input.wav');

    // Write the audio data to a temporary WAV file synch
    fs.writeFileSync(filePath, buffer);

    // Use your OpenAI helper to transcribe the audio fil
    const transcription = await transcribeAudio(filePath)

    let result;

    switch (mode) {
      case 'audio':
        result = await talkToGpt({ content: transcription
        break;
```

9. UTILDADES

9.1. Codificador de OPENIA - archivo OpenAiv1.ts

9.1.1. Componente ChatToGPT

La función chatToGPT en este componente acepta content, fileContent y fileType como propiedades. Recolecta mensajes basados en el contenido proporcionado y el contenido del archivo. Si se proporciona contenido de archivo, verifica si el tipo de archivo comienza con "image/" y ajusta el contenido en consecuencia. Luego, crea una solicitud de completado a la API de OpenAI utilizando el método openai.chat.completions.create().

La función talkToGpt es otra función que interactúa con chatToGPT. Llama a chatToGPT para obtener una respuesta y luego utiliza esa respuesta para generar voz utilizando el método audio.speech.create() de OpenAI. El archivo de audio resultante se guarda en la ruta especificada en speechFile. Esta función devuelve un objeto con información sobre el archivo de audio generado.

La función transcribeAudio transcribe audio desde la ruta de archivo proporcionada mediante el método audio.transcriptions.create() de OpenAI. Lee el archivo de audio como una secuencia y devuelve la transcripción.

Finalmente, la función imageGeneration genera una imagen basada en el contenido proporcionado utilizando el método images.generate() de OpenAI.

9.1.2. React Hooks y Dependencias Externas

Este componente no utiliza ganchos específicos de React ya que no está destinado a ser un componente React.

Las dependencias externas cruciales incluyen:

- fs: Un módulo Node.js para operaciones del sistema de archivos.
- path: Un módulo Node.js para gestionar rutas de archivos.
- OpenAI: Una biblioteca externa para interactuar con la API de OpenAI.

9.1.3. Interfaz ChatToGPTProps

Define la estructura de las propiedades que se pasan a las funciones chatToGPT, talkToGpt y imageGeneration. Incluye content como obligatorio, y fileContent y fileType como opcionales.

9.1.4. Uso

- chatToGPT: Proporcione un mensaje del usuario y contenido de archivo opcional para generar una respuesta utilizando la API de OpenAI.
- talkToGpt: Utiliza chatToGPT para generar voz a partir de la respuesta y guarda el resultado como un archivo de audio.
- transcribeAudio: Transcribe contenido de audio desde una ruta de archivo proporcionada.
- imageGeneration: Genera una imagen basada en el contenido proporcionado.

```
import fs from "fs";
import path from "path";
import OpenAI from "openai"; // Import the OpenAI library

const openai = new OpenAI(); // Create an instance of the

// Define the structure of the properties for the ChatToG

/**
 * Sends a user's message to the OpenAI API and retrieves
 *
 * @param {ChatToGPTProps} props - The properties contain
 * @returns {Promise<any>} - The response from the OpenAI
 */

interface ChatToGPTProps {
  content: string;
  fileContent?: string;
  fileType?: string;
}

export async function chatToGPT({
  content,
  fileContent,
  fileType,
}: ChatToGPTProps) {
  let messages = [
    {
      role: "user",
      content: [{ type: "text", text: content }],
    },
  ];
```

9.2. Codificador de API - archivo Ayudante de API.ts

Este componente de React utiliza dos funciones auxiliares del archivo utils/apiHelpers.js para interactuar con API mediante el envío de mensajes o datos:

9.2.1. Funciones Auxiliares:

- sendMessageToAPI(apiUrl, userInput, file?): Esta función envía un mensaje a un endpoint específico de API. Utiliza fetchAPI para hacer una solicitud POST y envía la entrada del usuario junto con un archivo opcional, si se proporciona. El archivo se adjunta a los datos del formulario antes de enviar la solicitud. Si la respuesta de la API no es exitosa (estado incorrecto), se genera un error. La función devuelve una promesa que se resuelve con la respuesta JSON de la API.

- sendAudioToAPI(apiUrl, formData): Similar a sendMessageToAPI, pero diseñada específicamente para enviar datos de audio a un endpoint de API. El objeto formData proporcionado contiene los datos de audio que se enviarán. También utiliza fetchAPI para realizar una solicitud POST a la API especificada. Si la respuesta no es exitosa, se genera un error. Al igual que sendMessageToAPI, esta función devuelve una promesa que se resuelve con la respuesta JSON de la API.

Estas funciones abstraen el proceso de interacción con las API, manejando la construcción de las solicitudes, el envío de datos y la gestión de errores. Simplifican la comunicación de la API para el componente React.

9.2.2. Uso y Características:

- No se utilizan ganchos de React directamente en este código. En su lugar, se centra en funciones asincrónicas para realizar llamadas a la API y gestionar las respuestas.
- Estas funciones son esperadas por el componente que las utiliza cuando se necesite interactuar con las API. Facilitan la integración de la comunicación de API en los componentes, abstrayendo la complejidad de las solicitudes API, manejo de errores y análisis de respuestas.
- Ambas funciones están tipadas en TypeScript para especificar los tipos de datos esperados para sus parámetros.
- Dependencia de fetchAPI, una característica estándar del navegador para realizar solicitudes de red.

```
// utils/apiHelpers.js

/**
 * Sends a message to the specified API and returns the r
 *
 * @param {string | URL | Request} apiUrl - The URL of th
 * @param {any} userInput - The user input to be sent to
 * @returns {Promise<any>} - The response from the API as
 * @throws Will throw an error if the response status is
 */
export async function sendMessageToAPI(apiUrl: string | U
  const formData = new FormData();

  formData.append("userInput", userInput);
  if (file) {
    formData.append("file", file);
  }

  const response = await fetch(apiUrl, {
    method: "POST",
    cache: "no-store",
    body: formData,
  });

  if (!response.ok) {
    throw new Error(`HTTP error! Status: ${response.statu
  }

  return response.json();
}
```

9.3. Codificador de Imagen - archivo codificarImagenABase64.ts

Este archivo no muestra código específico de React, sino que contiene una función de utilidad escrita en TypeScript.

9.3.1. Función de Utilidad: encodeImageToBase64

- **Propósito:** Esta función auxiliar toma una ruta de archivo de imagen y su tipo MIME como entradas, y devuelve una representación base64 de la imagen.

- **Parámetros:**
  - imagePath (string): La ruta del archivo de imagen que se desea codificar.
  - mimeType (string): El tipo MIME del archivo de imagen.

9.3.2. Dependencias:

- fs: Este módulo es básico en Node.js y se utiliza para operaciones de sistema de archivos. En este contexto, se emplea para leer el contenido del archivo de imagen de manera síncrona.

9.3.3. Operaciones:

1. Lee el archivo de imagen de forma síncrona en un búfer utilizando fs.readFileSync(imagePath).
2. Convierte el búfer de imagen a una cadena en formato base64.
3. Formatea y devuelve la URI de datos de la imagen en base64, incluyendo el tipo MIME especificado.

9.3.4. Uso:

Esta función puede emplearse en un componente React u otras partes de una aplicación Node.js donde sea necesario convertir archivos de imágenes a base64. Es especialmente útil cuando se trabaja con datos de imágenes que deben integrarse en contenido web, como para mostrar imágenes recibidas desde un servidor o almacenadas localmente.

```
// utils/encodeImageToBase64.ts
import fs from "fs";

export const encodeImageToBase64 = (imagePath: string, mi
  const imageBuffer = fs.readFileSync(imagePath);
  return `data:${mimeType};base64,${imageBuffer.toString(
};
```

9.4. Codificador de Documento - archivo extraerTextoDeDoc.ts

Este archivo contiene una función de utilidad llamada extractTextFromDoc, que utiliza la biblioteca mammoth para extraer contenido de texto de un documento de Word de manera asincrónica. A continuación se detallan los elementos clave de esta función:

9.4.1. Dependencias:

La función de utilidad depende de la biblioteca mammoth, que está diseñada para analizar documentos de Word y extraer su contenido.

9.4.2. Funcionalidad:

- extractTextFromDoc es una función asincrónica que acepta un parámetro filePath (una cadena que representa la ruta al documento de Word).
- Utiliza mammoth.extractRawText para extraer los datos de texto sin procesar del documento de Word ubicado en la ruta proporcionada.
- Retorna una promesa que se resuelve con el contenido de texto extraído del documento de Word.
- Detecta y maneja cualquier error que pueda ocurrir durante el proceso de extracción, generando un mensaje de error correspondiente.

```
// utils/extractTextFromDoc.ts
import mammoth from "mammoth";

export const extractTextFromDoc = async (filePath: string
  try {
    const result = await mammoth.extractRawText({ path: f
    return result.value;
  } catch (error) {
    throw new Error(`Error extracting text from Word docu
  }
};
```

9.5. Archivo extraerTextoDePdf.ts

Este fragmento de código presenta una función de utilidad TypeScript llamada extractTextFromPdf, diseñada para extraer datos de texto de archivos PDF. A continuación se detalla su funcionamiento:

9.5.1. Dependencias:

- Utiliza el módulo fs de Node.js para interactuar con el sistema de archivos.
- Dependencia de pdf2json para analizar el contenido del archivo PDF.

9.5.2. Funcionalidad:

- La función acepta un parámetro filePath, que es una cadena representando la ruta al archivo PDF.
- Retorna una Promise que se resuelve con el contenido de texto extraído del PDF.
- Internamente, crea una instancia de PDFParser para manejar el análisis del PDF.
- Configura listeners de eventos en la instancia pdfParser para capturar eventos pdfParser_dataError y pdfParser_dataReady.
- Si ocurre un error durante el análisis del PDF, la Promise se rechaza con el error del analizador.
- Si los datos se analizan correctamente, itera sobre las páginas y elementos de texto del PDF, decodificando el texto y acumulándolo en una variable text.
- Finalmente, resuelve la Promise con el texto extraído del PDF.

9.5.3. Uso:

- Para utilizar esta función, simplemente proporciona la ruta al archivo PDF como argumento.
- Retorna una Promise que proporciona el texto extraído del PDF una vez que el proceso de análisis se completa.

