
**ESCUELA SUPERIOR POLITÉCNICA AGROPECUARIA DE MANABÍ MANUEL FÉLIX LÓPEZ**

## CARRERA DE COMPUTACIÓN

**INFORME DE TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACIÓN**

**MECANISMO: SISTEMATIZACIÓN DE EXPERIENCIAS PRÁCTICAS DE INVESTIGACIÓN Y/O INTERVENCIÓN**

**TEMA:**

# DESARROLLO DE UNA INTERFAZ CONVERSACIONAL INTEGRADA A UN ASISTENTE VIRTUAL

**AUTORES:**

**JÉSSICA JOHANA MONTES VERA**
**CARLOS PIERRE QUIJIJE VERA**

**TUTOR:**

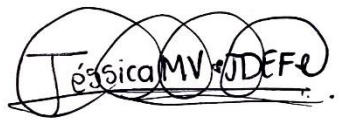
**ING. FERNANDO RODRIGO MOREIRA MOREIRA, MGTR.**

**CALCETA, OCTUBRE DE 2023**

# DECLARACIÓN DE AUTORÍA

Nosotros **JÉSSICA JOHANA MONTES VERA** y **CARLOS PIERRE QUIJIJE VERA**, con cédulas de ciudadanía 1351043961 y 1316462124 respectivamente, declaramos bajo juramento que el Trabajo de Integración Curricular titulado: "**DESARROLLO DE UNA INTERFAZ CONVERSACIONAL INTEGRADA A UN ASISTENTE VIRTUAL**" es de nuestra autoría, que no ha sido previamente presentado para ningún grado o calificación profesional, y que hemos consultado las referencias bibliográficas que se incluyen en este documento.

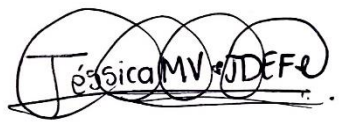
A través de la presente declaración, concedemos a favor de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos, conservando a nuestro favor todos los derechos patrimoniales de autor sobre la obra, en conformidad con el Artículo 114 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación.



| | |
|---|---|
| **JÉSSICA J. MONTES VERA** | **CARLOS P. QUIJIJE VERA** |
| **CC: 1351043961** | **CC: 1316462124** |

# AUTORIZACIÓN DE PUBLICACIÓN

**JÉSSICA JOHANA MONTES VERA** y **CARLOS PIERRE QUIJIJE VERA**, con cédulas de ciudadanía 1351043961 y 1316462124 respectivamente, autorizamos a la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, la publicación en la biblioteca del Trabajo de Integración Curricular titulado: "**DESARROLLO DE UNA INTERFAZ CONVERSACIONAL INTEGRADA A UN ASISTENTE VIRTUAL**", cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y total autoría.



**JÉSSICA J. MONTES VERA**
**CC: 1351043961**

**CARLOS P. QUIJIJE VERA**
**CC: 1316462124**

# CERTIFICACIÓN DEL TUTOR

**MGTR. FERNANDO RODRIGO MOREIRA MOREIRA**, certifica haber tutelado el Trabajo de Integración Curricular titulado: "**DESARROLLO DE UNA INTERFAZ CONVERSACIONAL INTEGRADA A UN ASITENTE VIRTUAL**", que ha sido desarrollada por **JÉSSICA JOHANA MONTES VERA** y **CARLOS PIERRE QUIJIJE VERA**, previo a la obtención del título de Ingeniero en Ciencias de la Computación de acuerdo al **REGLAMENTO DE LA UNIDAD DE INTEGRACIÓN CURRICULAR DE CARRERAS DE GRADO** de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López.

Firmado electrónicamente por:
**FERNANDO RODRIGO MOREIRA MOREIRA**

**MGTR. FERNANDO R. MOREIRA**

**CC: 1311726689**

**TUTOR**

# APROBACIÓN DEL TRIBUNAL

Los suscritos integrantes del Tribunal correspondiente, declaramos que hemos **APROBADO** el Trabajo de Integración Curricular titulado: "**DESARROLLO DE UNA INTERFAZ CONVERSACIONAL INTEGRADA A UN ASITENTE VIRTUAL**", que ha sido desarrollado por **JÉSSICA JOHANA MONTES VERA** y **CARLOS PIERRE QUIJIJE VERA**, previo a la obtención del título de "**INGENIERO EN CIENCIAS DE LA COMPUTACIÓN**", de acuerdo al **REGLAMENTO DE LA UNIDAD DE INTEGRACIÓN CURRICULAR DE CARRERAS DE GRADO** de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López.

Firmado electrónicamente por:
LUIS CRISTOBAL CEDENO VALAREZO

**MGTR. LUIS C. CEDEÑO VALAREZO**
**CC: 1306246651**
**PRESIDENTE DEL TRIBUNAL**

ANGEL ALBERTO VELEZ MERO
Firma digital.
avelez@espam.edu.ec

**MGTR. ÁNGEL A. VÉLEZ MERO**
**CC: 1308648565**
**MIEMBRO DEL TRIBUNAL**

Firmado digitalmente por ALFONSO TOMAS LOOR VERA

**MGTR. ALFONSO T. LOOR VERA**
**CC: 1311655938**
**MIEMBRO DEL TRIBUNAL**

# AGRADECIMIENTO

A la prestigiosa Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López que nos brindó la oportunidad de forjar nuestros conocimientos profesionales con compromiso ético y social, permitiendo de esta manera una excelente educación de calidad.

Al magister Fernando Rodrigo Moreira Moreira, tutor de nuestro trabajo de titulación, por la contribución y guía, no solo en la elaboración del mismo, sino a lo largo de nuestra carrera universitaria y habernos brindado todo el conocimiento enriquecedor para potenciar nuestras habilidades informáticas.

Al Mgtr. Joffre Moreira Pico, director de la Carrera de Computación por su motivación y apoyo en nuestro proceso universitario.

A la Ing. Jessica Morales Carrillo, por ser una excelente guía en los dos últimos semestres de estudio de la carrera.

A los distinguidos docentes, que durante el transcurso de la carrera nos compartieron conocimientos significativos, consejos, amistad y tiempo a lo largo de la preparación de nuestra profesión. De manera especial, a los docentes: Mgtr. Tomás Loor Vera, Ing. Yimmy Loor Vera, y Ph.D. Javier López Zambrano, profesionales con gran calidad humana que en colaboración nos ayudaron a esclarecer dudas que suscitaron durante el proceso de nuestro trabajo de titulación.

**LOS AUTORES**

# DEDICATORIA

A mi Dios amado, por su amor inefable, el milagro de vivir, ser la fuente de mi inspiración, por brindarme su paz y bendecir mi camino. Él me enseñó por medio de Jesús mi Señor y Salvador, que el amor es sacrificado y servicial, que los planes tienen éxito cuando confiamos en Su Voluntad y que las recompensas son más grandes en el cielo.

A mi hermosa madre Sra. Carmen Vera Loor, por su fortaleza y amor incondicional, creer en mí y ser mi apoyo constante, por estar siempre presente en cada meta de mi vida, ella es mi motivación para seguir adelante. A mi querido padre Sr. José Montes Moreira, por brindarme su cariño y apoyarme en mis estudios, por enseñarme con entusiasmo a ser un buen líder y trabajar con esmero y dedicación.

A mi recordada abuelita y gran mujer, mi bella Sra. Argentina del Jesús Loor Loor, por enseñarme a vivir con valentía, humildad y alegría. A mi abuelito Sr. Humberto Vera Mendoza, a quien recuerdo con mucho amor como “Mi Tito”, por sus sabios consejos e instruirme en la fe y los valores. Ellos me dieron amor del bueno y sincero, conservo por siempre los mejores recuerdos en mi corazón.

A mis hermanos: Génesis Montes Vera y Yandry Cedeño Vera, por las risas, el cariño y la complicidad. Ellos siempre me animaron y creyeron en mi potencial. A mis hermosas sobrinas Ashley y Amy Cedeño, por los dulces abrazos y enseñarme a tener esperanza. Ellas son el amor más puro que tengo.

A mi lorito y amigo de la infancia, mi Erick que añoro y recuerdo con gran cariño. A toda mi querida familia, por darme ánimos y compartir mis pequeños y grandes logros. A cada una de las personas que en el transcurso de mi vida me brindaron amistad, empatía y momentos inolvidables. No terminaría de mencionarlas, por eso quiero expresar que fueron de gran bendición en mi vida.

**JÉSSICA JOHANA MONTES VERA**

# DEDICATORIA

A mi madre, por ser la persona más importante y por demostrarme siempre su cariño y apoyo incondicional. A mi padre, por apoyarme constantemente y brindarme valiosos consejos.

A mis hermanos, por estar siempre apoyándome en las diferentes etapas de este proceso universitario. A mi novia que siempre me apoyó en los momentos más difíciles.

A todas aquellas personas, que gracias a su apoyo me permitieron permanecer con empeño y dedicación para lograr culminar mis estudios.

**CARLOS PIERRE QUIJIJE VERA**

# TABLA DE CONTENIDO

## CONTENIDO DE TABLAS

# CONTENIDO DE FIGURAS

## RESUMEN

El presente trabajo de titulación, consistió en desarrollar una interfaz conversacional que se integre con un asistente virtual para que pueda brindar una mejor interacción con el usuario y las aplicaciones de la Unidad de Docencia, Investigación y Vinculación (UDIV) de Infraestructuras correspondientes a la Carrera de Computación. Para cumplir con la ejecución del mismo, fue necesario emplear la técnica de la entrevista para recabar todos los datos esenciales para el desarrollo del producto, así mismo se utilizaron los métodos: Bibliográficos y Extreme Programming. Por medio del método bibliográfico se definieron las técnicas asociadas para el desarrollo del producto tecnológico. Por lo consiguiente, al utilizar la metodología Extreme Programming, por medio de la fase de diseño se llevó a cabo el prototipado y desarrollo de la interfaz, así pues permitiendo la interacción entre el usuario y el sistema; posteriormente se llevó a cabo la fase de codificación, la cual permitió que se acoplaran las APIs que proveyó el asistente para establecer el contrato de conexión y de esta manera integrarse con las aplicaciones de la UDIV de Infraestructura; en la última fase de pruebas unitarias se determinó el comportamiento de la interfaz con el asistente virtual. Con el procedimiento antes indicado, se ejecutaron las pruebas finales de producción realizadas a un grupo de 50 personas y se determinó que el 90% de los usuarios que realizaron las peticiones, la interfaz mostraba de manera correcta los resultados o coincidencias específicas de forma automatizada con base en las necesidades del usuario final.



## PALABRAS CLAVE

Node.js, websockets, APIs, sistema web, aplicaciones.

## ABSTRACT

The present degree work consisted of developing a conversational interface that integrates with a virtual assistant so that it can provide better interaction with the user and the applications of the Teaching, Research and Liaison Unit (UDIV) of Infrastructures corresponding to the Computer Major. To comply with its execution, it was necessary to use the interview technique to collect all the essential data for the development of the product, likewise the following methods were used: Bibliographic and Extreme Programming. Through the bibliographic method, the associated techniques for the development of the technological product were defined. Therefore, by using the Extreme Programming methodology, through the design phase the prototyping and development of the interface was carried out, thus allowing the interaction between the user and the system; subsequently, the coding phase was carried out, which allowed the APIs provided by the wizard to be coupled to establish the connection contract and thus integrate with the applications of the Infrastructure UDIV; in the last phase of unit tests, the behavior of the interface with the virtual assistant was determined. With the procedure indicated above, the final production tests were executed on a group of 50 people and it was determined that 90% of the users who made the requests, the interface correctly showed the results or specific matches in an automated way based on the needs of the end user.



## KEYWORDS

Node.js, websockets, APIs, web system, applications.

# CAPITULO I. ANTECEDENTES

## 1.1. DESCRIPCIÓN DE LA INSTITUCIÓN

La Escuela Superior Politécnica Agropecuaria de Manabí "Manuel Félix López" está localizada en la ciudad de Calceta del cantón Bolívar, perteneciente a la provincia de Manabí; fue fundada en abril de 1999 cuyo propósito es de participar junto a otras entidades, en el auge y avance de la provincia de Manabí y del país, a través de la enseñanza universitaria, la investigación científica y el emprendimiento (ESPAM MFL, 2016).

La filosofía de la institución tiene como misión: “formar profesionales pertinentes con compromisos ético y social, desde la calidad de los procesos sustantivos” (ESPAM MFL, 2022). Asimismo, la visión de la institución es “ser un centro de referencia en la formación de profesionales que contribuyan al desarrollo agropecuario regional” (ESPAM MFL, 2019).

La ESPAM MFL, cuenta con varias ofertas académicas de tercer nivel y entre ellas se encuentra la Carrera de Computación que tiene como objetivo: “Formar profesionales que aporten innovaciones computacionales para la solución de problemas sociales, regionales y nacionales, vinculados al modelo constructivista y desarrollador productivo, dentro de equipos multidisciplinares e interdisciplinares, con énfasis en el sector agropecuario y agroindustrial, que actúen con responsabilidad económica, ambiental, ética y social, en sintonía con los planes y políticas públicas” (ESPAM MFL, 2023).

En la carrera de Computación existe la Unidad de Docencia, Investigación y Vinculación (UDIV) de Infraestructuras, la cual es la encargada de ofrecer una formación sólida, teórica, metodológica y práctica a los estudiantes de la ESPAM MFL en el análisis de problemas relacionados con los procesos del computador, auditoría y redes (UDIV de Infraestructura, 2022).

## 1.2. DESCRIPCIÓN DE LA INTERVENCIÓN

La influencia que ha comenzado a adquirir la Interacción Humano - Computador (IHC) junto al de usabilidad para el desarrollo de aplicaciones efectivas, eficientes y que satisfacen el gusto de los usuarios, compactando con ello el concepto de calidad, ha abierto a debate la finalidad que le corresponde a la ingeniería de software como el marco conceptual más influyente en la construcción de aplicaciones que automatizan procesos y que a su vez son considerados de calidad (Toledo et al., 2018). Por su esencia, es un área multidisciplinaria donde convergen diversos tipos de especialistas (Molero et al., 2017). Justamente por ello, este nuevo rol trae como consecuencia la importancia cada vez más creciente de las interfaces y, de manera general, de la Interacción Humano - Computador (Gamboa, 2019) (Chanchí et al., 2019).

De acuerdo con Madou et al. (2020), la interfaz utiliza un conjunto de componentes visuales, interactivos y de diseño para representar la información y acciones disponibles. El desarrollo de estos sistemas interactivos se ha dado en gran medida gracias al avance y aplicación de técnicas de Inteligencia Artificial (Martínez & Cely, 2018).

El uso de la interfaz conversacional conectado a un asistente virtual, permite que los tutores y estudiantes obtengan atención rápida, precisa y eficaz a sus necesidades de información o solicitudes (Alfaro, 2022), mejorando así el proceso de atención al usuario con respecto a fomentar la identidad y credibilidad de la institución mediante el uso de la tecnología. Coincidiendo con Chen et al. (2020) citado por Ogosi (2021), “los avances en tecnologías informáticas, en particular la inteligencia artificial, los sistemas informáticos son capaces de proporcionar apoyos educativos de una manera más amigable e inteligente”.

Hoy en día, las aplicaciones que se trabajan en la Unidad de Docencia, Investigación y Vinculación (UDIV) de Infraestructuras pertenecientes a la carrera de Computación, no cuentan con la disposición de componentes interactivos de comunicación entre las aplicaciones y los usuarios que permitan resolver los problemas de la interacción entre el humano y el computador, y a su vez encontrar soluciones que mejoren la usabilidad del producto software. Esto

trae como consecuencia la importancia de optar por el desarrollo de las interfaces que son el medio central entre la Interacción Humano - Computador.

Precisamente se usan las interfaces conversacionales conjuntamente con asistentes virtuales, para permitir al usuario una comunicación directa con el sistema informático mediante comandos de voz o texto, para obtener información.

Justamente por ello, surge la necesidad de integrar estas tecnologías a partir del planteamiento de innovar nuevos métodos de interacción entre el usuario y las aplicaciones existentes en la institución por medio de la inteligencia artificial y sistemas conversacionales. Por lo tanto, se planteó este proyecto, el cual consiste en desarrollar una interfaz conversacional integrada a un asistente virtual, de manera que impulse a la institución a mejorar la calidad educativa y aprovechar los grandes avances de la tecnología.

La interfaz conversacional se desarrollará bajo el uso de Node.js como entorno de ejecución, ya que permite utilizar un solo lenguaje para programar tanto el cliente como el servidor (Flores, 2017). Fue creado en 2009 por Ryan Dahl, diseñado para construir aplicaciones en red escalables y de rápida comunicación y ejecución de las entradas y salidas (Guardado, 2017). También se optó por utilizar Express, el componente de Node.js que se encarga de la gestión de conexiones HTTP, lo que proporciona un carácter de servidor web a Node (Peralbo, 2019) (Carrillo, 2016).

De este modo, la implementación de la interfaz conversacional en las aplicaciones correspondientes a la UDIV de Infraestructura de la Carrera de Computación, tales como, Gestión de Tareas y Administración de Espacios; permitirá que los usuarios realicen las peticiones de forma escrita al asistente virtual desarrollado en el Trabajo de Integración Curricular Complementario denominado “Desarrollo de un asistente virtual empleando técnicas de comprensión de lenguaje natural” por los estudiantes Jesús Stefano Cajape Bravo y Sandro Antonio Palau Delgado, y de esta manera obtener los resultados específicos solicitados para cubrir las necesidades del usuario dentro de la ESPAM MFL.

## 1.3. OBJETIVOS

### 1.3.1. OBJETIVO GENERAL

Desarrollar una interfaz conversacional que se integre con un asistente virtual para brindar interacción entre el usuario y las aplicaciones correspondientes al área de la UDIV de Infraestructura de la Carrera de Computación de la ESPAM MFL.

### 1.3.2. OBJETIVOS ESPECÍFICOS

- Establecer los requisitos e información específica para la interfaz conversacional.
- Identificar las técnicas asociadas para el desarrollo de interfaces conversacionales que estén integradas con asistentes virtuales.
- Diseñar la interfaz conversacional mediante el uso de un framework de desarrollo.
- Integrar el asistente virtual a la interfaz conversacional.
- Implementar la interfaz conversacional en las aplicaciones de la UDIV de Infraestructura, previo al proceso de evaluación.

# CAPITULO II. DESARROLLO METODOLÓGICO DE LA INTERVENCIÓN

Para la ejecución del presente trabajo de titulación se lo realizó a través del cumplimiento de cada fase de la metodología Extreme Programming (XP). De acuerdo a Jiménez et al. (2019), las fases de la metodología XP son cuatro: Planificación, Diseño, Codificación y Pruebas.

## 2.1. EXTREME PROGRAMMING

Como dice Sánchez et al. (2020) “la metodología de desarrollo XP captura los requerimientos en un alto nivel, se basa en iteraciones, y es en cada iteración donde se profundiza en los requerimientos, además busca desarrollar aplicaciones enfocadas en los usuarios por lo que se fomenta la comunicación y la integración de usuarios expertos al equipo de desarrollo”. En este sentido, el desarrollo de la interfaz conversacional se realizó con base en la metodología Extreme Programming o XP (por sus siglas en inglés), cuyo requirente es la UDIV de Infraestructura de la Carrera de Computación, ya que el funcionamiento engloba un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades estructurales que se adaptan perfectamente a las necesidades del proyecto, las cuales son: planificación, diseño, codificación y pruebas (Jiménez et al., 2019). A continuación, se describen las actividades de los objetivos que se realizaron en las fases que componen esta metodología de desarrollo de software.

### 2.1.1 PLANIFICACIÓN

En esta fase se desarrollaron los dos primeros objetivos del proyecto; como primer punto, “Establecer los requisitos e información específica para la interfaz conversacional”, y luego, “Identificar las técnicas asociadas para el desarrollo de interfaces conversacionales que estén integradas con asistentes virtuales”. En consecuencia, se realizó una visita técnica en el área requirente ubicada en la UDIV de Infraestructura de la Carrera de Computación. Dicho lo anterior, se optó por seguir lo recomendado por la técnica de la entrevista, dado que, según

Fonseca (2017), es un instrumento técnico que adopta la forma de un diálogo coloquial, y sirve para obtener información sobre el tema de investigación.

Por lo consiguiente, se hizo una entrevista informal a dos docentes de la carrera: Ing. Javier López, quien fue el anterior responsable del área, y el Ing. Yimmy Loor, quien actualmente tiene el cargo del área antes mencionada. Así pues, se describió la funcionalidad y procedimientos requeridos para la realización de la interfaz conversacional integrada a un asistente virtual. Este conversatorio también permitió establecer el consentimiento de acceso a la información de las aplicaciones conocidas como Gestión de Tareas y Administración de Espacios, pertenecientes a la UDIV de Infraestructura de la Carrera de Computación.

Luego, se definió los requisitos funcionales y no funcionales del producto mediante las historias de usuario, para poder describir las características y funcionalidades de la interfaz conversacional. Para esto, se utilizó un formato conforme con el estándar IEEE 830, el cual permitió obtener un listado de requerimientos desde el punto de vista del usuario, cliente y desarrollador.

Por lo que sigue, se empezó a realizar el segundo objetivo del proyecto (mencionado al inicio de la planificación), conforme a ello también fue necesario realizar una revisión bibliográfica en la que se buscó información en diferentes repositorios de base de datos científicos como Redalyc, Scielo, Springer Link, entre otros; posteriormente, se escogieron artículos desde el año 2018 hasta la actualidad sobre interfaces conversacionales y asistentes virtuales utilizadas, sobre todo, en el campo de la educación para la optimización de procesos.

Además, toda esta información obtenida es resumida en un cuadro comparativo en el que se especificaba el título de la investigación, año, técnica empleada, lenguaje de programación y métricas de calidad; con el fin de determinar las herramientas tecnológicas a utilizar en el proceso del desarrollo de la interfaz conversacional.

### 2.1.2 DISEÑO

Para empezar la fase de diseño, se fueron desarrollando los Diagramas UML de comportamiento a partir del listado de requerimientos mencionados en la fase

anterior, tales como: Diagramas de Casos de Uso y Diagramas de Arquitectura, ya que representan la forma en la que se comporta la interfaz conversacional y cómo este interactúa dentro de sí mismo, así como con los usuarios u otros sistemas.

Mediante el proceso de diseño se desarrolló el tercer objetivo “Diseñar la interfaz conversacional mediante el uso de un framework de desarrollo”, donde se analizó el levantamiento de requerimientos para determinar el funcionamiento de la interfaz, del mismo modo se analizaron los diagramas UML de comportamiento.

En consecuencia, se realizaron prototipos de la interfaz conversacional mediante el uso de frameworks. Estos mismos abarcan todos los requerimientos específicos del producto, y todo esto logró obtener una visión general de cómo se visualiza el producto final.

### 2.1.3 CODIFICACIÓN

Al momento de codificar es aún más necesaria la participación del cliente en el equipo de desarrollo, debido a que son los que crean las historias de usuario y especifican los tiempos de implementación. Las historias de usuario permiten a los equipos de desarrollo codificar la interfaz con un enfoque centrado en el usuario. Esto significa que los equipos pueden diseñar diálogos útiles para los usuarios, así como cada paso del proceso de conversación, en función de la información recopilada de las historias de usuario.

Con base en la fase de codificación, se emplearon algunas herramientas tecnológicas, como por ejemplo Visual Studio Code (VSC) que permite trabajar con diversos lenguajes de programación. También, se utilizaron varios frameworks para el desarrollo del frontend y backend de la interfaz conversacional según las necesidades del usuario. Adicionalmente, en esta fase también se acoplaron las APIs que el asistente virtual proveyó, así como la integración de las aplicaciones de la UDIV de Infraestructura (conocidas como Gestión de Tareas y Administración de Espacios) mediante el desarrollo de algoritmos a la interfaz conversacional.

Del mismo modo, para consumir al asistente virtual se utilizó WebSocket, una tecnología avanzada para crear una conexión entre un cliente y un servidor, y permitir la comunicación entre ellos en tiempo real (Gómez, 2018).

