Implementación de una aplicación de chat en tiempo real

José Fernando Revelo Benítez

Asesor

MSc. Edgar Rodrigo Enriquez Rosero

Universidad Nacional Abierta y a Distancia UNAD

Escuela de Ciencias Básicas, Tecnología e Ingeniería ECBTI

Ingeniería de Telecomunicaciones

2025

Dedicatoria

Dedico este proyecto a mi familia, que me ha acompañado a lo largo de todo este proceso. Con su esfuerzo, apoyo incondicional, han sido fundamentales para la realización de esta meta.

José F. Revelo B.

Agradecimientos

Agradezco, primero y, sobre todo, a Dios por permitirme llevar a feliz término el presente proyecto. A mi madre, por brindarme su confianza y apoyo incondicional en todos los momentos del pregrado; al asesor del proyecto, MSc. Edgar Rodrigo Enriquez Rosero, y al Esp. Edgar Dulce Villarreal, por sus acertadas orientaciones; a los tutores, por sus valiosos aportes en mi formación; y a la Universidad Nacional Abierta y a Distancia, por ser la institución que me ha brindado la oportunidad de formarme como Ingeniero de Telecomunicaciones. Finalmente, agradezco a todas aquellas personas que, de una u otra manera, han contribuido a la culminación satisfactoria de esta meta.

Resumen

El proyecto *Real-Time Chat App* tiene como objetivo desarrollar una aplicación de chat en tiempo real que facilite la comunicación instantánea y eficiente entre múltiples usuarios. Utiliza tecnologías modernas como *TypeScript*, *React*, *Node.js* y *Socket.io*. *TypeScript* aporta tipado estático, mejorando la calidad del código; *React* se encarga de la interfaz de usuario, ofreciendo una experiencia interactiva; *Node.js* actúa como el entorno del servidor, gestionando solicitudes y conexiones; y *Socket.io* permite la transmisión instantánea de mensajes entre el cliente y el servidor. La aplicación permite la comunicación simultánea entre varios clientes, garantizando una interacción sin latencia perceptible. Está diseñada para ser robusta, capaz de manejar múltiples conexiones de manera eficiente, y se implementan medidas de seguridad para proteger la información de los usuarios.

El proyecto también abarca soluciones en otros sectores, centradas en mejorar la experiencia del cliente y optimizar tiempos de respuesta. En el área de atención al cliente en tiempo real, se espera lograr una resolución de consultas en menos de 5 minutos, con un nivel de satisfacción superior al 80%, beneficiando a empresas de servicios que buscan mejorar su atención. Para los servicios de emergencia médica, el sistema permitirá una comunicación rápida y efectiva, se espera obtener un tiempo de respuesta menor a 2 minutos y una satisfacción de usuarios por encima del 80%, ayudando a optimizar la respuesta en situaciones críticas. En cuanto a la plataforma de soporte técnico, se proyecta una resolución de problemas en menos de 10 minutos, manteniendo la satisfacción del cliente por encima del 80%, lo que favorecerá a empresas tecnológicas que necesitan mejorar su soporte. Por último, se puede utilizar como una plataforma de asesoramiento financiero en tiempo real, que ofrecerá respuestas en menos de 5 minutos y garantizará una satisfacción del cliente superior al 80%, dirigida a empresas de

asesoría financiera que deseen perfeccionar su servicio al cliente.

De esta manera, Real-Time Chat App no solo optimiza la comunicación entre usuarios en entornos digitales, sino que también se convierte en una herramienta versátil para mejorar la eficiencia en la atención de diversos sectores, asegurando respuestas rápidas y personalizadas en tiempo real.

El informe presenta un marco teórico y conceptual sobre el desarrollo de aplicaciones de chat en tiempo real, abarcando tanto los aspectos técnicos como su aplicación en diversas industrias. Se exploran los protocolos de comunicación utilizados y se describen las características y funcionalidades clave que esta aplicación incorpora. Asimismo, se detalla el uso de lenguajes de programación en el frontend y backend, como TypeScript, React y Node.js, junto con la integración de una base de datos MySQL y la herramienta de almacenamiento UploadThing. Además, se explica la configuración del servidor y las medidas implementadas de seguridad, cifrado y autenticación, que protegen la información y aseguran la integridad del sistema.

Palabras claves: Chat, Tiempo real, Comunicación instantánea, Socket.IO.

Abstract

The Real-Time Chat App project aims to develop a real-time chat application that facilitates instant and efficient communication between multiple users. It uses modern technologies such as TypeScript, React, Node.js and Socket.io. TypeScript provides static typing, improving the quality of the code; React handles the user interface, providing an interactive experience; Node.js acts as the server environment, handling requests and connections; and Socket.io allows instant transmission of messages between the client and the server. The application allows simultaneous communication between multiple clients, ensuring interaction with no perceptible latency. It is designed to be robust, capable of handling multiple connections efficiently, and security measures are implemented to protect user information.

The project also includes solutions in other sectors, focused on improving customer experience and optimizing response times. In the area of real-time customer service, it is expected to achieve a resolution of queries in less than 5 minutes, with a satisfaction level of over 80%, benefiting service companies seeking to improve their customer service. For medical emergency services, the system will enable fast and effective communication, with a response time of less than 2 minutes and a user satisfaction level of over 80%, helping to optimize response in critical situations. As for the technical support platform, it is projected that problems will be resolved in less than 10 minutes, maintaining customer satisfaction above 80%, which will favor technology companies that need to improve their support. Finally, it can be used as a real-time financial advisory platform, which will provide answers in less than 5 minutes and ensure customer satisfaction above 80%, targeting financial advisory firms that want to improve their customer service. In this way, Real-Time Chat App not only optimizes communication between users in digital environments, but also becomes a versatile tool to improve efficiency in

serving various sectors, ensuring fast and personalized responses in real time.

The report presents a theoretical and conceptual framework for the development of real-time chat applications, covering both technical aspects and their application in various industries. It explores the communication protocols used and describes the key features and functionalities that this application incorporate. The use of frontend and backend programming languages, such as TypeScript, React and Node.js, along with the integration of a MySQL database and the UploadThing storage tool, is also detailed. In addition, the server configuration and the implemented security, encryption and authentication measures, which protect the information and ensure the integrity of the system, are explained.

Keywords: Chat, Real-time, Instant communication, Socket.IO.

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Tabla de Apéndices

Introducción

En la actualidad, la comunicación instantánea es esencial tanto en el ámbito personal como profesional. La evolución tecnológica ha propiciado que las aplicaciones de mensajería en tiempo real se conviertan en herramientas clave para facilitar la interacción entre los usuarios, eliminando las barreras de tiempo y espacio (Castells, 2009). Este entorno ha generado la necesidad de desarrollar aplicaciones más eficientes y seguras que puedan soportar múltiples usuarios conectados simultáneamente, garantizando la protección de la información que se intercambia (Bauman, 2013). En este contexto, el proyecto se propuso crear una aplicación de chat en tiempo real, denominada *Real-Time Chat App*, utilizando tecnologías actuales como *TypeScript*, *React*, *Node.js* y *Socket.io* para proporcionar una plataforma robusta y efectiva.

La selección de herramientas como *TypeScript* y *React* responde a la demanda de contar con tecnologías que mejoren la calidad del desarrollo en el *frontend*. *TypeScript* ofrece un sistema de tipado estático que ayuda a reducir errores y a mantener un código más limpio y estructurado (Hejlsberg, 2012). Por su parte, *React* es una de las bibliotecas más populares para la creación de interfaces de usuario interactivas, lo que asegura una experiencia de usuario dinámica y accesible (Abramov, 2017). Estas tecnologías son ampliamente utilizadas en el desarrollo de aplicaciones modernas debido a su flexibilidad y a las herramientas que facilitan la creación de interfaces intuitivas.

Además*, Node.js* se encarga de gestionar el procesamiento del lado del servidor. Este entorno de ejecución permite manejar múltiples solicitudes de manera eficiente gracias a su modelo no bloqueante, lo que es crucial en aplicaciones de chat donde se requiere la conexión simultánea de varios usuarios (Ejsmont, 2015). A su vez, *Socket.io* es una biblioteca fundamental en este tipo de proyectos, ya que habilita la transmisión bidireccional de datos en tiempo real

entre el cliente y el servidor (Geraint, 2014). Gracias a estas tecnologías, la *Real- Time Chat App* garantiza una comunicación fluida sin interrupciones ni latencia perceptible, ofreciendo una plataforma confiable para los usuarios.

El desarrollo de esta aplicación no solo implementó una solución eficiente para la comunicación en tiempo real, sino que también busca establecer un entorno seguro para los usuarios. Puesto que la privacidad y la seguridad de los datos son aspectos críticos, la implementación de medidas de protección, como la encriptación de mensajes y la prevención de accesos no autorizados, se ha vuelto imprescindible (Schneier, 2015). Por esta razón, se otorgó especial atención en aspectos de seguridad y confiabilidad durante la creación de la aplicación, asegurando que la información de los usuarios se encuentre protegida y que la plataforma cumpla con los estándares de seguridad actuales (Stallings, Criptografía y seguridad en redes: Principios y práctica. Pearson., 2017).

Finalmente, el proyecto *Real-Time Chat App* se distingue por su código fuente abierto disponible en *GitHub*, lo que permite a otros desarrolladores revisar, mejorar y adaptar la aplicación a diferentes contextos (Raymond, 2001). Esta característica no solo contribuye al intercambio de conocimiento dentro de la comunidad de desarrolladores, sino que también facilita su implementación en otros proyectos con fines educativos o empresariales. De esta forma, este proyecto no solo busca satisfacer una necesidad tecnológica, sino también fomentar el desarrollo del software libre y la innovación en el ámbito de la comunicación digital (GitHub., 2021).

Planteamiento del Problema

Situación Problemática

En la actualidad, la falta de una comunicación efectiva en entornos de trabajo remoto puede dar lugar a malentendidos, disminución de la productividad y dificultades en la coordinación entre equipos (Nardi, 2002). En contextos donde es esencial mantener una comunicación abierta y constante entre el cliente y el servidor para la transmisión de datos en tiempo real, las tecnologías tradicionales que se basan en interacciones intencionadas resultan insuficientes (Schmidt, 2018). Esta limitación se vuelve especialmente evidente en aplicaciones que requieren actualizaciones instantáneas, como en los sistemas de chat, donde la inmediatez en la comunicación es fundamental para garantizar una experiencia satisfactoria para el usuario (Rosenberg J. , 2019).

Para abordar esta problemática, han emergido tecnologías *push* que permiten una comunicación más dinámica y continua. Dentro de estas, *WebSocket* se ha consolidado como un estándar ampliamente aceptado que habilita la comunicación bidireccional y persistente entre el cliente y el servidor (Fette, 2011). A diferencia de las solicitudes HTTP tradicionales, que son unidireccionales y requieren de un nuevo ciclo de conexión para cada intercambio de datos, *WebSocket* establece un canal de comunicación constante, permitiendo que ambos extremos envíen y reciban mensajes en cualquier momento (W3C, 2021). Esta característica es crucial en aplicaciones que requieren una interacción fluida, como las salas de chat en tiempo real.

En este sentido, en el ecosistema de *Node.js, Socket.IO* ha emergido como la herramienta preferida para implementar *WebSockets*, ofreciendo una solución robusta para crear aplicaciones que requieren comunicación instantánea (Rasband, 2020). Con *Socket.IO*, es posible mantener un chat abierto y accesible para todos los usuarios conectados de forma continua y eficiente, lo que

optimiza la interacción y mejora la experiencia general del usuario (Mason, Socket.IO para desarrolladores de Node.js: Construcción de aplicaciones en tiempo real. , 2018). Sin embargo, a pesar de la efectividad de estas tecnologías, aún persiste la

necesidad de investigar cómo implementarlas de manera óptima en entornos específicos y adaptarlas a diferentes escenarios de uso.

Formulación del problema

¿Cómo puede la implementación de tecnologías de comunicación en tiempo real, como WebSocket y Socket.IO, mejorar la efectividad de la comunicación en aplicaciones de chat, en diferentes entornos como atención al cliente, soporte técnico y emergencias médicas?