```ts
// utils/extractTextFromPdf.ts
import fs from 'fs';
import PDFParser from 'pdf2json';

export const extractTextFromPdf = (filePath: string): Pro
  return new Promise((resolve, reject) => {
    const pdfParser = new PDFParser();
    pdfParser.on("pdfParser_dataError", errData => reject
    pdfParser.on("pdfParser_dataReady", pdfData => {
      let text = '';
      pdfData.Pages.forEach(page => {
        page.Texts.forEach(textItem => {
          textItem.R.forEach(r => {
            text += decodeURIComponent(r.T) + ' ';
          });
        });
      });
      resolve(text.trim());
    });
    pdfParser.loadPDF(filePath);
  });
};
```

9.6. Archivo Transición de desvanecimiento.tsx

Este componente de React, FadeInTransition, utiliza la biblioteca framer-motion para crear una animación simple de aparición gradual cuando sus componentes secundarios entran en la pantalla. A continuación se detallan los elementos clave en el código del componente:

9.6.1. Dependencias externas:

- framer-motion: Utilizado para animaciones y transiciones. Se importan LazyMotion, domAnimation, y m.div de esta biblioteca para construir la animación.
- react-intersection-observer: Utiliza el gancho useInView de esta biblioteca para detectar cuándo el componente entra en la vista y activar la animación de aparición gradual.

9.6.2. Propiedades del componente:

- children: Representa el contenido que se desvanecerá.
- delay: Propiedad opcional que establece el retraso antes de que comience la animación de entrada gradual (valor predeterminado: 0).
- className: Propiedad opcional para especificar clases CSS adicionales para el estilo (valor predeterminado: "flex flex-1").

9.6.3. Ganchos de React utilizados:

- useInView: Detecta cuándo el componente está dentro de la ventana gráfica, proporcionando un ref para adjuntarlo al elemento observado y un valor booleano inView que indica su visibilidad.

9.6.4. Configuración de la animación:

- fadeInVariant: Objeto que define las variantes de animación para el efecto de aparición gradual, con estados hidden y visible. La animación incluye un retraso configurado por delay segundos y tiene una duración de 0,5 segundos con suavizado "easeInOut".

9.6.5. Estructura del componente:

- Envuelto en un componente LazyMotion con domAnimation como una de sus características.
- Utiliza m.div como elemento para encapsular el contenido que debe aparecer gradualmente.
- El ref del gancho useInView se adjunta a este elemento m.div, controlando la animación basada en el estado booleano inView.
- La propiedad variants de m.div se establece con fadeInVariant para definir el comportamiento de la animación.
- La propiedad className permite la aplicación de estilos personalizados al contenido que aparece gradualmente.

9.6.6. Uso:

Para usar este componente, impórtalo en tu aplicación React y coloca el contenido que quieres que aparezca como elemento secundario. Opcionalmente, puedes ajustar los elementos delayy classNamepara personalizar el estilo y el tiempo de la animación, respectivamente. La animación de aparición gradual se activará cuando el componente se desplace hasta la vista en la página.

```
"use client";
import React from "react";
import { LazyMotion, domAnimation, m } from "framer-motic
import { useInView } from "react-intersection-observer";

type FadeInTransitionProps = {
  children: React.ReactNode;
  delay?: number;
  className?: string;
};

const FadeInTransition: React.FC<FadeInTransitionProps> =
  children,
  delay = 0,
  className = "flex flex-1",
}) => {
  const { ref, inView } = useInView({
    triggerOnce: true, // Optional: Trigger animation onl
    threshold: 0.1, // Customize threshold as needed
  });

  const fadeInVariant = {
    hidden: { opacity: 0 },
    visible: {
      opacity: 1,
      transition: { delay, duration: 0.5, ease: "easeInOu
    },
  };
```

9.7. Interfaz ModeState:

Define la forma del objeto de estado mode, que incluye:

- type: Una propiedad que especifica el tipo de contenido siendo procesado ('texto', 'audio' o 'imagen').
- apiUrl: Una cadena que representa la URL de la API correspondiente al tipo de contenido seleccionado.

9.7.1. Interfaz TextOrVoiceContextType:

Define la estructura del valor de contexto proporcionado por el proveedor de contexto, que incluye el estado mode y la función setMode para actualizar dicho estado.

9.7.2. Texto o contexto de voz:

Es un contexto de React creado con createContext, que gestiona el estado y los métodos relacionados con el contexto del tipo de contenido (texto, audio o imagen).

9.7.3. Gancho useTextOrVoice:

Un gancho personalizado (useTextOrVoice) creado para acceder fácilmente al contexto dentro de componentes funcionales utilizando useContext. Este gancho arroja un error si se usa fuera del proveedor TextOrVoiceProvider.

9.7.4. Componente TextOrVoiceProvider:

Es un proveedor que envuelve a sus elementos secundarios con el proveedor TextOrVoiceContext. Este componente:

- Inicializa el estado del modo utilizando el gancho useState, con el modo predeterminado establecido en "texto" y una URL de API predeterminada.
- Proporciona una función setMode para actualizar dinámicamente el estado del modo basado en el tipo de contenido seleccionado.

9.7.5. Interfaz TextOrVoiceProviderProps:

Define los tipos de propiedad esperados por el componente TextOrVoiceProvider, que incluyen children de tipo ReactNode para renderizar los elementos secundarios dentro del proveedor.

9.7.6. Uso:

Para utilizar este contexto:

- Envuelve la parte del árbol de componentes que necesita acceso al estado del modo y a la función setMode con TextOrVoiceProvider.
- Utiliza el gancho useTextOrVoice dentro de cualquier componente funcional secundario para acceder y manipular el estado del modo según sea necesario.

9.7.7. Dependencias externas:

Este componente depende únicamente de React para su funcionalidad y no tiene otras dependencias externas más allá de las proporcionadas por React mismo.

```
'use client';
import React, { createContext, useContext, useState, Reac

// Define the mode state interface
interface ModeState {
  type: 'text' | 'audio' | 'image';
  apiUrl: string;
}

// Define the context type
interface TextOrVoiceContextType {
  mode: ModeState;
  setMode: (type: ModeState['type']) => void;
}

// Create the context
const TextOrVoiceContext = createContext<TextOrVoiceConte

// Export a hook to use the context
export const useTextOrVoice = () => {
  const context = useContext(TextOrVoiceContext);
  if (!context) throw new Error('useTextOrVoice must be u
  return context;
};

// Define the provider component props
interface TextOrVoiceProviderProps {
  children: ReactNode;
}

// Define the provider component
export const TextOrVoiceProvider: React.FC<TextOrVoicePro
  const [mode, setModeState] = useState<ModeState>({ type
```

10. INTERFAZ DE USUARIO

10.1. Audio

10.1.1. Reproductor de audio.tsx

Este componente de React, AutoPlayAudio, es responsable de reproducir un archivo de audio automáticamente cuando se renderiza. Aprovecha la API de audio web para el procesamiento y la visualización de audio. Analicemos los elementos clave:

Ganchos de reacción :

- useEffect:Se utilizan dos useEffectganchos. El primero configura el contexto de la API de audio web y crea un nodo analizador. También se encarga de la limpieza cuando se desmonta el componente. Este gancho se ejecuta cuando cambia audioContexto setLoading. El segundo useEffectactualiza la fuente de audio cuando audiocambia la propiedad. Esto garantiza que el archivo de audio se cargue y se reproduzca siempre que se proporcione un nuevo archivo de audio.

- useRef: audioRefes una referencia al <audio>elemento. Permite el acceso directo al nodo DOM sin activar una nueva representación. Esto es fundamental para manipular el elemento de audio directamente cuando sea necesario.

- useState:Se utilizan dos variables de estado:

  - audioContext:Administra el contexto de la API de audio web.
  - analyser:Almacena la referencia al nodo analizador creado para la visualización.

Dependencias externas :

- VisualizerComponente: este componente se importa y se renderiza de manera condicional para brindar una representación visual del audio que se está reproduciendo. Toma el analysernodo como un elemento para crear la visualización.

Funcionalidad del componente :

- El componente configura el contexto de la API de audio web y crea un nodo analizador para visualizar el audio que se está reproduciendo.

- Carga el archivo de audio especificado en la audiopropiedad, inicia la reproducción y detecta cualquier error que ocurra durante la reproducción.

- Cuando finaliza la reproducción de audio, indica que el estado de carga es falso.

- El <audio>elemento se configura con la fuente de audio proporcionada y los atributos para la precarga y el control. Se audioRefadjunta a este elemento para acceso directo.

Accesorios :

- audio: Cadena que representa el archivo de audio que se reproducirá. Es una propiedad obligatoria para el componente.

En resumen, este AutoPlayAudiocomponente encapsula la lógica para reproducir automáticamente archivos de audio mediante la API de audio web y proporciona una representación visual del audio mediante un Visualizercomponente. Se encarga de configurar el contexto de audio, cargar y reproducir el archivo de audio y limpiar los recursos cuando se desmonta el componente.