### 2.1.4 PRUEBAS

Las pruebas unitarias del producto, según lo expuesto por Cárdenas & Quimbita (2017), “se realizan con el fin de lograr el cumplimiento de los objetivos, y se lo realizan en un marco de trabajo que permita automatizar los procesos para la validación de datos y realizar pruebas de aceptación, integración y validaciones diarias que permitan corregir los errores conforme vayan apareciendo en las respectivas pruebas realizadas a cada módulo del sistema”. Por supuesto, al realizar las pruebas unitarias de la interfaz conversacional, permitió detectar y corregir los fallos que se presentaban en la fase de codificación.

De este modo, al estar integrado el asistente virtual a la interfaz conversacional se procedió a realizar las pruebas del funcionamiento del producto final para determinar que se cumplan todas las especificaciones correspondientes. Adicional, se presentó un informe de evaluación de pruebas, así como los manuales de programador y de usuario.

En consecuencia, una vez aprobado el proceso de evaluación del producto se procedió a implementar la interfaz conversacional con el asistente virtual en la UDIV de Infraestructura para optimizar los procesos y cubrir las necesidades de los usuarios.

# CAPITULO III. DESCRIPCIÓN DE LA EXPERIENCIA

## 3.1. PLANIFICACIÓN

Para la ejecución del trabajo de titulación sobre el desarrollo de la interfaz conversacional integrada a un asistente virtual, se procedió a realizar la primera actividad, el cual consistió en solicitar la colaboración de los estudiantes encargados del desarrollo del presente proyecto dentro de la Unidad de Docencia, Investigación y Vinculación de Infraestructuras, y así atender la necesidad de la integración entre un asistente virtual y aplicativos de la carrera, y por medio de un oficio dirigido al director de la Carrera de Computación (**Anexo 1**), permitió establecer el consentimiento de acceso a la información de las aplicaciones conocidas como Gestión de Tareas y Administración de Espacios.

Conforme a ello, se desarrolló la entrevista informal y se recabaron los datos obtenidos en este conversatorio para su posterior análisis, donde se establecieron los requisitos funcionales y no funcionales que se resumen en las Tablas 3.1 y 3.2 respectivamente. Este proceso está basado en el formato del estándar IEEE 830 (**Anexo 2**), de acuerdo con la necesidad de optimización de procesos a solicitudes por parte de los usuarios dentro de las aplicaciones mencionadas en el párrafo anterior pertenecientes a la UDIV de Infraestructura.

**Tabla 3. 1** Requisitos funcionales.

| Número de requisito | Nombre de requisito |
|---|---|
| **RF01** | Interacción en tiempo real |
| **RF02** | Usuarios y solicitudes |
| **RF03** | Modo oscuro |
| **RF04** | Configurar las alertas de notificaciones |
| **RF05** | Permitir funcionalidades ocultas |
| **RF06** | Envío de mensajes |
| **RF07** | Lenguaje de los mensajes |
| **RF08** | Revisión ortográfica |

| | |
|---|---|
| **RF09** | Estado de notificaciones |
| **RF010** | Gestión de conversación |
| **RF011** | Emisión de enlaces |
| **RF012** | Integración con las aplicaciones |
| **RF013** | Seguimiento de la conversación |
| **RF014** | Denunciar fallos en respuestas |

***Fuente.*** Los Autores.

**Tabla 3. 2** Requisitos no funcionales.

| Número de requisito | Nombre de requisito |
|---|---|
| **RNF01** | Interfaz del Sistema |
| **RNF02** | Ayuda en el uso del sistema |
| **RNF03** | Seguridad y Lógica de Datos |
| **RNF04** | Seguridad |
| **RNF05** | Fiabilidad |
| **RNF06** | Disponibilidad |
| **RNF07** | Contraste |
| **RNF08** | Adaptar interfaz a diferentes dispositivos |
| **RNF09** | Longitud de los mensajes |

***Fuente.*** Los Autores.

A continuación, se llevó a cabo la búsqueda de información bibliográfica en diferentes repositorios de base de datos científicos, como: Redalyc, Scielo, Springer Link, entre otros; luego, se escogieron artículos desde el año 2018 hasta la actualidad relacionados al presente proyecto. Además, toda esta información obtenida es resumida en un cuadro comparativo, la misma que se muestra en la Tabla 3.3, y en la que se detalla una referencia sobre la formulación de los campos necesarios para realizar la clasificación de información propuesta en la revisión bibliográfica.

**Tabla 3. 3** Información Bibliográfica.

| ID | Año | Titulo | Autor | Entorno de la página web | Tipo de herramienta para integrar el asistente al software |
|---|---|---|---|---|---|
| 1 | 2017 | "Desarrollo e implementación de un sistema web para mejorar la administración de los procesos internos y el servicio al cliente de la pyme gráficas Rivas, implementando también una herramienta de inteligencia artificial chatbot." | Manuel Humberto Rivas Fuentes | Php 5.6 + framework codeigniter | Dialogflow |
| 2 | 2017 | "Desarrollo de un prototipo de aplicación web y móvil para la gestión de obras de instalación de fibra óptica." | Alejandro Pérez Martín | Node.js | - |
| 3 | 2017 | "Desarrollo de aplicaciones web con Node.js" | Jesús Octavio Guardado Osuna | Node.js | - |
| 4 | 2018 | "Introducción de un diseño de una plataforma virtual para la interacción entre docente y estudiante con la integración de un asistente virtual (chatbot); orientada a los estudiantes del 2do y 3ro de Bachillerato en la especialización de Informática del Colegio Fiscal Técnico Provincia de Bolívar." | Mario Enrique Santos Méndez | Laravel | Dialogflow |
| 5 | 2018 | "Diseño de un servicio de respuesta automático sobre el gasto público de chile mediante un asistente virtual de interfaz conversacional." | Orlando Andrés Rojas Romero | Flutter | NODE RED |
| 6 | 2018 | "Implementación de un chatbot con Botframework: caso de estudio, servicios a clientes del área de fianzas de seguros equinoccial." | Omar Humberto Zarabia Zuñiga | - | C# y Visual Basic. |
| 7 | 2018 | "Desarrollo del sistema de requisiciones para la Empresa Hidroeléctrica Abanico S.A. aplicando el entorno de programación Node.js" | Diana Elizabeth Gómez García | Node.js | Python |
| 8 | 2019 | "Desarrollo de una aplicación web alternativa para videoconferencia y compartición de pantalla con el uso de WebRTC." | Peralbo Velasco Michelle Alejandra | Node.js y Express | WebRTC |
| 9 | 2020 | "Desarrollo e implementación de una plataforma web con chatbot para la comunicación activa entre usuario e información del portafolio de servicio de la empresa Electricsystems de la ciudad de Guayaquil." | Espinoza Hoyos Sonia Elizabeth | Angular | Dialogflow |
| 10 | 2020 | "Implementación de una aplicación web con servicio de chatbot con inteligencia artificial que permita la autogestión de cuentas por pagar de los proveedores de la Universidad Autónoma de Bucaramanga." | Julián David Nieto Cortés | Flutter | Dialogflow |
| 11 | 2020 | "Sistema Web con Asistente Chatbot aplicando la metodología ICONIX, para las Ventas Online en la Empresa Comercial Sandra SAC de Tarapoto." | Gutiérrez Pizarro, Gonzalo Daniel | Angular | - |
| 12 | 2021 | "Diseño y desarrollo de la interfaz de usuario de una aplicación web para la mejora de la salud basada en un chatbot." | Pablo Juan Fernández Cotrina | Angular | Dialogflow |
| 13 | 2021 | "Desarrollo e implementación de sitio web con agente virtual a través de un chatbot para la empresa “El Rey del Embrague S.A.”" | Coppiano Ramírez Jhonny Maverick | Wordpress | Chatcompose |
| 14 | 2021 | "Desarrollo e implementación de una página web con Chatbot, para el proceso de solicitud de exámenes de laboratorio de la empresa “SANLAC S.A.”" | Sánchez Díaz, Kléber Andrés | WordPress | Dialogflow |
| 15 | 2022 | "Sistema web con Chatbot para la gestión de ventas de productos de limpieza con aporte en quechua en la empresa Gea Chemical." | Chung Ku, Daniel Gustavo; Fiestas Ramirez,Gino Bismarck | Laravel | Dialogflow |
| 16 | 2022 | "Implementación de un chatbot para la gestión de incidentes en la Plataforma Virtual de Educación a Distancia de la Universidad Inca Garcilaso de la Vega." | Zavaleta Zegarra, Naysha Medalit | PHP | Dialogflow |

***Fuente.*** Los Autores.

Estas variables son útiles para analizar tendencias en el uso de chatbots en diferentes entornos de desarrollo web y para identificar las herramientas más populares para su implementación. Esta búsqueda de artículos permitió analizar que los autores de los trabajos han utilizado diferentes entornos de programación y herramientas de inteligencia artificial, como Angular, Flutter, Node.js, Express, Laravel, WordPress, Dialogflow, Node-RED y Python, entre otros. Con lo que se puede observar que existe una tendencia en la implementación de chatbots para mejorar la eficiencia y la satisfacción del cliente.

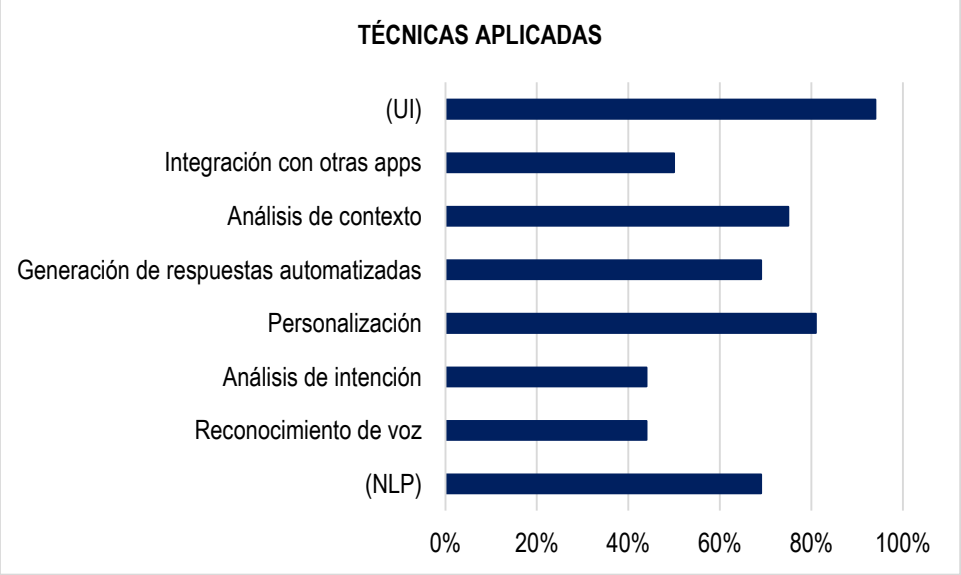
Además, los artículos recopilados compartían similares frameworks de desarrollo, y comparando nuestro alcance del proyecto, esto ayudó a determinar el uso de Node.js (entorno de tiempo de ejecución de JavaScript), ya que es uno de los más utilizados en la elaboración de aplicaciones escalables en red, el mismo en la que está desarrollado la interfaz conversacional. Además, una de las características clave de Node.js es su modelo de entrada/salida sin bloqueo y basado en eventos, que le permite ser altamente eficiente y adecuado para aplicaciones en tiempo real, como chats, juegos en línea, intercambio de archivos y aplicaciones colaborativas en línea. Esto se debe a que Node.js utiliza un solo subproceso de trabajo para procesar múltiples solicitudes, lo que lo hace ideal para aplicaciones que requieren una gran cantidad de interacciones simultáneas con el servidor.

Agregando a lo anterior, se procedió al análisis de las técnicas empleadas y métricas de calidad en los diferentes proyectos revisados de los artículos (**Anexo 3**). En el gráfico 1 se muestra el uso de diferentes tecnologías en el ámbito de procesamiento del lenguaje natural (NLP) y la interfaz de usuario (UI). Respecto a la interfaz de usuario, se hace referencia a dos tipos: Interfaz de preguntas y respuestas, que tiene como elemento simplificar procesos y responder a las expectativas del usuario; por otro lado, Interfaz gráfica de usuario (GUI).

- En el ámbito de NLP, se ve un uso mayor de tecnologías como el reconocimiento de voz, personalización, generación de respuestas automatizadas y análisis de contexto, con un uso promedio del 75%.
- Por otro lado, el análisis de intención y la integración con otras aplicaciones tienen un uso más bajo, con un promedio del 44%.

- En el ámbito de UI, se ve un alto uso de la tecnología de integración con otras aplicaciones, con un promedio del 94%.

**Figura 3. 1** Análisis estadístico de las técnicas empleadas en los diferentes proyectos revisados de los artículos.



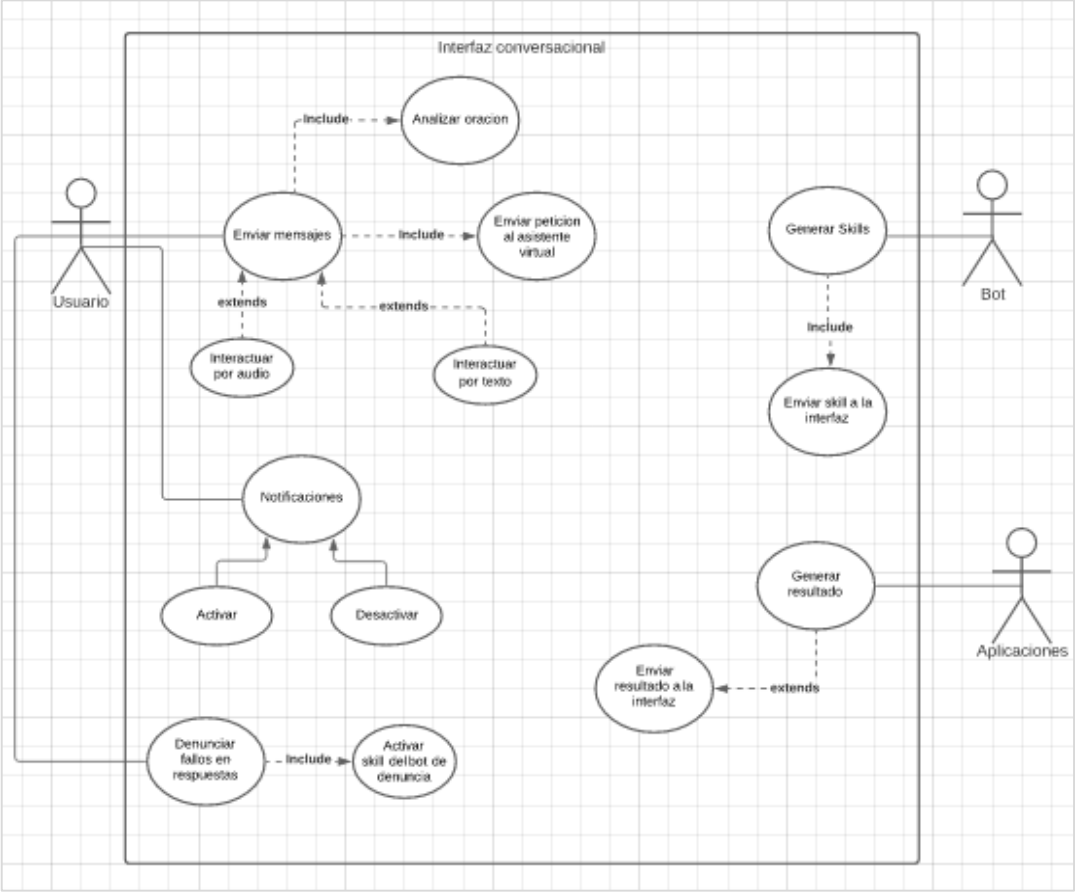

***Fuente.*** Los Autores.

En general, se puede resumir que se está haciendo un uso más intenso de tecnologías de NLP y UI para mejorar la interacción con los usuarios y la personalización de la experiencia del usuario.

## 3.2. DISEÑO

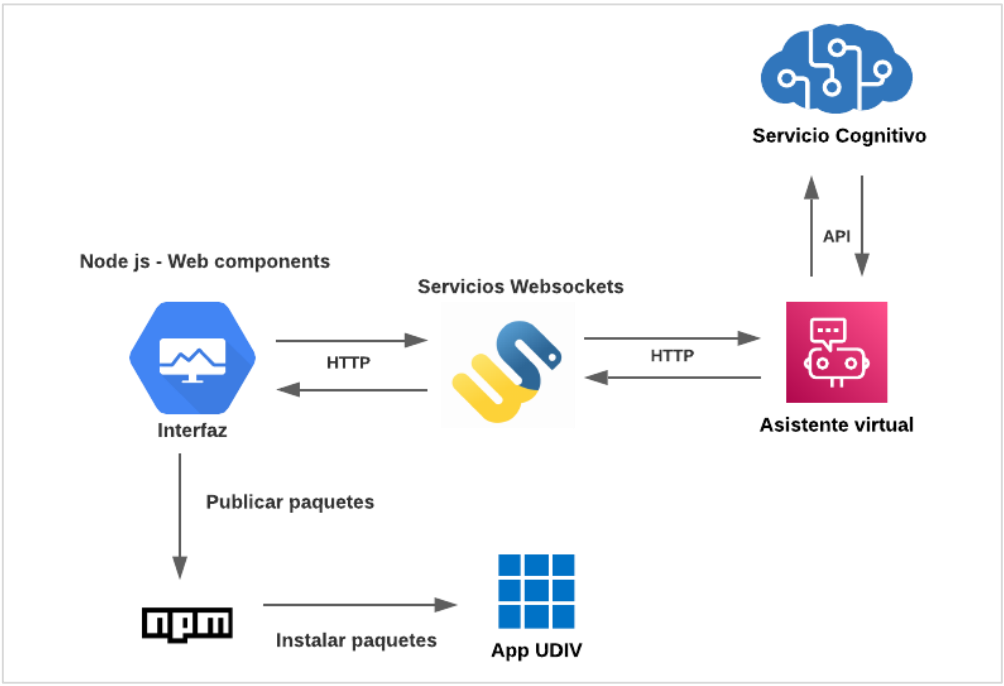
Con la obtención de los requisitos funcionales y no funcionales, se procedió a la elaboración de los diagramas UML de comportamiento con el fin de modelar la forma en que la interfaz conversacional interactúa con los usuarios y otros sistemas. Para ello, se utilizó la herramienta de diagramación online “Lucidchart”, obteniendo como resultado un diseño de diagrama de casos de uso y un diagrama de arquitectura. El diagrama de casos de uso (Ver Figura 3.2), permitió representar los diferentes escenarios en los que la interfaz conversacional es utilizada por los usuarios, y cómo ésta interactúa con el sistema. Por otro lado, el diagrama de arquitectura que se visualiza en la Figura 3.3, proporciona una visión clara de los componentes y la estructura del sistema.

**Figura 3. 2** Diagrama de casos de uso.




***Fuente.*** Los Autores.

**Figura 3. 3** Diagrama de arquitectura.




***Fuente.*** Los Autores.

Por otra parte, durante la fase de diseño, se determinó la interacción de la interfaz conversacional con el usuario, ya que proporciona una experiencia de usuario óptima. Además, esto ayuda a comprender la intención del usuario, proporcionar respuestas adecuadas y asegurar que la conversación sea fluida y satisfactoria. Por lo tanto, se han determinado los siguientes puntos:

**Tabla 3. 4** Lista de interacción de la interfaz con el usuario.

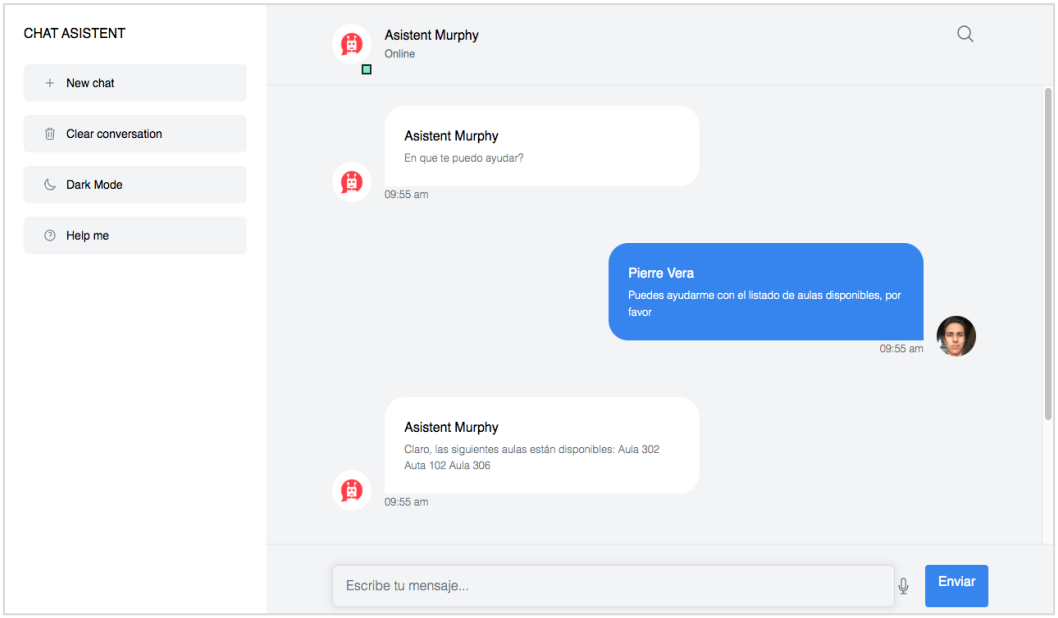
| Interacción | Descripción |
|---|---|
| Reconocimiento de voz | Utilizar herramientas de reconocimiento de voz para convertir la entrada del usuario de voz a texto. |
| Entender el texto | Se deben utilizar herramientas de NLP para analizar el texto y obtener una representación de intenciones, entidades y slots. |
| Generar la respuesta | Generar una respuesta apropiada para la intención del usuario a partir de la información de entrada. |
| Iniciar un diálogo | Mantener una conversación fluida con el usuario mediante herramientas de diálogo. |
| Administrar el diálogo | Controlar el flujo de la conversación para garantizar coherencia y relevancia. |
| Evaluar el diálogo | Utilizar herramientas de evaluación de diálogo para medir la satisfacción del usuario con la conversación. |

***Fuente.*** Los Autores.

Luego de esto, como se observa en la Figura 3.4, se procedió a desarrollar una parte significativa del prototipo de la interfaz conversacional mediante el uso del framework Tailwind CSS. El objetivo principal de esta elección fue obtener una vista previa del modelo, aprovechando las múltiples clases pre construidas que ofrece la herramienta para diseñar la apariencia de una página web o aplicación.

Gracias a esta funcionalidad, es posible acelerar el proceso de diseño y desarrollo, ya que se cuenta con una amplia variedad de opciones preestablecidas que pueden ser combinadas y modificadas según las necesidades específicas de la interfaz. En resumen, se optó por una metodología que permite obtener resultados eficientes sin comprometer la calidad final del producto.

**Figura 3. 4** Prototipo de la vista general de la interfaz conversacional.




***Fuente.*** Los Autores

De este modo, se elaboró una tabla para evaluar las características principales que cumple el prototipo para garantizar la viabilidad de la interfaz conversacional y garantizar que cumpla con los requisitos, asimismo, corregir las áreas críticas o posibles errores de rendimiento. Este mecanismo de evaluación surgió a raíz de la lista de requisitos que se estudió anteriormente y consta de tres rangos: Alto, Medio o Bajo, como se muestra a continuación.

**Tabla 3. 5** Evaluación del prototipo de la interfaz conversacional.

| Características | Alto | Medio | Bajo |
|---|---|---|---|
| Saludo | X | | |
| Finalización | X | | |
| Pregunta | X | | |
| Reconocimiento | X | | |
| Declaración Informativa | | X | |
| Sugerencia | | X | |
| Disculpa | X | | |
| Comando | | X | |
| Confirmación | X | | |
| Marcador de discurso | | X | |
| Error | X | | |
| Botones | X | | |
| Elementos audiovisuales | | | X |

***Fuente.*** Los Autores.

## 3.3. CODIFICACIÓN

En la fase de codificación se utilizó los protocolos de seguridad HTTP y Websockets para establecer una conexión bidireccional entre la aplicación y el servidor. Estas tecnologías permitieron la implementación de una interfaz conversacional que ofrece una comunicación fluida y en tiempo real, mejorando la experiencia de usuario.