Justificación

La creciente necesidad de herramientas de comunicación eficientes, confiables y accesibles en un entorno donde la inmediatez es clave para la interacción entre usuarios ha impulsado el desarrollo de sistemas de chat (Duarte, 2020). De igual manera y considerando un mundo cada vez más globalizado y digital, la capacidad de intercambiar información de manera instantánea se ha convertido en un pilar fundamental tanto en entornos profesionales como personales.

Por otra parte, las soluciones tradicionales de mensajería que dependen de solicitudes HTTP unidireccionales no son suficientes para garantizar la fluidez de la comunicación requerida en el contexto actual (Schmidt, 2018). Por esta razón, tecnologías como *WebSocket* han surgido como alternativas viables para la implementación de plataformas que requieren comunicación bidireccional y en tiempo real, mejorando significativamente la experiencia del usuario (W3C, 2021). *WebSocket* es una tecnología de comunicación que opera en la capa de aplicación del modelo OSI, pero utiliza TCP como protocolo de transporte subyacente (Fette, 2011). Lo que le permite una comunicación constante y simultánea entre el cliente y el servidor, siendo ideal para aplicaciones que requieren actualizaciones rápidas y frecuentes, como las plataformas de chat. A diferencia de otros métodos que dependen de múltiples solicitudes HTTP para cada mensaje, WebSocket establece una conexión persistente que permite el intercambio de información en ambas direcciones de manera continua y eficiente (Mason, Socket.IO para desarrolladores de Node.js: Construcción de aplicaciones en tiempo real. , 2018). Esto es crucial en el ámbito de las telecomunicaciones, donde la calidad del servicio y la experiencia del usuario dependen de la rapidez y fiabilidad en la transmisión de datos (Schmidt, 2018).

Un claro ejemplo de la implementación exitosa de *WebSocket* en una plataforma de

comunicación es *Slack*, una herramienta utilizada a nivel global para la coordinación en equipos de trabajo (Jackson, 2020). *Slack* emplea *WebSocket* para ofrecer un chat en tiempo real y notificaciones instantáneas, lo que permite a los usuarios mantenerse conectados e informados de forma constante. Este tipo de soluciones no solo facilita la colaboración, sino que también optimiza la eficiencia operativa, reduciendo los tiempos de espera y mejorando la calidad de las interacciones (Rosenberg J. , 2019).

El desarrollo de una aplicación de chat en tiempo real personalizada permite a los usuarios interactuar y colaborar de manera eficaz, resaltando la importancia de los protocolos de comunicación en la mejora de la calidad de la experiencia del usuario (Schmidt, 2018). En el contexto de las telecomunicaciones, esta implementación también proporciona una visión más clara de cómo los protocolos como *WebSocket y TCP* pueden influir directamente en el rendimiento y la fiabilidad de las aplicaciones modernas (Fette, 2011) Además, este proyecto no solo contribuirá a cubrir una necesidad actual en la comunicación remota, sino que también servirá como base para futuras investigaciones en diversos entornos. En atención al cliente, estas tecnologías permiten interacciones instantáneas y fluidas, mejorando la experiencia del usuario y reduciendo los tiempos de espera. En soporte técnico, la comunicación en tiempo real facilita la resolución de problemas de manera más eficiente, ya que los técnicos pueden recibir y responder consultas inmediatamente. En el contexto de emergencias médicas, la capacidad de transmitir información crítica al instante puede ser vital para la toma de decisiones rápidas y efectivas. Así, el uso de estas tecnologías no solo optimiza la comunicación, sino que también tiene el potencial de impactar positivamente en la efectividad de los servicios ofrecidos en múltiples sectores.

Objetivos

Objetivo General

Desarrollar e implementar una aplicación de chat que permita a múltiples usuarios comunicarse simultáneamente de manera segura a través de una red, ofreciendo una experiencia de chat en tiempo real.

Objetivos Específicos

Crear un servidor central que gestione de manera eficiente la comunicación entre múltiples clientes conectados.

Implementar un sistema de autenticación para garantizar la seguridad de los usuarios.

Establecer una comunicación confiable y eficiente entre los clientes y el servidor.

Desarrollar una interfaz amigable y de fácil uso para los usuarios.

Realizar pruebas para asegurar el correcto funcionamiento de la aplicación.

Revisión de la Literatura

Marco Contextual

El desarrollo de la aplicación Real-Time Chat App se ha llevado a cabo en un equipo de cómputo con características específicas que aseguran un rendimiento óptimo y la capacidad de manejar múltiples conexiones simultáneas. La robustez del hardware es crucial en entornos donde se requiere una comunicación en tiempo real, ya que influye directamente en la capacidad del servidor para gestionar cargas de trabajo, garantizar tiempos de respuesta rápidos y mantener la estabilidad del sistema durante picos de uso (Silberschatz, 2018).

Infraestructura de Hardware

Procesador. Intel(R) Core i5-10400 CPU @ 2.90 GHz. Este procesador es capaz de manejar múltiples conexiones en tiempo real, lo cual es esencial para aplicaciones de chat donde se espera que varios usuarios interactúen simultáneamente sin latencia significativa. Su arquitectura permite un equilibrio eficiente entre rendimiento y consumo energético (Hennessy, 2017).

Memoria RAM (Memoria de acceso aleatorio). 24 GB. La capacidad de RAM es fundamental para gestionar varias instancias de prueba y ejecución simultánea. Una mayor cantidad de memoria permite el almacenamiento temporal de datos y mejora el rendimiento general del sistema, evitando cuellos de botella durante las operaciones de la aplicación (Silberschatz, 2018).

Disco SSD (Disco de estado sólido). M.2 de 480 GB. El uso de un disco SSD garantiza tiempos de respuesta rápidos, lo que es crítico para la carga eficiente de la aplicación y el acceso veloz a los datos necesarios para el funcionamiento en tiempo real (Stallings, Computer Organization and Architecture: Designing for Performance., 2019).

Disco HDD (Disco rígido). 1 TB. Este disco se utiliza para almacenamiento adicional y respaldo de datos, asegurando que la información se conserve de manera segura y accesible, lo que es fundamental para mantener la integridad de las conversaciones y otros datos relevantes (Hennessy, 2017).

La combinación de estos componentes facilitó que la Real-Time Chat App funcione de manera efectiva en un entorno de producción, asegurando que los usuarios disfruten de una experiencia fluida y confiable.

***Herramientas* de Software**

El desarrollo de la Real-Time Chat App necesitó un conjunto de herramientas de software que son esenciales para garantizar un proceso de creación eficiente y eficaz.

IDE (Entorno de Desarrollo Integrado). Un IDE (Integrated Development Environment) o Entorno de Desarrollo Integrado es una aplicación de software que proporciona a los programadores y desarrolladores un conjunto de herramientas y características para facilitar el proceso de desarrollo de software. Un IDE típicamente incluye un editor de código, herramientas de depuración, un compilador o intérprete, y un sistema de control de versiones, todo integrado en una sola interfaz de usuario. Las principales características de un IDE incluyen:

Editor de Código. Proporciona resaltado de sintaxis, autocompletado y herramientas de refactorización para ayudar a los desarrolladores a escribir código de manera más eficiente.

Depurador. Permite a los desarrolladores identificar y corregir errores en su código, proporcionando capacidades para ejecutar el código paso a paso y observar el estado de las variables en tiempo real.

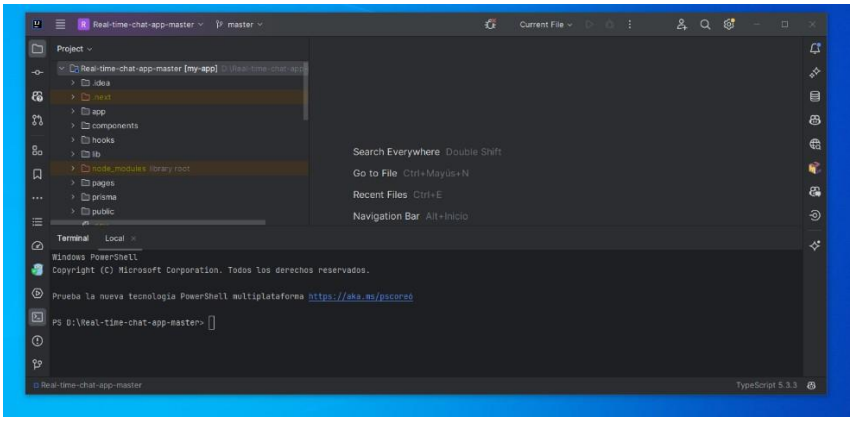
Compilador/Intérprete. Convierte el código fuente en un formato ejecutable, permitiendo que el software se ejecute en el entorno adecuado.

Integración con Sistemas de Control de Versiones: Facilita la colaboración entre desarrolladores al permitir el seguimiento de cambios en el código y la gestión de versiones del mismo (Wikipedia, 2024).

Los IDEs están diseñados para simplificar el proceso de desarrollo, reducir el tiempo necesario para completar tareas y mejorar la calidad del código final. Herramientas como IntelliJ IDEA son ejemplos de IDEs que ofrecen un amplio rango de características y soporte para múltiples lenguajes de programación, lo que los convierte en una opción popular entre los desarrolladores de software (Wikipedia, 2024).

IntelliJ IDEA. Es un IDE desarrollado por JetBrains, conocido por su robustez y características avanzadas que mejoran la productividad del desarrollador. Este entorno de desarrollo proporciona potentes herramientas de edición, refactorización de código y depuración, así como integración con sistemas de control de versiones, lo que facilita el trabajo colaborativo y la gestión del código fuente. Durante el desarrollo de mi proyecto, Como se muestra en la Figura 1, IntelliJ IDEA se utilizó para escribir, depurar y gestionar el código de manera eficiente. La herramienta permite a los desarrolladores centrarse en la lógica del negocio al automatizar tareas repetitivas y proporcionar sugerencias contextuales, lo que resulta en una programación más rápida y efectiva (JetBrains, 2025).

Figura 1
IntelliJ IDEA



Nota. Captura de pantalla propia de IntelliJ IDEA.
Fuente. Autoría propia.

Licencia Educativa. Una licencia educativa es un tipo de licencia que permite a estudiantes y educadores acceder a software profesional de forma gratuita o a un costo reducido. Este tipo de licencia es especialmente importante en el ámbito de la educación, ya que brinda a los estudiantes la oportunidad de aprender y practicar con herramientas que son ampliamente utilizadas en la industria, lo que mejora su preparación profesional y su empleabilidad. Las licencias educativas están diseñadas para facilitar el acceso a software que, de otro modo, podría ser costoso para los estudiantes. Por ejemplo, la licencia educativa de IntelliJ IDEA permite a los estudiantes y profesores utilizar todas las características del IDE sin costo alguno, lo que es fundamental para proyectos académicos y el aprendizaje de habilidades prácticas en desarrollo de software. Al proporcionar acceso a herramientas avanzadas, se fomenta la innovación y se permite a los estudiantes experimentar con tecnologías actuales y relevantes para su campo de estudio (JetBrains, 2025) .

Windows 10. Es un sistema operativo desarrollado por Microsoft, lanzado en julio de 2015 como parte de la familia de sistemas operativos Windows NT. Es conocido por su interfaz de usuario amigable y su capacidad para soportar una amplia variedad de

aplicaciones, desde software de productividad hasta juegos. Windows 10 ofrece características como el asistente virtual Cortana, el navegador *Microsoft Edge*, y la compatibilidad con aplicaciones universales de Windows (UWP), lo que lo convierte en una plataforma versátil para desarrolladores y usuarios. El sistema operativo está diseñado para funcionar en diferentes dispositivos, incluidos PCs, tabletas y consolas, lo que permite a los desarrolladores crear aplicaciones que pueden ejecutarse en múltiples dispositivos con una sola base de código. Esta flexibilidad es una de las razones por las cuales Windows 10 es una opción popular entre los desarrolladores de software (Microsoft, 2025).