```
const AutoPlayAudio: React.FC<{ audio: string }> = ({ aud
  const audioRef = useRef<HTMLAudioElement>(null);
  const { setLoading } = useMessagesData();
  const [audioContext, setAudioContext] = useState<AudioC
  const [analyser, setAnalyser] = useState<AnalyserNode |

  useEffect(() => {
    if (!audioContext) {
      const audioCtx = new AudioContext();
      setAudioContext(audioCtx);
    }

    if (audioRef.current && audioContext) {
      const source = audioContext.createMediaElementSourc
      const analyserNode = audioContext.createAnalyser();

      source.connect(analyserNode);
      analyserNode.connect(audioContext.destination);
      setAnalyser(analyserNode);

      audioRef.current.onended = () => {
        setLoading(false);
      };
```

10.1.2. Grabador de audio.tsx

Componente AudioRecorder

Este componente utiliza los siguientes ganchos de React:

useState : administra el estado de isRecording, un valor booleano que indica si la grabación de audio está actualmente activa o no.

useEffect : maneja los efectos secundarios relacionados con el inicio y la detención de la grabación de audio y la limpieza de recursos cuando se desmonta el componente.

useRef : crea una referencia para mediaRecorderRef, que almacena la instancia de MediaRecorder para grabar audio.

Funciones y operaciones auxiliares:

- **toggleRecording** : alterna el valor de isRecordingestado entre truey false, iniciando o deteniendo así la grabación de audio según corresponda.

Bibliotecas externas/Dependencias:

- **useAudioSender** : este enlace personalizado useAudioSenderparece gestionar el envío de datos de audio, posiblemente a un servidor u otro destino. Se importa, pero no se muestra explícitamente en el código proporcionado.

Uso del componente:

- Cuando el componente se renderiza, configura un efecto useEffectque escucha los cambios en la isRecordingvariable de estado.
- Si isRecordinges true, solicita acceso al micrófono del usuario, crea una instancia de MediaRecorder, comienza a grabar audio y actualiza la mediaRecorderRef.currentcon la referencia de instancia. También define la lógica para recopilar datos de audio durante la grabación y detiene la grabación cuando es necesario.
- Cuando se detiene la grabación o se desmonta el componente, limpia los recursos deteniendo la instancia de MediaRecorder y sus pistas asociadas.
- La toggleRecordingfunción se llama cuando se hace clic en el botón representado en el componente, lo que permite a los usuarios iniciar o

detener la grabación de audio.

- La apariencia del botón cambia entre un ícono de micrófono (cuando no se está grabando) y un ícono de pausa (cuando se está grabando) para proporcionar información visual al usuario.

Tipificaciones de TypeScript:

- El componente es un componente funcional de tipo React.FCy no recibe ninguna propiedad explícitamente.

```
useEffect(() => {
  if (isRecording) {
    console.log("Starting recording");

    navigator.mediaDevices
      .getUserMedia({ audio: true })
      .then((stream) => {
        console.log("Microphone access granted");
        const mediaRecorder = new MediaRecorder(stream)
        mediaRecorderRef.current = mediaRecorder;

        const audioChunks: BlobPart[] = [];
        mediaRecorder.ondataavailable = (event) => {
          audioChunks.push(event.data);
          console.log("Data available:", event.data);
        };

        mediaRecorder.onstop = () => {
          console.log("Recording stopped");
          const audioBlob = new Blob(audioChunks, { typ
          sendAudio(audioBlob);
        };

        mediaRecorder.start();
        console.log("Recording started");
      })
      .catch((error) => {
        console.error("Error accessing the microphone:"
        setIsRecording(false);
      });
```

10.1.3. Visualizador.tsx

Este componente de React, Visualizer, es un componente de visualización que representa un gráfico de barras de frecuencia en un lienzo en función de los datos de audio recibidos de un AnalyserNode.

- **Ganchos** :
    - useEffect:Se utiliza para realizar efectos secundarios en componentes de funciones. En este componente, useEffectse utiliza para configurar y actualizar la visualización en el lienzo.
    - useRef:Se utiliza para almacenar una referencia al elemento de lienzo.
- **Función auxiliar** :
    - ctx.roundRect:Esta función es una extensión de la API CanvasRenderingContext2D. Dibuja un rectángulo con esquinas redondeadas en el lienzo. Incluye comprobaciones de parámetros para valores opcionales radiusy un valor predeterminado si no se proporciona ninguno.
- **Uso** :
    - Se necesita un soporte analyser, que es un AnalyserNodeque proporciona datos de frecuencia en tiempo real para la visualización.
    - Dentro del useEffectgancho, se configura el contexto del lienzo y una función para renderizar el marco de forma continua usando requestAnimationFrame.
    - La visualización consta de barras que representan los datos de frecuencia recibidos. La altura de cada barra está determinada por la amplitud de la frecuencia y el color se basa en el color del tema obtenido de las variables CSS.
- **Dependencias** :
    - Este componente no depende de ninguna biblioteca o dependencia externa, y utiliza únicamente la API estándar de React y Canvas para la visualización.
- **Estilo y tamaño del lienzo** :
    - El elemento de lienzo se redimensiona a width="600"y height="200"y se le asignan estilos en línea para capacidad de respuesta (style={{ width: '100%', maxWidth: '600px', height: '200px', display: 'block' }}).

El componente actualiza continuamente la visualización en función de los datos de frecuencia recibidos desde el AnalyserNode, proporcionando una representación en vivo de la frecuencia de audio en un formato de gráfico de barras en el lienzo.

```
const Visualizer: React.FC<VisualizerProps> = ({ analyser
  const canvasRef = useRef<HTMLCanvasElement | null>(null

  useEffect(() => {
    const canvas = canvasRef.current;
    if (!canvas) return;

    const ctx = canvas.getContext('2d');
    if (!ctx) return;

    const WIDTH = canvas.width;
    const HEIGHT = canvas.height;

    // Add type check and default value for radius
    ctx.roundRect = function (x: number, y: number, width
      if (radius === undefined) {
        radius = 5; // Provide a default radius if none i
      }

      this.beginPath();
      this.moveTo(x + radius, y);
      this.lineTo(x + width - radius, y);
      this.quadraticCurveTo(x + width, y, x + width, y +
      this.lineTo(x + width, y + height - radius);
      this.quadraticCurveTo(x + width, y + height, x + wi
      this.lineTo(x + radius, y + height);
      this.quadraticCurveTo(x, y + height, x, y + height
      this.lineTo(x, y + radius);
      this.quadraticCurveTo(x, y, x + radius, y);
      this.closePath();
      return this;
    };

    const getThemeColor = (): string => {
      const computedStyle = getComputedStyle(document.doc
      return computedStyle.getPropertyValue('--fallback-p
    };
```

10.2. BANNER

10.2.1. Banner.tsx

Este componente React Banneres un componente funcional simple que muestra un banner con una imagen, un mensaje de saludo y un párrafo corto de texto de de bienvenida
El componente importa el Imagecomponente del next/imagepaquete, que optimiza las imágenes en las aplicaciones Next.js.

Dentro del componente:

- El elemento externo divtiene clases CSS para el diseño de flexbox (flexpara contenedor flexible, flex-colpara apilamiento vertical, gap-2para espaciado entre elementos secundarios, items-centerpara justify-centercentrar contenido horizontal y verticalmente y text-centerpara alinear texto en el centro).

- El Imagecomponente se utiliza para mostrar una imagen con los siguientes accesorios:

  - src: Especifica la fuente de la imagen (“/logo.svg” en este caso).
  - alt:Texto alternativo para la imagen (establecido en el mismo valor que srcaquí).
  - className:Clases para darle estilo a la imagen (define el ancho, el color de fondo y las esquinas redondeadas).
  - width:Define el tamaño de la imagen para pantallas de escritorio grandes (1800 px en este caso).
  - height:Mantiene una relación de aspecto de 1:1 para la imagen (1800px también).
  - quality:Especifica la calidad de la imagen (aquí se establece en 75).
  - loading:Controla el comportamiento de carga de imágenes (establecido en “ansioso” para una carga inmediata).
  - priority:Indica que la imagen debe considerarse de alta prioridad y precargarse.

- El h1elemento muestra un mensaje de saludo con el nombre de la persona formateado en negrita. El pelemento muestra un párrafo de presentacion con algún texto de marcador de posición.

En general, este componente es un banner simple que tiene como objetivo mostrar una imagen, un saludo con un nombre y un texto introductorio. La imagen está optimizada para pantallas grandes y el diseño está diseñado para ser responsivo y centrado mediante las propiedades de flexbox.