Ahora bien, para lograr una conexión confiable entre la interfaz conversacional y el asistente virtual, se acoplaron las APIs que este último provee mediante una solución técnica que utiliza Node.js, HTTP y Websockets. Esta combinación de tecnologías permitió la gestión eficiente de múltiples solicitudes simultáneas y la actualización de información en tiempo real, lo que brindó una experiencia de usuario satisfactoria y una comunicación fluida.

A continuación, se presentará una tabla descriptiva de las tecnologías empleadas, incluyendo detalles sobre su funcionamiento y características que permitieron la integración de APIs.

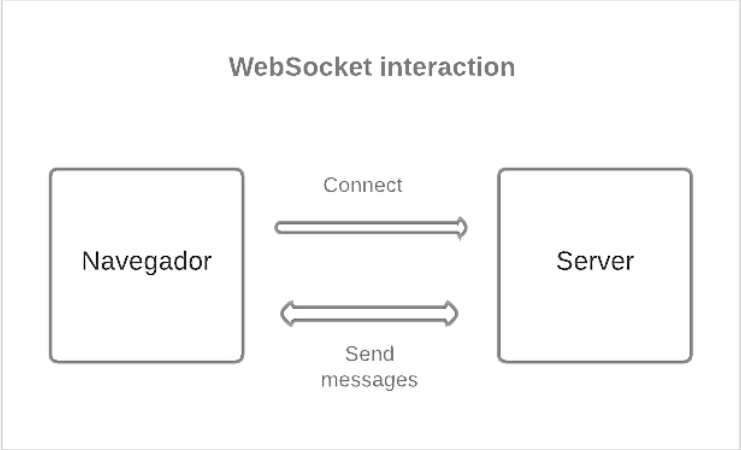
**Tabla 3. 6** Tecnologías empleadas para el desarrollo de la interfaz conversacional.

| TECNOLOGÍA | DESCRIPCIÓN |
|---|---|
| Node.js | Es un entorno de ejecución de JavaScript que permite ejecutar código JavaScript en el servidor. Tiene una amplia variedad de módulos y librerías que puedes utilizar para implementar diferentes funcionalidades en tus aplicaciones. |
| Módulo https | Es un módulo nativo de Node.js que permite hacer solicitudes HTTP a servidores web y recibir las respuestas en formato JSON. |
| Módulo Websocket | Es una librería de Node.js que permite implementar la funcionalidad de Websockets en tus aplicaciones. Los Websockets son una tecnología que te permite establecer una conexión bidireccional entre el cliente y el servidor, lo que significa que el cliente y el servidor pueden enviar y recibir mensajes en cualquier momento. Con el módulo Websocket, se puede implementar la funcionalidad de Websockets en una aplicación y recibir actualizaciones en tiempo real desde el servidor. |

***Fuente.*** Los Autores.

De este modo, el protocolo utilizado opera a través de conexiones TCP con el servidor, lo que permite mantener una conexión activa y bidireccional entre el servidor y el navegador. Por lo tanto, para establecer una conexión WebSocket, se requieren dos partes fundamentales: un WebSocket Server (servidor WebSocket) y un WebSocket Cliente (cliente WebSocket). El WebSocket server es responsable de aceptar las conexiones y representa el backend, mientras que el cliente sería el navegador. En este contexto, es el cliente el que inicia la conexión con el servidor. Una vez establecida la conexión, tanto el servidor como el cliente pueden intercambiar mensajes simultáneamente.

**Figura 3. 5** Arquitectura Cliente-Servidor.




***Fuente.*** Los Autores

Como se observa en la Figura 3.5, el Navegador establece una conexión con el Servidor. El servidor acepta la conexión y luego se inicia un intercambio de mensajes entre el servidor y el navegador. Es importante destacar que los WebSockets mantienen la conexión activa mientras la pestaña del navegador esté abierta, pero tanto el cliente como el servidor tienen la capacidad de cerrar la conexión en cualquier momento.

### 3.3.1 FLUJO DE LA INTERFAZ CONVERSACIONAL PARA CONSUMIR LOS SERVICIOS DEL ASISTENTE VIRTUAL

Así que, con respecto a la implementación de la interfaz conversacional para consumir los servicios del asistente virtual se basó en una arquitectura compuesta por Flask y ngrok. Flask se encarga de crear las APIs necesarias para la comunicación, mientras que ngrok proporciona un enlace seguro para

acceder a la API del asistente desde la interfaz a través de Internet. El proceso comienza cuando un usuario se conecta a la interfaz, estableciendo una conexión bidireccional en tiempo real mediante Socket.IO. Esta tecnología permitió una comunicación eficiente entre el cliente y el servidor.

Cuando un usuario envía un mensaje a través de la interfaz, se activa el evento "*send message*". En respuesta a este evento, se realiza una solicitud HTTP al servidor del asistente virtual utilizando la API generada por ngrok. Esta solicitud incluye los detalles necesarios, como la URL del servidor, el puerto y la ruta de la API.

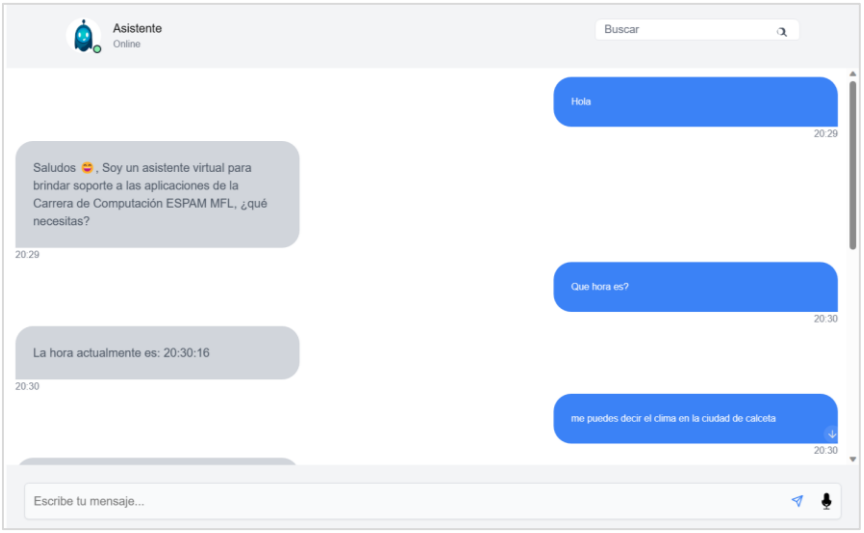
**Figura 3. 6** Conectar el asistente a la interfaz conversacional.

```
module.exports = function(io) {
  io.on('connection', (socket) => {
    console.log('Un usuario se ha conectado');
    socket.on('Send message', function(data) {
      io.emit('new user message', { message: data });
      const https = require('https');
      const options = {
        hostname: '72d6-200-24-154-53.ngrok.io',
        port: 443,
        path: '/webhooks/rest/webhook',
        method: 'POST',
        headers: { 'Content-Type': 'application/json' }
      };

      const req = https.request(options, (res) => {
        let responseBody = '';

        res.on('data', (chunk) => {
          responseBody += chunk;
        });

        res.on('end', () => {
          try {
            const response = JSON.parse(responseBody);
            const responseMessages = response.map((item) => item.text);
            io.emit('new bot message', { messages: responseMessages });  console.log('Mensajes insertados correctamente!');
          } catch (error) {
            console.error(error);
            const errorMessage = 'El asistente no está disponible en este momento. Por favor, inténtalo más tarde.';
            io.emit('new bot message', { error: true, message: errorMessage });
          }
        });
      });
```

***Fuente.*** Los Autores

Una vez que se recibe la respuesta del servidor del asistente, se procesa el cuerpo de la respuesta, que generalmente se encuentra en formato JSON. A partir de este cuerpo, se extraen los mensajes generados por el asistente, los cuales se emiten al cliente mediante Socket.IO utilizando el evento "*new bot message*". Esto permitió que el usuario vea las respuestas del asistente en tiempo real, creando una experiencia de conversación fluida.

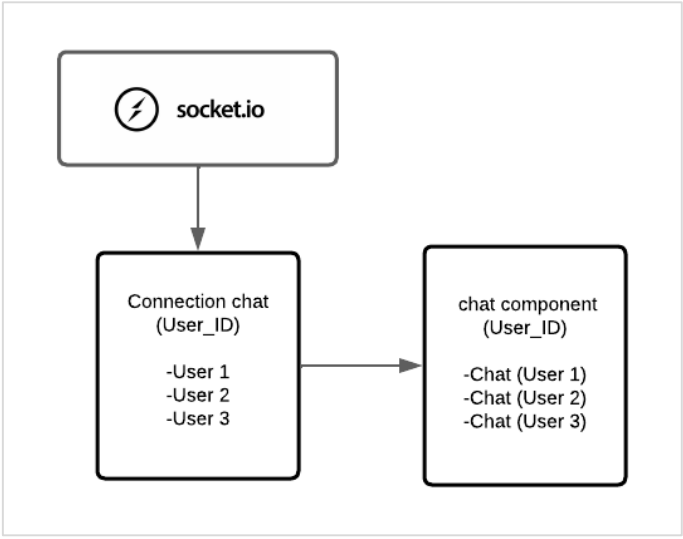
**Figura 3. 7** Conversación del usuario con el asistente virtual.




***Fuente.*** Los Autores

Cabe destacar, que para garantizar que los mensajes de los usuarios no se mezclen cuando varios usuarios consumen los servicios del asistente al mismo tiempo, se implementó un mecanismo de manejo de conexiones individualizado en Socket.IO. Cada conexión se identifica de manera única, lo que permite que cada usuario reciba las respuestas del asistente de forma independiente, sin interferencias de otros usuarios.

**Figura 3. 8** Asignación de chat para distintos usuarios.




***Fuente.*** Los Autores

En el diagrama se puede observar cómo, durante el proceso de conexión, Socket.IO genera automáticamente un ID único que se asigna exclusivamente a esa conexión en particular. Una vez que se ha establecido una conexión exclusiva para cada usuario, se le asigna un chat independiente correspondiente al ID de su conexión. De esta manera, cada usuario puede interactuar con el asistente de manera personalizada, sin afectar el rendimiento de la interfaz conversacional.

### 3.3.2 IMPLEMENTACIÓN DE LA BASE DE DATOS CON MONGODB

MongoDB es una base de datos NoSQL ampliamente utilizada en aplicaciones modernas debido a su flexibilidad y escalabilidad. Por lo tanto, en este contexto se presenta como una solución ideal para almacenar las conversaciones del usuario con el asistente virtual de manera eficiente y efectiva.

**Figura 3. 9** Estructura de la base de datos MongoDB.

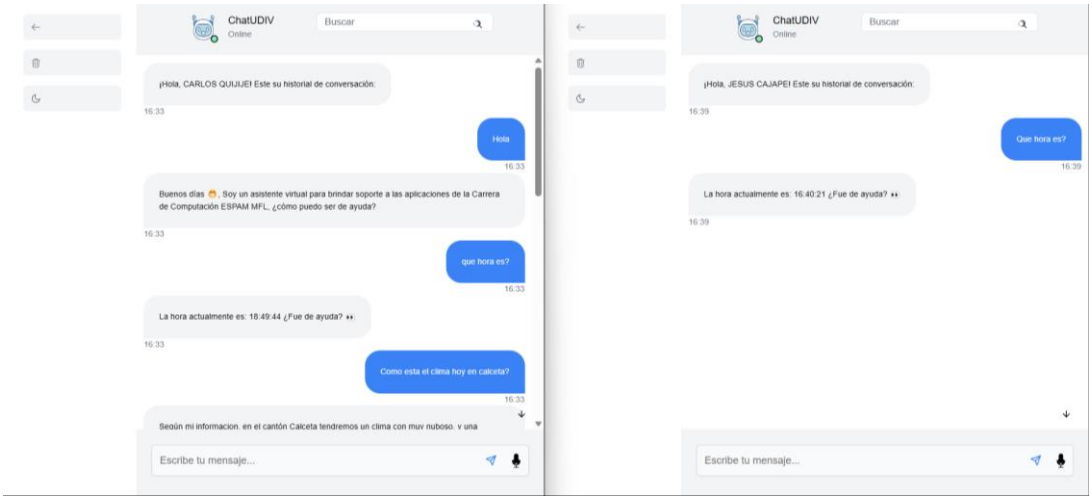

***Fuente.*** Los Autores

En el código proporcionado, se está utilizando Mongoose para definir dos esquemas: conversationSchema y userSchema. Estos esquemas permiten establecer la estructura que tendrán los datos almacenados en la base de datos MongoDB para representar las conversaciones de los usuarios con el asistente virtual.

- **conversationSchema**: Este esquema define cómo se estructurará la información de una conversación.
- **userSchema**: Este esquema define cómo se estructurará la información del usuario y sus conversaciones.
- **Modelo y conexión a la base de datos**: Después de definir los esquemas, se crea el modelo **User** utilizando el esquema userSchema. Este modelo se utiliza para interactuar con la colección de usuarios en la base de datos. Al establecer la conexión con MongoDB mediante **mongoose.connect()**, todas las operaciones realizadas con el modelo **User** se reflejarán en la base de datos.

Como se puede observar en la Figura 3.10, cada vez que un usuario se verifica correctamente, la interfaz retornará todo el historial de conversación que ha tenido con el asistente virtual. En este caso, se realizaron pruebas con dos usuarios distintos para verificar que las conversaciones se retornen adecuadamente y sin mezclarse con los distintos chats que puedan existir en ese momento.

**Figura 3. 10** Historial de distintas conversaciones con el asistente virtual.




***Fuente.*** Los Autores

### 3.3.3 IMPLEMENTACIÓN DE LOS WEBCOMPONENTS

Los web componentes se refieren a un conjunto de tecnologías que permiten la creación de nuevos elementos en HTML que son encapsulados y reutilizables en diversas páginas o aplicaciones web. Su objetivo principal es facilitar la

reutilización de código en diferentes proyectos sin generar dependencias que puedan afectar la escalabilidad del sistema. Estas tecnologías se dividen en tres elementos principales: Custom elements, Shadow DOM y HTML templates. Mediante el uso de estos componentes, se busca simplificar el desarrollo y mejorar la modularidad de las aplicaciones web, permitiendo su fácil consumo y adaptación en otros proyectos.

Para crear los webs components en la interfaz conversacional se llevó a cabo la implementación de estos tres elementos, lo cual encapsulara cada una de las funcionalidades de la interfaz conversacional. En este caso se implementó el web component de la ventana del chat, que no solo agrega los mensajes al chat, sino que contiene la conexión con el websockets para la comunicación del usuario con el asistente virtual.

- **HTML templates:** La utilización de un template para definir la estructura del componente de chat personalizado, junto con las funcionalidades de manipulación del DOM y eventos, proporciona una forma eficiente y modular de implementar una funcionalidad de chat en una página web. Esto permite una mayor reutilización de código, facilita el mantenimiento y la escalabilidad del sistema, y garantiza que los estilos y el comportamiento definidos para el componente no afecten a otras partes de la aplicación.

**Figura 3. 11** Encapsular el cuerpo HTML del chat en un template.

```
<template id="chat-template">
  <div id = "chat" class="py-6 px-20 overflow-auto h-3/4 card-budy">
    <div id="chat-container"></div>
    <button id="scrollDownBtn" class="cursor-pointer absolute right-6 bott
      <svg stroke="black" fill="none" stroke-width="2" viewBox="0 0 24 24"
        <line x1="12" y1="5" x2="12" y2="19"></line>
        <polyline points="19 12 12 19 5 12"></polyline>
      </svg>
    </button>
  </div>
</template>
```

***Fuente.*** Los Autores

- **Custom elements:** Básicamente, este se trata de un conjunto de interfaces de programación de aplicaciones (API) en JavaScript que

posibilitan la creación de etiquetas HTML personalizadas o la mejora de las ya existentes.

**Figura 3. 12** Encapsulamiento de toda la lógica JS del Chat.

```
class ChatComponent extends HTMLElement {
  constructor() {
    super();

    const template = document.querySelector('#chat-template');
    const content = template.content.cloneNode(true);
    this.appendChild(content);

    const chatContainer = this.querySelector('#chat-container');
    const scrollDownButton = this.querySelector('#scrollDownBtn');

    scrollDownButton.addEventListener('click', () => {···
    });

    // Listen for new user messages and add them to the chat
    socket.on('new user message', function(data) {···
    });

    // Listen for new bot messages and add them to the chat
    socket.on('new bot message', function(data) {···
    });
  }
}

customElements.define('chat-component', ChatComponent);
```

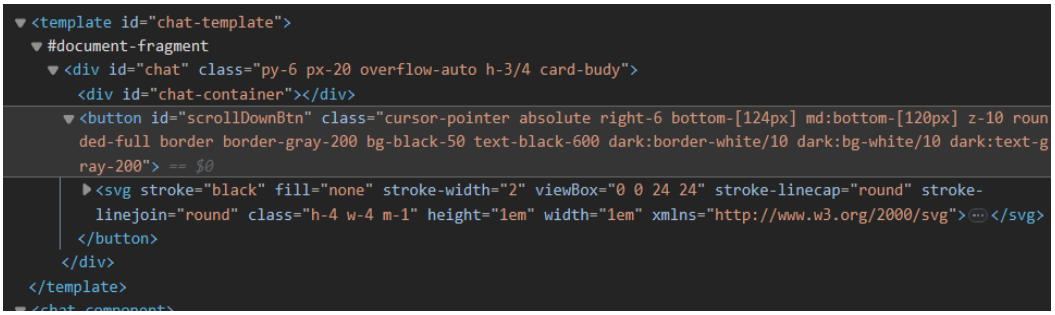
***Fuente.*** Los Autores

El código presentado implementa un componente de chat personalizado en la página web utilizando la clase "ChatComponent" que hereda de "HTMLElement". Este componente puede ser utilizado en el documento HTML mediante la etiqueta personalizada <chat-component>.

El componente posee funcionalidades específicas para mostrar mensajes de usuario y mensajes de Bot en un contenedor de chat. Al recibir nuevos mensajes desde el usuario o el Bot a través de una conexión de socket, dichos mensajes son agregados de forma dinámica al contenedor de chat. Además, se asegura que el contenido se desplace automáticamente hacia abajo para mostrar siempre los mensajes más recientes en pantalla. La utilización de este componente permite añadir una funcionalidad de chat de manera sencilla en la página web, aprovechando la capacidad de reutilización y encapsulamiento que proporciona el componente personalizado.

- **DocumentFragment:** Además, se utilizó DocumentFragment, el cual se crea utilizando el método "createDocumentFragment()" del objeto document. Esto permite la creación y manipulación eficiente de una colección de nodos sin afectar directamente el DOM principal.

**Figura 3. 13** Construcción del DocumentFragment.

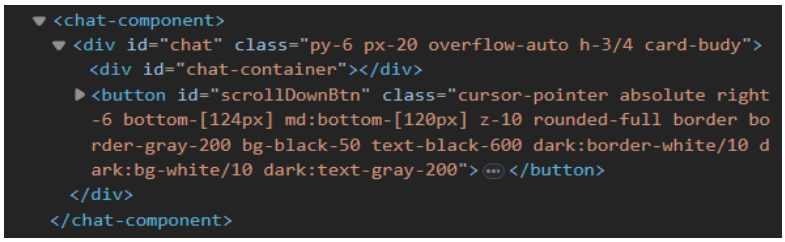



***Fuente.*** Los Autores

Como se observa en la Figura 3.13, una vez que se ha construido la estructura deseada en el "DocumentFragment", se puede insertar en el DOM utilizando métodos como "appendChild ()" o "insertBefore ()". Al hacer esto, los nodos contenidos en el fragmento se añaden al árbol DOM en el lugar especificado, sin afectar el rendimiento o la respuesta visual del documento principal.

Después de haber creado el componente web del chat, se logra la reutilización en distintos proyectos. Esto se consigue mediante la simple inclusión de la etiqueta <chat-component></chat-component> en el cuerpo del documento HTML.

**Figura 3. 14** Etiqueta que contiene el web component del Chat.




***Fuente.*** Los Autores

El fragmento de código presentado en la Figura 3.14 muestra la forma en que, al insertar la etiqueta mencionada dentro del cuerpo del archivo

HTML, se logra incluir todo el contenido en la plantilla asociada. Además, dicha etiqueta incorpora tanto el código JavaScript como los estilos CSS necesarios para garantizar el correcto funcionamiento del Web component. Esta solución proporciona una mayor modularidad del código, permitiendo una fácil integración del componente en una amplia gama de proyectos. Además, al mantener la lógica y los estilos encapsulados en la misma etiqueta, se mejora la legibilidad, mantenibilidad y escalabilidad del código, facilitando su gestión y actualización en el futuro.

### 3.3.4 IMPLEMENTACIÓN DEL RECONOCIMIENTO DE VOZ

La incorporación del reconocimiento de voz en la interfaz conversacional brinda una experiencia más intuitiva y conveniente para los usuarios. Al utilizar la tecnología de reconocimiento de voz, se permite a los usuarios enviar mensajes hablados en lugar de tener que escribirlos manualmente.

**Figura 3. 15** Configuración de la librería webkitSpeechRecognition.

```
const recognition = new webkitSpeechRecognition();
recognition.lang = 'es-ES';
recognition.continuous = true;
recognition.interimResults = false;
```

***Fuente.*** Los Autores

Como se puede observar en la Figura 3.15, al crear una instancia mediante el **webkitSpeechRecognition**, se establecen ciertas configuraciones relacionadas con el reconocimiento de voz. La propiedad **lang** define el idioma utilizado para el reconocimiento, en este caso, el español de España. La propiedad **continuous**, indica si el reconocimiento de voz debe continuar escuchando después de que el usuario haya hablado, y en este caso está configurada como verdadera. Por último, la propiedad **interimResults** determina si se deben devolver resultados intermedios durante el proceso de reconocimiento.

De este modo, cuando se produce un resultado de reconocimiento de voz, el evento **onresult** se activa. En este evento, se obtienen los resultados del reconocimiento de voz a través de event.results. Luego, se extrae la última frase

reconocida mediante results [results.length - 1][0].transcript y se agrega al valor actual del campo de entrada de mensajes y a la variable recognizedMessage.

El evento **onend** se activa cuando el reconocimiento de voz ha finalizado. Aquí se realizan varias acciones, como restablecer la interfaz visual si el estado de grabación está activo, mostrar el campo de entrada de mensajes, actualizar los iconos de micrófono y pausa, y mostrar una notificación indicando que la grabación de voz se ha detenido. Posteriormente, se toma el contenido de recognizedMessage y se elimina cualquier espacio en blanco alrededor. Si el mensaje no está vacío, se envía al servidor a través del evento 'Send message' y se restablecen los valores del campo de entrada de mensajes y recognizedMessage.

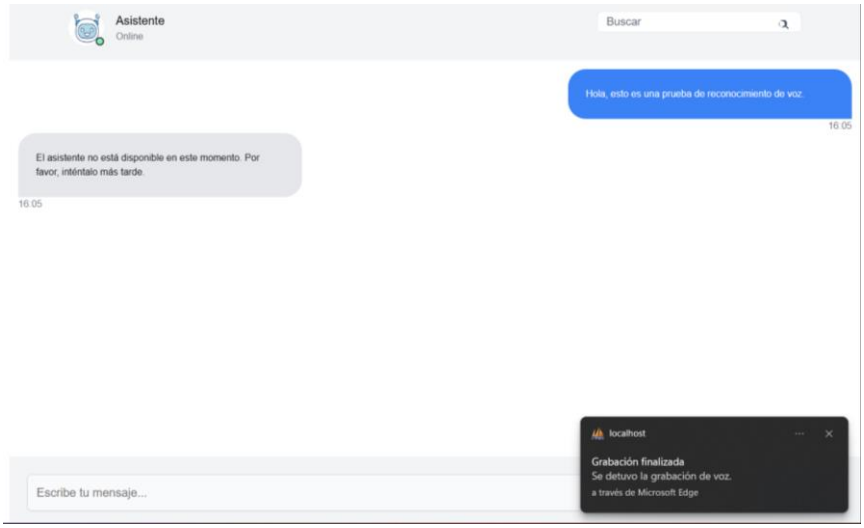
**Figura 3. 16** Evento para activar el reconocimiento de voz.

```
recognition.onresult = (event) => {
  const results = event.results;
  const frase = results[results.length - 1][0].transcript;
  this.messageInput.value += frase;
  recognizedMessage += frase;
};

recognition.onend = () => {
  if (recordingStatus && microIcon && pauseIcon) {
  this.messageInput.style.display = 'block';
  recordingStatus.classList.remove('recording');
  recordingStatus.innerText = '';
  microIcon.classList.remove('hidden');
  pauseIcon.classList.add('hidden');
};
new Notification('Grabación finalizada', {
  body: 'Se detuvo la grabación de voz.'
});

const message = recognizedMessage.trim();
if (message !== '') {
  socket.emit('Send message', message);
  this.messageInput.value = '';
  recognizedMessage = '';
}
}
```

***Fuente.*** Los Autores

Finalmente, una vez que el mensaje de voz se ha convertido en texto, se utiliza para interactuar con el asistente virtual y obtener respuestas pertinentes. Este enfoque elimina la necesidad de que los usuarios redacten mensajes o instrucciones extensas, lo cual ahorra tiempo y esfuerzo.