Uso De Windows 10 Como Servidor. Se utilizó Windows 10 como servidor para desarrollar y ejecutar la Real-Time Chat App por varias razones clave:

Compatibilidad Con Herramientas De Desarrollo. Windows 10 es compatible con una amplia variedad de herramientas y entornos de desarrollo, incluyendo Node.js, npm y frameworks de frontend como React y Next.js. Esto facilita la instalación y configuración de todos los componentes necesarios para el desarrollo de la aplicación (Microsoft, 2025).

Facilidad De Uso Y Familiaridad. Al ser un sistema operativo ampliamente utilizado, Windows 10 ofrece una interfaz de usuario intuitiva que es familiar para muchos desarrolladores. Esto permite un acceso más rápido y eficiente a las herramientas y recursos necesarios para el desarrollo (ZDNet., 2021).

Ambiente Estable Y Seguro. Windows 10 proporciona un entorno estable que es esencial para ejecutar aplicaciones en tiempo real. Sus características de seguridad y actualizaciones regulares ayudan a proteger el sistema contra amenazas, lo que es

fundamental al operar un servidor (Microsoft, 2025) .

Soporte Para Aplicaciones Web. Windows 10 permite la ejecución de aplicaciones web modernas y proporciona el soporte necesario para tecnologías como WebSocket y Socket.IO, que son esenciales para la comunicación en tiempo real en mi aplicación de chat (Microsoft, 2025).

Acceso A Recursos Y Comunidad. Dado que Windows 10 es uno de los sistemas operativos más populares, hay una gran cantidad de recursos, tutoriales y foros de soporte disponibles, lo que facilita la solución de problemas y la obtención de ayuda cuando sea necesario (ZDNet., 2021).

Sistema De Base De Datos En La Nube. se refiere a la capacidad de almacenar y gestionar datos en servidores remotos accesibles a través de Internet, en lugar de depender de hardware local. Esto permite a los desarrolladores aprovechar la infraestructura de nube, garantizando una mayor disponibilidad, seguridad y flexibilidad en la gestión de datos. Los sistemas de bases de datos en la nube eliminan la necesidad de administración de servidores físicos y permiten a los equipos enfocarse en el desarrollo de la aplicación, en lugar de en la infraestructura subyacente (AWS, 2024).

PlanetScale. (Plan Hobby gratuito): Es un sistema de base de datos en la nube que ofrece una plataforma escalable y de alto rendimiento para el almacenamiento y gestión de datos. Diseñada para aplicaciones modernas, *PlanetScale* se basa en la tecnología de *Vitess*, que permite manejar grandes volúmenes de datos y tráfico de manera eficiente. Este sistema de base de datos proporciona características avanzadas como la replicación automática, la escalabilidad horizontal y una arquitectura sin servidor, lo que facilita el desarrollo y la operación de aplicaciones en entornos

dinámicos (PlanetScale, 2025).

Vitess. Es un sistema de gestión de bases de datos que se centra en escalar aplicaciones SQL en la nube y en entornos de alta demanda. Originalmente desarrollado por YouTube, Vitess permite gestionar grandes volúmenes de datos mediante técnicas como la fragmentación (*sharding*) y ofrece compatibilidad con MySQL, lo que facilita su adopción en aplicaciones existentes (Kumar, 2021).

MySQL. MySQL es un sistema de gestión de base de datos que actualmente cuenta con más de seis millones de clientes en todo el mundo. Un software libre que se engloba en el grupo de licencias GNU GPL.

El uso del Plan Hobby gratuito de PlanetScale fue ideal para el desarrollo, ya que proporciona suficiente capacidad para gestionar bases de datos pequeñas sin costo alguno durante un tiempo de prueba establecido. Esto es especialmente beneficioso para proyectos en etapas iniciales, como la Real-Time Chat App, donde los costos deben mantenerse bajos mientras se validan conceptos y se construye la funcionalidad básica. Al utilizar *PlanetScale*, puedo aprovechó la infraestructura de nube escalable sin comprometer los recursos económicos, lo que facilitó el desarrollo ágil y la iteración rápida en mi aplicación.

Herramientas De Gestión De Archivos Y Seguridad

Una herramienta de gestión de archivos es, básicamente, un software diseñado para almacenar información y permitir a los usuarios un acceso compartido a la misma a través de aplicaciones desarrolladas para el uso de datos y el control de estos (Bitrix24, 2024).

Uploadthing

Es una herramienta de gestión de archivos que permite a los desarrolladores cargar, almacenar y gestionar archivos en tiempo real. Su plan gratuito ofrece un límite de 2 GB, lo que

es suficiente para manejar la carga de archivos en aplicaciones en fase de prueba o desarrollo. Uploadthing proporciona una interfaz sencilla para la gestión de archivos, lo que facilita su integración en aplicaciones web y móviles. Esta herramienta es especialmente útil para aplicaciones que requieren la subida de documentos, imágenes o archivos multimedia, permitiendo a los usuarios interactuar con contenido dinámico de manera efectiva (Uploadthing, 2024).

Clerk

Se centra en la gestión de la autenticación de usuarios. Esta herramienta garantiza que los usuarios se autentiquen de manera segura, protegiendo así los datos personales y las credenciales. Clerk proporciona un flujo de trabajo de autenticación completo, incluyendo inicio de sesión, registro y recuperación de contraseñas, además de ofrecer opciones de autenticación de múltiples factores. Al utilizar Clerk, se puede asegurar que solo los usuarios autorizados tengan acceso a funciones críticas de la aplicación, lo que contribuye a la seguridad general del sistema (Clerk, 2024).

Entorno De Servidor Y Gestión Local

Un entorno local se refiere a un ambiente de desarrollo en una computadora local donde se escribe, prueba y depura el código antes de que se implemente en un entorno de producción en línea. Es decir, es una instalación de un servidor web y otras herramientas de desarrollo de software en una computadora local para permitir el desarrollo (Real, 2024).

Laragon

Es una herramienta que se utilizó para la configuración y administración del entorno local de desarrollo. Es una solución ligera y potente que simplifica la instalación y gestión de servidores locales para PHP y MySQL. Aunque mi aplicación se basa principalmente en Node.js,

se utilizó *Laragon* como opción para facilitar la gestión de proyectos y entornos de desarrollo. *Laragon* permite configurar rápidamente un entorno de trabajo con características avanzadas, como soporte para múltiples versiones de PHP y la posibilidad de crear proyectos nuevos con un solo clic. Gracias a su interfaz amigable y su capacidad para manejar diversas configuraciones, *Laragon* es ideal para el desarrollo de aplicaciones web (Laragon., 2024).

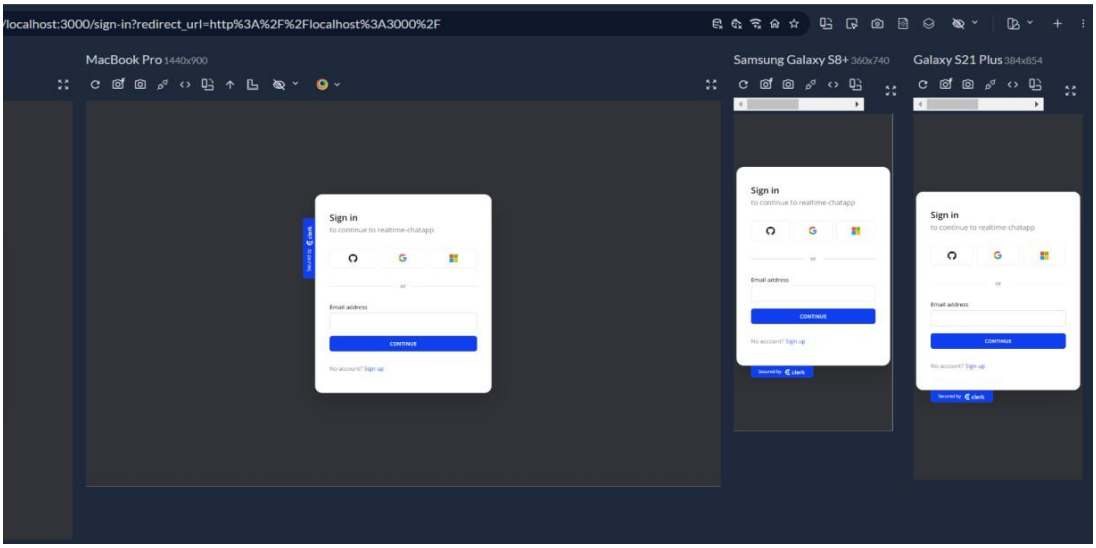
Phpmyadmin

Es una herramienta de gestión de bases de datos que permite interactuar con bases de datos MySQL a través de una interfaz gráfica web. Aunque mi aplicación utiliza una base de datos gestionada a través de PlanetScale, se utilizó PHPMyAdmin para gestionar bases de datos locales y realizar tareas comunes de administración, como la creación, modificación y eliminación de tablas, así como la ejecución de consultas SQL. PHPMyAdmin facilita la gestión de bases de datos locales al proporcionar una forma accesible y visual de manipular datos, lo que es especialmente útil durante el desarrollo y las pruebas de aplicaciones (Phpmyadmin, 2025).

Pruebas De Responsividad

Para asegurar que la interfaz de usuario de la Real-Time Chat App se adapte adecuadamente a diferentes dispositivos y tamaños de pantalla, se realizaron pruebas de responsividad.

Figura 2
Responsively App



Nota. Captura de pantalla propia de Responsively app.
Fuente. Autoría propia.

Es una herramienta eficaz que se ha utilizado para probar la responsividad del diseño de la aplicación. Esta herramienta permite a los desarrolladores visualizar cómo se comporta la interfaz de usuario en múltiples dispositivos simultáneamente. Con Responsively App, se pueden ajustar y verificar elementos de diseño en tiempo real, facilitando la identificación de problemas de visualización y la adaptación de estilos según las necesidades específicas de cada dispositivo. Además, Como se muestra en la Figura 2, Responsively App proporciona la posibilidad de simular diferentes resoluciones de pantalla y navegadores, lo que resulta esencial para garantizar una experiencia de usuario coherente y atractiva. Al emplear esta herramienta, se puede asegurar que todos los usuarios, independientemente del dispositivo que utilicen, tengan acceso a una interfaz funcional y amigable (responsively.app, 2025).

Tipo De Red

La Real-Time Chat App se diseñó para operar principalmente en redes de área amplia (WAN), como Internet, como se muestra en la Figura 3, esto permite que los usuarios se comuniquen en tiempo real desde diferentes ubicaciones geográficas. Para garantizar una

comunicación eficiente y rápida, la aplicación utilizó WebSocket sobre el protocolo TCP/IP, estableciendo una conexión bidireccional y persistente entre el servidor y los clientes, lo que asegura una baja latencia y un intercambio rápido de datos.