```jsx
import Image from "next/image";

export default function Banner() {
  return (
    <div className="flex flex-col gap-2 items-center just

      <Image
            src={"/logo.svg"}
            alt={"/logo.svg"}

            className="w-[80px] bg-primary rounded-xl"
            width={1800} // Define the size of the image
            height={1800} // This maintains a 1:1 aspect
            quality={75} // General quality setting, can
            loading="eager"
            priority

          />
      <h1 className="text-4xl sm:text-7xl font-light ">
        Hola,{" "}
        <span className="font-bold text-primary ">
          Jhon Doe
        </span>
      </h1>
      <p className="text-base-content/80 max-w-[500px]">
        Lorem ipsum dolor sit amet, consectetur adipiscin
        tempor incididunt ut labore et dolore magna aliqu
      </p>
    </div>
  );
}
```

10.3. TARJETAS

10.3.1. Tarjeta.tsx

Este componente de React es un componente de tarjeta simple (Card) que muestra un título y una descripción. Analicemos los aspectos clave de este componente:

Accesorios :

- CardProps:Esta es una interfaz que define las propiedades esperadas por el Cardcomponente.
  - title:Una cadena que representa el título de la tarjeta.
  - description:Una cadena que representa la descripción de la tarjeta.
  - onclick:Una función que no toma ningún argumento y se activa cuando se hace clic en la tarjeta.

Estructura del componente :

- El Cardcomponente es un componente funcional que toma las propiedades especificadas (title, description, y onclick) como argumentos.
- Devuelve un divelemento que representa la carta. divContiene el título y la descripción de la carta.
- El titlese muestra en un h3elemento con la clase font-semibold text-base-content-100.
- El descriptionse muestra en un pelemento con la clase text-sm text-base-content/70.
- Tiene divuna cursor-pointerclase y un onClickcontrolador de eventos que activa la onclickfunción cuando se hace clic en la tarjeta.

Estilo :

- El componente utiliza clases CSS Tailwind para el estilo.
- Se utilizan clases como min-w-[200px], p-4, sm:w-[280px], shadow-xl, text-start, grid, glassy para darle estilo a la tarjeta.rounded-2xl

Dependencias :

- Este componente no tiene ninguna dependencia externa ni utiliza ninguna biblioteca externa aparte de React mismo.

```tsx
import React from "react";

export interface CardProps {
  title: string;
  description: string;
  onclick: () => void;
}

const Card: React.FC<CardProps> = ({ title, description,
  return (
    <div
      className="min-w-[200px] p-4 sm:w-[280px] shadow-xl
      onClick={onclick}
    >
      <div className="flex items-center gap-2">
        <h3 className="font-semibold text-base-content-10
      </div>
      <p className="text-sm text-base-content/70">{descri
    </div>
  );
};

export default Card;
```

10.4. TARJETAS

ChatBubble.tsx

Este ChatBubblecomponente es un elemento de burbuja de chat versátil que puede manejar varios tipos de mensajes, como texto, imágenes y audio. El comportamiento del componente se representa de forma dinámica en función de las propiedades proporcionadas role, message, typey opcionales .file

Ganchos de reacción:

- **useState Hook** : No se utiliza ningún gancho useState en este componente.

Accesorios:

- **rol (cadena)** : representa el rol del remitente del mensaje, distinguiendo entre un usuario y un bot.
- **mensaje (cadena)** : contiene el contenido del mensaje que debe mostrarse dentro de la burbuja de chat.
- **tipo (cadena | indefinido)** : especifica el tipo de mensaje, como “audio”, “imagen” o “texto”. El valor predeterminado es “texto” si no se especifica.
- **file (File | null)** : una propiedad opcional que puede contener un objeto File. Se utiliza para mostrar una vista previa del archivo cuando se adjunta un archivo a un mensaje.

Funciones auxiliares:

- **renderContent()** : Esta función se utiliza para representar condicionalmente el contenido en función del typemensaje. Devuelve diferentes componentes para distintos tipos de mensajes: AutoPlayAudioun componente para el tipo “audio”, un elemento de imagen para el tipo “imagen” y un TypingEffectcomponente para cualquier otro tipo o tipo faltante.

Dependencias:

- **Componente de imagen Next.js (Image)** : importado desde "next/image"para optimizar la representación de imágenes. - **Componentes externos** :

  - AutoPlayAudiodesde "./AudioPlayer": Se utiliza para representar un componente de reproductor de audio para mensajes de audio.
  - FilePreviewdesde "./FilePreview": Se utiliza para mostrar una vista previa de los archivos adjuntos dentro de la burbuja de chat.
  - TypingEffectdesde "./TypingEffect": Proporciona una animación de efecto de escritura para mensajes de texto.

Estructura del componente:

1. **Estilo condicional** : aplica diferentes estilos (userMessageStyleso botMessageStyles) según si el remitente del mensaje es un usuario o un bot.

2. **Representación condicional :**

    - **Mensaje del bot** : si el mensaje no es del usuario, muestra componentes específicos del bot, como un ícono de bot y el contenido del mensaje representado según el tipo.
    - **Mensaje de usuario** : para los mensajes de usuario, muestra el contenido del mensaje, con una vista previa de archivo opcional para los archivos adjuntos y un reproductor de audio para los mensajes de audio.

3. **Representación de contenido:** utiliza la renderContent()función para mostrar contenido apropiado según el tipo de mensaje.

4. **Manejo de archivos adjuntos**: si filese proporciona un, se muestra un FilePreviewpara el archivo adjunto, que el usuario puede eliminar (aunque la funcionalidad no está implementada aquí).

5. **Reproducción de audio:** cuando el tipo de mensaje es “audio”, reproduce el archivo de audio mediante un <audio>elemento; de lo contrario, muestra el texto del mensaje.

Este ChatBubblecomponente proporciona flexibilidad en el manejo de diferentes tipos de mensajes y formatos de chat, mejorando la experiencia del usuario en una interfaz de chat.

```tsx
import Image from "next/image";
import AutoPlayAudio from "./AudioPlayer";
import FilePreview from "./FilePreview";
import TypingEffect from "./TypingEffec";


interface ChatBubbleProps {
  role: string;
  message: string;
  type?: string;
  file?: File | null; // Add the file property here
}

export default function ChatBubble({ role, message, type,
  const isUser = role === "user";
  const userMessageStyles = "chat chat-end text-primary-c
  const botMessageStyles = "flex items-start gap-4 p-4 te

  const renderContent = () => {
    switch (type) {
      case "audio":
        return <AutoPlayAudio audio={message} />;
      case "image":
        return <a href={message} target="_blank"> <img sr
      case "text":
      default:
        return <TypingEffect text={message} />;
    }
  };

  return (
    <div className={isUser ? userMessageStyles : botMessa
      {!isUser && (
        <>
          <Image
            width={40}
            height={40}
            alt="bot"
            src="/logo.svg"
            className="w-9 bg-primary rounded-md p-1"
          />
          {renderContent()}
        </>
      )}
```

10.5. Vista previa de Documentos

ArchivoPreview.tsx

Este componente de React es un FilePreviewcomponente que muestra una vista previa de un archivo en un elemento con estilo junto con un botón para eliminar. Representa de manera condicional una vista previa de la imagen (si el tipo de archivo comienza con “image/”) utilizando el next/imagecomponente o una vista previa predeterminada que muestra el nombre del archivo y un ícono para los archivos que no son imágenes.

Ganchos usados

- No se utilizan ganchos en este componente.

Dependencias externas

- El componente utiliza el next/imagecomponente para mostrar imágenes de manera eficiente.

Funciones auxiliares

- No se definen funciones auxiliares específicas en este componente.

Funcionalidad del componente

Accesorios :

- FilePreviewEl componente espera dos propiedades:
  - file:Un objeto de archivo o null.
  - onRemove:Una función para activar la eliminación del archivo.

Representación condicional :

- Si filees null, el componente retorna nully no representa nada.
- Determina si el archivo es una imagen basándose en su type(comenzando con “image/”).

Representación :

- Si el archivo es una imagen, muestra la imagen utilizando el next/imagecomponente con un ancho y alto especificados.
- Para los archivos que no son imágenes, muestra el nombre del archivo junto con un ícono.

- Se proporciona un botón “Eliminar” que activa la onRemovefunción pasada como propiedad.

Estilo :

- Utiliza clases CSS Tailwind para diseñar el contenedor, la imagen y otros elementos.

Este componente es particularmente útil en escenarios donde los usuarios necesitan obtener una vista previa de los archivos antes de cargarlos, lo que les permite confirmar visualmente el contenido y posiblemente eliminar el archivo si es necesario.

```
interface FilePreviewProps {
  file: File | null;
  onRemove: () => void;
}

const FilePreview: React.FC<FilePreviewProps> = ({ file,
  if (!file) return null;

  const isImage = file.type.startsWith("image/");