**Figura 3. 17** Implementación del reconocimiento de voz en el chat.




***Fuente.*** Los Autores

Consecutivamente, se verificó que la interfaz conversacional cumpla con todos los requerimientos establecidos en la fase de planificación. En la Tabla 3.7, se muestra una visión detallada de los diferentes requisitos de la interfaz conversacional y su estado actual de cumplimiento. Cada uno de los elementos de la tabla representa un componente crítico de la interfaz y su cumplimiento garantiza que la experiencia del usuario sea lo más eficiente y satisfactoria posible.

**Tabla 3. 7** Requisitos y su estado de cumplimiento.

| Requisito | Estado Actual | Motivo |
|---|---|---|
| Interacción en tiempo real | Correcto | Se implementó la tecnología de Websockets para habilitar una comunicación bidireccional en tiempo real entre el usuario y el asistente virtual. Debido a su capacidad puede manejar conexiones en tiempo real, lo que permite una experiencia de usuario fluida y una interacción conversacional más natural. |
| Capacidad de manejar múltiples solicitudes simultáneamente | Correcto | Esta solución técnica permite manejar múltiples solicitudes al mismo tiempo. |
| Mantener conversaciones separadas entre usuarios | Correcto | El WebSocket se configuro de tal manera para que cada usuario tenga su propia conversación con el asistente. |
| Integrar con otras aplicaciones | Correcto | Se utilizó la tecnología de Web Components para modularizar y encapsular la lógica de la interfaz conversacional, permitiendo su fácil integración en otros proyectos. |

| | | |
|---|---|---|
| Envío de mensajes de voz | Correcto | Se implementó la funcionalidad de envío de mensajes de voz utilizando la librería webkitSpeechRecognition de Node.js. |
| Integración con el asistente virtual. | Correcto | Se utiliza solicitudes http para la comunicación, cuando el usuario envía un mensaje a través del socket, se envía una solicitud POST a la API del asistente virtual. |
| Modo oscuro | Correcto | Se implementó la funcionalidad de modo oscuro utilizando la librería de CSS llamada Tailwind. Esta biblioteca permitió la definición de estilos de manera rápida y eficiente, lo que facilitó el desarrollo de la interfaz. |
| Limpiar conversación | Correcto | Se implementó la funcionalidad para limpiar los mensajes. |
| Filtro de búsqueda | Correcto | Se implementó la funcionalidad de filtro de búsqueda. |
| Corrección ortográfica | Correcto | Se implementó la corrección ortográfica en tiempo real utilizando un código que utiliza XMLHttpRequest para comunicarse con un servicio de revisión ortográfica externo. |
| Prohibición de emojis | Correcto | Se implementó la prohibición de emojis utilizando una expresión regular que verifica si hay caracteres que pertenecen a un rango específico de caracteres de emojis, y se muestra una notificación al usuario si intenta enviar un mensaje con un emoji. |

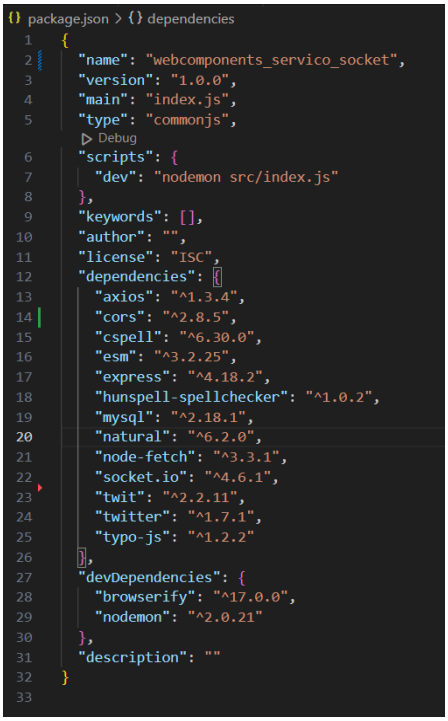
***Fuente.*** Los Autores

### 3.3.5 CONSTRUCCIÓN Y PUBLICACIÓN DE LA DE LA INTERFAZ CONVERSACIONAL UTILIZANDO NPM

Una vez desarrollada la interfaz conversacional utilizando web components, se procedió a su construcción y empaquetado. Esta etapa implicó una serie de pasos técnicos para garantizar la optimización y la generación de un paquete finalizado apto para su distribución.

De este modo, se llevó a cabo la compilación de los archivos de código fuente, asegurando que estuvieran libres de errores y listos para su implementación. Para lograr esto, se configuró adecuadamente el archivo package.json del producto con las dependencias y scripts necesarios como se ve a continuación:

**Figura 3. 18** Configuración del package.json

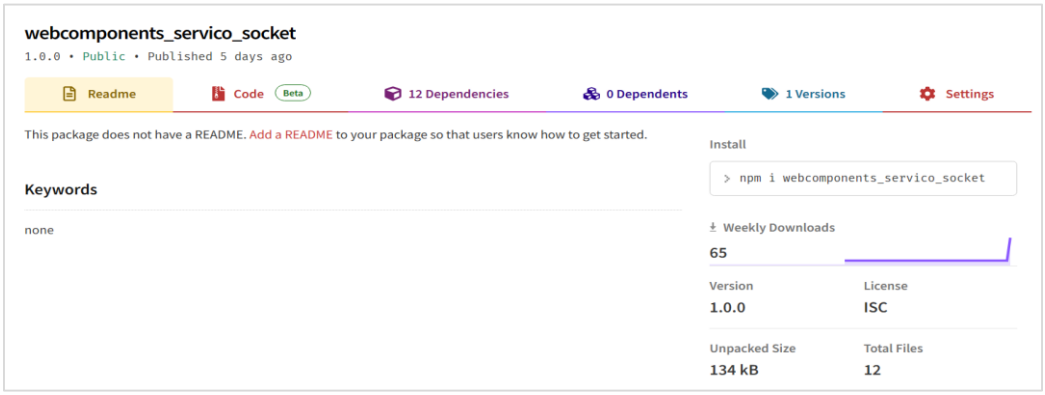



***Fuente.*** Los Autores

Una vez finalizada la construcción y configuración del package.json de la interfaz conversacional, se procedió a generar un paquete listo para su distribución. Para ello, se utilizó un registro de paquetes confiable, como NPM (Node Package Manager). NPM proporciona una plataforma centralizada que permite gestionar las dependencias de paquetes de software en proyectos de desarrollo.

Para publicar la interfaz conversacional, se creó una cuenta en el registro correspondiente de NPM y se utilizó la línea de comandos para iniciar sesión y realizar la publicación del paquete.

**Figura 3. 19** Verificación del paquete en NPM.




***Fuente.*** Los Autores

En la ilustración se puede apreciar que el paquete ha sido publicado y está listo para ser instalado en otros proyectos mediante la ejecución del comando "*npm i webcomponents_servico_socket*". Es importante destacar que, al instalar el paquete en un nuevo proyecto, todas las dependencias de la interfaz conversacional se instalarán automáticamente para garantizar su correcto funcionamiento. Esto proporciona una forma sencilla y conveniente de incorporar la funcionalidad de la interfaz conversacional en diferentes proyectos, sin la necesidad de preocuparse por instalar y configurar manualmente cada una de las dependencias relacionadas.

## 3.4. PRUEBAS

En esta fase de la metodología, las pruebas unitarias fueron parte fundamental para la documentación del proyecto (**Anexo 4**) y sirvieron para probar el código base, también permitieron validar la correcta implementación y comunicación del protocolo de transmisión de mensajes en tiempo real a través de los WebSockets entre el cliente y el servidor, asimismo se logró validar la funcionalidad de la conexión con la API del asistente virtual y la coherencia de las respuestas recibidas; de tal forma que se pudo verificar el funcionamiento eficiente de la interfaz conversacional.

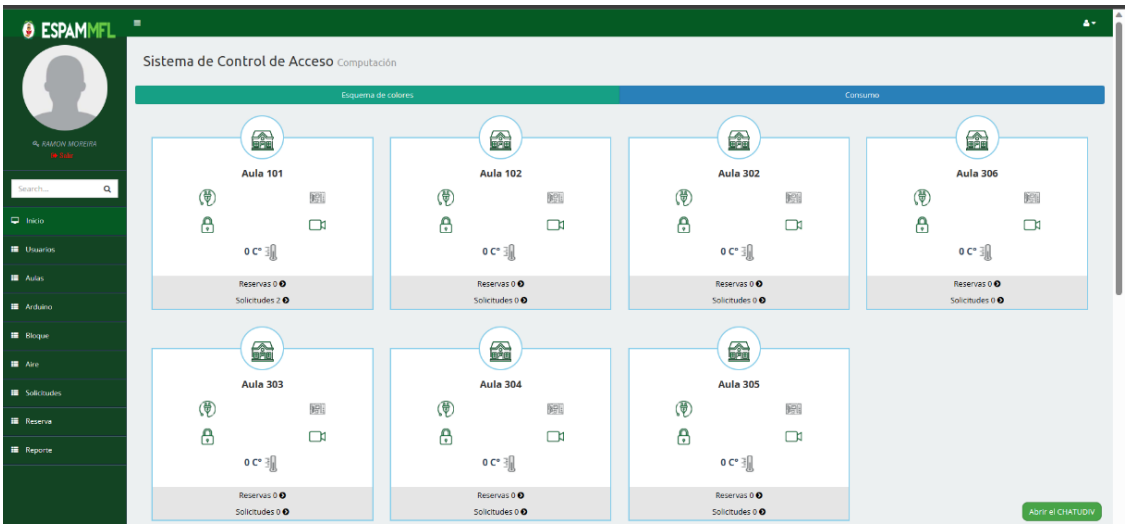
Además de las pruebas unitarias, también se ejecutaron las respectivas pruebas de integración, estas son fundamentales para garantizar un comportamiento consistente y de alta calidad en diferentes situaciones de uso en el desarrollo de aplicaciones modernas, se detallan en un informe (**Anexo 5**).

### 3.4.1 INTEGRAR LA INTERFAZ CONVERSACIONAL FUNCIONAL A LA APLICACIÓN DE LA UDIV

Para esto se utiliza el objeto XMLHttpRequest para enviar una solicitud GET a la dirección donde se encuentra alojado el proyecto de la interfaz conversacional. Una vez que se recibe una respuesta exitosa a la solicitud, se procede a actualizar el contenido del contenedor especificado con el HTML devuelto por el proyecto de la interfaz conversacional. Esta respuesta HTML se obtiene a través de la propiedad responseText del objeto XMLHttpRequest.

Dado esto, como se puede observar en la Figura 3.20, la integración del botón "Abrir ChatUDIV" en la aplicación de gestión de tareas permite a los usuarios acceder a una interfaz conversacional con el asistente virtual. Al ingresar a la aplicación, los usuarios pueden interactuar de forma directa y fluida con el asistente virtual, estableciendo una comunicación en tiempo real y eficiente.

**Figura 3. 20** Integración del botón en la aplicación “Administración de espacios”.

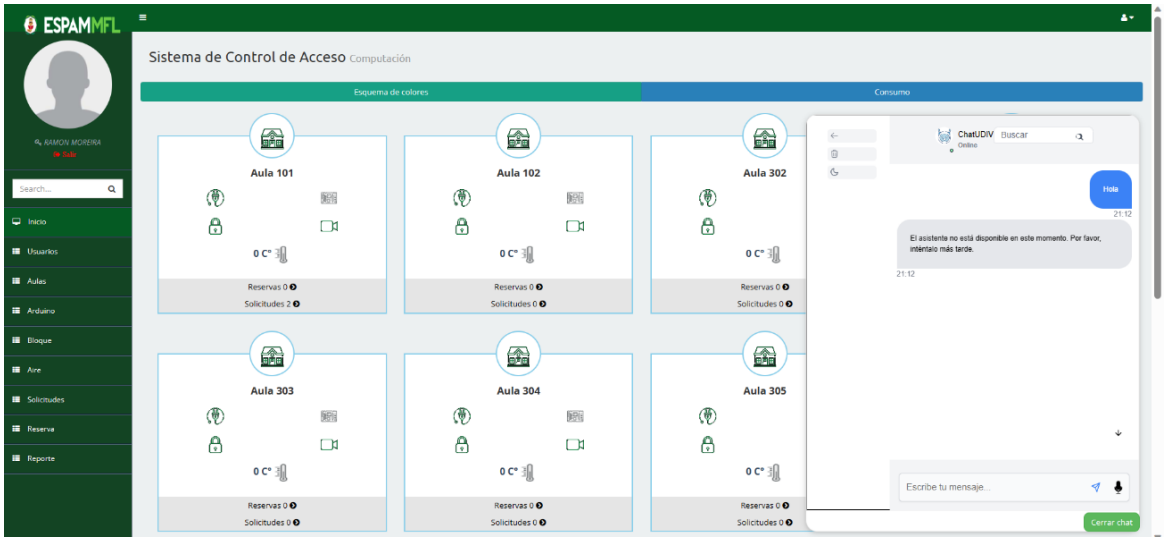



***Fuente.*** Los Autores

La interfaz conversacional está diseñada con una funcionalidad de validación que garantiza la disponibilidad adecuada del asistente virtual para llevar a cabo conversaciones con los usuarios. Esta validación permite evitar la entrega de respuestas incorrectas o inapropiadas que podrían afectar negativamente las necesidades y expectativas del usuario final.

Además de verificar la disponibilidad del asistente, la interfaz conversacional también incorpora un sistema de detección y manejo de errores en las respuestas. Esto asegura que cualquier error en el procesamiento de la información sea identificado y gestionado adecuadamente. De esta manera, se garantiza la entrega de respuestas precisas y confiables, mejorando la experiencia del usuario y aumentando la eficacia del asistente virtual en la aplicación de gestión de tareas.

**Figura 3. 21** Pruebas de funcionalidad de la Interfaz.

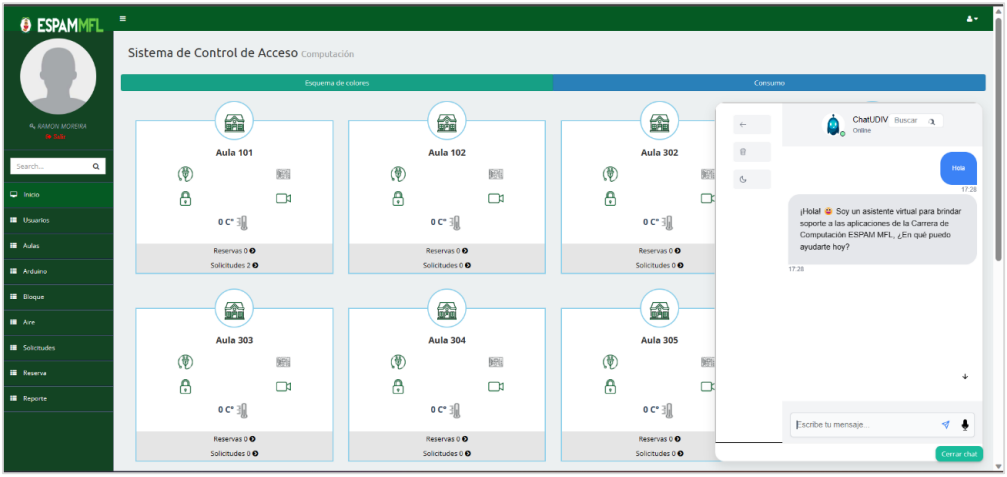



***Fuente.*** Los Autores

La implementación de esta técnica de integración permite cargar de forma dinámica el proyecto de la interfaz conversacional en el contenedor designado dentro de la aplicación de la UDIV de Infraestructura.

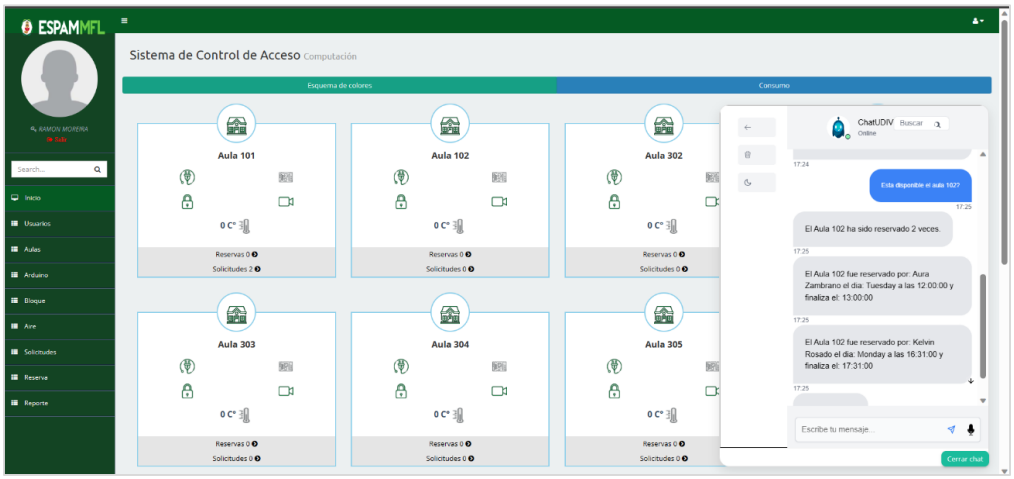
Por lo tanto, se procedió a establecer la conexión con el asistente virtual a la interfaz conversacional implementada en la aplicación de la UDIV para someterla a un proceso de prueba y validación para garantizar su correcto funcionamiento y que no haya problemas con su rendimiento:

**Figura 3. 22** Prueba de funcionalidad de la Interfaz conectado con el asistente.



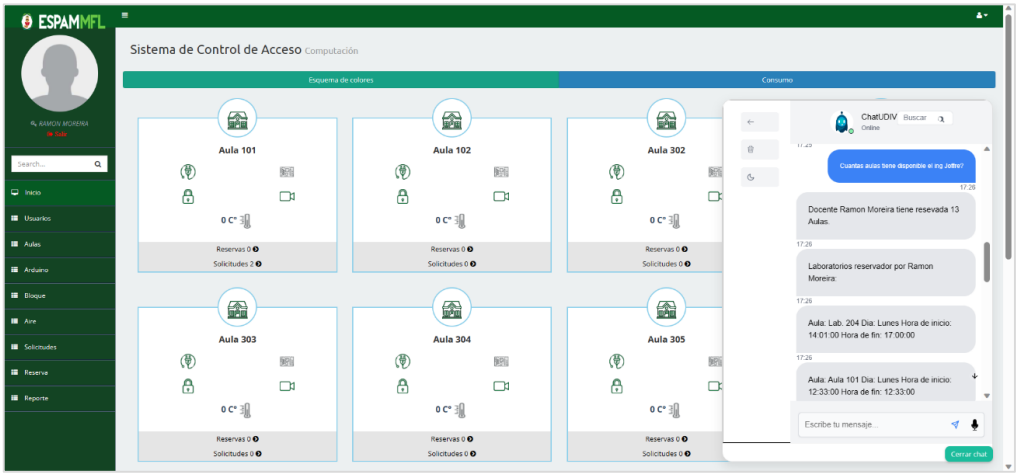

***Fuente.*** Los Autores

**Figura 3. 23** Evaluación de múltiples respuestas emitidas al Socket.




***Fuente.*** Los Autores

**Figura 3. 24** Evaluación de la capacidad de conversación del asistente a través de la Interfaz.




***Fuente.*** Los Autores

Durante las pruebas y validaciones realizadas, se pudo constatar que la interfaz conversacional implementada en la App de Administración de Espacios, conectada con el asistente virtual, es capaz de mantener conversaciones fluidas y sin interrupciones. Esto asegura una experiencia satisfactoria para los usuarios, sin que se produzca una disminución en el rendimiento del producto.

Cabe destacar, que los Web Components desarrollados se subieron a NPM para que puedan ser instalados en otros proyectos de la UDIV de Infraestructura que sean compatibles con la misma tecnología que fue desarrollada para la interfaz conversacional. De esta manera, se permite la reutilización del web components

en proyectos futuros que deseen incorporar la interfaz conversacional, siempre y cuando cumplan con los requisitos y tecnología compatibles.

Por último, se procedió a realizar el documento de manual de programador (**Anexo 6**) con el fin de proporcionar información del proyecto realizado, respecto al código fuente que sirva a los nuevos desarrolladores. Cabe mencionar, que se realizó el documento de manual de usuario (**Anexo 7**) que proporciona instrucciones detalladas sobre cómo utilizar y aprovechar al máximo la interfaz conversacional. Este manual está diseñado para ayudar a los usuarios a comprender cómo interactuar con la interfaz, qué preguntas pueden realizar y cómo obtener los resultados deseados.

# CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

## 4.1. CONCLUSIONES

- El proceso de levantamiento de requerimientos, mediante entrevistas y encuestas a usuarios clave, brindó una comprensión clara de sus necesidades y expectativas. Esta información fue fundamental para el desarrollo exitoso de la interfaz conversacional y la codificación precisa de los requisitos. Además, facilitó el manejo efectivo de la información y contribuyó al diseño adecuado de los diagramas UML.
- Mediante la búsqueda bibliográfica, permitió obtener información relevante sobre el uso de reconocimiento de voz como técnica predominante en la mejora de eficiencia y la interacción natural en interfaces conversacionales con asistentes virtuales. Además, se destacó la importancia de integrar estos productos con otras aplicaciones para agilizar la gestión de datos y tareas. Desde el punto de vista técnico, se observó que Node.js, en combinación con el framework Express.js y Websockets, prevalecieron como las tecnologías más utilizadas para proporcionar una experiencia de chat fluida y en tiempo real para los usuarios.
- El uso del framework Tailwind ha demostrado ser una solución efectiva y viable para la creación rápida y eficiente de interfaces de usuario flexibles y adaptables. Su implementación ha permitido establecer una estructura de diseño sólida y óptima, simplificando las tareas de mantenimiento y mejora a largo plazo de la interfaz. Además, gracias a la identificación de interacciones con el usuario y la generación de prototipos, se ha definido y validado de manera efectiva la experiencia de usuario en la interfaz conversacional, asegurando así la satisfacción para el usuario final.
- Por medio de los protocolos de seguridad HTTP y Websockets, se logró obtener una comunicación fluida y en tiempo real entre la aplicación y el servidor, mejorando la experiencia del usuario. Además, la integración de las APIs mediante Node.js, HTTP y Websockets facilitó una conexión confiable con el asistente virtual, gestionando eficientemente múltiples

solicitudes simultáneas y garantizando una comunicación fluida y sin interrupciones.

- Mediante las pruebas exhaustivas con Postman, se evaluó la funcionalidad del producto, asegurando el cumplimiento de los requisitos planteados desde el inicio del proyecto y evitando posibles errores en su ejecución. La integración de la interfaz conversacional en la aplicación "Administración de Espacios" correspondientes a la UDIV de Infraestructuras, determinó una interacción directa, fluida y eficiente con el asistente virtual en tiempo real. Además, los componentes web desarrollados son reutilizables en futuros proyectos que cumplan con los requisitos y tecnología compatibles.