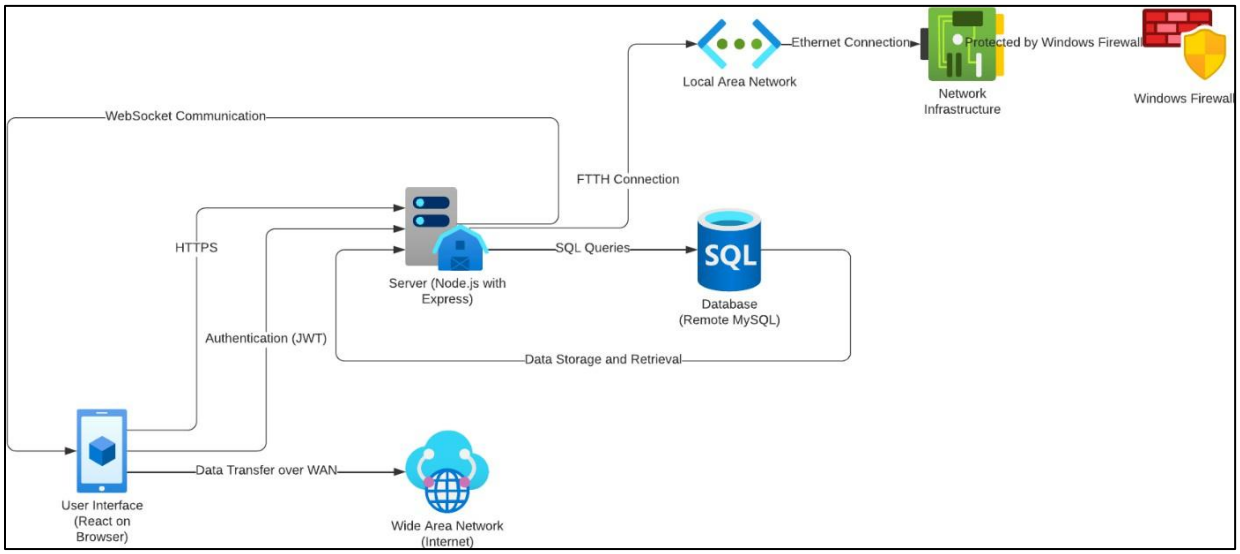
Tcp/Ip (Transmission Control Protocol/Internet Protocol)

Es el conjunto de protocolos fundamentales que permiten la comunicación en redes de área amplia (WAN), como Internet. Este protocolo asegura que los datos se transmitan de manera confiable entre dispositivos en una red, dividiendo la información en pequeños paquetes que luego son enviados, recibidos y ensamblados nuevamente en el destino correcto. TCP garantiza la fiabilidad y la corrección en la entrega de los datos, mientras que IP se encarga de direccionar y enrutar estos paquetes a través de la red, asegurando que lleguen a su destino incluso si pasan por diferentes rutas en la red (Kurose, 2017).

Websocket

Es un protocolo de comunicación en tiempo real que funciona sobre TCP. A diferencia del modelo tradicional de comunicación HTTP, donde el cliente debe hacer solicitudes repetidas al servidor (lo que se conoce como *polling*), WebSocket permite una conexión bidireccional entre el cliente y el servidor. Esto significa que, una vez establecida la conexión WebSocket, tanto el cliente como el servidor pueden enviar mensajes de manera simultánea sin necesidad de realizar nuevas solicitudes, lo que hace que sea ideal para aplicaciones que requieren una comunicación continua y en tiempo real, como las aplicaciones de chat (Lubbers, 2010).

Figura 3
Diagrama de red



Nota. Elaboración propia haciendo uso de lucid chart.
Fuente. Autoría propia.

Presupuesto Utilizado

La asignación de estos recursos fue gestionada de manera eficiente para garantizar que el proyecto cumpliera con los objetivos establecidos, optimizando la relación costo-beneficio.

Como ilustra la tabla 1, la asignación de estos recursos fue gestionada de manera eficiente para garantizar que el proyecto cumpliera con los objetivos establecidos, optimizando la relación costo-beneficio.

Tabla 1
Recursos necesarios

RECURSO	DESCRIPCIÓN	PRESUPUESTO
1. Equipo Humano	Estudiante	Contribución personal
2. Equipos y Software	Hardware: Equipo de cómputo PC: • Procesador: Intel(R) Core (TM) i5-10400 CPU @ 2.90GHz 2.90 GHz • Memoria RAM: 24GB • Disco SSD: M.2 480GB • Disco HHD: 1TB Software: • IntelliJ IDEA (License For educational use) • Windows 10 • PlantetScale free Hobby plan (10 GB storage) • Uploadthing (plan limited 2GB) • Clerk • Laragon • PHPmyAdmin • Responsively App	2’500.000 COP
3. Viajes y Salidas de Campo	No aplica	No aplica
4. Materiales y suministros	No aplica	No aplica

5. Bibliografía	Base de datos de la universidad, artículos académicos de Google Scholar, informes de investigación, libros electrónicos de la biblioteca digital de la universidad, fuentes de internet, sitios web, artículos de blogs especializados.	No aplica
TOTAL: 2’500.000		

Nota. Tabla que describe los recursos necesarios para realizar el proyecto.
Fuente. Autoría propia.

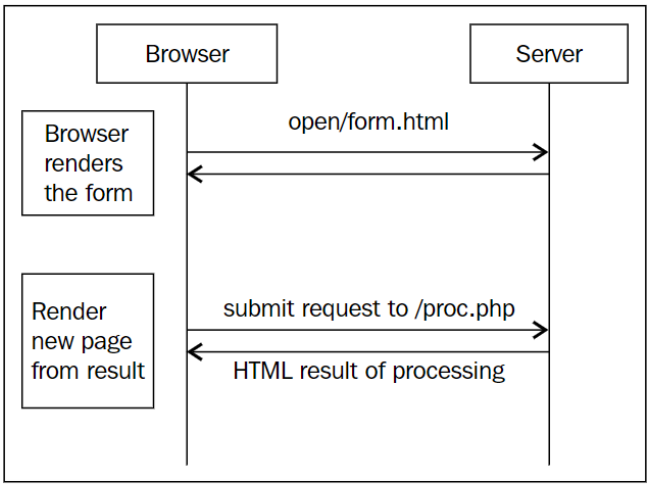
Marco Teórico

Internet y las aplicaciones web, tal y como las conocen los usuarios, se basan en el protocolo HTTP, que es un sistema de petición-respuesta en el que el «cliente» solicita información al «servidor», que responde proporcionando la información solicitada al «cliente». En la mayoría de los casos, la información solicitada pueden ser páginas HTML, o información en forma de XML o JSON (Universidad del Pireo , 2013).

La Figura 4 ilustra esta interacción entre el buscador (cliente) y el servidor, mostrando las etapas clave del proceso: desde la emisión de una solicitud por parte del cliente, el tratamiento de esta en el servidor, hasta la devolución de la respuesta correspondiente.

Figura 4

Interacción buscador - servidor



Fuente. Tomado de Universidad del Pireo – Departamento de Informática (Universidad del Pireo , 2013).

Html

HTML (HyperText Markup Language) es el lenguaje estándar utilizado para crear páginas web. Define la estructura y el contenido de un documento web mediante el uso de

elementos y etiquetas. Cada elemento HTML tiene un propósito específico, como definir encabezados, párrafos, enlaces, imágenes, entre otros. Además, HTML se combina con tecnologías como CSS para dar estilo a las páginas y con JavaScript para añadir interactividad, lo que lo convierte en un lenguaje fundamental para el desarrollo web y constituye la base de cualquier sitio web (Flanagan, HTML5: The Definitive Guide. , 2011).

Http

En relación con esto, el Protocolo de Transferencia de Hipertexto (HTTP, por sus siglas en inglés) es el protocolo de comunicación que permite la transferencia de datos en la web. Este protocolo sigue un sistema de solicitud-respuesta entre un cliente y un servidor, donde el cliente solicita un recurso y el servidor lo entrega (Fielding, 1999).

Get

Este método se utiliza para recuperar datos de un servidor, sin alterar su estado. Las solicitudes GET son idempotentes, lo que significa que realizar la misma solicitud varias veces no cambia el estado del servidor. Se emplea comúnmente para solicitar recursos como páginas HTML o archivos multimedia (Flanagan, HTML5: The Definitive Guide. , 2011).

La Figura 5 representa gráficamente el funcionamiento del método GET.

Figura 5
Método GET

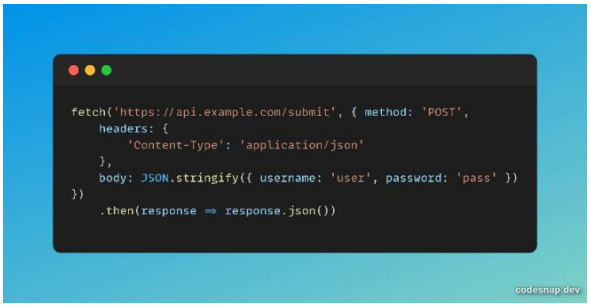
Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Post

Utilizado para enviar datos al servidor y provocar un cambio en su estado, como la creación de nuevos recursos. En comparación con GET, POST incluye un cuerpo en la solicitud que contiene los datos que el cliente quiere enviar (Flanagan, HTML5: The Definitive Guide. , 2011).

La Figura 6 representa gráficamente el funcionamiento del método POST.

Figura 6
Método POST



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Put

Se emplea para actualizar un recurso existente en el servidor. Al igual que POST, incluye un cuerpo en la solicitud, pero se espera que reemplace completamente el recurso en lugar de solo modificarlo parcialmente (Flanagan, HTML5: The Definitive Guide. , 2011).

La Figura 7 representa gráficamente el funcionamiento del método PUT.

Figura 7
Método PUT

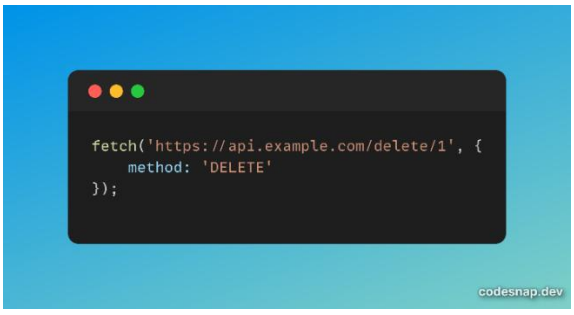
Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Delete

Este método elimina un recurso existente en el servidor. Las solicitudes DELETE son también idempotentes, lo que significa que, si se envía la misma solicitud varias veces, el estado del servidor no cambiará después de la primera ejecución (Flanagan, HTML5: The Definitive Guide. , 2011).

La Figura 8 representa gráficamente el funcionamiento del método DELETE.

Figura 8
Método DELETE



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Una solicitud HTTP está compuesta por varios elementos clave:

Método

Especifica el tipo de operación (GET, POST, PUT, DELETE).

Url

Indica la dirección del recurso en el servidor.

Cabeceras (Headers)

Transmiten información adicional, como el tipo de contenido o datos de autenticación.

Cuerpo (Body)

Incluye los datos que el cliente quiere enviar al servidor (presentes en POST y PUT) (Flanagan, HTML5: The Definitive Guide. , 2011).

Respuesta HTTP

Cuando el servidor recibe una solicitud, responde con un código de estado y, en muchos casos, con datos. Algunos de los códigos de estado más comunes incluyen:

200Ok. La solicitud fue exitosa.

404 Not Found. El recurso solicitado no pudo ser encontrado.

500 Internal Server Error. Hubo un problema en el servidor al procesar la solicitud.

Aunque los WebSockets se usan para la comunicación en tiempo real en aplicaciones como un Real-Time Chat App, HTTP sigue siendo fundamental. HTTP se utiliza para manejar solicitudes iniciales, como autenticación de usuarios o la carga de recursos estáticos. Una vez que se ha autenticado al usuario, WebSockets toman el relevo para manejar la comunicación en tiempo real, sin necesidad de seguir enviando solicitudes HTTP continuas (Flanagan, HTML5: The Definitive Guide. , 2011).

Server Sent Events (SSE) Y Websocket

A medida que los navegadores evolucionaron rápidamente, impulsados inicialmente por Firefox y más tarde por Chrome, la tan esperada actualización de HTML, HTML5, fue finalmente adoptada de manera generalizada. Dentro de HTML5, surgieron dos nuevos métodos para enviar datos desde el servidor al cliente: *Server Sent Events (SSE*) y el protocolo *WebSocket*, ambos importantes para la comunicación en tiempo real en la web (Universidad del Pireo , 2013).

De estos dos métodos, el SSE es el más sencillo y utiliza una comunicación simplex, en la que los datos se envían exclusivamente del servidor al cliente. Para implementar este enfoque, se utiliza una interfaz de programación (API) en *JavaScript* que permite crear una fuente de eventos continua, es decir, un flujo de datos que el servidor transmite al cliente. En este contexto, la petición XHR sigue siendo utilizada para establecer la conexión inicial (Universidad del Pireo , 2013).

Por otro lado, el protocolo *WebSocket* permite una comunicación bidireccional (full dúplex), lo que significa que tanto el cliente como el servidor pueden enviar y recibir datos de forma simultánea. En este modelo, el navegador establece una conexión socket con el servidor, que también soporta WebSocket, y, a partir de esa conexión, se intercambian mensajes entre ambas partes de manera continua (Universidad del Pireo , 2013).