  return (
    <div className="flex items-center gap-2 p-2 border ro
      {isImage ? (
        <Image
          src={URL.createObjectURL(file)}
          alt={file.name}
          width={500}
          height={500}
          className="h-16 w-16 object-cover rounded-lg"
        />
      ) : (
        <div className="flex items-center gap-2">
          <span className="icon-doc"></span>
          <span>{file.name}</span>
        </div>
      )}
      <button onClick={onRemove} className="text-red-600"
        Remove
      </button>
    </div>
```

10.6. APORTE

10.6.1. Entrada.tsx

Este Inputcomponente representa un cuadro de entrada de mensajes de chat con varias funciones, como enviar mensajes de texto, adjuntar archivos y grabar mensajes de audio. Analicemos los aspectos clave de este componente:

Ganchos de React utilizados:

useRef : inputRefse utiliza para acceder y centrarse en el elemento de entrada cuando se activa un atajo de teclado.

useEffect : Inicializa un detector de eventos para manejar un atajo de teclado (Ctrl/Cmd + S) para centrarse en el campo de entrada.

useState : Administra el estado del texto de entrada del mensaje y el archivo seleccionado a enviar.

Funciones auxiliares:

handleSubmit : gestiona el envío del formulario enviando el contenido del mensaje y el archivo adjunto (si lo hay).

handleInputKeyDown : escucha la pulsación de la tecla Enter para enviar el mensaje si el usuario presiona Enter en el campo de entrada.

handleFileChange : actualiza el archivo seleccionado para enviar cuando un usuario adjunta un archivo.

handleFileRemove : borra el archivo seleccionado cuando el usuario decide eliminarlo antes de enviarlo.

Estructura y funcionalidad del componente:

- El componente contiene un elemento de formulario con campos de entrada para mensajes de texto, un botón para enviar el mensaje, un área para grabación de audio y controles para adjuntar archivos.
- El envío de formulario se deshabilita cuando hay una operación de carga en curso, indicada por el isLoadingestado.

- Los archivos adjuntos se muestran como vistas previas debajo del campo de entrada con una opción para eliminarlos.
- El atajo de teclado (Ctrl/Cmd + S) se centra en el campo de entrada para la entrada rápida de mensajes.
- ModeToggleSe representa un componente para cambiar entre los modos de mensajes de texto y de voz.

Accesorios:

- **marcador de posición (cadena)** : especifica el texto del marcador de posición predeterminado para el campo de entrada, normalmente “Escriba su mensaje”.

Dependencias externas:

@heroicons/react : proporciona íconos como PaperAirplaneIcony PaperClipIconpara representación visual.

../Context/MessageDataContext : Contiene datos de contexto relacionados con los mensajes, probablemente utilizados para verificar el estado de carga (isLoading).

../Hooks/useMessageSender : gancho personalizado que maneja la funcionalidad de envío de mensajes, utilizando la URL de API proporcionada.

../Context/TextOrVoiceContext : Administra el modo de envío de mensajes (texto o voz).

./AudioRecorder : Representa la interfaz de usuario para la funcionalidad de grabación de audio, probablemente para mensajes de voz.

./FilePreview : muestra una vista previa del archivo adjunto con una opción para eliminarlo antes de enviarlo.

En resumen, este Inputcomponente maneja la entrada del usuario para mensajes de texto, archivos adjuntos y grabaciones de audio, proporcionando una interfaz integral para enviar mensajes en una aplicación de chat.

```tsx
import React, { useRef, useEffect, useState } from "react
import { PaperAirplaneIcon, PaperClipIcon } from "@heroic
import { useMessagesData } from "../Context/MessageDataCo
import useMessageSender from "../Hooks/useMessageSender";
import ModeToggle from "./ToggleSwitch";
import { useTextOrVoice } from "../Context/TextOrVoiceCon
import AudioRecorder from "./AudioRecorder";
import FilePreview from "./FilePreview";

export default function Input({ placeholder = "Type your
  const { mode } = useTextOrVoice();
  const { message, setMessage, sendMessage, file, setFile
    mode.apiUrl
  );
  const { isLoading } = useMessagesData();
  const inputRef = useRef<HTMLInputElement | null>(null);

  useEffect(() => {
    const handleKeyDown = (event: KeyboardEvent) => {
      if ((event.ctrlKey || event.metaKey) && event.key =
        event.preventDefault();
        if (inputRef.current) {
          inputRef.current.focus();
        }
      }
    };

    document.addEventListener("keydown", handleKeyDown);

    return () => {
      document.removeEventListener("keydown", handleKeyDo
    };
  }, []);

  const handleSubmit = async (event: React.FormEvent<HTML
    event.preventDefault();
    if (isLoading) return;

    await sendMessage(message, file ?? undefined);
    setFile(null); // Clear the file input after sending
  };

  const handleInputKeyDown = (event: React.KeyboardEvent<
    if (event.key === "Enter") {
      event.preventDefault();
      if (isLoading) return;
      handleSubmit(event as any);
    }
```

10.7. CargandoLogo.tsx

Este componente de React, LoadingLogo, muestra una animación de giro y salto de una imagen, que normalmente se utiliza como indicador de carga. Utiliza la framer-motionbiblioteca para funciones de animación y el next/imagecomponente para una carga y optimización eficientes de imágenes.

Ganchos de reacción

Este componente no utiliza ganchos de React. Es un componente funcional que define las animaciones y transiciones para el giro y el salto del logotipo y, luego, lo renderiza utilizando motion.divfrom framer-motion.

Funciones y operaciones auxiliares

Animaciones :

rotateJumpAnimation: Define las animaciones para rotar el logotipo 360 grados y moverlo hacia arriba 20 píxeles y nuevamente hacia abajo.

Transiciones :

rotateJumpTransition: Administra el tiempo y el comportamiento de las animaciones de rotación y salto.

Uso

- Importar el LoadingLogocomponente a un componente principal.
- Cuando LoadingLogose renderiza, muestra una imagen de logotipo girando y saltando, lo cual es ideal para indicar actividades de carga en la interfaz de usuario.

Tipificaciones de TypeScript para accesorios

No hay tipificación de TypeScript para las propiedades de este componente. Todas las funcionalidades y propiedades se definen internamente sin ninguna entrada externa.

Dependencias

- **framer-motion** : esta biblioteca es fundamental para animar elementos en aplicaciones React. Proporciona una forma sencilla de crear animaciones con componentes como motion.div.
- **next/image** : Se utiliza para optimizar y cargar imágenes de forma diferida en aplicaciones Next.js, lo que garantiza una representación y un rendimiento eficientes al mostrar imágenes. En este caso, se trata de la representación de la imagen del logotipo.

```
'use client'
import React from 'react';
import { motion } from 'framer-motion';
import Image from 'next/image';

// Define the animations for rotation and vertical moveme
const rotateJumpAnimation = {
    rotate: [0, 360],  // Rotate from 0 to 360 degrees
    translateY: [0, -20, 0]  // Move up 20px and then bac
};

const rotateJumpTransition = {
    rotate: {
        repeat: Infinity,
        duration: 2,
        ease: "linear",
        repeatDelay: 0.6  // Wait for the jump to finish
    },
    translateY: {
        duration: 0.6,
        ease: "easeInOut",
        delay: 2,  // Start the jump right after the rota
        repeat: Infinity,  // Repeat the jump indefinitel
        repeatDelay: 1.4  // Wait for the rotation to nea
    }
};

const LoadingLogo = () => {
    return (
        <div className='p-4'>
            <motion.div
                animate={rotateJumpAnimation}
                transition={rotateJumpTransition}
                style={{ display: 'inline-block' }}
            >
                <Image width={40} height={40} alt="bot" s
            </motion.div>
        </div>
    );
};

export default LoadingLogo;
```

10.8. BARRA LATERAL

10.8.1. Barra lateral.tsx

Este componente de React es una barra lateral que contiene varios elementos, como un menú de barra lateral con elementos, un botón para abrir o cerrar la barra lateral en pantallas más pequeñas, un logotipo y un controlador de tema. Analicemos los aspectos clave de este componente:

Estado

- **Estado abierto** : esta variable de estado opense utiliza para determinar si la barra lateral está abierta o cerrada en pantallas más pequeñas. La función activa o desactiva esta toggleOpenfunción.

Efectos

- **Efecto de cambio de tamaño** : el useEffectgancho se utiliza para configurar un detector de eventos para el cambio de tamaño de la ventana. La barra lateral se contrae de forma predeterminada en dispositivos con un ancho inferior a 1024 píxeles y la handleResizefunción establece el estado inicial en función del tamaño de la ventana.

Manejo de eventos

- **toggleOpen** : esta función alterna el valor del openestado, que controla si la barra lateral está abierta o cerrada en pantallas más pequeñas.

Componentes y bibliotecas

- **Bibliotecas externas** :
  - **Heroicons/React** : se utiliza para íconos como CubeIcony Bars4Icon.
  - **Siguiente/Imagen** : se utiliza para mostrar imágenes de manera eficiente.

Uso

- El componente Sidebar representa su contenido de forma condicional en función de si el openestado es verdadero o falso.
- La barra lateral incluye un botón de alternancia (Bars4Icon) para abrir/cerrar la barra lateral en pantallas más pequeñas.
- Tiene un logotipo, 'Ai Chat', y un controlador de tema que se muestra en la parte superior.
- Los elementos de la barra lateral se asignan a través del componente SidebarItem, cada uno de los cuales representa un enlace de navegación a una página diferente.

```tsx
"use client";
import { useState, useEffect } from "react";
import SidebarAnnouncement from "./SideBarAnnauncement";
import SidebarItem from "./SideBarItem";
import { CubeIcon } from "@heroicons/react/24/outline";
import Image from "next/image";
import { Bars4Icon } from "@heroicons/react/24/outline";
import ThemeController from "../ThemeController";

export default function Sidebar() {
  const [open, setOpen] = useState(false);

  useEffect(() => {
    // Collapse the sidebar by default on devices with a
    const handleResize = () => {
      setOpen(window.innerWidth >= 1024);
    };

    // Set the initial state based on the window size
    handleResize();

    // Add event listener for window resize
    window.addEventListener("resize", handleResize);