## 4.2 RECOMENDACIONES

- Se recomienda continuar utilizando los métodos de Elicitación de Requerimientos que demostraron ser eficaces en el desarrollo de interfaces conversacionales en este proyecto. Los métodos empleados incluyeron la metodología XP (Programación Extrema), técnicas de entrevistas y visitas de campo. Estos enfoques permitieron recopilar información valiosa de los usuarios y stakeholders, contribuyendo al logro exitoso de los objetivos especificados. Mantener la aplicación de estas prácticas en futuros proyectos relacionados con el desarrollo de interfaces conversacionales garantizará una sólida base para obtener requisitos claros y precisos que respalden el éxito del proyecto.
- Es importante determinar el análisis de variables de cada información recopilada que tenga similitud con el proyecto a desarrollar, pues de dicha manera, se establece las técnicas y métricas más utilizadas. De tal manera que sea posible optar o decidir sobre las herramientas software que se utilizará para el desarrollo de su producto final.
- Se recomienda utilizar un framework de desarrollo, como Tailwind CSS, para la elaboración de prototipos, ya que permite una evaluación temprana y continua de la experiencia de usuario. Esta práctica facilita la identificación y corrección de problemas de diseño antes de una implementación completa, lo que ahorra tiempo y recursos al asegurar

una interfaz intuitiva y fácil de usar para los usuarios finales. Además, se puede aprovechar las capacidades de Tailwind CSS para crear componentes reutilizables, se acelera la creación de prototipos y se reduce la duplicación de código, lo que conduce a un desarrollo más ágil y sostenible a largo plazo.

- Para agilizar la fase de codificación del proyecto y mejorar la calidad del producto se recomienda utilizar el uso de framework de desarrollo como Node.js y a su vez utilizar instrumentos de trabajo colaborativos como GitHub.
- Para asegurar el correcto funcionamiento de la interfaz conversacional, se recomienda implementar un proceso de pruebas exhaustivas de carga y rendimiento que permita evaluar la capacidad del sistema en la gestión de múltiples solicitudes simultáneas. Estas pruebas ayudarán a identificar posibles cuellos de botella y garantizar una comunicación fluida y sin interrupciones.

# BIBLIOGRAFÍA


Alfaro, H. (2022). Propuesta de implementación de chatbot para recepción de pagos en la Universidad Iberoamericana Puebla.

Cárdenas Tutillo, C. J., & Quimbita Quingaluisa, E. F. (2017). Análisis, diseño y construcción de un prototipo de una red social orientada a la seguridad para la empresa CEFOSEG. Universidad Politécnica Salesiana. http://dspace.ups.edu.ec/handle/123456789/14123

Carrillo, I. A. (2016). Aplicación web de gestión financiera personal desarrollada con Mean.js. Universidad Politécnica de Madrid.

Chen, X., Xie, H., Zou, D. y Hwang, G. (2020). Application and theory gaps during the rise of Artificial Intelligence in Education. Computers and Education: Artificial Intelligence, 1 (100002), 100002. https://doi.org/10.1016/j.caeai.2020.100002

ESPAM MFL (2022). Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López. http://www.espam.edu.ec/web/universidad/filosofia.aspx

ESPAM MFL. (2016). Modelo Educativo Escuela Superior Politécnica Agropecuaria de Manabí “Manuel Félix López”. Editorial Humus (ed.).

ESPAM MFL. (2019). Obtenido de: http://www.espam.edu.ec/recursos/sitio/espam/EstatutoESPAMMFL.pdf

ESPAM MFL (Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López). (2023). Carrera de Computación. Obtenido de: http://www.espam.edu.ec/web/oferta/grado/computacion.aspx

Flores, G. R. (2017). Desarrollo de una aplicación web con Node.js para la monitorización en tiempo real de un electrocardiograma. 59.

Fonseca, R. (2017). Universidad Politécnica Salesiana Sede Quito. Tesis, 1–100. http://dspace.ups.edu.ec/bitstream/123456789/5081/1/UPS-CYT00109.pdf

Gabriel Elías Chanchí, G., Acosta-Vargas, P., & Wilmar Yesid Campo, M. (2019). Construcción de recursos educativos para la temática de accesibilidad en el curso de interacción humano computador. RISTI - Revista Ibérica de Sistemas e Tecnologías de Información, 2019(E23), 171–183.

Gamboa, E. D. (2019). Prototipo de un Chatbot para compras online utilizando Bot Framework. Universidad Internacional de La Rioja, 137. https://repositorio.uta.edu.ec/bitstream/123456789/30105/1/Tesis_t1634si.pdf

Gómez García, D. E. (2018). Desarrollo del sistema de requisiciones para la empresa hidroeléctrica abanico S.A. aplicando el entorno de programación Node.js. Escuela Superior Politécnica de Chimborazo.

Guardado Osuna, J.O. (2017). Desarrollo de aplicaciones web con Node.js. Universidad Politécnica de Sinaloa.

Jiménez Builes, J. A., Ramirez Bedoya, D. L., & Branch Bedoya, J. W. (2019). Metodología de desarrollo de software para plataformas educativas robóticas usando ROS-XP. Revista Politécnica, 15(30), 55–69. https://doi.org/10.33571/rpolitec.v15n30a6

Madou, R., Guerrero, F. N., & Spinelli, E. M. (2020). Interfaz humano-máquina web amigable para dispositivo IoT. XXVI Congreso Argentino de Ciencias de La Computación (CACIC). (Modalidad Virtual, 5 Al 9 de octubre de 2020), 440–449.

Martínez, Y. A., & Cely, C. A. C. (2018). Diseño de Interfaz de Usuario para la Creación de Sistemas Multimedia para Apoyar el Desarrollo del Lenguaje.

Molero, G., Benítez, E., & Mezura, C. (2017). Interacción humano computadora y minería de datos. Ciencias de La Información, 48(1), 3–10. https://www.redalyc.org/pdf/1814/181454538001.pdf

Ogosi Auqui, J. A. (2021). Chatbot del proceso de aprendizaje universitario: Una revisión sistemática. Alpha Centauri, 2(2), 29–43.

https://doi.org/10.47422/ac.v2i2.33

Peralbo Velasco, M. A. (2019). Desarrollo de una aplicación web alternativa para videoconferencia y compartición de pantalla con uso de WebRTC. Escuela Politécnica Nacional.

Sánchez Hernández, D., Lizano Madriz, F., & Sandoval Carvajal, M. (2020). Integración de Pruebas de Usabilidad Remota en Programación Extrema: Una Revisión de la Literatura. Uniciencia, 34 (1), 20-31. https://doi.org/10.15359/ru.34-1.2

Toledo Toledo, G., Arellano Pimentel, J. J., Aguilar Acevedo, F., & Molina Rodríguez, E. W. (2018). Aprendizaje Basado en Proyectos Dentro de un Curso Universitario de Interacción Humano Computadora. ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica, 7(2), 65-91.

UDIV de Infraestructura. (2022). Documento de aprobación.

# ANEXOS

# ANEXO 1. OFICIO PARA LA COLABORACIÓN DEL PROYECTO EN LA UDIV DE INFRAESTRUCTURAS.

**UDIV DE INSFRAESTRUCTURA DE LA CARRERA DE COMPUTACIÓN**

Calceta, 01 de Julio del 2022

Mgtr. Moreira Pico Ramón Joffre
**DIRECTOR DE LA CARRERA DE COMPUTACIÓN**
Ciudad.

De mi consideración.

Reciba un atento saludos y éxitos en su gestión como director de la Carrera de Computación de nuestra institución.

En virtud de conocer la excelencia académica de nuestros estudiantes de la carrera, solicito de la manera más cordial que bajo su digno intermedio se gestione la colaboración de, Jéssica Johana Montes Vera y Carlos Pierre Quijije Vera, estudiantes de noveno semestre de la Carrera de Computación de la ESPAM MFL, para el desarrollo de trabajo curricular en modalidad de sistematización de experiencia dentro de la unidad de docencia, investigación y vinculación de infraestructuras, y así atender la necesidad de la integración entre un asistente virtual y aplicativos de la carrera, denominando a la investigación como: **DESARROLLO DE UNA INTERFAZ CONVERSACIONAL INTEGRADA A UN ASISTENTE VIRTUAL.**

Agradeciendo su apoyo y contribución, y esperando una respuesta positiva, anticipo mis agradecimientos.

Atentamente,

PhD. Javier López Zambrano
**RESPONSABLE DE LA UDIV DE INFRAESTRUCTURA**
**CARRERA DE COMPUTACIÓN ESPAM MFL**

# ANEXO 2. DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE.

ChatUDIV

# Documento de Requerimientos de Software

***DESARROLLO DE UNA INTERFAZ CONVERSACIONAL INTEGRADA A UN ASISTENTE VIRTUAL***

***Fecha: 04/11/2022***

# Tabla de contenido

# Información del Proyecto

| | |
|---|---|
| Institución | ESPAM MFL |
| Proyecto | Desarrollo de una interfaz conversacional integrada a un asistente virtual |
| Fecha de preparación | 04 de noviembre del 2022 |
| Área requirente | Unidad de Docencia, Investigación y Vinculación de Infraestructura |
| Carrera | Computación |
| Responsable del proyecto | Carlos Pierre Quijije Vera<br>Jéssica Johana Montes Vera |
| Tutor del proyecto | Mtr. Fernando Moreira Moreira |
| Responsable de área | Ing. Yimmy Loor Vera<br>Ph.D. Javier López Zambrano |

## 1. Propósito

Este documento de Especificación de Requerimientos de Software (ERS) está basado en el "**Desarrollo de una interfaz conversacional integrada a un asistente virtual**" con el fin de brindar interacción entre el usuario y las aplicaciones correspondientes al área de la UDIV de Infraestructura de la Carrera de Computación de la ESPAM MFL. También, permite contar con la información necesaria para la construcción de la interfaz conversacional integrada a un asistente virtual. Este documento va dirigido hacia las partes interesadas como el responsable del proyecto, desarrolladores, entre otros, para servir como guía sobre el desarrollo, funcionamiento y posterior implementación de la interfaz conversacional.

## 2. Alcance del producto / Software

El proyecto tiene como objetivo principal "Desarrollar una interfaz conversacional integrada a un asistente virtual para brindar interacción entre el usuario y las aplicaciones de la UDIV de Infraestructura de la ESPAM MFL".

- **Objetivos específicos**
    - Diseñar una interfaz conversacional intuitiva y fácil de usar que permita a los usuarios interactuar de manera natural y fluida con las aplicaciones de la UDIV de Infraestructura.
    - Implementar una funcionalidad de reconocimiento de voz confiable y precisa.
    - Adaptar la interfaz conversacional para funcionar de manera eficiente en varios dispositivos
    - Integrar la interfaz conversacional con un asistente virtual.
    - Realizar pruebas exhaustivas de la interfaz conversacional para identificar y solucionar posibles errores y mejorar la experiencia del usuario.

# 3. Referencias


- Díaz, AFA & Ramos, ARR (2019). Especificación de Requisitos de Software.
- UDIV (Unidades de Docencia, Investigación y Vinculación). (2022). Documento de aprobación.


# 4. Funcionalidades del producto

- **Requisitos Funcionales**.

| Número de requisito | Nombre de requisito |
|---|---|
| **RF01** | Interacción en tiempo real |
| **RF02** | Usuarios y solicitudes |
| **RF03** | Modo oscuro |
| **RF04** | Configurar las alertas de notificaciones |
| **RF05** | Permitir funcionalidades ocultas |
| **RF06** | Envío de mensajes |
| **RF07** | Lenguaje de los mensajes |
| **RF08** | Revisión ortográfica |
| **RF09** | Estado de notificaciones |
| **RF010** | Gestión de conversación |
| **RF011** | Emisión de enlaces |
| **RF012** | Integración con las aplicaciones |
| **RF013** | Seguimiento de la conversación |
| **RF014** | Denunciar fallos en respuestas |

- **Requisitos No Funcionales.**

| Número de requisito | Nombre de requisito |
|---|---|
| **RNF01** | Interfaz del Sistema |
| **RNF02** | Ayuda en el uso del sistema |
| **RNF03** | Seguridad y Lógica de Datos |
| **RNF04** | Seguridad |
| **RNF05** | Fiabilidad |
| **RNF06** | Disponibilidad |
| **RNF07** | Contraste |
| **RNF08** | Adaptar interfaz a diferentes dispositivos |
| **RNF09** | Longitud de los mensajes |

# 5. Características de usuarios

El acceso a la interfaz conversacional está restringido solo para usuarios de la ESPAM MFL de la Carrera de Computación.

# 6. Requerimientos funcionales

## 6.1. Credenciales de usuario

**ACCESO**

| | |
|---|---|
| Número de requisito | RF01 |
| Nombre de requisito | Interacción en tiempo real |
| Tipo | ■ Requisito ☐ Restricción |
| Descripción del requisito | La interfaz conversacional debe ocupar el logueo del usuario cuando éste haya iniciado sesión en las aplicaciones de la UDIV (Gestión de Tareas y gestión de espacios). |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.2. Usuarios y solicitudes

| | |
|---|---|
| Número de requisito | RF02 |
| Nombre de requisito | Usuarios y solicitudes |
| Tipo | ■ Requisito ☐ Restricción |
| Descripción del requisito | La interfaz conversacional deberá proporcionar múltiples respuestas al usuario, sin afectar su rendimiento. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.3. Configurar el asistente virtual

**CONFIGURACIÓN DE LA INTERFAZ CONVERSACIONAL**

| | |
|---|---|
| Número de requisito | RF03 |
| Nombre de requisito | Modo oscuro |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | La interfaz conversacional debe cambiar de apariencia cada vez que el usuario lo requiera. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.4. Configurar las alertas de notificaciones

| | |
|---|---|
| Número de requisito | RF04 |
| Nombre de requisito | Configurar las alertas de notificaciones |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Los usuarios podrán activar/desactivar las notificaciones de la interfaz. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.5. Permitir funcionalidades ocultas

**GESTIÓN DE LOS MENSAJES**

| | |
|---|---|
| Número de requisito | RF05 |
| Nombre de requisito | Permitir funcionalidades ocultas |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Permitir a los usuarios mostrar u ocultar ciertas funcionalidades de la interfaz conversacional. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ☐Alta/Esencial ☐Media/Deseado ■ Baja/ Opcional |

## 6.6. Envío de mensajes

| | |
|---|---|
| Número de requisito | RF06 |
| Nombre de requisito | Envío de mensajes |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | El usuario podrá realizar las peticiones en texto y audio a la interfaz, en caso de que sea texto es importante restringir los emoticones. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.7. Lenguaje de los mensajes

| | |
|---|---|
| Número de requisito | RF07 |
| Nombre de requisito | Lenguaje de los mensajes |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | El usuario solo podrá realizar las peticiones en el idioma español e inglés. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.8. Revisión ortográfica

| | |
|---|---|
| Número de requisito | RF08 |
| Nombre de requisito | Revisión ortográfica |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | La interfaz conversacional será capaz de detectar faltas ortográficas cometidas por los usuarios y realizar sugerencias. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.9. Estado de notificaciones

| | |
|---|---|
| Número de requisito | RF09 |
| Nombre de requisito | Estado de notificaciones |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Los usuarios deben ser informados con una notificación en cada respuesta de la interfaz. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.10. Gestión de conversación

| | |
|---|---|
| Número de requisito | RF010 |
| Nombre de requisito | Gestión de conversación |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Debe ser capaz de mantener una conversación fluida con el usuario. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.11. Emisión de ficheros o enlaces

| | |
|---|---|
| Número de requisito | RF011 |
| Nombre de requisito | Emisión de ficheros o enlaces |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | La interfaz conversacional tendrá la disposición de dar como respuesta en forma de texto, con base en la petición que le realizó el usuario. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.12. Integración con las aplicaciones

| | |
|---|---|
| Número de requisito | RF012 |
| Nombre de requisito | Integración con las aplicaciones |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | La interfaz conversacional debe ser capaz de integrarse a las aplicaciones de la UDIV de Infraestructura de la carrera de Computación, para poder obtener la información que el usuario esté solicitando. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 6.13. Seguimiento de la conversación

| | |
|---|---|
| Número de requisito | RF013 |
| Nombre de requisito | Seguimiento de la conversación |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Debe poder realizar un seguimiento del historial de conversaciones. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

### 6.14. Denunciar fallos en respuestas

| | |
|---|---|
| Número de requisito | RF014 |
| Nombre de requisito | Denunciar fallos en respuestas |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Debe permitirle al usuario realizar denuncias sobre fallos de respuestas/coincidencias que la interfaz le esté proporcionando. En este caso habría un Bot de denuncia; el usuario podría informar de un problema y se desplegará una lista de los problemas más comunes que puede suceder al momento de que la interfaz proporciona una respuesta. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

# 7. Requerimientos de interfaces externas

## 7.1. Interfaces de usuario

En la interfaz conversacional cuenta con un diseño amigable e intuitivo para los usuarios, algunas de las características de la interfaz serán:

- Botones para ejecutar los diferentes procesos, como consultar, enviar, mensaje de voz, nueva conversación, buscar, etc.
- Ingreso por texto o por voz para realizar las peticiones que requiera el usuario.

## 7.2. Interfaces de hardware

El usuario puede utilizar la interfaz conversacional sin necesidad de instalación de cualquier sistema operativo adicional, excepto el navegador web.

## 7.3. Interfaces de software

Para poder acceder a todas las funcionalidades del producto y que este se ejecute correctamente el equipo de trabajo deberá instalar en su equipo las siguientes herramientas:

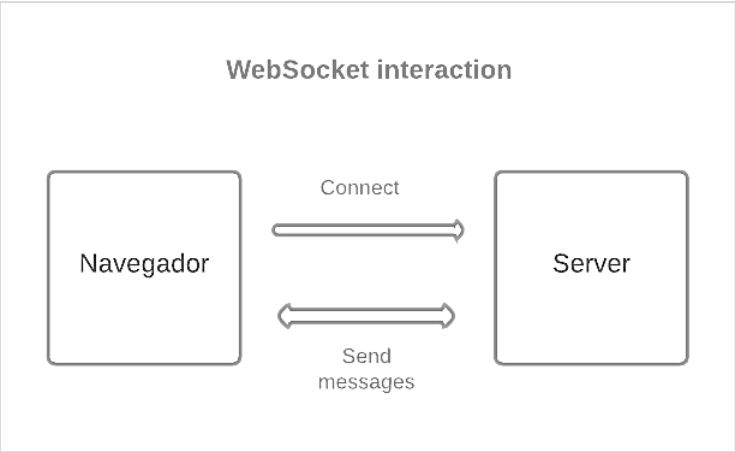
- Node.js
- Express.js
- Tailwind CSS
- WebSockets
- Web Components

- HTML y CSS
- Git y GitHub
- Sistema de paquetes NPM
- Editor de código VSC (recomendado)

### 7.4. Interfaces de comunicación

El protocolo de comunicación a usar es TCP, para permitir una conexión activa y bidireccional entre el servidor y el navegador.

Por lo tanto, se utilizará dos partes fundamentales para establecer una conexión WebSocket: WebSocket Server y WebSocket Cliente.




# 8. Requerimientos no funcionales

## 8.1. Interfaz del Sistema

| | |
|---|---|
| Número de requisito | RNF01 |
| Nombre de requisito | Interfaz del Sistema |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | La interfaz conversacional presentará una interfaz de usuario sencilla para que sea de fácil manejo a los usuarios. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 8.2. Ayuda en el uso del sistema

| | |
|---|---|
| Número de requisito | RNF02 |
| Nombre de requisito | Ayuda en el uso del sistema |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | La interfaz debe estar complementada con un buen sistema de ayuda. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 8.3. Seguridad y Lógica de Datos

| | |
|---|---|
| Número de requisito | RNF03 |
| Nombre de requisito | Seguridad y Lógica de Datos |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos de cada una de las aplicaciones. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 8.4. Seguridad

| | |
|---|---|
| Número de requisito | RNF04 |
| Nombre de requisito | Seguridad |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Todas las comunicaciones externas entre servidores de datos, aplicación y los usuarios del sistema deben estar encriptadas. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 8.5. Fiabilidad

| | |
|---|---|
| Número de requisito | RNF05 |
| Nombre de requisito | Fiabilidad |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | La interfaz debe poseer consistencia entre los datos de entrada versus los datos de salidas. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 8.6. Disponibilidad

| | |
|---|---|
| Número de requisito | RNF06 |
| Nombre de requisito | Disponibilidad |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | La interfaz conversacional debe tener una disponibilidad del 99,99% de las veces en que la interfaz esté en uso. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

## 8.7. Contraste

| | |
|---|---|
| Número de requisito | RNF07 |
| Nombre de requisito | Contraste |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Debe existir suficiente contraste de color entre los distintos elementos que componen la vista de la interfaz conversacional. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional |

### 8.8. Adaptar interfaz a diferentes dispositivos

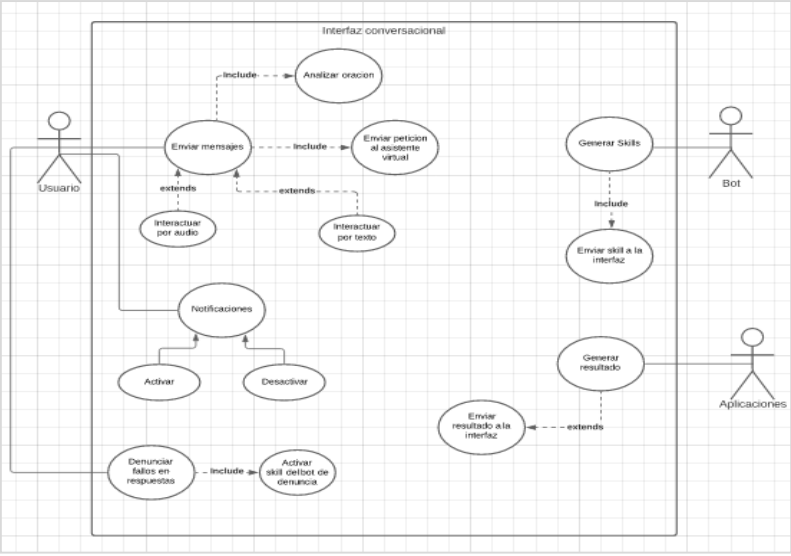
| | |
|---|---|
| Número de requisito | RNF08 |
| Nombre de requisito | Adaptar interfaz a diferentes dispositivos |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | Permitir que la aplicación sea responsive a cualquier dispositivo. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐Media/Deseado ☐Baja/ Opcional |

### 8.9. Longitud de los mensajes

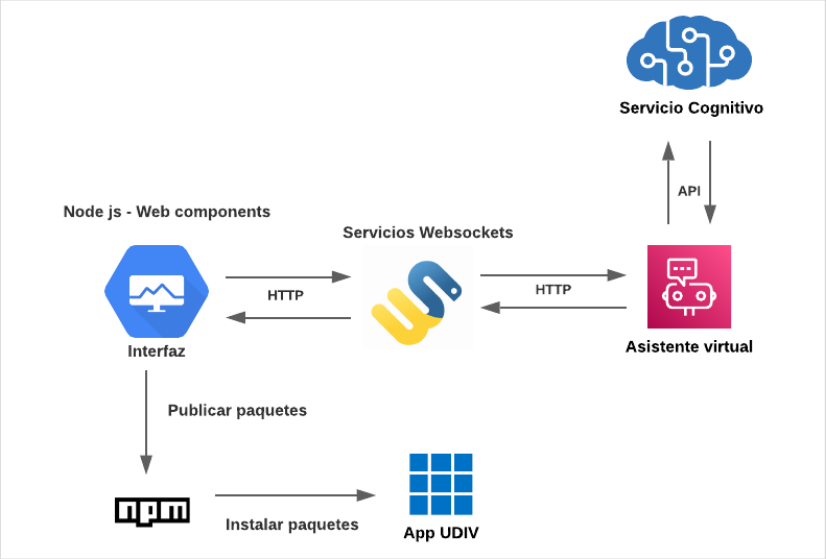
| | |
|---|---|
| Número de requisito | RNF09 |
| Nombre de requisito | Longitud de los mensajes |
| Tipo | ■Requisito ☐ Restricción |
| Descripción del requisito | El mensaje que redacte el usuario no tiene un límite de palabras. |
| Fuente del requisito | Usuario final |
| Prioridad del requisito | ■ Alta/Esencial ☐Media/Deseado ☐Baja/ Opcional |

## 9. Diagramas

### 9.1. Diagrama de casos de uso



### 9.2. Diagrama de arquitectura




# 10. Glosario

## 10.1. Términos

- **Usuario:** Persona que interactúa con el software
- **Interfaz conversacional:** Una interfaz conversacional es una forma de interacción de usuario que utiliza el lenguaje natural para permitir realizar actividades.
- **Node.js:** Node.js es un entorno de tiempo de ejecución de JavaScript en el servidor que ofrece velocidad, rendimiento y un modelo de programación basado en eventos.
- **NPM:** NPM es el administrador de paquetes de Node.js. NPM permite instalar y gestionar fácilmente paquetes de terceros, lo que facilita la reutilización de código y acelera el desarrollo de aplicaciones.
- **WebSocket:** WebSocket es un protocolo de comunicación bidireccional en tiempo real, que permite una conexión persistente y de baja latencia entre un cliente y un servidor a través de la web.
- **Web Components:** Los Web Components son un conjunto de tecnologías web que permiten crear componentes reutilizables y encapsulados para construir aplicaciones web.