Comunicación En Tiempo Real

WebSockets es un protocolo estandarizado en 2012 que permite la comunicación full dúplex mediante el Protocolo de Control de Transmisión (TCP) (Declan Williamson, 2023). En este contexto, los protocolos que sustentan esta comunicación han tenido que evolucionar para responder al aumento del uso y la demanda. Muchos de los protocolos ampliamente utilizados hoy en día, como TCP y HTTP, se desarrollaron hace décadas, antes de la adopción generalizada

de Internet (Declan Williamson, 2023) .

Además, el protocolo de comunicación predominante para el tráfico de Internet sigue siendo el TCP, que proporciona una comunicación ordenada y fiable. No obstante, un segundo protocolo, el Protocolo de Datagramas de Usuario (UDP), ofrece una comunicación desordenada y poco fiable. UDP se emplea en aplicaciones donde la pérdida de datos no es un problema y la velocidad es crítica, como en las búsquedas DNS (Domain Name System) y en las aplicaciones de streaming de vídeo. Las búsquedas DNS son fundamentales en la infraestructura de Internet, ya que permiten traducir los nombres de dominio legibles para los humanos en direcciones IP numéricas, necesarias para localizar recursos en línea (Declan Williamson, 2023).

Asimismo, las aplicaciones modernas de Internet han evolucionado para mantener conexiones abiertas entre el servidor y el cliente, lo que permite el envío de mensajes sin necesidad de inicializar una nueva conexión. Un ejemplo de este avance es *WebTransport*, un protocolo diseñado para este propósito. De esta manera, se ahorra tiempo en la apertura de la conexión y se permite que el servidor envíe mensajes sin una solicitud previa del cliente.

A partir de esta evolución, *WebTransport* puede considerarse una mejora sobre *WebSockets*. Mientras que WebSockets admiten un único flujo bidireccional entre el cliente y el servidor, *WebTransport* permite múltiples flujos bidireccionales, tanto para una comunicación fiable como no fiable (Declan Williamson, 2023). En el caso de WebSockets, se establece una única conexión en la que el cliente o el servidor pueden iniciar la comunicación. Sin embargo, si se envían varios elementos, puede ser necesario abrir un socket adicional o multiplexar la información en el mismo flujo (Declan Williamson, 2023).

Este enfoque puede generar problemas como el bloqueo de la cabecera de línea (HoL) si algún mensaje se pierde en la transmisión. Para evitar este inconveniente, algunas aplicaciones

requieren más de un flujo de comunicación, como en los juegos, donde se necesita un flujo para la información del juego y otro para los mensajes entre jugadores. En este sentido, *WebTransport* permite abrir simultáneamente flujos fiables y no fiables, respondiendo a las necesidades modernas de comunicación entre cliente y servidor, no como un simple sustituto de WebSockets, sino como una evolución significativa (Declan Williamson, 2023).

Aplicaciones

Las aplicaciones son programas de software diseñados para realizar tareas específicas, ya sea en el ámbito empresarial, el entretenimiento o la comunicación. Dependiendo de su propósito, las aplicaciones pueden estar orientadas a gestionar datos, facilitar la interacción entre usuarios, o incluso monitorear procesos en tiempo real. En los últimos años, la evolución de las aplicaciones ha sido impulsada por la necesidad de brindar experiencias más dinámicas y en tiempo real, especialmente en el contexto de la comunicación y la colaboración en línea.

A continuación, se presentan ejemplos de diferentes tipos de aplicaciones, incluyendo aquellas que hacen uso de WebSockets (Rai, Socket.io Real-time Web Application Development., 2013).

Aplicaciones Empresariales

En el ámbito empresarial, uno de los componentes más relevantes es la Gestión de Relaciones con el Cliente (CRM), ya que permite a las empresas gestionar y analizar interacciones con clientes. Estas plataformas han evolucionado desde simples sistemas de seguimiento de incidencias hacia soluciones más sofisticadas que incluyen capacidades en tiempo real y funcionalidades sociales. Además, las soluciones de Gestión de Procesos de Negocio (BPM) también han integrado componentes en tiempo real para controlar el estado de los procesos y realizar actualizaciones en tiempo real (Rai, Socket.io Real-time Web Application

Development., 2013). Aunque no siempre utilizan WebSockets, algunas de estas aplicaciones se benefician de la comunicación en tiempo real para mejorar la interacción entre el usuario y el sistema.

Gaming

En el mundo de los videojuegos, *WebSockets* juegan un papel crucial en la creación de experiencias de juego inmersivas y colaborativas. Por ejemplo, en *Among Us*, los jugadores se comunican en tiempo real a través de un chat de texto mientras intentan descubrir al impostor entre ellos. Este tipo de juego requiere una comunicación fluida y en tiempo real, donde WebSockets ayudan a mantener la conexión entre los jugadores. Otro ejemplo es *League of Legends* (LoL), un juego de estrategia en línea donde los jugadores coordinan sus tácticas a través de un sistema de chat en tiempo real. La capacidad de comunicar de forma instantánea es clave para el éxito del equipo en cada partida, y los *WebSockets* facilitan este intercambio rápido de información entre jugadores (Rai, Socket.io Real-time Web Application Development., 2013).

Actualizaciones De Flujos Sociales

Las aplicaciones de mensajería y redes sociales también han adoptado tecnologías en tiempo real como *WebSockets* para mejorar la experiencia del usuario. *WhatsApp*, una de las aplicaciones de mensajería más utilizadas, emplea WebSockets para permitir la entrega instantánea de mensajes, tanto en chats individuales como en grupos. De manera similar, *Slack* y *Discord* son herramientas que permiten la comunicación en tiempo real, ya sea para equipos de trabajo o comunidades de usuarios. Ambas plataformas aprovechan *WebSockets* para garantizar

que los mensajes se entreguen de manera rápida y eficiente, además de permitir el uso de múltiples canales de comunicación (Rai, Socket.io Real-time Web Application Development., 2013).

Monitores Basados En La Web

En cuanto a la supervisión en tiempo real, plataformas como Google Analytics y Splunk también se benefician del uso de WebSockets. Google Analytics permite a los administradores de sitios web ver en tiempo real las interacciones de los usuarios, lo que facilita la toma de decisiones basadas en datos actuales. Por otro lado, *Splunk*, utilizado para supervisar eventos en infraestructuras y datos de máquinas, utiliza *WebSockets* para proporcionar actualizaciones instantáneas en los gráficos y datos visualizados por los usuarios, mejorando la capacidad de respuesta ante eventos en tiempo real (Rai, Socket.io Real-time Web Application Development., 2013).

Marco conceptual

Creación De Una Sala De Chat Sencilla

La creación de una sala de chat sencilla implica el desarrollo de un entorno virtual donde múltiples usuarios pueden intercambiar mensajes en tiempo real. En una aplicación de chat en tiempo real, este tipo de salas permite la interacción simultánea entre usuarios mediante el uso de tecnologías como *WebSockets*, que facilitan la comunicación bidireccional entre el cliente y el servidor sin la necesidad de establecer nuevas conexiones repetidamente. Un servidor de WebSocket se encarga de recibir los mensajes de los usuarios y de retransmitirlos a todos los participantes de la sala de chat en tiempo real, garantizando una experiencia de comunicación fluida y sin demoras perceptibles (Flanagan, JavaScript: The Definitive Guide, 2011).

En la aplicación, el código del servidor actuó como el centro de los mensajes entrantes.

Cuando llegaban nuevos mensajes, los emitía a todos los sockets conectados.

Los mensajes se enviaron desde el formulario en el lado del cliente. También se renderizaron los nuevos mensajes cuando eran emitidos desde el servidor. De esta forma, el cliente que emitía el mensaje escuchaba el mismo evento "message" que el resto de los clientes.

Gestión Del Ciclo De Vida De Los Sockets

El servidor mantiene una lista de sockets conectados, debe estar siempre al tanto de cuándo se desconecta un socket. Un socket puede desconectarse por diversas razones:

El usuario puede salir de la página web en la que se encuentra la conexión *WebSocket*.

La conexión a Internet del usuario puede interrumpirse (Cadenhead, 2015).

Cuando estas situaciones ocurren, se aprovecha el evento de desconexión para notificar al cliente que el socket ya no está disponible (Cadenhead, 2015).

Typescript

TypeScript es en realidad dos tecnologías distintas pero relacionadas: un lenguaje y un compilador:

El lenguaje es un lenguaje de programación estáticamente tipado y rico en funciones que añade a *JavaScript* verdaderas capacidades orientadas a objetos.

El compilador convierte el código *TypeScript* en *JavaScript* nativo, pero también ayuda al programador a escribir código con menos errores (Choi, 2020).

TypeScript es una tecnología de desarrollo. No existe ningún componente de tiempo de ejecución y ningún código *TypeScript* se ejecuta nunca en ninguna máquina. En su lugar, el compilador de *TypeScript* convierte *TypeScript* en *JavaScript* y ese código se despliega y ejecuta en navegadores o servidores. La extensión de archivo .ts es una extensión específica de *TypeScript* y permite que el compilador de *TypeScript* reconozca el archivo y lo transforme a *JavaScript* (Choi, 2020).

TypeScript se ha utilizado en el desarrollo de la aplicación para proporcionar una serie de beneficios que mejoran la calidad del código y la experiencia de desarrollo.

Tipado Dinámico Frente A Tipado Estático

Todo lenguaje de programación tiene y hace uso de tipos. Un tipo es simplemente un conjunto de reglas que describen un objeto y que pueden reutilizarse. *JavaScript* es un lenguaje de tipado dinámico. En *JavaScript*, las nuevas variables no necesitan declarar su tipo e incluso después de ser establecidas, pueden ser restablecidas a un tipo diferente. Esta característica añade una flexibilidad impresionante al lenguaje, pero también es la fuente de muchos errores. *TypeScript* utiliza una alternativa mejor llamada tipado estático. El tipado estático obliga al desarrollador a indicar el tipo de una variable desde el principio, cuando la crea. Esto elimina la ambigüedad y elimina muchos errores de

conversión entre tipos (Choi, 2020).

Backend

El *backend* es un término que utilizamos para referirnos a la arquitectura interna de un sitio web. Esta área lógica, que no es visible a los ojos del usuario y no incluye elementos de tipo gráfico, permite que todos los elementos de una web desarrollen la función correcta. La palabra, en inglés, se refiere a 'lo que está detrás', 'lo que no se ve'. De ahí su término 'back'. Cuando hablamos de *backend*, hablamos únicamente del código interno de la página. Quien lo desarrolla, desarrolla la funcionalidad del sitio y la seguridad y la optimización de los recursos (Bottega, 2020).

Node.Js

Node.js, es un entorno en tiempo de ejecución multiplataforma para la capa del servidor (en el lado del servidor) basado en *JavaScript. Node.j*s es un entorno controlado por eventos diseñado para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo (Simões, 2021).

La librería de E/S es lo suficientemente versátil como para implementar servidores que utilicen cualquier protocolo basado en *TCP o UDP*, como *DNS (Domain Name System), HTTP (Hypertext Transfer Protocol), IRC (Internet Relay Chat) o FTP (File Transfer Protocol).*

Aunque admite el desarrollo de servidores o clientes para cualquier protocolo de red, su caso de uso más destacado es en sitios web regulares en lugar de tecnologías como una pila Apache/PHP o Rails, o para complementar sitios web existentes. Por ejemplo, añadir chat en tiempo real o monitorizar sitios web existentes se puede lograr fácilmente con la biblioteca Socket.IO para Node.js (Herron D. , 2016).

El *File* Transfer *Protocol (FTP)* o protocolo de transferencia de archivos es uno de los

protocolos más antiguos de la historia de Internet: ya en 1974 se empezó a trabajar con la técnica de transmisión de archivos completos y, en 1985, se definió finalmente el *FTP* en el RFC 959 de manera concreta. Este protocolo está pensado para provocar cargas y descargas a través de comandos, de manera que se puedan transferir archivos desde el propio ordenador (portátil, smartphone, etc.) hacia un servidor y viceversa: el FTP también permite descargar archivos de un servidor al propio dispositivo (IONOS., 2023).