    // Cleanup the event listener on component unmount
    return () => window.removeEventListener("resize", han
  }, []);

  const toggleOpen = () => {
    setOpen(!open);
  };

  return (
    <>
      <div
        onClick={toggleOpen}
        className="fixed  left-2 top-2 rounded-full flex
      >
        <Bars4Icon className="w-4 h-4 text-primary" />
      </div>
      <div
        className={` sidebar glass bg-[url('/bg.svg')] bg
          open ? "translate-x-0" : "-translate-x-full"
        } lg:translate-x-0 fixed inset-y-0 left-0  shadow
      >
```

10.8.2. Anuncio de barra lateral.tsx

Este componente de React, SidebarAnnouncement, es un componente funcional responsable de mostrar un anuncio en la barra lateral con un título, un mensaje, imágenes y botones de acción. A continuación, se muestra un desglose del componente:

- El componente recibe propiedades que se ajustan a la SidebarAnnouncementPropsinterfaz. Estas propiedades incluyen:
    - title:Una cadena que representa el título del anuncio.
    - message:Una cadena que representa el contenido/mensaje principal del anuncio.
    - mainImageSrc:Una cadena que contiene la URL de origen de la imagen principal que se muestra en el anuncio.
    - actionImageSrc:Una cadena que contiene la URL de origen de la imagen de acción que se muestra en el anuncio.
    - onDismiss:Una función que se llama cuando se hace clic en el botón “Descartar”.
    - onLearnMore:Una función que se llama cuando se hace clic en el botón "¿Qué hay de nuevo?"
- La estructura JSX del componente consta de:
    - Un divelemento con clases CSS que dan estilo al contenedor del anuncio.
    - Elementos para representar el título, el mensaje, la imagen principal, los botones de acción y los controladores de eventos asociados para invocar las funciones proporcionadas al hacer clic en los botones.
- El componente utiliza el Imagecomponente del next/imagepaquete para manejar imágenes. Sin embargo, parece haber una discrepancia entre el uso de la mainImageSrcpropiedad y la propiedad fija src={"/hero.png"}en el Imagecomponente. Asegúrese de corregir esto para utilizar la mainImageSrcpropiedad de manera adecuada.
- Los botones de acción permiten a los usuarios activar eventos (onDismissy onLearnMore) para descartar el anuncio o aprender más sobre él, respectivamente.

En resumen, el SidebarAnnouncementcomponente es un elemento de interfaz de usuario reutilizable para mostrar anuncios en una barra lateral, proporcionando un título, contenido del mensaje, imágenes y opciones de interacción para que los usuarios los descarten u obtengan más información.

```tsx
import Image from "next/image";

// SidebarAnnouncement.tsx
interface SidebarAnnouncementProps {
  title: string;
  message: string;
  mainImageSrc: string;
  actionImageSrc: string;
  onDismiss: () => void; // Function to handle dismiss ac
  onLearnMore: () => void; // Function to handle learn mo
}

const SidebarAnnouncement: React.FC<SidebarAnnouncementPr
  title,
  message,
  mainImageSrc,
  actionImageSrc,
  onDismiss,
  onLearnMore,
}) => {
  return (
    <div className="flex glass flex-col px-4 py-5 mw-full
      <div className="flex gap-1  items-start">
        <span className="flex-1 text-gray-900 dark:text-z

      </div>
      <p className=" text-xs text-base-content/70">{messa
      <Image
        width={500}
        height={500}
        src={"/hero.png"}
        className="w-full bg-primary my-4"
        alt="logo"
      />
      <div className="flex gap-3 ">
        <button
          onClick={onDismiss}
          className="text-slate-600 hover:text-slate-800"
        >
          Dismiss
        </button>
        <button
          onClick={onLearnMore}
          className="text-violet-700 hover:text-violet-90
        >
          What's new?
        </button>
```

10.8.3. Elemento de barra lateral.tsx

Este componente de React, SidebarItem, es un componente funcional que se utiliza para representar un solo elemento en un menú de la barra lateral. A continuación, se muestra un desglose de sus características principales:

React Hooks: este componente no utiliza ningún gancho de React.

Accesorios :

- Icon:Propiedad obligatoria que espera un componente React que represente un ícono SVG. Se escribe con React.ComponentType<SVGProps<SVGSVGElement>>para garantizar que el componente de ícono pueda recibir propiedades específicas de SVG.
- label:Una cadena que representa la etiqueta de texto del elemento de la barra lateral.
- href:Una cadena que sirve como URL de enlace para el elemento de la barra lateral.
- isActive:Un indicador booleano que indica si el elemento de la barra lateral está actualmente activo o no.

Funciones auxiliares :

- No hay funciones auxiliares presentes en este componente.

Bibliotecas externas :

- El componente se importa Linkdesde Next.js, lo que sugiere que es probable que este componente se use dentro de una aplicación Next.js para el enrutamiento del lado del cliente.
- El componente también importa SVGPropsdesde React. Esta importación sirve para garantizar la tipificación correcta de los elementos relacionados con SVG que se pasan al Iconcomponente.

Funcionalidad del componente :

- El componente genera un enlace dinámico basado en la hrefpropiedad proporcionada utilizando el Linkcomponente importado desde Next.js.
- El estilo del componente cambia según la isActivepropiedad. Cuando está activo, aplica estilos específicos, como un borde rojo y una fuente en negrita, y al pasar el mouse sobre él, cambia el color del fondo y del texto sin problemas.
- Representa el Iconcomponente proporcionado seguido del labeltexto dentro de un contenedor flexible, lo que permite que un ícono y una etiqueta se muestren uno al lado del otro.

```tsx
import Link from "next/link";
import { SVGProps } from 'react';
interface SidebarItemProps {
  Icon: React.ComponentType<SVGProps<SVGSVGElement>>;
  label: string;
  href: string; // Assuming you might want to link somewh
  isActive: boolean;
}

const SidebarItem: React.FC<SidebarItemProps> = ({
  Icon,
  label,
  href,
  isActive,
}) => {
  return (
    <Link
      href={href}
      className={`text-sm p-2 hover:bg-primary/80 hover:t
        isActive
          ? " border-2 border-red-200 font-semibold"
          : "text-base-content/80"
      }`}
      style={{
        transition:
          "background-color 0.3s, color 0.3s, border-colo
      }}
    >
      <div className="flex gap-2 items-center">
      {Icon && <Icon className="w-4 h-4 " /> }

        <span >{label}</span>
      </div>
    </Link>
  );
};

export default SidebarItem;
```

10.9. CONTROLADOR DE TEMA

Controlador de temas.tsx

Este componente de React, ThemeController, muestra un menú desplegable para seleccionar de una lista de temas predefinidos. El componente recorre una matriz de nombres de temas y crea elementos de entrada de radio para cada tema dentro de una lista.

Ganchos de React utilizados

En este componente no se utilizan ganchos de React. Es un componente funcional que utiliza el manejo básico de eventos y la representación JSX.

Función auxiliar

- handleClick: Esta función toma un objeto de evento como parámetro y detiene su propagación mediante stopPropagation(). Se utiliza para evitar una mayor propagación del evento de clic a los elementos principales.

Uso general del componente

- El componente muestra un menú desplegable con una etiqueta “Seleccionar tema” y una lista de botones de opción que representan diferentes temas.
- Los botones de opción se completan en función de la themesmatriz y cada uno tiene un valor único.
- Cuando se selecciona un botón de opción, se activa un evento de cambio de tema que se puede capturar en un nivel superior en la jerarquía de componentes para aplicar el tema seleccionado.

Tipificación de TypeScript

No se proporcionan tipificaciones de TypeScript en el fragmento de código, pero si se incluyeran, las posibles tipificaciones de propiedades podrían incluir:

- Accesorios: No se necesitan para este componente, ya que no recibe ningún dato ni configuración externa.

Dependencias externas

- El componente no tiene dependencias ni bibliotecas externas. Depende únicamente de React para representar el menú desplegable de selección de temas.

```jsx
import React from 'react';

const themes = [
  "light", "dark", "cupcake", "bumblebee", "emerald", "co
  "retro", "cyberpunk", "valentine", "halloween", "garden
  "lofi", "pastel", "fantasy", "wireframe", "black", "lux
  "autumn", "business", "acid", "lemonade", "night", "cof
  "nord", "sunset",
];

const ThemeController = () => {
  // Handler to stop propagation of click events
  const handleClick = (event: { stopPropagation: () => vo
    event.stopPropagation();
  };

  return (
    <div className="dropdown" onClick={handleClick}>
      <label tabIndex={0} className="btn m-1">Select Them
      <ul tabIndex={0} className="dropdown-content z-[1]
        {themes.map((theme) => (
          <li key={theme}>
            <input
              type="radio"
              name="theme-dropdown"
              className="theme-controller btn btn-sm btn-
              aria-label={theme.charAt(0).toUpperCase() +
              value={theme}
            />
          </li>
        ))}
      </ul>
    </div>
  );
};

export default ThemeController;
```

10.10. INTERRUPTOR DE PALANCA

Interruptor de palanca.tsx

Este componente de React es un ModeTogglecomponente que permite al usuario alternar entre diferentes modos: texto, audio e imagen. Utiliza el useTextOrVoicegancho personalizado para acceder al modo actual y la función para actualizar el modo desde el TextOrVoiceContext.

Ganchos :

- useTextOrVoice:Este gancho personalizado se importa desde '../Context/TextOrVoiceContext'. Proporciona acceso al modo actual (mode) y la función para actualizar el modo (setMode).

Funcionalidad del componente :

- El ModeTogglecomponente representa un conjunto de botones de opción, cada uno de los cuales representa un modo diferente (texto, audio, imagen).
- En función de lo moderecibido del contexto, se marca el botón de opción correspondiente.
- setModese llama con el valor de modo respectivo ('texto', 'audio', 'imagen') cuando se selecciona un modo diferente usando los botones de opción.

Uso del componente :

- Puede incluir este ModeTogglecomponente en un componente principal que proporcione el TextOrVoiceContexto cualquier contexto que contenga las funciones modey setMode.
- Al interactuar con los botones de opción, el usuario puede seleccionar un modo específico, lo que desencadena un cambio en la funcionalidad de la aplicación en función del modo seleccionado.

Dependencias :

- El componente se basa en la Reactbiblioteca para crear la interfaz de usuario.

Tipo de escritura :

- El componente no tiene ningún tipado TypeScript para propiedades ya que es un componente funcional que no recibe ninguna propiedad.