## 10.2. Siglas

- **ERS:** Especificación de requerimientos de software.
- **XP:** Extreme Programming o Programación extrema.
- **UDIV:** Unidades de Docencia, Investigación y Vinculación.
- **NPM:** Node Package Manager.
- **VSC:** Visual Studio Code.

# ANEXO 3. ANÁLISIS DE LAS TÉCNICAS EMPLEADAS Y MÉTRICAS DE CALIDAD.

## ANÁLISIS DE LAS TÉCNICAS EMPLEADAS EN LOS ARTÍCULOS

- ✓ **Procesamiento del lenguaje natural (NLP):** utilizado para comprender el lenguaje natural humano y responder de manera adecuada.
- ✓ **Reconocimiento de voz:** utilizado para permitir al usuario interactuar con el asistente virtual mediante comandos de voz.
- ✓ **Análisis de intención:** utilizado para identificar la intención detrás de la pregunta del usuario para responder de manera precisa.
- ✓ **Generación de respuestas automatizadas:** utilizado para generar respuestas automatizadas para preguntas comunes y predefinidas.
- ✓ **Análisis de contexto:** utilizado para identificar el contexto de la conversación para poder mantener una conversación coherente con el usuario.
- ✓ **Personalización:** utilizado para adaptarse a las preferencias y necesidades individuales del usuario.
- ✓ **Integración con otras aplicaciones:** utilizado para integrarse con otras aplicaciones y servicios para proporcionar una experiencia de usuario más completa.
- ✓ **Diseño de Interfaz de usuario (UI):** se refiere al diseño visual de la interfaz conversacional. Se deben tener en cuenta aspectos como la claridad, la simplicidad y la facilidad de uso.

# TABLA DE CLASIFICACIÓN DE ARTÍCULOS

| ID | Año | Titulo | Autor | Entorno de la pagina web | Tipo de herramienta para integrar el asistente al software | Procesamiento del lenguaje natural (NLP) | Reconocimiento de voz | Análisis de intención | Personalización | Generación de respuestas automatizadas | Análisis de contexto | Integración con otras aplicaciones | Diseño de Interfaz de usuario (UI): |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2020 | Desarrollo e implementación de una plataforma web con chatbot para la comunicación activa entre usuario e información del portafolio de servicio de la empresa Electricsystems de la ciudad de Guayaquil. | Espinoza Hoyos Sonia Elizabeth | Angular | Dialogflow | X | X | X | X | X | X | X | X |
| 2 | 2022 | Sistema web con Chatbot para la gestión de ventas de productos de limpieza con aporte en quechua en la empresa Gea Chemical. | Chung Ku, Daniel Gustavo; Fiestas Ramirez,Gino Bismarck | Laravel | Dialogflow | X | X | - | X | X | X | X | X |
| 3 | 2021 | Diseño y desarrollo de la interfaz de usuario de una aplicación web para la mejora de la salud basada en un chatbot | Pablo Juan Fernández Cotrina | Angular | Dialogflow | - | - | - | X | X | X | - | X |
| 4 | 2021 | Desarrollo e implementación de sitio web con agente virtual a través de un chatbot para la empresa "El Rey del Embrague S.A." | Coppiano Ramírez Jhonny Maverick | Wordpress | Chatcompose | - | - | - | - | - | - | X | X |
| 5 | 2017 | Desarrollo e implementación de un sistema web para mejorar la administración de los procesos internos y el servicio al cliente de la Pyme Gráficas Rivas, implementando también una herramienta de inteligencia artificial chatbot. | Manuel Humberto Rivas Fuentes | Php 5.6 + framework codeigniter | Dialogflow | X | X | - | X | X | X | X | X |
| 6 | 2018 | Introducción de un diseño de una plataforma virtual para la interacción entre docente y estudiante con la integración de un asistente virtual (chatbot); orientada a los estudiantes del 2do y 3ro de Bachillerato en la especialización de Informática del Colegio Fiscal Técnico Provincia de Bolívar. | Mario Enrique Santos Méndez | Laravel | Dialogflow | - | - | - | - | - | - | - | - |
| 7 | 2020 | Implementación de una aplicación web con servicio de chatbot con inteligencia artificial que permita la autogestión de cuentas por pagar de los proveedores de la Universidad Autónoma de Bucaramanga. | Julián David Nieto Cortés | Flutter | Dialogflow | | - | x | x | x | x | - | x |
| 8 | 2022 | Implementación de un chatbot para la gestión de incidentes en la Plataforma Virtual de Educación a Distancia de la Universidad Inca Garcilaso de la Vega. | Zavaleta Zegarra, Naysha Medalit | PHP | Dialogflow | | - | x | x | - | x | x | x |
| 9 | 2018 | Diseño de un servicio de respuesta automático sobre el gasto público de chile mediante un asistente virtual de interfaz conversacional. | Orlando Andrés Rojas Romero | Flutter | NODE RED | | x | x | x | x | x | x | x |
| 10 | 2021 | Desarrollo e implementación de una página web con Chatbot, para el proceso de solicitud de exámenes de laboratorio de la empresa "SANLAC S.A." | Sánchez Díaz, Kléber Andrés | WordPress | Dialogflow | | x | x | x | x | x | x | x |
| 11 | 2020 | Sistema Web con Asistente Chatbot aplicando la metodología ICONIX, para las Ventas Online en la Empresa Comercial Sandra SAC de Tarapoto. | Gutiérrez Pizarro, Gonzalo Daniel | Angular | - | | - | - | - | x | x | - | x |
| 12 | 2018 | Implementación de un chatbot con botframework: caso de estudio, servicios a clientes del área de fianzas de seguros equinoccial. | Omar Humberto Zarabia Zuñiga | - | C# y visual basic. | | x | x | x | x | x | - | x |
| 13 | 2019 | Desarrollo de una aplicación web alternativa para videoconferencia y compartición de pantalla con el uso de WebRTC. | Peralbo Velasco Michelle Alejandra | Node.js y Express | WebRTC | x | x | x | x | x | x | x | x |
| 14 | 2017 | Desarrollo de un prototipo de aplicación web y móvil para la gestión de obras de instalación de fibra óptica. | Alejandro Pérez Martín | Node.js | - | - | - | - | - | x | - | - | - | x |
| 15 | 2017 | Desarrollo de aplicaciones web con Node.js | Jesús Octavio Guardado Osuna | Node.js | - | - | - | - | - | x | - | - | - | x |
| 16 | 2018 | Desarrollo del sistema de requisiciones para la Empresa Hidroeléctrica Abanico S.A. aplicando el entorno de programación Node.js | Diana Elizabeth Gómez García | Node.js | Python | x | - | - | x | x | x | - | x |

Promedio de uso de las técnicas empleadas en los artículos.

| | (NLP) | Reconocimiento de voz | Análisis de intención | Personalización | Generación de respuestas automatizadas | Análisis de contexto | Integración con otras apps | (UI) |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| **Total** | 11 | 7 | 7 | 13 | 11 | 12 | 8 | 15 |
| **Promedio de uso** | 69% | 44% | 44% | 81% | 69% | 75% | 50% | 94% |

# ANEXO 4. REALIZAR PRUEBAS UNITARIAS DE LA INTERFAZ INTEGRADA CON EL ASISTENTE VIRTUAL.

## PRUEBAS UNITARIAS DE LA INTERFAZ CONVERSACIONAL

Para garantizar el correcto funcionamiento de cada componente de la interfaz conversacional de manera individual, es esencial llevar a cabo pruebas unitarias. Estas pruebas permiten asegurar que cada componente funcione correctamente y que cumpla con los requisitos establecidos. Además, son una parte fundamental del desarrollo de software y permiten detectar errores y fallos de manera temprana, lo que facilita su corrección y reduce los costos asociados.

En el caso específico de la interfaz integrada con el asistente virtual, es necesario asegurarse de que la conexión con la API del asistente se establezca correctamente y que las respuestas recibidas sean las esperadas. Para lograr esto, se utilizó código específico que establece la conexión con la API del asistente virtual y envía mensajes de prueba para verificar las respuestas. De esta manera, se podrá verificar que la conexión se establezca correctamente y que el asistente virtual responda adecuadamente a los mensajes enviados.

Además, se realizaron pruebas para validar la implementación y comunicación del protocolo de transmisión de mensajes en tiempo real a través de los WebSockets.

**OBJETIVO:** Validar la correcta implementación y comunicación del protocolo de transmisión de mensajes en tiempo real a través de los WebSockets entre el cliente y el servidor.

Para validar la correcta implementación y comunicación del protocolo de transmisión de mensajes en tiempo real a través de los WebSockets, se llevaron a cabo pruebas unitarias que incluyeron la implementación del código para enviar y recibir mensajes a través del objeto "socket" y el evento "socket.on".

En la función "io.on", se estableció el evento "connection" para detectar cuándo un usuario se conecta al servidor. Posteriormente, en el evento "socket.on", se implementó la función "Send message" que emite un mensaje al socket con los datos del mensaje enviado.

Además, se agregó el evento "new message" para recibir los mensajes enviados por el usuario y agregarlos al Chat con un formato adecuado. Esto incluyó la obtención de la hora actual mediante la creación de un objeto "Date" y la generación de una cadena de caracteres para mostrar la hora de manera legible.

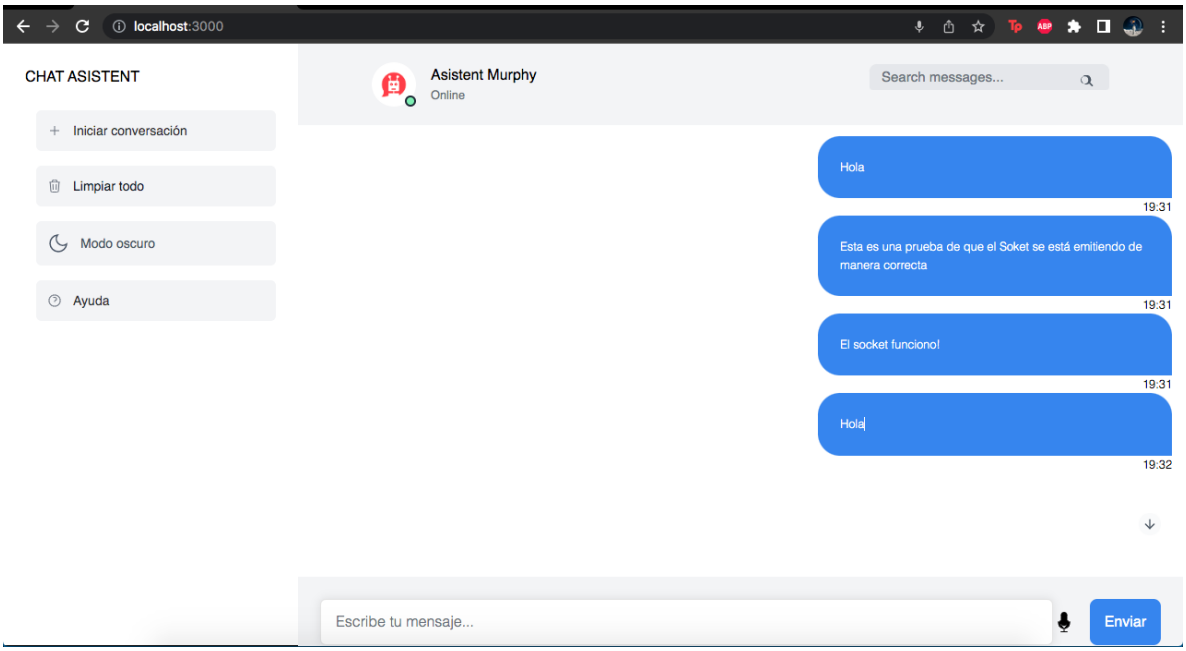
El código utilizado para realizar estas pruebas unitarias es el siguiente:

```
socket.on('Send message', function(data) {

    socket.emit('new message', data);


    const now = new Date();
    const time = now.getHours() + ':' + (now.getMinutes() < 10 ? '0' : '') + now.getMinutes();

    $chat.append('<div class="flex flex-col message-text">' +
      '<div class="bg-blue-500 p-6 w-96 rounded-3xl rounded-br-none self-end">' +
      '<small class="text-white rounded font-light">' + data + '</small>' +
      '</div>' +
      '<small id="TimeDark" class="text-black font-light self-end">' + time + '</small>' +
      '</div>'
    );
  });
```

**RESULTADOS**

```
> nodemon src/index.js

[nodemon] 2.0.21
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
server on port 3000
Un usuario se ha conectado
Un usuario se ha conectado
```

En la ilustración, se muestra la salida de consola generada durante la implementación de la funcionalidad para escuchar las conexiones entrantes utilizando la librería Socket.IO. El mensaje de consola "Un usuario se ha conectado" confirma que el proceso de conexión se realizó exitosamente y que la interfaz conversacional está lista para transmitir y recibir mensajes.




Durante la ejecución de las pruebas unitarias, se pudo verificar que los mensajes enviados por el cliente se agregaron correctamente al chat y se transmitieron de manera efectiva a través de los sockets. La ejecución de diferentes mensajes de prueba permitió comprobar la correcta implementación del protocolo de transmisión de mensajes en tiempo real a través de los WebSockets, lo que garantiza una comunicación efectiva y estable entre el cliente y el servidor.

En resumen, los resultados obtenidos en las pruebas unitarias confirman la capacidad de la interfaz conversacional para manejar múltiples conexiones simultáneas sin comprometer su rendimiento y estabilidad, lo que es fundamental para brindar una experiencia de usuario óptima.

**OBJETIVO:** Validar la funcionalidad de la conexión con la API del asistente virtual y la coherencia de las respuestas recibidas.

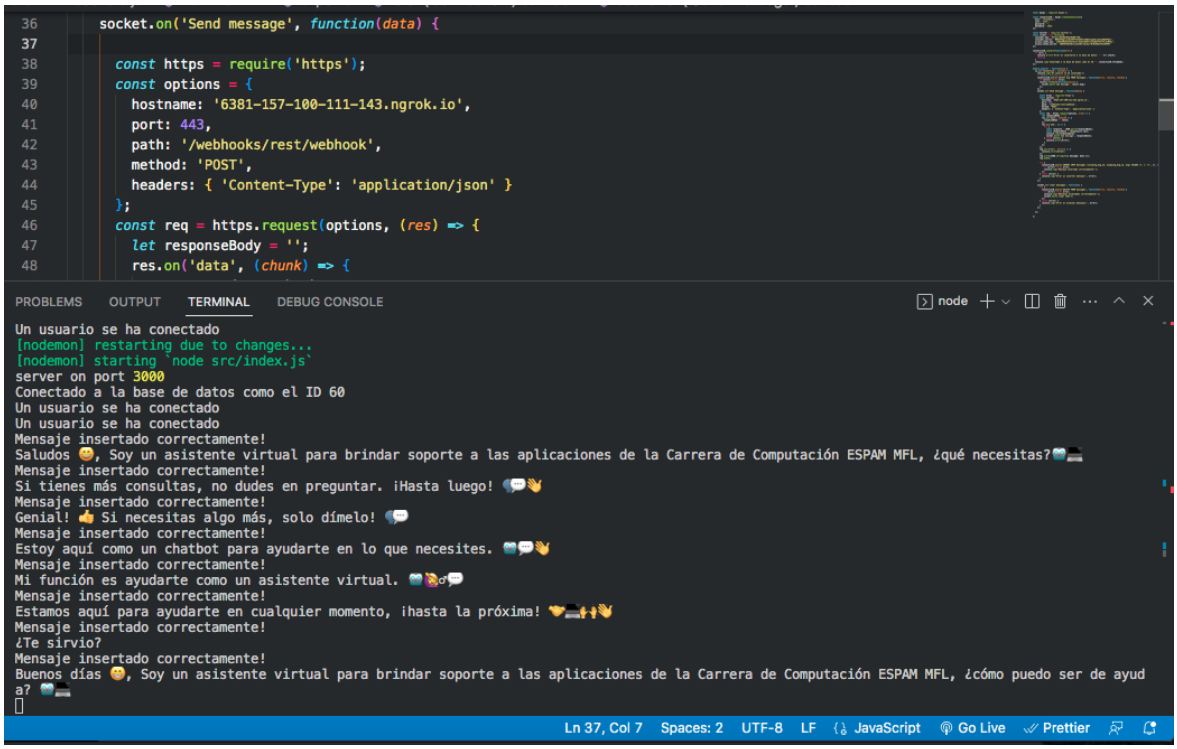
Para comprobar la conectividad con la API del asistente virtual, se diseñaron pruebas unitarias utilizando la librería HTTP. La función implementada establece una conexión segura con el servidor del asistente virtual, especificado en la variable "options", y envía un mensaje de prueba mediante el método POST. Posteriormente, se almacena la respuesta del asistente virtual en la variable "responseData" y se muestra en la consola mediante la función "console.log()".

Después de llevar a cabo múltiples pruebas unitarias con diferentes mensajes de prueba, se validó que la conexión con la API del asistente virtual se establece correctamente y que las respuestas recibidas son las esperadas. Gracias a este enfoque, se garantiza el correcto funcionamiento de la conexión con la API y la calidad de la comunicación en la interfaz.

El código utilizado para realizar estas pruebas unitarias es el siguiente:

```
const https = require('https');
const options = {
  hostname: '25a0-200-24-154-53.ngrok.io/',
  port: 443,
  path: '/webhooks/rest/webhook',
  method: 'POST',
  headers: { 'Content-Type': 'application/json' }
};

const req = https.request(options, (res) => {
  let responseBody = '';
  res.on('data', (chunk) => {
    responseBody += chunk;
  });
  res.on('end', () => {
    try {
      const response = JSON.parse(responseBody);
      const responseData = response[0].text;
      console.log(responseData);
    } catch (error) {
      console.error(error);
    }
  });
});

req.on('error', (error) => {
  console.error(error);
});

req.write(JSON.stringify({ message: 'Test message' }));
req.end();
```

# RESULTADOS




Como se puede observar en la ilustración, se ha establecido una conexión exitosa con la API del asistente virtual mediante el protocolo HTTP y la dirección proporcionada en la variable de opciones. Se ha confirmado que el envío de mensajes al chat mediante el socket se realiza de manera correcta, enviando el mensaje a la API del asistente virtual y mostrando la respuesta correspondiente en la consola de forma coherente con las preguntas formuladas.

Además, se llevaron a cabo diferentes pruebas unitarias con mensajes de prueba para evaluar el correcto funcionamiento, lo cual indica que la interfaz integrada con el asistente virtual proporciona una experiencia satisfactoria al usuario en la mayoría de los casos.

# ANEXO 5. REALIZAR PRUEBAS DE INTEGRACIÓN PARA DETERMINAR CÓMO SE COMPORTA LA INTERFAZ CONVERSACIONAL CON EL ASISTENTE VIRTUAL.

# INTRODUCCIÓN

Las pruebas de integración son fundamentales para garantizar un comportamiento consistente y de alta calidad en diferentes situaciones de uso en el desarrollo de aplicaciones modernas. En este proyecto, se creó una interfaz conversacional integrada con un asistente virtual, y se evaluó exhaustivamente su comportamiento en distintos escenarios.

El objetivo principal de estas pruebas fue asegurar que la interfaz conversacional ofreciera un comportamiento óptimo en situaciones complejas y cambiantes. Para lograr esto, se realizó un análisis detallado de su comportamiento, evaluando la precisión y eficacia en la interpretación de los comandos de voz o texto del usuario y la transmisión de estas solicitudes al asistente virtual.

Se puso especial atención en la capacidad de la interfaz para manejar múltiples solicitudes simultáneamente y garantizar la coherencia y pertinencia de las respuestas a cada solicitud. Además, se llevaron a cabo pruebas exhaustivas de las solicitudes utilizando la herramienta Postman para asegurarse de que los resultados generados por la interfaz fueran coherentes con las respuestas del asistente virtual y no hubiera modificaciones en las respuestas al ser transmitidas a través de los sockets.

# DESARROLLO

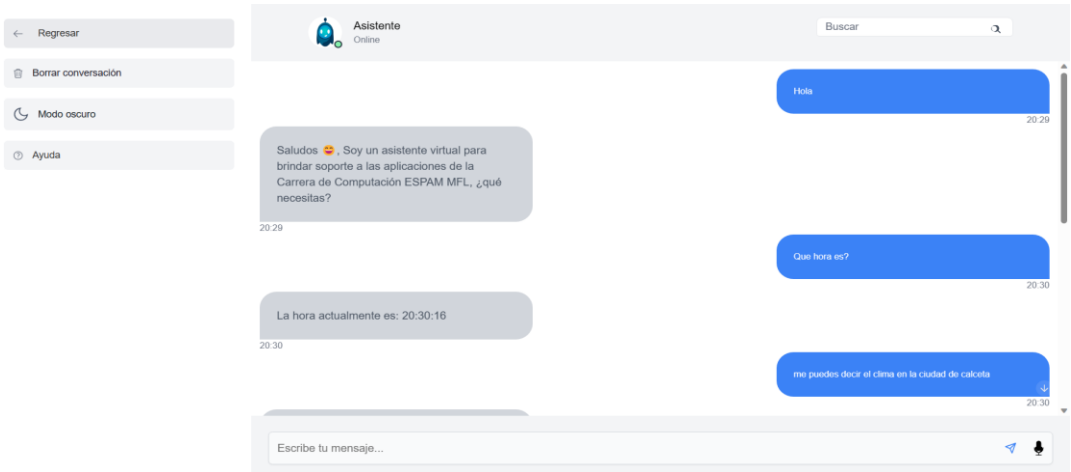
**Evaluación del comportamiento de la interfaz conversacional en diferentes escenarios de uso.**

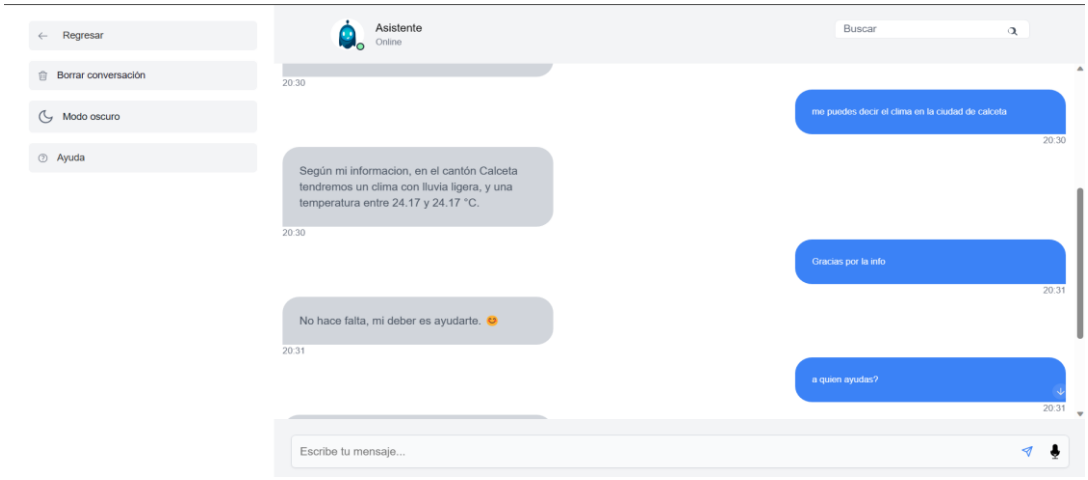
Es necesario definir diferentes escenarios de uso en los que se evalúe el comportamiento de la interfaz conversacional con el asistente virtual. Entre estos escenarios se definió lo siguiente:

- Conversaciones básicas en las que el usuario solicita información como la hora, el clima, saludos y despedidas.
- Conversaciones más complejas en las que el usuario intenta confundir al asistente virtual con preguntas o solicitudes ambiguas o inesperadas.