El término IRC, abreviatura de *Internet Relay Chat*, se refiere a un sistema de chat que permite enviarse mensajes de texto con otras personas (ubicadas en otros países) casi en tiempo real a través de *Internet*. Para ello, los usuarios del IRC se conectan a una de las numerosas redes y se unen a uno o varios de los canales que contienen. Una vez en ellos, tienen la opción de enviar mensajes de texto a todos los presentes o de mantener una conversación con solo una persona en el modo query (IONOS, 2022).

Capacidades De Node.Js. Más allá de su capacidad nativa para ejecutar JavaScript, los módulos incluidos proporcionan capacidades de diversos tipos:

Herramientas de línea de comandos (estilo script de Shell).

Un programa interactivo estilo TTY (REPL, que significa Read-Eval-Print Loop).

Excelentes funciones de control de procesos para supervisar procesos hijos.

Un objeto buffer para manejar datos binarios.

Sockets TCP o UDP con llamadas de retorno completas basadas en eventos.

Búsqueda DNS.

Un cliente/servidor HTTP y HTTPS superpuesto al acceso al sistema de archivos de la biblioteca TCP.

Soporte integrado de pruebas unitarias rudimentarias mediante aserciones (Herron,2016) .

La capa de red de Node.js es de bajo nivel y a la vez sencilla de usar. Por ejemplo, los módulos HTTP permiten escribir un servidor HTTP (o cliente) con solo unas pocas líneas de código. Esto es potente, pero se acerca mucho a las peticiones del protocolo y obliga a implementar con precisión las cabeceras HTTP que se deben incluir en las respuestas a las peticiones (Herron D. , 2016).

En la aplicación de chat en tiempo real, *Node.js* se utilizó principalmente para las siguientes funciones:

Entorno De Ejecución. *Node.js* proporciona un entorno de ejecución para *JavaScript* en el lado del servidor. Esto permite que el código *JavaScript* se ejecute fuera del navegador, lo que es fundamental para construir aplicaciones web modernas.

Manejo De Solicitudes HTTP. Aunque la aplicación utilizó Next.js, que maneja automáticamente las solicitudes HTTP, Node.js es la base sobre la cual se ejecuta *Next.js.* Esto significa que todas las solicitudes que llegan al servidor son procesadas por el motor de Node.js.

Integración Con Socket.Io. *Node.js* permitió la integración con *Socket.io*, que es crucial para la funcionalidad de comunicación en tiempo real de la aplicación. Socket.io utiliza las capacidades de Node.js para manejar conexiones *WebSocket* y permitir la comunicación bidireccional entre el cliente y el servidor.

Manejo De Eventos Asíncronos. *Node.js* es conocido por su modelo de I/O no bloqueante, lo que significa que puede manejar múltiples conexiones simultáneamente sin bloquear el hilo principal.

Socket.IO

Socket.io es una capa de abstracción para *WebSockets*, con *Flash, XHR*, *JSONP*, y *HTMLFile fallbacks*. Socket.io proporciona una sencilla librería de servidor y cliente para realizar

actualizaciones en tiempo real y *streaming* entre un servidor web y un cliente de navegador. Socket.io es un módulo de *node* disponible a través de npm (Rai, Socket.io Real-time Web Application Development. , 2013).

XMLHttpRequest (XHR) es un objeto especial de *Javascript* que permite realizar peticiones HTTP asíncronas (AJAX) de forma nativa desde *Javascript* (Manz.dev, s.f.).

JSONP o *JSON* con padding es una API para el intercambio de datos que pueden estar alojados en nuestro servidor o en servidores remotos (Zúñiga, 2022).

HTML fallback se refiere a un diseño básico con elementos web que se le presenta al usuario en caso de error. Por esto se llama *fallback*, puesto que es un diseño en el que se puede

«caer» en caso de que exista problemas con el diseño original. En este sentido, *fallback* es una forma de decir que algo es un plan B (Keepcoding., 2024).

En la aplicación creada, Socket.io permitió que el servidor y los clientes se comunicaran de manera eficiente y bidireccional. Cuando un usuario enviaba un mensaje, este se transmitía instantáneamente a todos los demás usuarios conectados mediante *WebSockets*, asegurando que todos vieran los mensajes en tiempo real. Además, si *WebSockets* no estaba disponible, *Socket.io* utilizaba alternativas como XHR para mantener la funcionalidad.

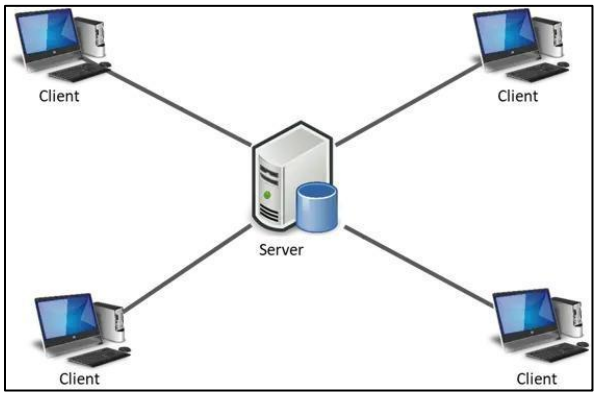
Arquitectura

En una arquitectura Cliente-Servidor existe un servidor y múltiples clientes que se conectan al servidor para recuperar todos los recursos necesarios para funcionar, en este sentido, el cliente solo es una capa para representar los datos y se detonan acciones para modificar el estado del servidor, mientras que el servidor es el que hace todo el trabajo pesado (Blancarte, 2021).

La Figura 9 muestra la arquitectura cliente-servidor, destacando los componentes

principales y la forma en que interactúan para lograr una comunicación eficiente.

Figura 9
Arquitectura cliente servidor



Fuente. Tomado de Reactive Programming. (Blancarte, 2021).

La aplicación se desarrolló siguiendo un patrón de arquitectura cliente-servidor, donde el cliente (*frontend*) se comunica con el servidor (*backend*) para enviar y recibir mensajes.

Next.Js Como Frontend

Interfaz De Usuario. *Next.js* es un framework de React que permitió construir interfaces de usuario interactivas y dinámicas. Proporciona una estructura para crear componentes, manejar el estado y gestionar la navegación entre diferentes páginas de la aplicación.

Renderizado Híbrido. *Next.js* permitió tanto el renderizado del lado del servidor (SSR) como el renderizado del lado del cliente (CSR). Esto significa que algunas páginas pueden ser generadas en el servidor y enviadas al cliente, mejorando el rendimiento y la SEO.

Api Routes. Next.js también permitió crear rutas de API dentro del mismo proyecto, lo que facilita la comunicación entre el frontend y el backend. Esto puede ser útil para manejar solicitudes simples sin necesidad de un servidor backend separado.

Node.Js. El backend de la aplicación está construido con *Node.js,* que es un

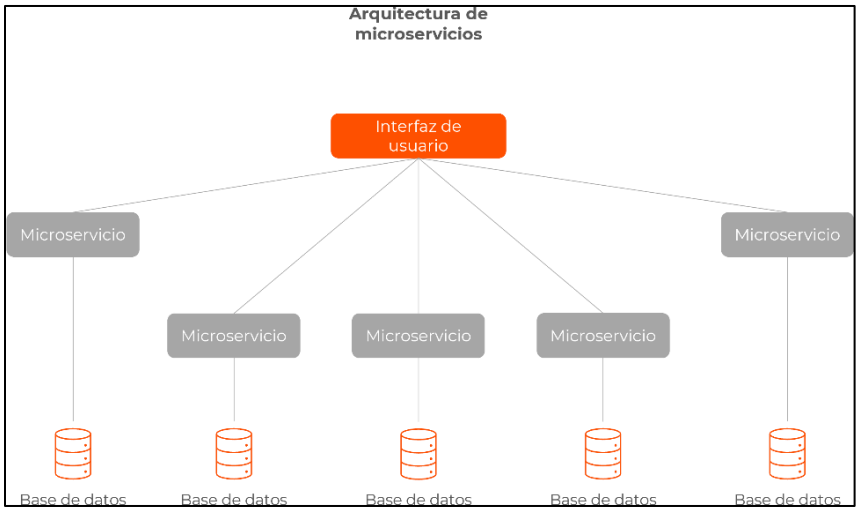
entorno de ejecución para *JavaScript* en el servidor. *Node.js* permitió manejar solicitudes HTTP, interactuar con bases de datos y gestionar la lógica del servidor.

Integración Con Next.Js. Si bien Next.js puede manejar algunas funciones del backend a través de sus API routes, en este caso, se utilizó un servidor Node.js separado para manejar la lógica de Socket.IO y otras funcionalidades del backend.

Los Microservicios son un enfoque arquitectónico que compone el software en pequeños componentes o servicios independientes. Cada servicio realiza una única función y se comunica con otros servicios a través de una interfaz bien definida. Como se ejecutan de forma independiente, se puede actualizar, modificar, implementar o escalar cada servicio según sea necesario (Amazon., 2024).

La Figura 10 muestra la arquitectura basada en microservicios, representando cómo distintos servicios autónomos interactúan con el cliente y entre sí, permitiendo una mayor modularidad y capacidad de adaptación frente a cambios o fallos individuales.

Figura 10
Arquitectura Microservicios



Fuente. Tomado de Decide Soluciones (Decidesoluciones., 2024).

Microservicios

Aunque en la aplicación no se implementó una arquitectura de microservicios en el

sentido estricto, se alinearon varios elementos con este enfoque:

Se descompusieron las funcionalidades en módulos independientes.

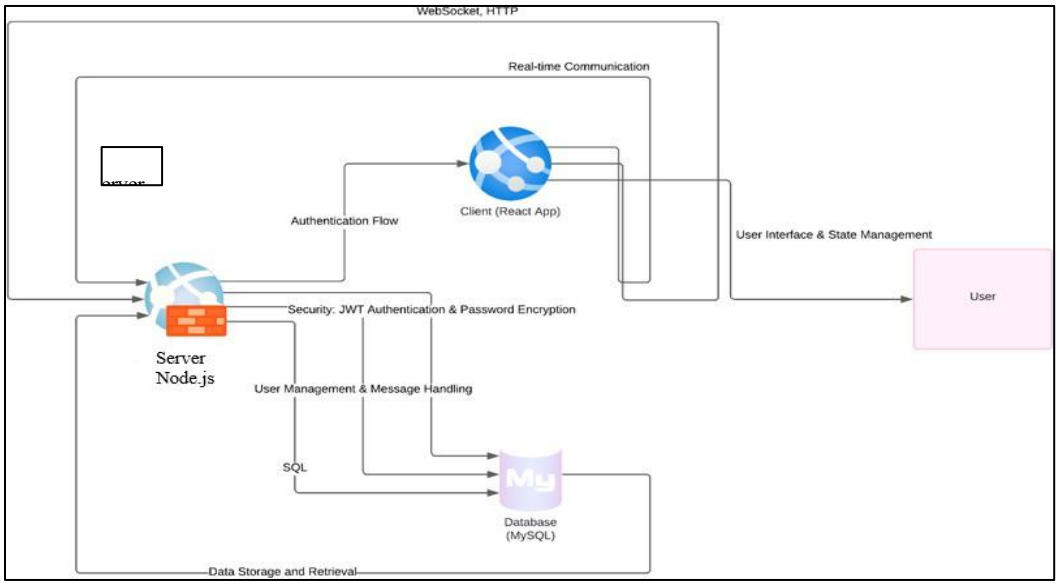
Se utilizó *Socket.IO* para manejar la comunicación en tiempo real.

Se implementó la interacción a través de APIs, lo que permitió un desacoplamiento entre el frontend y el backend.

Se contempló la posibilidad de desplegar y escalar componentes de manera independiente.

La Figura 11 muestra el flujo completo de una arquitectura moderna de aplicación web, destacando la comunicación en tiempo real mediante WebSockets y HTTP, el uso de JWT para autenticación y cifrado de contraseñas, así como la interacción entre cliente, servidor y base de datos para ofrecer una experiencia de usuario eficiente y segura.