```
'use client';
import React from 'react';
import { useTextOrVoice } from '../Context/TextOrVoiceCon

const ModeToggle: React.FC = () => {
  const { mode, setMode } = useTextOrVoice();

  return (
    <div className="form-control flex flex-row">
      <label className="label cursor-pointer">
        <input
          type="radio"
          name="mode"
          className="radio radio-primary"
          checked={mode.type === 'text'}
          onChange={() => setMode('text')}
        />
        <span className="label-text ml-2">Text</span>
      </label>

      <label className="label cursor-pointer">
        <input
          type="radio"
          name="mode"
          className="radio radio-accent"
          checked={mode.type === 'audio'}
          onChange={() => setMode('audio')}
        />
        <span className="label-text ml-2">Audio</span>
      </label>

      <label className="label cursor-pointer">
        <input
          type="radio"
          name="mode"
          className="radio radio-secondary"
          checked={mode.type === 'image'}
          onChange={() => setMode('image')}
        />
        <span className="label-text ml-2">Image</span>
      </label>
    </div>
```

11. **GANCHO**

useAudioSender.ts

Este componente de React es un gancho personalizado llamado useAudioSenderque se encarga de enviar datos de audio a una API para su transcripción. Analicemos los elementos clave presentes:

Dependencias:

- next/navigation:Importaciones useRoutery usePathnameganchos de Next.js para navegación y manejo de rutas.
- ../Context/MessageDataContext:Importa el useMessagesDataganchо personalizado desde un proveedor de contexto para administrar datos de mensajes.
- ../Context/TextOrVoiceContext:Importa el useTextOrVoiceganchо personalizado desde un proveedor de contexto para administrar si el usuario está usando entrada de texto o voz.
- ../utils/apiHelper:Importa la sendAudioToAPIfunción desde un archivo de utilidad para manejar el envío de datos de audio a la API.

Estado y contexto:

- **Extracción de estado:** desestructura addMessage, setBotThinking, y setLoadingdesde useMessagesData()para acceder a funciones relacionadas con el mensaje desde el contexto.
- **Enrutador:** se utiliza useRouterpara acceder al objeto enrutador Next.js a través de router.
- **Ruta:** se utiliza usePathnamepara obtener la ruta actual de la aplicación.
- **Modo de texto o voz:** se desestructura modepara useTextOrVoice()determinar si el usuario está en modo de texto o de voz.

Funcionalidad:

Función sendAudio:

- **Parámetros:** Toma un audioBlob (objeto Blob) que representa los datos de audio que se enviarán.
- **Funcionalidad:**
  - Establece los estados de carga y de pensamiento del bot como verdaderos.
  - Agrega el mensaje de audio al contexto de los mensajes.
  - Maneja el enrutamiento si la ruta actual es /y realiza la transición a /message.
  - Construye el formData y añade el audioBlob y el tipo de modo.

- Llama a la sendAudioToAPIfunción para enviar los datos de audio.
- Actualiza los estados y agrega el mensaje de transcripción al contexto de los mensajes según el tipo de mensaje relacionado con el modo.

Devolver:

- **Devuelve:** Un objeto con la sendAudiofunción que se utiliza para activar el envío de datos de audio a la API.

Este enlace encapsula la lógica para enviar datos de audio, administrar estados de carga y de pensamiento, actualizar mensajes y manejar respuestas de transcripción. Proporciona una forma limpia y reutilizable de manejar operaciones relacionadas con el audio para la aplicación.

```
const apiUrl = '/api/transcribe'
const { addMessage, setBotThinking, setLoading } = us
const router = useRouter(); // Use the useRouter hook
const pathname = usePathname(); // Use the usePathnam
const { mode } = useTextOrVoice();

// Function to send audio
const sendAudio = async (audioBlob: Blob) => {
    if (!audioBlob) return;
    setLoading(true);
    setBotThinking(true);
    const audioURL = URL.createObjectURL(audioBlob);
    addMessage({ message: audioURL, role: 'user', typ

    if (pathname === '/') {
        router.push('/message');
    }
    try {
        console.log('Sending the audio logic');
        console.log(mode.type);

        const formData = new FormData();
        formData.append('audio', audioBlob, 'audio.we
        formData.append('mode', mode.type);

        const transcription = await sendAudioToAPI(ap

        setBotThinking(false);
        setLoading(false)
```

11.2. usoMessageSender.ts

Este componente de React es un gancho personalizado useMessageSenderque se encarga de enviar mensajes en una aplicación de chat. Utiliza varios ganchos de React y dependencias externas:

Ganchos de React utilizados:

usoEstado:

- Se utiliza para gestionar el estado de messagey file.
- message:Contiene el mensaje de entrada del usuario.
- file: Representa un archivo adjunto opcional.

utilizarMessagesData:

- Gancho personalizado del MessageDataContextcontexto, utilizado para acceder y administrar los datos de los mensajes de chat.
- Desestructura addMessage, setBotThinkingy setLoadingfunciona a partir del contexto.

utilizarRouter:

- Proporcionado por Next.js para acceder al objeto enrutador.
- Se utiliza para navegar entre diferentes páginas.

usePathname:

- Gancho personalizado del next/navigationmódulo para acceder a la ruta actual.

Dependencias externas:

Next.js:

- El uso de useRoutery usePathnamesugiere que este código es parte de una aplicación Next.js.

Funciones y operaciones auxiliares:

- La sendMessagefunción es responsable de enviar un mensaje, manejar archivos adjuntos, actualizar el estado de la interfaz de usuario y comunicarse con la API.
- Activa los estados de carga y de pensamiento del bot para indicar el proceso asincrónico.

- Actualiza el mensaje de chat con la entrada del usuario y maneja la redirección si el usuario está en la página de inicio.
- Se invoca sendMessageToAPIdesde apiHelperpara enviar el mensaje y el archivo a la API.
- Procesa la respuesta de la API, actualiza el chat con el mensaje recibido y maneja las excepciones.

Valor de retorno:

- El useMessageSendergancho devuelve un objeto con variables de estado y funciones necesarias para administrar el envío de mensajes:

    1. message:El mensaje de entrada del usuario actual.
    2. setMessage:Función para actualizar el estado del mensaje.
    3. file: El archivo adjunto.
    4. setFile:Función para actualizar el estado del archivo.
    5. sendMessage:Función para iniciar el proceso de envío de mensajes.

Este gancho encapsula la lógica para enviar mensajes en una aplicación de chat, proporcionando una forma reutilizable y organizada de administrar la entrada y el envío de mensajes.

```
import { useState } from "react";
import { useRouter, usePathname } from "next/navigation";
import { useMessagesData } from "../Context/MessageDataCo
import { sendMessageToAPI } from "../utils/apiHelper";


function useMessageSender(apiUrl: string | URL | Request)
  const [message, setMessage] = useState<string>("");
  const [file, setFile] = useState<File | null>(null);
  const { addMessage, setBotThinking, setLoading } = useM
  const router = useRouter();
  const pathname = usePathname();

  const sendMessage = async (messageToSend: string, fileT
    if (!messageToSend.trim() && !fileToSend) return;

    setLoading(true);
    setBotThinking(true);

    addMessage({
      message: messageToSend,
      role: "user",
      file:file
    });

    if (pathname === "/") {
      router.push("/message");
    }

    setMessage("");

    try {
      const result = await sendMessageToAPI(apiUrl, messa
      setBotThinking(false);
      setLoading(false)