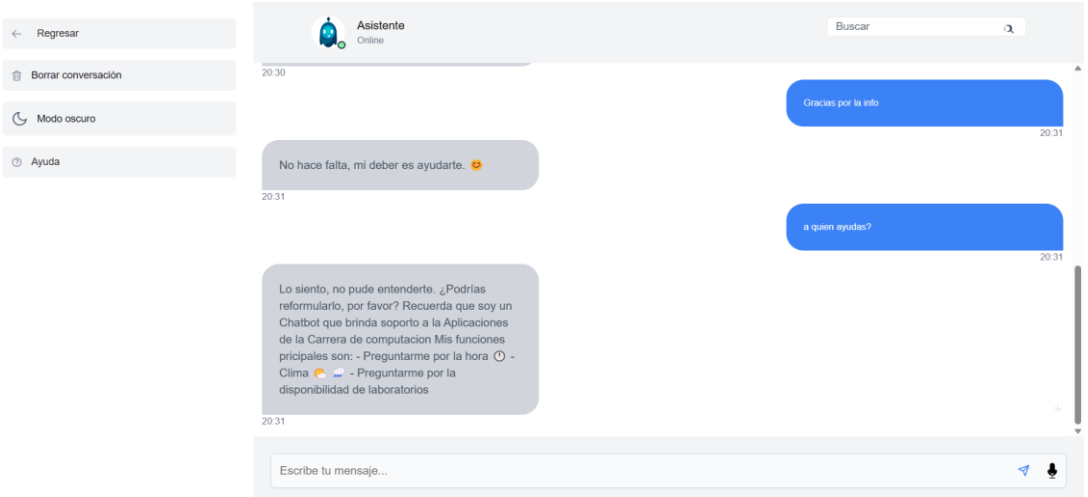
En este sentido, al realizar estos escenarios, se llevó a cabo una evaluación de la capacidad de la interfaz conversacional para manejar múltiples solicitudes sin que esto afecte su rendimiento, mediante la simulación de una conversación prolongada con el asistente virtual. Ya que, es esencial evitar que la interfaz se sature o colapse, ya que esto podría comprometer la calidad de la experiencia del usuario y, por lo tanto, afectar negativamente la percepción del servicio. Por lo tanto, es importante asegurar que la interfaz sea capaz de manejar una alta carga de solicitudes sin afectar su rendimiento y garantizar una experiencia de usuario fluida y satisfactoria.

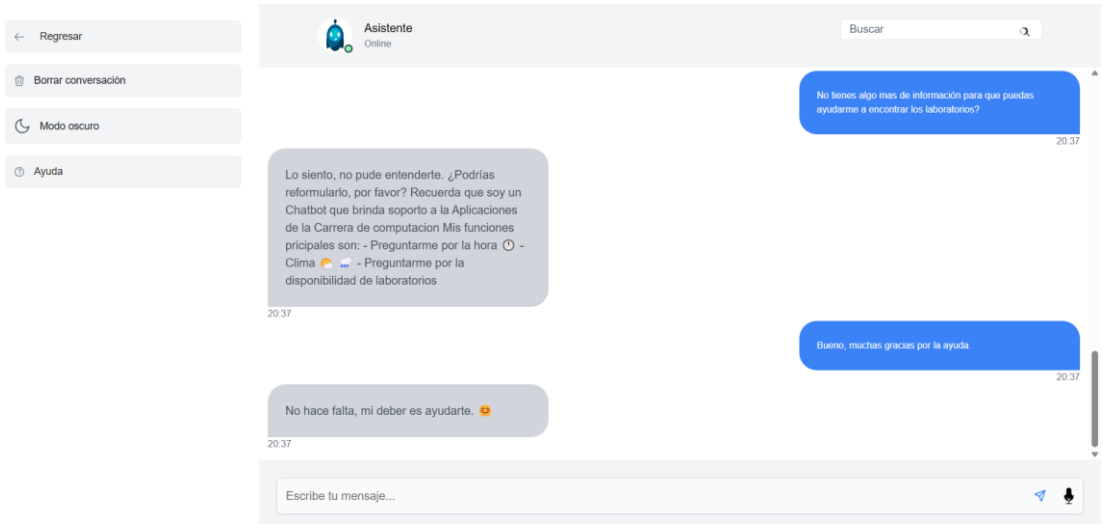
**Evidencias:**




Regresar
Borrar conversación
Modo oscuro
Ayuda
Asistente
Online
Buscar
20:30
me puedes decir el clima en la ciudad de calceta
20:30
Según mi informacion, en el cantón Calceta tendremos un clima con lluvia ligera, y una temperatura entre 24.17 y 24.17 °C.
20:30
Gracias por la info
20:31
No hace falta, mi deber es ayudarte.
20:31
a quien ayudas?
20:31
Escribe tu mensaje...


Regresar
Borrar conversación
Modo oscuro
Ayuda
Asistente
Online
Buscar
reformularlo, por favor? Recuerda que soy un Chatbot que brinda soporto a la Aplicaciones de la Carrera de computacion Mis funciones pricipales son: - Preguntarme por la hora - Clima - Preguntarme por la disponibilidad de laboratorios
20:36
No tienes algo mas de información para que puedas ayudarme a encontrar los laboratorios?
20:37
Lo siento, no pude entenderte. ¿Podrías reformularlo, por favor? Recuerda que soy un Chatbot que brinda soporto a la Aplicaciones de la Carrera de computacion Mis funciones pricipales son: - Preguntarme por la hora - Clima - Preguntarme por la disponibilidad de laboratorios
20:37
localhost
Grabación iniciada
Comenzó la grabación de voz.
a través de Microsoft Edge
Bueno, muchas gracias por la ayuda.

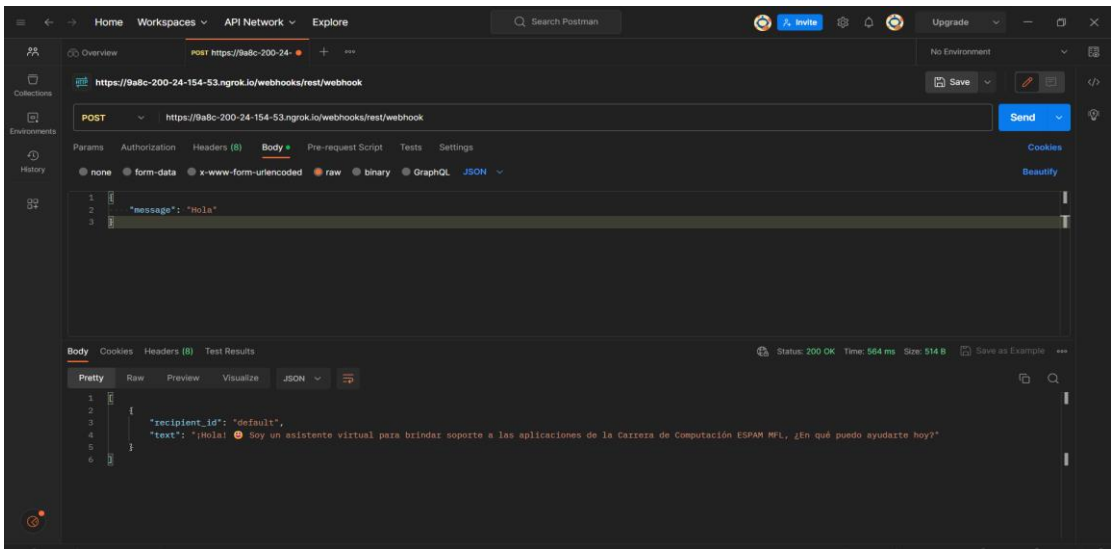

Regresar
Borrar conversación
Modo oscuro
Ayuda
Asistente
Online
Buscar
20:30
Gracias por la info
20:31
No hace falta, mi deber es ayudarte.
20:31
a quien ayudas?
20:31
Lo siento, no pude entenderte. ¿Podrías reformularlo, por favor? Recuerda que soy un Chatbot que brinda soporto a la Aplicaciones de la Carrera de computacion Mis funciones pricipales son: - Preguntarme por la hora - Clima - Preguntarme por la disponibilidad de laboratorios
20:31
Escribe tu mensaje...

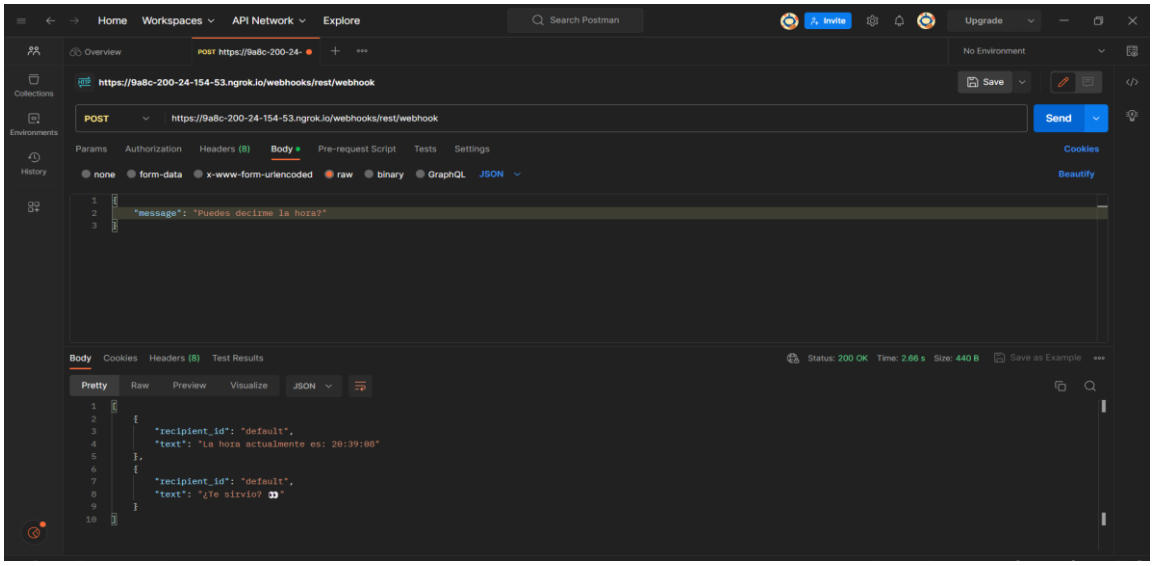


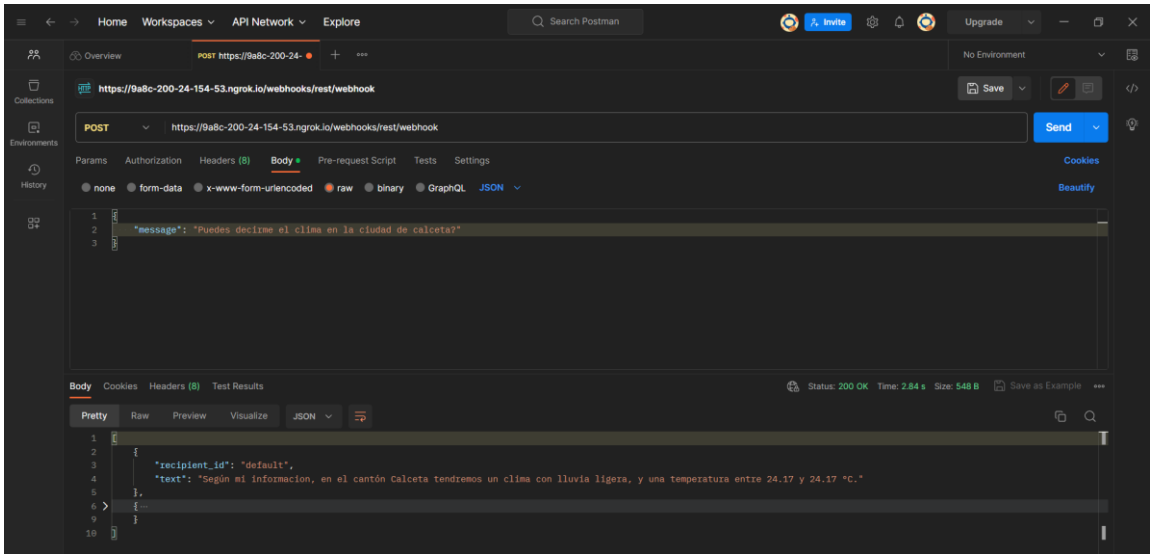

**Implementación de pruebas exhaustivas de solicitudes con Postman para garantizar la integridad de las respuestas de la interfaz.**

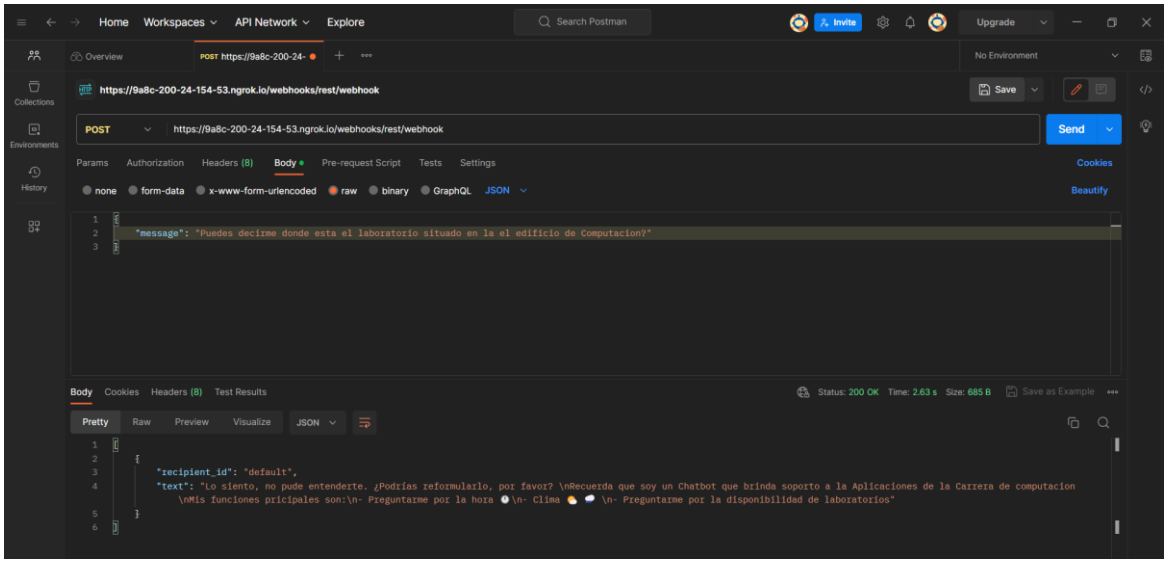
Una parte importante de esta tarea es asegurarse de que las solicitudes realizadas por los usuarios se procesen correctamente y que las respuestas proporcionadas por el asistente virtual sean precisas y consistentes. En este contexto, la realización de pruebas exhaustivas de solicitudes con herramientas como Postman se convierte en una tarea fundamental para garantizar la integridad de las respuestas del asistente y que estas coincidan cuando se agreguen al chat.

**Evidencias:**




Home Workspaces API Network Explore
Search Postman
Invite
Upgrade
Overview
POST https://9a8c-200-24-
No Environment
https://9a8c-200-24-154-53.ngrok.io/webhooks/rest/webhook
Save
POST
https://9a8c-200-24-154-53.ngrok.io/webhooks/rest/webhook
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"message": "Puedes decirme la hora?"
}
Body Cookies Headers (8) Test Results
Status: 200 OK Time: 2.66 s Size: 440 B
Save as Example
Pretty Raw Preview Visualize JSON
[
{
"recipient_id": "default",
"text": "La hora actualmente es: 20:39:00"
},
{
"recipient_id": "default",
"text": "¿Te sirvio? 👀"
}
]


Home Workspaces API Network Explore
Search Postman
Invite
Upgrade
Overview
POST https://9a8c-200-24-
No Environment
https://9a8c-200-24-154-53.ngrok.io/webhooks/rest/webhook
Save
POST
https://9a8c-200-24-154-53.ngrok.io/webhooks/rest/webhook
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"message": "Puedes decirme el clima en la ciudad de calceta?"
}
Body Cookies Headers (8) Test Results
Status: 200 OK Time: 2.84 s Size: 548 B
Save as Example
Pretty Raw Preview Visualize JSON
[
{
"recipient_id": "default",
"text": "Según mi información, en el cantón Calceta tendremos un clima con lluvia ligera, y una temperatura entre 24.17 y 24.17 °C."
},
{...}
]


Home Workspaces API Network Explore
Search Postman
Invite
Upgrade
Overview
POST https://9a8c-200-24-
No Environment
https://9a8c-200-24-154-53.ngrok.io/webhooks/rest/webhook
Save
POST
https://9a8c-200-24-154-53.ngrok.io/webhooks/rest/webhook
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"message": "Puedes decirme donde esta el laboratorio situado en la el edificio de Computacion?"
}
Body Cookies Headers (8) Test Results
Status: 200 OK Time: 2.63 s Size: 685 B
Save as Example
Pretty Raw Preview Visualize JSON
[
{
"recipient_id": "default",
"text": "Lo siento, no pude entenderte. ¿Podrías reformularlo, por favor? \nRecuerda que soy un Chatbot que brinda soporte a la Aplicaciones de la Carrera de computacion
\nMis funciones pricipales son:\n- Preguntarme por la hora \n- Clima \n- Preguntarme por la disponibilidad de laboratorios"
}
]

# CONCLUSIONES

- La realización de pruebas exhaustivas de integración y de solicitudes con herramientas como Postman permitió asegurar un comportamiento consistente y de alta calidad de la interfaz conversacional integrada al asistente virtual en diversos escenarios de uso. Como resultado, se logró garantizar la precisión y eficacia en la interpretación de los comandos de voz o texto del usuario, así como la capacidad de la interfaz para manejar múltiples solicitudes simultáneamente y garantizar la coherencia y pertinencia de las respuestas a cada solicitud, lo que asegura una experiencia satisfactoria para el usuario final.
- La evaluación del comportamiento de la interfaz conversacional en diferentes escenarios de uso permitió detectar y corregir errores y fallos en el funcionamiento del asistente virtual y en la capacidad de la interfaz para manejar solicitudes complejas. Como resultado, se logró mejorar la calidad y la eficacia del servicio prestado al usuario final, asegurando una experiencia fluida y sin contratiempos en la interacción con el asistente virtual, lo que aumenta la satisfacción del usuario y la eficiencia del servicio.

# ANEXO 6. MANUAL DEL PROGRAMADOR.

UNIDAD DE TECNOLOGÍA

# MANUAL TÉCNICO DE PROGRAMADOR

**OBJETIVO**

Registrar la metodología, estructura lógica y características determinantes con las que se ha desarrollado el proyecto para que sirva como una guía de consulta que facilitará la realización de correcciones, actualizaciones y mejoras en el sistema, así como brindar soporte técnico y servir de documentación y evaluación del producto.

ChatUDIV

# DESARROLLO DE UNA INTERFAZ CONVERSACIONAL INTEGRADA A UN ASITENTE VIRTUAL

## Manual de Programador

**Autores:**

Carlos Pierre Quijije Vera

Jéssica Johana Montes Vera

**Versión 1.0.0**

**2023/06/08**

# TABLA DE CONTENIDO

# 1 INTRODUCCIÓN

El propósito de este manual es proporcionar a los desarrolladores una comprensión detallada de la arquitectura, los procedimientos, las funciones, las configuraciones y las dependencias utilizadas en la codificación del proyecto de Titulación titulado "Desarrollo de una interfaz conversacional integrada a un asistente virtual". Lo cual servirá como una guía exhaustiva que permitirá a los desarrolladores familiarizarse con todos los aspectos técnicos clave del proyecto y facilitará el mantenimiento, las actualizaciones y el soporte continuo del sistema.

# 2 IDENTIFICACIÓN DEL SISTEMA

| PARAMETROS | DETALLE |
|---|---|
| Nombre del Sistema | ChatUDIV |
| Versión | V1.0 |
| Logotipo |   |
| Área de desarrollo | Esta plataforma fue desarrollada para la UDIV de Infraestructura de la Carrera de Computación de la ESAPM MFL |
| Desarrollador / Equipo | Carlos Pierre Quijije Vera<br>Jéssica Johana Montes Vera |
| Modelo de desarrollo | Extreme Programming (XP) |
| Paradigma de programación | Programación lógica, Funcional, basado en eventos y asincrónica. |

# 3 ALCANCE

Este documento proporcionará información y directrices sobre la interfaz conversacional integrada a un asistente virtual. Esta misma describe información sobre los requerimientos y requisitos para acciones futuras sobre el desarrollo del producto. Básicamente, se centra en la configuración de los servidores, estructura lógica y características determinantes utilizadas para desarrollar la interfaz conversacional y su integración con el asistente virtual.

# 4 REQUERIMIENTOS

## 4.1 REQUERIMIENTOS DE HARDWARE

Las características que debe poseer el ordenador mediante el cual se interactuará con la solución software son las siguientes:

- Sistema operativo Windows, Linux o Mac
- 8 GB RAM mínimo
- Procesador Intel Core I3 o superior.
- Disco solido de 20 GB
- Acceso a Internet

## 4.2 HERRAMIENTAS DE SOFTWARE

Para poder acceder a todas las funcionalidades del producto y que este se ejecute correctamente el equipo de trabajo deberá instalar en su equipo las siguientes herramientas:

Dependencias necesarias:

- "axios": "^1.3.4",
- "cors": "^2.8.5",
- "cspell": "^6.30.0",
- "esm": "^3.2.25",
- "express": "^4.18.2",
- "hunspell-spellchecker": "^1.0.2",
- "mysql": "^2.18.1",
- "natural": "^6.2.0",
- "node-fetch": "^3.3.1",
- "socket.io": "^4.6.1",
- "typo-js": "^1.2.2"

Sistema de paquetes NPM

Editor de código VsCode (recomendado)

Node JS

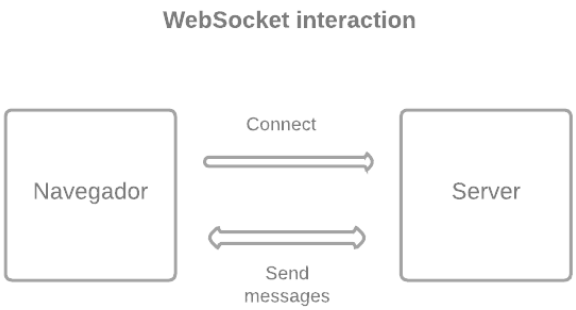
## 4.3 CONOCIMIENTOS PREVIOS

Para que un equipo de trabajo realice cambios necesarios en la estructura de la interfaz conversacional se deben tener conocimientos acerca de:

- Node JS
- Express
- WebSockets
- Web Components
- HTML y CSS
- Tailwind CSS
- Git y GitHub

# 5 ANÁLISIS Y DISEÑO DE LA APLICACIÓN

## 5.1 Arquitectura Cliente-Servidor




# 6 Descripción de la configuración de la Interfaz Conversacional

## 6.1 Configuración del servidor:

```
src > JS index.js > ...
1   const express = require('express');
2   const app = express();
3   const http = require('http');
4   const path = require('path');
5   const server = http.createServer(app);
6   const socketio = require('socket.io');
7   const io = socketio(server);
8   const cors = require('cors');
9
10  app.set('port', process.env.PORT || 3000);
11
12  app.use(cors());
13
14  const sockets = require('./sockets');
15  sockets(io);
16  app.use(express.static(path.join(__dirname, 'Public')));
17
18  server.listen(app.get('port'), () => {
19    console.log('server on port', app.get('port'))
20  });
```

Este fragmento de código configura un servidor utilizando Express y Socket.IO. A continuación, se explica línea por línea:

- Se importan los módulos necesarios para el servidor: express, http, path, socket.io y cors.
- app representa la instancia de la aplicación Express.
- **http.createServer(app)** crea un servidor HTTP utilizando la instancia de la aplicación Express.
- **socketio(server)** crea una instancia de Socket.IO asociada al servidor HTTP.
- cors se utiliza como middleware para permitir solicitudes de origen cruzado (CORS) en el servidor.