Figura 11
Esquema Arquitectura



Nota. Esquema de arquitectura de aplicación diseñado por José Revelo usando Miro.
Fuente. Autoría propia.

Patrón De Diseño

Es un patrón de diseño que se estructura mediante tres componentes: modelo, vista y controlador. Este patrón tiene como principio que cada uno de los componentes se encuentre

separado en diferentes objetos, esto significa que los componentes no se pueden combinar dentro de una misma clase. Sirve para clasificar la información, la lógica del sistema y la interfaz que se le presenta al usuario (Easyappcode., 2020).

En la aplicación se implementó de la siguiente manera:

Modelo. Define la estructura de los datos y la lógica de acceso a la base de datos (usando Prisma).

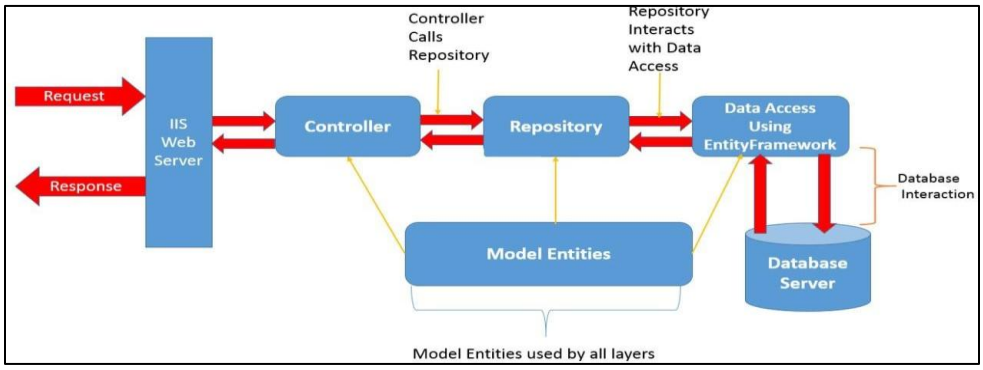
Vista. Es la parte de la aplicación que se encarga de la presentación de los datos. En una aplicación Next.js, las vistas se implementan a través de componentes *React*.

Controlador. Maneja la lógica de negocio y la comunicación entre el modelo y la vista.

En Next.js, esto se puede manejar en las API *routes* o en los métodos de los componentes.

La Figura 12 ilustra el funcionamiento del patrón MVC, mostrando cómo fluye la información entre los tres componentes principales y cómo cada uno cumple un rol específico dentro de la aplicación.

Figura 12
Patrón de diseño



Fuente. Tomada de: Microsoft Learn, 'Using a Simple Repository Pattern for Performing Database Operations in ASP.NET MVC 5 (DotNetCurry.com., s.f.)**.**

Otros Patrones Y Tecnologías

Se utilizó otros patrones en el desarrollo de la aplicación de la siguiente manera.

Singleton. Es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia **(Refactoring, 2024)**. utilizado para la conexión a la base de datos con *Prisma*.

La Figura 13 representa el funcionamiento del patrón Singleton, destacando cómo se evita la creación múltiple de instancias mediante una verificación previa y cómo se retorna siempre la misma instancia al ser solicitada.

Figura 13
Patrón Singleton

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Pub/Sub (Publicador-Suscriptor). Es un patrón de mensajería que promueve el acoplamiento flexible y la escalabilidad. Este patrón gira en torno al envío de mensajes de los publicadores a un número no especificado de suscriptores u oyentes, lo que promueve una dependencia de varios a varios entre objetos. **(Superviz & Pearson, 2024)** este patrón es común en aplicaciones de chat en tiempo real, especialmente cuando se utiliza una biblioteca como *Socket.io*. En este caso, el servidor actúa como un publicador que envía mensajes a todos los clientes suscritos a un canal específico.

La Figura 14 muestra gráficamente el funcionamiento del patrón Pub/Sub, donde un publicador envía mensajes a un canal común y los suscriptores registrados en ese canal reciben

automáticamente las notificaciones.

Figura 14
Patrón Pub/sub

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Observer. Es un patrón de diseño de comportamiento que te permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda al objeto que están observando. se utilizó a través de la suscripción a eventos de conexión y desconexión del socket. Cuando estos eventos ocurren, se notifica a los componentes que están utilizando el contexto del socket, permitiéndoles reaccionar a los cambios en el estado de conexión **(Refactoring., 2024)**.

La Figura 15 ilustra el funcionamiento del patrón Observer, mostrando cómo el sujeto central mantiene una lista de observadores que son notificados cada vez que ocurre un cambio.

Figura 15
Patrón Observer

```
"use client";

import React, {
    createContext,
    useContext,
    useEffect,
    useState
} from "react";
import {io as ClientIO} from "socket.io-client";

type SocketContextType = {
    socket: any | null;
    isConnected: boolean;
};

const SocketContext = createContext<SocketContextType>({
    socket: null,
    isConnected: false,
});

export const useSocket = () => {
    return useContext(SocketContext);
};

export const SocketProvider = ({
                                    children
                                }: {
    children: React.ReactNode
}) => {
    const [socket, setSocket] = useState(null);
    const [isConnected, setIsConnected] = useState(false);

    useEffect(() => {
        const socketInstance = new (ClientIO as any)(process.env.NEXT_PUBLIC_SITE_URL!, {
            path: "/api/socket/io",
            addTrailingSlash: false,
        });

        socketInstance.on("connect", () => {
            setIsConnected(true);
        });

        socketInstance.on("disconnect", () => {
            setIsConnected(false);
        });

        setSocket(socketInstance);

        return () => {
            socketInstance.disconnect();
        }
    }, []);

    return (
        <SocketContext.Provider value={{socket, isConnected}}>
            {children}
        </SocketContext.Provider>
    )
}
```

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Event-Driven Programming. Es un paradigma en el que la ejecución de un programa está determinada por eventos como acciones o mensajes del usuario (Geeksforgeeks, 2024).

Se utilizó para manejar eventos y notificaciones en tiempo real. Este enfoque se basa en la idea de que el flujo del programa se controla por eventos, que pueden ser acciones del usuario, mensajes del sistema, o cualquier otra señal que indique que algo ha ocurrido.

La Figura 16 representa gráficamente la arquitectura de la programación orientada a

eventos, mostrando cómo los eventos son detectados y gestionados por un event loop o bucle de eventos, que se encarga de distribuirlos a los controladores correspondientes.

Figura 16
Event-driven programming

```
"use client";

import * as z from "zod";
import axios from "axios";
import qs from "query-string";
import { useForm } from "react-hook-form";
import { zodResolver } from "@hookform/resolvers/zod";
import { Member, MemberRole, Profile } from "@prisma/client";
import { Edit, FileIcon, ShieldAlert, ShieldCheck, Trash } from "lucide-react";
import Image from "next/image";
import { useEffect, useState } from "react";
import { useRouter, useParams } from "next/navigation";

import { UserAvatar } from "@/components/user-avatar";
import { ActionTooltip } from "@/components/action-tooltip";
import { cn } from "@/lib/utils";
import {
    Form,
    FormControl,
    FormField,
    FormItem,
} from "@/components/ui/form";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { useModal } from "@/hooks/use-modal-store";

interface ChatItemProps {
    id: string;
    content: string;
    member: Member & {
        profile: Profile;
    };
    timestamp: string;
    fileUrl: string | null;
    deleted: boolean;
    currentMember: Member;
    isUpdated: boolean;
    socketUrl: string;
    socketQuery: Record<string, string>;
}

const roleIconMap = {
    "GUEST": null,
    "MODERATOR": <ShieldCheck className="h-4 w-4 ml-2 text-indigo-500" />,
    "ADMIN": <ShieldAlert className="h-4 w-4 ml-2 text-rose-500" />,
}

const formSchema = z.object({
    content: z.string().min(1),
});

export const ChatItem = ({
                             id,
                             content,
                             member,
                             timestamp,
                             fileUrl,
                             deleted,
                             currentMember,
                             isUpdated,
                             socketUrl,
                             socketQuery
                         }: ChatItemProps) => {
    const [isEditing, setIsEditing] = useState(false);
    const { onOpen } = useModal();
    const params = useParams();
    const router = useRouter();

    const onMemberClick = () => {
        if (member.id === currentMember.id) {
            return;
        }

        router.push(`/servers/${params?.serverId}/conversations/${member.id}`);
    }

    useEffect(() => {
        const handleKeyDown = (event: any) => {
            if (event.key === "Escape" || event.keyCode === 27) {
                setIsEditing(false);
            }
        };

        window.addEventListener("keydown", handleKeyDown);

        return () => window.removeEventListener("keyDown", handleKeyDown);
    }, []);

    const form = useForm<z.infer<typeof formSchema>>({
        resolver: zodResolver(formSchema),
        defaultValues: {
            content: content
        }
    });
```

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Asynchronous Programming. La programación asincrónica es una técnica que permite que el programa inicie una tarea potencialmente de larga duración y aún pueda responder a otros eventos mientras se ejecuta esa tarea, en lugar de tener que esperar hasta que esa tarea haya finalizado (Developer.mozilla.org, 2024). se utilizó a través de la

declaración de funciones async y el uso de await para manejar operaciones que pueden tardar en completarse, como la obtención de datos del perfil del usuario y la actualización de la base de datos.

La Figura 17 muestra un ejemplo de programación asíncrona utilizando async/await en un entorno con Next.js y Prisma.

Figura 17
Asynchronous programming

```
import {NextResponse} from "next/server";
import {currentProfile} from "@/lib/current-profile";
import {MemberRole} from "@prisma/client";
import {db} from "@/lib/db";

export async function POST(
    req: Request
) {
    try {
        const profile = await currentProfile();
        const {name, type} = await req.json();
        const {searchParams} = new URL(req.url);

        const serverId = searchParams.get("serverId");

        if (!profile) {
            return new NextResponse("Unauthorized", {status: 401});
        }
        if (!serverId) {
            return new NextResponse("Server ID missing", {status: 400});
        }
        if (name === "general") {
            return new NextResponse("Name cannot be 'general'", {status: 400});
        }
        const server = await db.server.update({
            where: {
                id: serverId,
                members: {
                    some: {
                        profileId: profile.id,
                        role: {
                            in: [MemberRole.ADMIN, MemberRole.MODERATOR]
                        }
                    }
                }
            },
            data: {
                channels: {
                    create: {
                       profileId: profile.id,
                        name,
                        type,
                    }
                }
            }
        });
        return NextResponse.json(server);

    } catch
        (error) {
        console.error("CHANNELS_POST", error);
        return new NextResponse("Internal Error", {status: 500});
    }
}
```

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Restful API. Es una interfaz que dos sistemas de computación utilizan para intercambiar información de manera segura a través de Internet. La mayoría de las aplicaciones para empresas deben comunicarse con otras aplicaciones internas o de terceros para llevar a cabo varias tareas

(Amazon, 2024).

Una Interfaz De Programa De Aplicación (Api). Define las reglas que se deben seguir para comunicarse con otros sistemas de software. Los desarrolladores exponen o crean API para que otras aplicaciones puedan comunicarse con sus aplicaciones mediante programación (Amazon, 2024) . En este contexto, la transferencia de estado representacional (REST) es una arquitectura de software que impone condiciones sobre cómo debe funcionar una API (Amazon, 2024).

De esta manera, las API que siguen el estilo arquitectónico de REST se llaman API REST. Los servicios web que implementan una arquitectura de REST son llamados servicios web RESTful (Amazon, 2024).

En la aplicación se implementó un endpoint de API RESTful que maneja solicitudes *POST* para crear un nuevo canal en un servidor. Utiliza métodos HTTP, maneja datos de entrada y salida en formato JSON, y realiza operaciones sobre recursos.

La Figura 18 presenta un fragmento de código en Next.js que ilustra una función asíncrona para gestionar la creación de canales en un servidor.