      if (result.message) {
        addMessage({
          message: result.message.content,
          role: result.message.role,
          type: result.message.type,
        });
```

12. CONTEXTO DE DATA

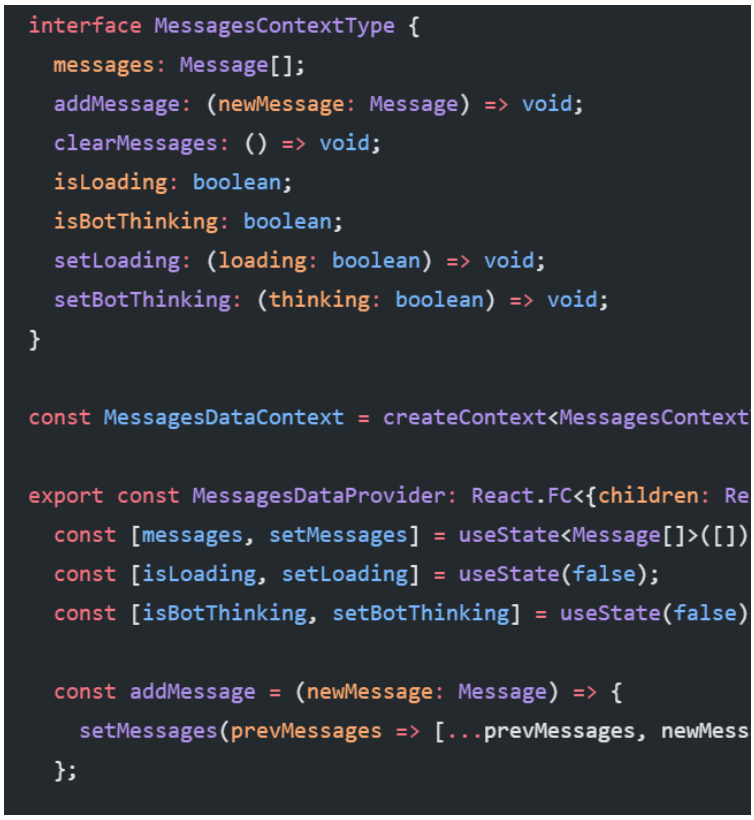
12.1. MensajeDataContext.tsx

Este componente de React es responsable de administrar y proporcionar datos de mensajería dentro de un contexto. Analicemos los elementos clave:

- **Dependencias** :
    - react:Este componente utiliza ganchos y contextos de React.
- **Ganchos** :
    - useState:Se utiliza para gestionar el estado de messages, isLoading, y isBotThinking.
    - createContexty useContext: Se utiliza para crear y acceder al contexto de los mensajes.
- **Tipos personalizados** :
    - Message:Define la estructura de un objeto de mensaje, incluidas propiedades como message, role, type, y file.
    - MessagesContextType:Describe la forma del valor de contexto, incluida la matriz de mensajes, funciones para agregar/borrar mensajes, estados de carga y funciones para configurar la carga y el pensamiento del bot.
- **Componente** :
    - **MessagesDataProvider:** Este componente es un proveedor que encapsula la lógica para administrar el estado de los mensajes y se la proporciona a sus elementos secundarios a través del contexto. Se utiliza useStatepara administrar los mensajes, el estado de carga y el estado de pensamiento del bot. Expone addMessage y clearMessages funciona para manipular los mensajes. El proveedor envuelve a sus elementos secundarios con el proveedor de contexto, pasando los valores y funciones necesarios como valor de contexto.
        - **Accesorios:**
            - **children:** Los componentes secundarios que tendrán acceso al contexto de los mensajes.

- **Contexto** :
    - MessagesDataContext: El contexto creado createContextcon el MessagesContextType como valor inicial. Contiene el estado y las funciones relacionadas con los mensajes.
- **Ganchos personalizados** :
    - useMessagesData: Un gancho personalizado diseñado para acceder a los valores de contexto de los mensajes. Se utiliza useContextpara recuperar el contexto. Si el contexto es undefined, lo que significa que el gancho se utiliza fuera de MessagesDataProvider, se genera un error.

Esta configuración permite que los componentes de la jerarquía de React accedan a los datos de mensajería y a las funciones relacionadas que proporciona a MessagesDataProvidertravés del useMessagesDatagancho. Refuerza la encapsulación y la separación de preocupaciones al centralizar la gestión del estado de la mensajería en un solo lugar. El componente está estructurado de una manera que promueve la reutilización y la capacidad de mantenimiento de las funcionalidades relacionadas con la mensajería.



```
interface MessagesContextType {
  messages: Message[];
  addMessage: (newMessage: Message) => void;
  clearMessages: () => void;
  isLoading: boolean;
  isBotThinking: boolean;
  setLoading: (loading: boolean) => void;
  setBotThinking: (thinking: boolean) => void;
}

const MessagesDataContext = createContext<MessagesContext

export const MessagesDataProvider: React.FC<{children: Re
  const [messages, setMessages] = useState<Message[]>([])
  const [isLoading, setLoading] = useState(false);
  const [isBotThinking, setBotThinking] = useState(false)

  const addMessage = (newMessage: Message) => {
    setMessages(prevMessages => [...prevMessages, newMess
  };
```

12.2. Texto o contexto de voz.tsx

Este componente de React se utiliza para administrar y brindar contexto para un valor de estado de modo que especifica el tipo de contenido (texto, audio o imagen) que se está procesando y la URL de API correspondiente. Consta de las siguientes partes:

Interfaz ModeState :

- Esta interfaz define la forma del objeto de estado de modo, especificando que debe tener una typepropiedad con un valor de 'texto', 'audio' o 'imagen' y una apiUrlpropiedad que contenga una cadena que represente la URL de la API para el modo elegido.

Interfaz TextOrVoiceContextType :

- Esta interfaz define la estructura del valor de contexto que proporcionará el proveedor de contexto.

Texto o contexto de voz :

- Este es el contexto de React creado con createContext. Contiene el estado y los métodos relacionados con el contexto del modo de texto o voz.

Gancho useTextOrVoice :

- Este gancho personalizado useTextOrVoicese creó para acceder fácilmente a los valores de contexto dentro de los componentes funcionales mediante useContext. Arroja un error si se usa fuera de TextOrVoiceProvider.

Componente TextOrVoiceProvider :

- Este componente es un proveedor que envuelve a sus elementos secundarios con el proveedor TextOrVoiceContext. Inicializa el estado del modo mediante el useStategancho con el modo predeterminado establecido en "texto" y la URL de API predeterminada.
- Proporciona una setModefunción que actualiza el estado del modo en función del tipo que se le pasa (audio, texto o imagen).

Interfaz TextOrVoiceProviderProps

- Esta interfaz define los tipos de propiedad esperados por el TextOrVoiceProvidercomponente, que incluye la childrenpropiedad de tipo ReactNode.

Uso:

- Para utilizar el contexto en un componente, envuelva la parte del árbol de componentes que necesita acceso al estado del modo y a la función setMode con TextOrVoiceProvider.
- Para acceder al estado del modo y setModela función dentro de un componente, utilice el useTextOrVoicegancho.

Dependencias externas:

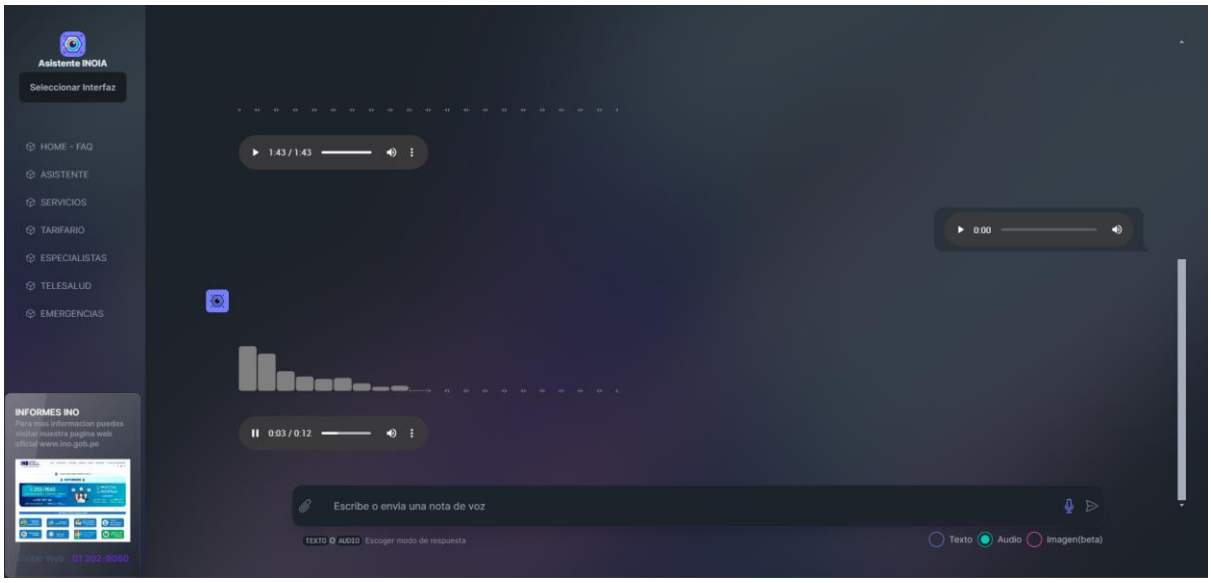
- Este componente depende de React para su funcionalidad y no tiene ninguna dependencia externa más allá del propio React.

```
'use client';
import React, { createContext, useContext, useState, React

// Define the mode state interface
interface ModeState {
  type: 'text' | 'audio' | 'image';
  apiUrl: string;
}

// Define the context type
interface TextOrVoiceContextType {
  mode: ModeState;
  setMode: (type: ModeState['type']) => void;
}

// Create the context
const TextOrVoiceContext = createContext<TextOrVoiceConte

// Export a hook to use the context
export const useTextOrVoice = () => {
  const context = useContext(TextOrVoiceContext);
  if (!context) throw new Error('useTextOrVoice must be u
  return context;
};
```

Interfaz del asistente Virtual en Producción

1. Home

2. Chat



Anexo 5. Carta de autorización

AUTORIZACIÓN PARA LA REALIZACIÓN Y DIFUSIÓN DE RESULTADOS

Yo, FÉLIX ANTONIO TORRES COTRINA, identificado con DNI N° 08134606, en mi calidad de DIRECTOR GENERAL EN OFTALMOLOGÍA Y REFRACCIÓN del área de CONSULTORIOS GENEREALES del Instituto Nacional de Oftalmología, con RUC N° 20131381094, ubicado en Av. Tingo María 398, Distrito de Cercado de Lima, Provincia de Lima, Departamento de Lima Metropolitana.

OTORGO LA SIGUIENTE AUTORIZACIÓN:

A Jaime Alexander Tarazona Rodríguez, identificado con DNI N° 70479656, estudiante de la Carrera Profesional de Ingeniería de Sistemas, para que realice su investigación titulada "Asistente virtual con inteligencia artificial para optimizar la satisfacción del paciente del Instituto Nacional de Oftalmología", y para que difunda los resultados obtenidos con la finalidad de desarrollar su proyecto de investigación.

Indicar si el Representante autoriza:

() Mantener en reserva el nombre o cualquier distintivo de la institución o
(x) Mencionar el nombre de la institución.

Firma y sello del Representante
DNI: 08134606
Médico Cirujano C.M.P. 38356
Especialista en Oftalmología R.N.E. 17309

Firma del Estudiante
DNI: 70479656