- **app.set('port', ...)** establece el puerto en el que se ejecutará el servidor. Si no se proporciona un puerto en la variable de entorno process.env.PORT, se utilizará el puerto 3000.
- **app.use(cors())** habilita el middleware CORS para permitir solicitudes de diferentes orígenes.
- **sockets** es un módulo que contiene la lógica para manejar los eventos del WebSocket.
- **app.use(express.static())** especifica una ruta estática para servir archivos estáticos, como archivos HTML, CSS o JavaScript.
- **server.listen()** inicia el servidor y lo hace escuchar en el puerto especificado. El mensaje de consola indica en qué puerto se ha iniciado el servidor.

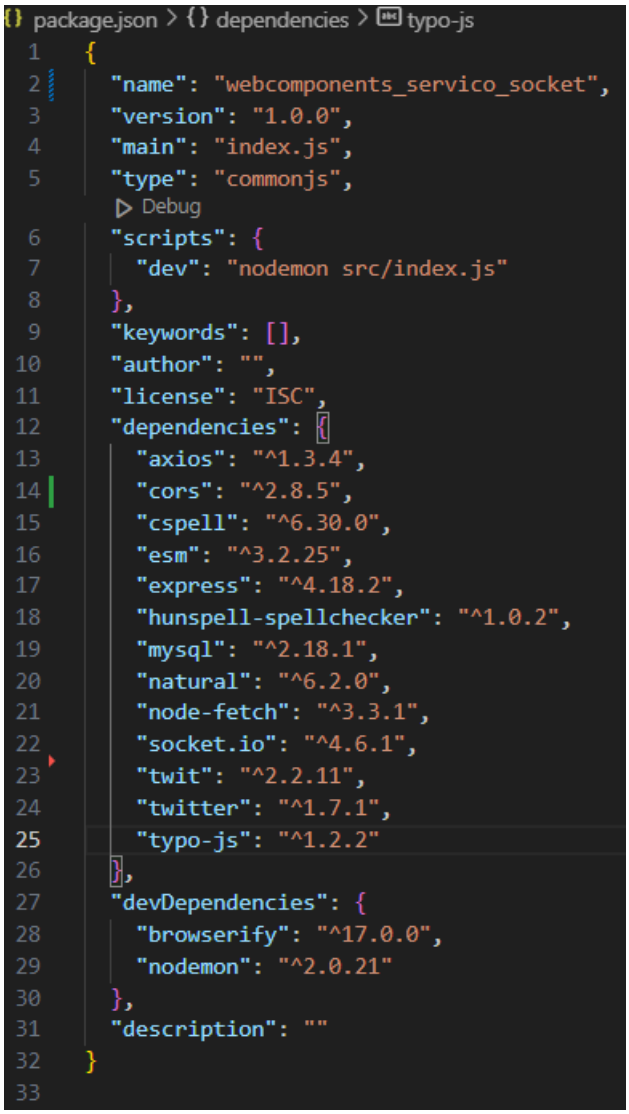
## 6.2 Configuración del WebSocket:

```
module.exports = function(io) {
  io.on('connection', (socket) => {
    socket.on('Send message', function(data) {
      io.emit('new user message', { message: data });
      const https = require('https');
      const options = {
        hostname: '72d6-200-24-154-53.ngrok.io',
        port: 443,
        path: '/webhooks/rest/webhook',
        method: 'POST',
        headers: { 'Content-Type': 'application/json' }
      };

      const req = https.request(options, (res) => {
        let responseBody = '';

        res.on('data', (chunk) => {
          responseBody += chunk;
        });

        res.on('end', () => {
          try {
            const response = JSON.parse(responseBody);
            const responseMessages = response.map((item) => item.text);
            io.emit('new bot message', { messages: responseMessages });
          } catch (error) {···
          }
        });
      });

      req.on('error', (error) => {···
      });

      req.write(JSON.stringify({ message: data }));
      req.end();
    });

    socket.on('clear messages', function() {···
    });
  });
};
```

- **module.exports = function(io) { ... }:** Exporta una función que recibe la instancia de Socket.IO io como argumento.
- **io.on('connection', (socket) => { ... }):** Maneja el evento de conexión, se ejecuta cuando un cliente se conecta al servidor.

- **socket.on('Send message', function(data) { ... }):** Maneja el evento 'Send message' enviado por el cliente. Recibe el mensaje enviado por el usuario en el parámetro data.
- **io.emit('new user message', { message: data });:** Emite el evento 'new user message' a todos los clientes conectados, enviando el mensaje del usuario como objeto { message: data }.
- **res.on('data', (chunk) => { ... }):** Recopila los datos recibidos en la respuesta del chatbot a medida que llegan.
- **res.on('end', () => { ... }):** Se ejecuta cuando la respuesta del chatbot ha finalizado. Aquí se procesa la respuesta recibida.
- **const response = JSON.parse(responseBody);:** Se analiza la respuesta JSON del chatbot y se guarda en la variable response.
- **const responseMessages = response.map((item) => item.text);:** Se extraen los mensajes de texto de la respuesta del chatbot y se guardan en la variable responseMessages.
- **io.emit('new bot message', { messages: responseMessages });:** Emite el evento 'new bot message' a todos los clientes conectados, enviando los mensajes de respuesta del chatbot como objeto { messages: responseMessages }..
- **req.on('error', (error) => { ... }):** Maneja los errores de la solicitud HTTP al chatbot.
- **req.write(JSON.stringify({ message: data }));:** Escribe el mensaje del usuario en el cuerpo de la solicitud HTTP al chatbot.
- **req.end();:** Finaliza la solicitud HTTP al chatbot.
- **socket.on('clear messages', function() { ... }):** Maneja el evento 'clear messages' enviado por el cliente para borrar los mensajes en el chat.
- **socket.emit('clear chat');:** Emite el evento 'clear chat' al cliente que envió el evento 'clear messages', lo que permite borrar los mensajes en el cliente.

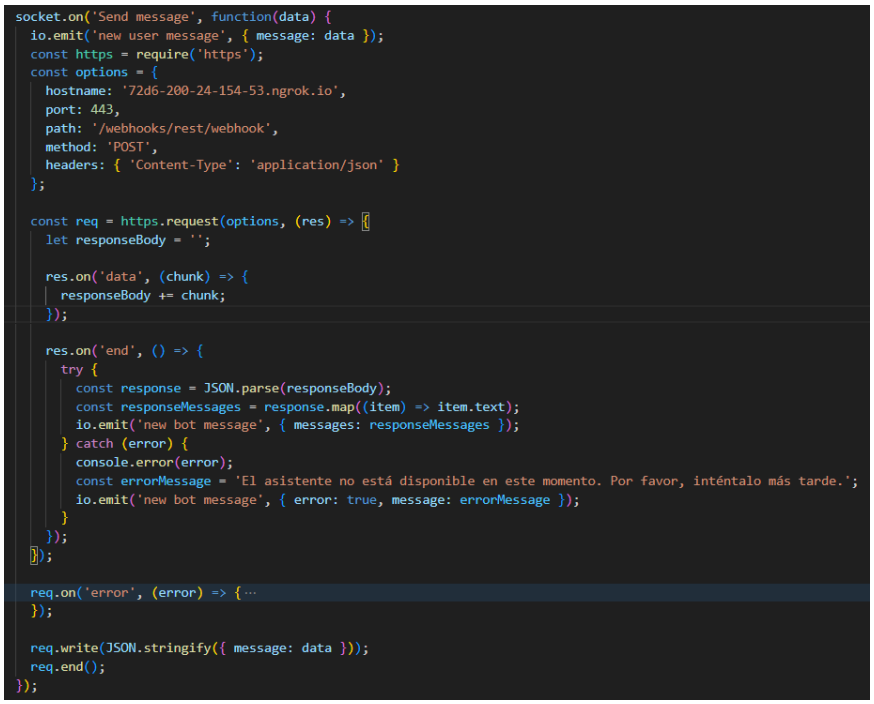
## 6.3 Configuración del Packajson para publicarlo a npm:




- En el código proporcionado muestra un archivo **package.json**. Este se configura de tal manera para que pueda ser publicado en npm y posteriormente instalado en otros proyectos si estos cumplen con los requisitos necesarios. A continuación, se proporciona una descripción general de los elementos presentes en el archivo:
- **"name":** Especifica el nombre del proyecto, en este caso "webcomponents_servico_socket".
- **"version":** Indica la versión del proyecto, en este caso "1.0.0".
- **"main":** Especifica el archivo principal del proyecto, en este caso "index.js".
- **"type":** Define el tipo de módulo utilizado, en este caso "commonjs".
- **"scripts":** Contiene los scripts de comandos para ejecutar tareas específicas. En este caso, solo se define un script llamado "dev" que utiliza "nodemon" para ejecutar el archivo "index.js" dentro de la carpeta "src".

- **"keywords":** Permite agregar palabras clave relacionadas con el proyecto.
- **"author":** Especifica el autor o los autores del proyecto.
- **"license":** Indica la licencia bajo la cual se distribuye el proyecto. En este caso, se utiliza la licencia "ISC".
- **"dependencies":** Enumera las dependencias del proyecto, que son los paquetes externos requeridos para que el proyecto funcione correctamente. Cada dependencia se especifica con su nombre y una versión mínima requerida.
- **"devDependencies":** Enumera las dependencias de desarrollo, que son los paquetes necesarios solo durante el desarrollo del proyecto. Estas dependencias no se incluirán en el paquete final del proyecto.

## 6.4 Configuración para que la interfaz consuma la API del asistente virtual



```
socket.on('Send message', function(data) {
  io.emit('new user message', { message: data });
  const https = require('https');
  const options = {
    hostname: '72d6-200-24-154-53.ngrok.io',
    port: 443,
    path: '/webhooks/rest/webhook',
    method: 'POST',
    headers: { 'Content-Type': 'application/json' }
  };

  const req = https.request(options, (res) => {
    let responseBody = '';

    res.on('data', (chunk) => {
      responseBody += chunk;
    });

    res.on('end', () => {
      try {
        const response = JSON.parse(responseBody);
        const responseMessages = response.map((item) => item.text);
        io.emit('new bot message', { messages: responseMessages });
      } catch (error) {
        console.error(error);
        const errorMessage = 'El asistente no está disponible en este momento. Por favor, inténtalo más tarde.';
        io.emit('new bot message', { error: true, message: errorMessage });
      }
    });
  });

  req.on('error', (error) => {···
  });

  req.write(JSON.stringify({ message: data }));
  req.end();
});
```

- El código muestra la configuración necesaria para que la interfaz de usuario consuma la API del asistente virtual a través de Socket.IO.
- **socket.on('Send message', function(data) { ... }):** Este código define un evento llamado 'Send message' que se activa cuando se envía un mensaje desde la interfaz de usuario al servidor a través del socket.
- **io.emit('new user message', { message: data }):** Se emite un evento llamado 'new user message' a todos los clientes conectados, enviando el mensaje del usuario como objeto

{ message: data }. Esto permite mostrar el mensaje enviado por el usuario en tiempo real en la interfaz.

- Se importa el módulo http para realizar una solicitud HTTP al servidor que contiene la API del asistente virtual.
- Se definen las opciones de la solicitud HTTP, como el hostname, el puerto, la ruta y el método de la solicitud. En este caso, el hostname es '72d6-200-24-154-53.ngrok.io', el puerto es 443 y la ruta es '/webhooks/rest/webhook'. Además, se especifica que la solicitud es de tipo POST y se establece el encabezado 'Content-Type' como 'application/json'.
- Se crea una instancia de la solicitud HTTP utilizando las opciones definidas.
- Se definen los eventos **'data'** y **'end'** en la respuesta de la solicitud HTTP para recopilar los datos recibidos y procesar la respuesta del asistente virtual.
- Dentro del evento **'end'**, se intenta analizar la respuesta JSON del asistente virtual y se guardan los mensajes de respuesta en la variable responseMessages.
- Si ocurre algún error durante el análisis de la respuesta JSON, se captura el error, se muestra un mensaje de error personalizado y se emite un evento 'new bot message' con el mensaje de error a todos los clientes conectados.
- En caso de que ocurra un error durante la solicitud HTTP, se captura el error, se muestra un mensaje de error personalizado y se emite un evento 'new bot message' con el mensaje de error a todos los clientes conectados.
- Se escribe el mensaje del usuario en el cuerpo de la solicitud HTTP utilizando req.write(JSON.stringify({ message: data })).
- Se finaliza la solicitud HTTP con req.end().

# ANEXO 7. MANUAL DE USUARIO.

# DESARROLLO DE UNA INTERFAZ CONVERSACIONAL INTEGRADA A UN ASISTENTE VIRTUAL

## Manual de Usuario

JÉSSICA JOHANA MONTES VERA

CARLOS PIERRE QUIJIJE VERA

Versión 1.0.0

2023/06/11

# TABLA DE CONTENIDO

# DESCRIPCIÓN DEL SISTEMA

## INTRODUCCIÓN

El uso de la interfaz conversacional integrada a un asistente virtual, permite que los tutores y estudiantes obtengan atención rápida, precisa y eficaz a sus necesidades de información o solicitud mejorando así el proceso de atención al usuario con respecto a fomentar la identidad y credibilidad de la institución mediante el uso de la tecnología.

## OBJETIVOS

El presente documento tiene como finalidad dar a conocer la interacción entre el usuario y las aplicaciones correspondientes al área de la UDIV de Infraestructura de la Carrera de Computación de la ESPAM MFL.

## ALCANCE

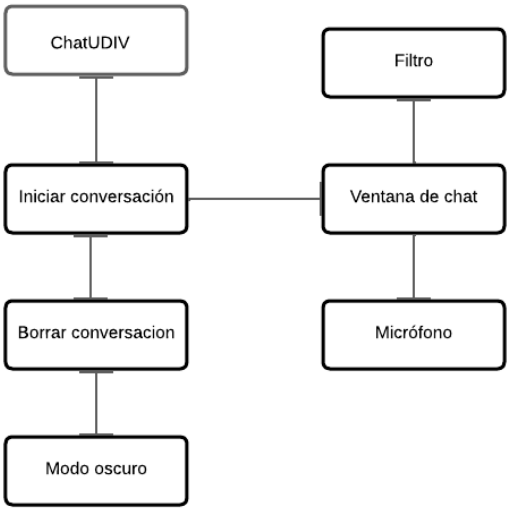
- Perspectiva general del sistema
- Detalle de cada funcionalidad

## ROLES DE USUARIO

- Miembros de la ESPAM MFL: Cualquier usuario que pertenezca a la institución tendrá acceso completo a todas las funcionalidades disponibles.

# MAPA DEL SISTEMA

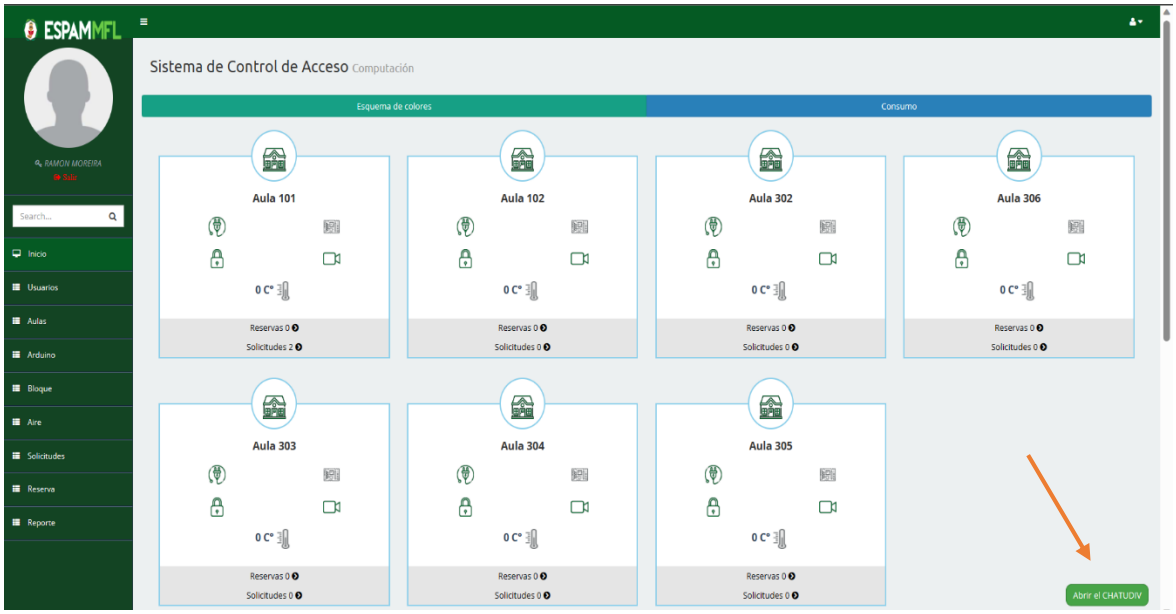
## MODELO LÓGICO



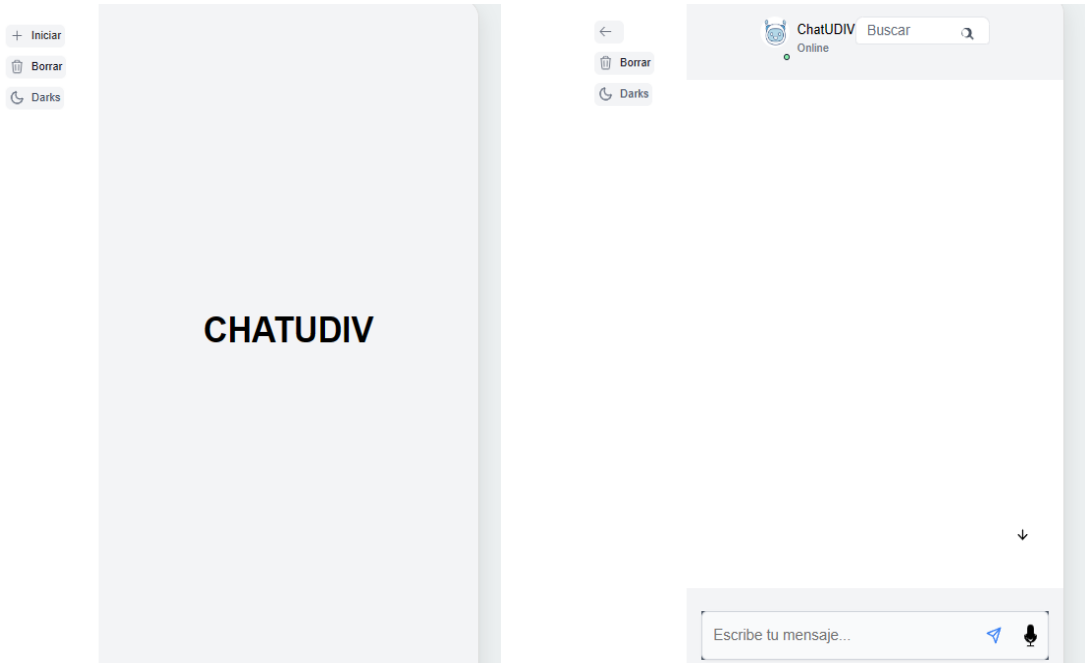
# CHATUDIV

## ACCESO AL CHATUDIV

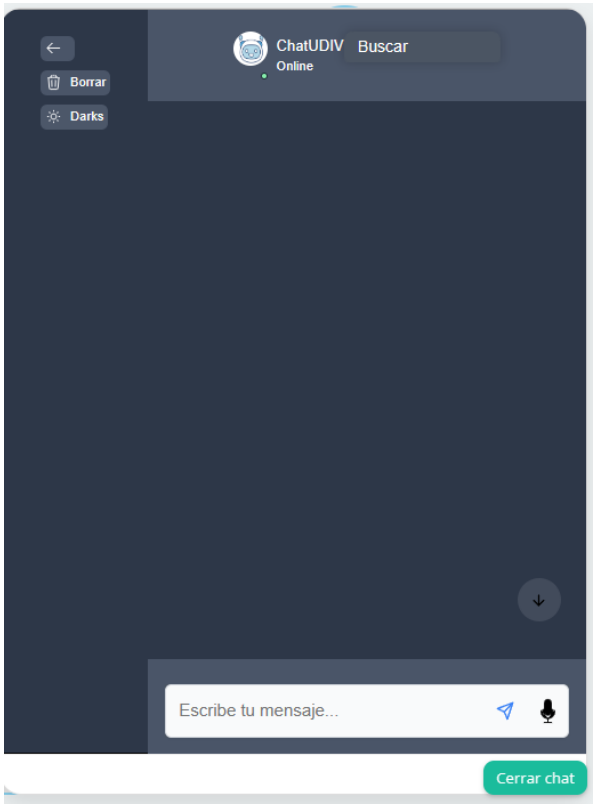
Para ingresare al Chat, primero el usuario deberá ingresar a la aplicación de la UDIV se infraestructura, en la pantalla principal, específicamente en la parte inferior derecha se encontrará con el botón que abrirá el chat con todas las funcionalidades disponibles.



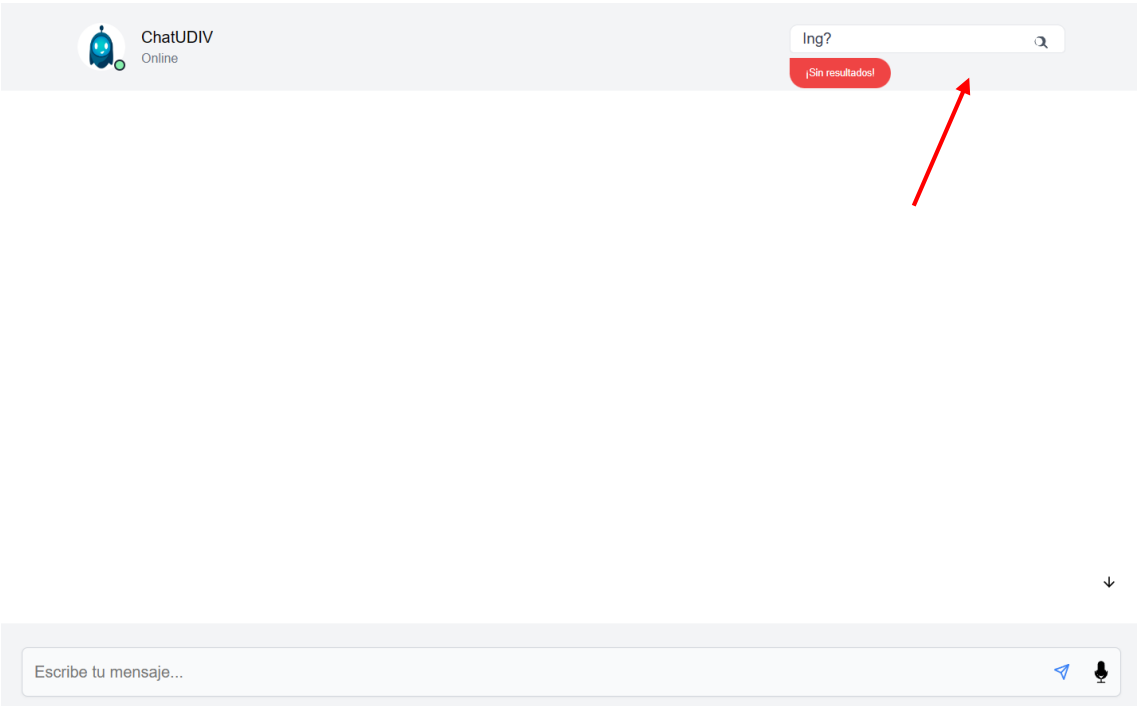

## INICIAR CONVERSACIÓN



## MODO OSCURO


## FILTRO DE BÚSQUEDA



## BORRAR CONVERSACIÓN


## MICRÓFONO