Figura 18
RESTful API

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Bases De Datos Y Almacenamiento

Una base de datos se define como un conjunto de datos organizados, los cuales guardan

una cierta relación entre sí. Así, los sistemas de información se encargan de recolectarlos y, posteriormente, gestionarlos y administrarlos para unos objetivos concretos (Axarnet, 2024)

MySQL es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS, por sus siglas en inglés) con un modelo cliente-servidor. RDBMS es un software o servicio utilizado para crear y administrar bases de datos basadas en un modelo relacional (Hostinger, 2024).

El ORM *(Object Relational Mapping)* es una técnica de programación que permite la relación de los objetos con los datos que ellos mismos representan. De esta forma, se consigue que las tareas de acceso a datos se simplifiquen para el programador (Neoattack., 2024).

Prisma ORM es un nuevo tipo de ORM que se diferencia fundamentalmente de los ORMs tradicionales y no sufre muchos de los problemas comúnmente asociados con estos.

En contraste, los ORMs tradicionales proporcionan una forma orientada a objetos de trabajar con bases de datos relacionales, mapeando las tablas a clases de modelo en tu lenguaje de programación (Prisma, 2024).

Sin embargo, Prisma ORM funciona de manera completamente diferente. Con *Prisma* ORM, defines tus modelos en el esquema declarativo de *Prisma*, el cual sirve como la única fuente de verdad para el esquema de la base de datos y los modelos en tu lenguaje de programación (Prisma, 2024).

Además, *Prisma* y *PlanetScale*, en conjunto, proporcionan un entorno de desarrollo que optimiza el desarrollo rápido y seguro de aplicaciones de acceso a datos, utilizando el ORM de Prisma y la plataforma MySQL altamente escalable de PlanetScale (Prisma, 2024).

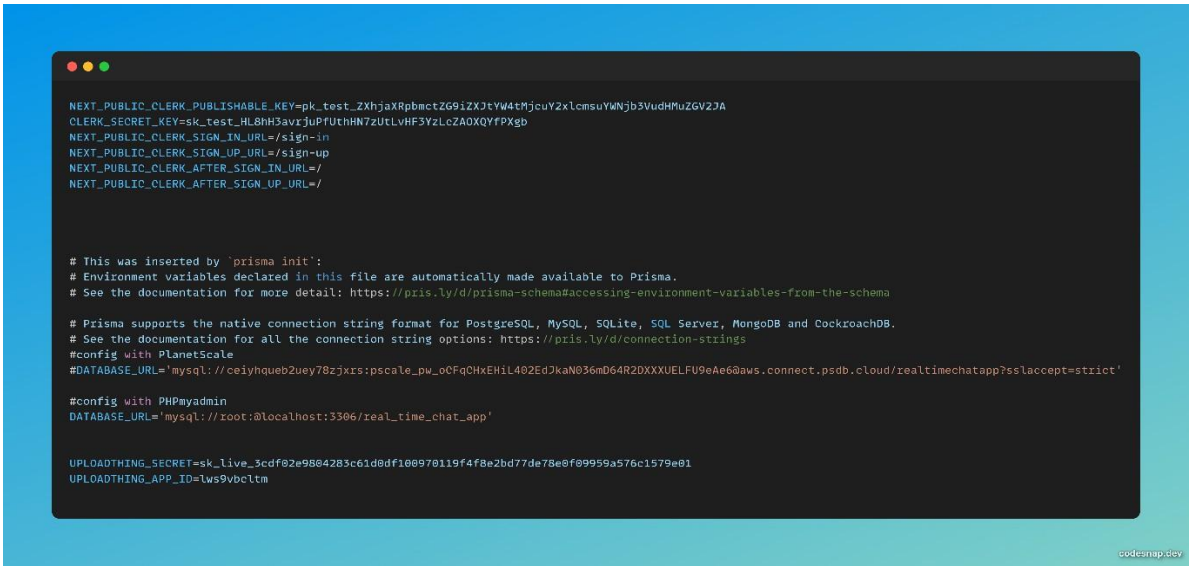
En la aplicación se utilizó Prisma para gestionar la base de datos, definiendo modelos que representan las entidades de la aplicación y utilizando una conexión configurada en un archivo.

env. Los datos, como los mensajes, se almacenan en la base de datos, lo que permite la persistencia y recuperación de información en tiempo real. Además, *Prisma Studio* facilita la visualización y manipulación de los datos almacenados.

PlanetScale se integró con la aplicación de chat en tiempo real como una solución de base de datos escalable y basada en la nube. Utilizando Prisma como ORM, conectando a *PlanetScale* y gestionar los modelos de datos de manera eficiente.

La Figura 19 muestra un archivo de configuración. env en un entorno de desarrollo moderno, el cual contiene variables de entorno esenciales para la conexión segura con servicios externos y la configuración del entorno de ejecución de una aplicación.

Figura 19
Archivo.env



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Implementación De Seguridad, Cifrado Y Autenticación

En la aplicación se implementaron medidas de seguridad avanzadas, incluyendo cifrado y autenticación, para proteger tanto los datos como la comunicación entre el cliente y el servidor. Estas estrategias se diseñaron para asegurar la confidencialidad, integridad y autenticación de la

información, utilizando protocolos de encriptación y métodos de autenticación confiables que resguardan los datos del usuario y previenen accesos no autorizados.

JWT es un objeto de JSON (notación de objeto de JavaScript), una herramienta de estándar abierto cuyo objetivo es establecer una transmisión de información entre dos o más campos. A partir de estos, se puede propagar información de forma segura y efectiva, que, además, es verificada, pues se firma de forma virtual (Keepcoding., 2024).

JSON Web Token es una cadena compuesta por tres partes separadas por un punto (.) y se serializa usando la base número 64. Las tres partes que componen este token son: *header***,** *paylod* **y** *signature* (Keepcoding., 2024).

La autenticación en la aplicación se lleva a cabo mediante el uso de tokens JSON Web Tokens (JWT). Al registrarse o iniciar sesión, se genera un *token JWT* que se envía al cliente. Este *token* es fundamental, ya que se utiliza para verificar la identidad del usuario en cada solicitud realizada a la aplicación.

Cifrado De Contraseñas

TLS (*Transport Layer Security*) es un protocolo de seguridad ampliamente adoptado, diseñado para facilitar la privacidad y la seguridad de los datos en las comunicaciones por Internet. Un caso de uso primario de TLS es la encriptación de las comunicaciones entre aplicaciones web y servidores, como los navegadores que cargan un sitio web (Cloudflare, 2024).

La aplicación utilizó el protocolo de cifrado TLS (*Transport Layer Security*) para proteger las comunicaciones entre el cliente y el servidor. Este protocolo asegura que los datos transmitidos permanezcan confidenciales y no sean interceptados por terceros no autorizados.

El algoritmo PBKDF2 es una función matemática de derivación de claves que permite cifrar contraseñas de forma segura para que se puedan almacenar debidamente en las bases de

datos de una aplicación móvil o web (Keepcoding., 2024).

Para fortalecer la seguridad de las contraseñas, se utilizó el algoritmo PBKDF2, que deriva una clave segura a partir de una contraseña. Esta clave derivada es luego empleada en el cifrado y la autenticación, dificultando los ataques de fuerza bruta. La función hash SHA512 se aplicó junto con PBKDF2 para generar un hash seguro de las contraseñas, añadiendo una capa adicional de protección.

SHA es el acrónimo de *Secure Hash Algorithm*, una familia de funciones hash criptográficas diseñadas por la Agencia de Seguridad Nacional (NSA). Esta función hash criptográfica desempeña un papel fundamental para garantizar la integridad y la seguridad de los datos digitales (Ssldragon., 2024).

Si pasamos a SHA-512, veremos que es un miembro más potente de la familia SHA-2 que produce un hash de 512 bits, lo que ofrece mayor seguridad, pero exige más recursos informáticos (Ssldragon., 2024).

Además, se implementó el cifrado AES para proteger la clave privada del usuario almacenada en la base de datos y para cifrar los mensajes de chat. Este estándar de cifrado es reconocido por su robustez y su uso en la protección de datos sensibles.

El cifrado AES, o *advanced encryption standard*, es un cifrado simétrico en bloques que se utiliza para cifrar datos confidenciales. Tanto por seguridad como por velocidad, AES se ha convertido en un estándar de seguridad para usuarios y aplicaciones que necesitan una encriptación fácil de usar (Pandasecurity, 2024).

El protocolo de intercambio de claves *Elliptic Curve Diffie-Hellman* (ECDH) es una variante del protocolo Diffie-Hellman que aprovecha las propiedades matemáticas de las curvas elípticas para proporcionar un método de intercambio de claves más eficiente y seguro (Eitca.,

2024).

Para el intercambio seguro de claves públicas entre dos partes, se empleó el método ECDH (Elliptic-Curve Diffie-Hellman). Esto permitió que ambas partes generaran una clave secreta compartida, utilizada para cifrar y descifrar mensajes de chat en canales privados, manteniendo la privacidad de las conversaciones.

Cada usuario tiene una sal única, lo que garantiza que, incluso si dos usuarios comparten la misma contraseña, los valores hash generados sean diferentes. Esta técnica evita la posibilidad de que contraseñas idénticas produzcan resultados de cifrado iguales.

Una sal de contraseña es un bit aleatorio de datos que se añade a la contraseña antes de pasarla por el algoritmo hash. Una sal de contraseña es diferente para cada usuario, lo que hace que los hashes asignados a cada contraseña también sean únicos (Nordpass, 2024).

Finalmente, se generó una frase de contraseña combinando la clave de autenticación con la contraseña del usuario. Esta frase se empleó para cifrar la clave privada del usuario almacenada en la base de datos, asegurando que los datos permanezcan protegidos incluso en caso de acceso no autorizado a la base de datos.

Validación De Datos

Zod es una poderosa biblioteca de validación y declaración de esquemas con una tracción significativa en la comunidad de *JavaScript*. Esto se debe a su capacidad para manejar estructuras de datos complejas y proporcionar una lógica de validación personalizada, lo que lo convierte en una excelente opción para la validación de formularios en *React* (Dhiwise, 2024).

En la aplicación, se utilizó la biblioteca Zod para definir y validar esquemas de datos.

Esto permite asegurarse de que los datos que se reciben cumplan con ciertas reglas antes de ser procesados.

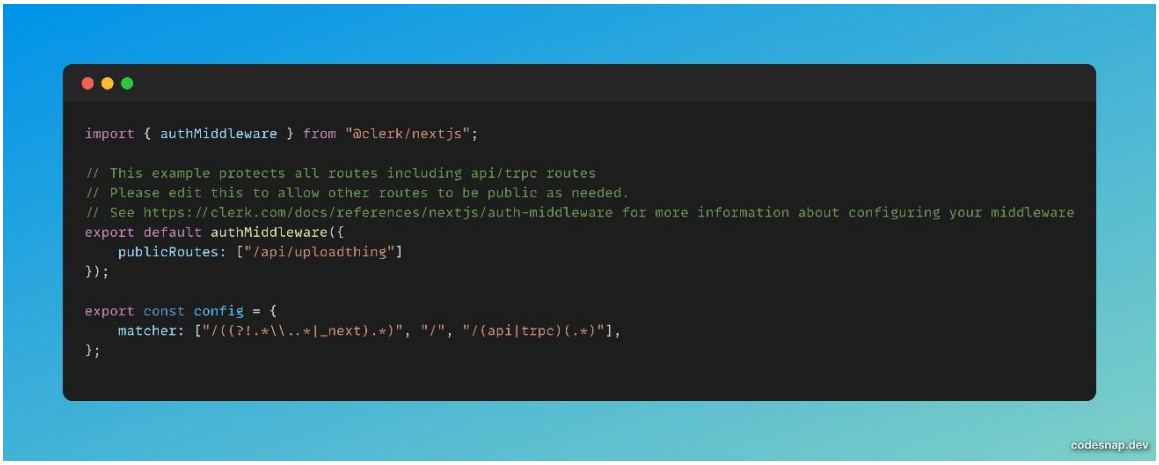
Seguridad Adicional

El middleware es un software con el que las diferentes aplicaciones se comunican entre sí. Brinda funcionalidad para conectar las aplicaciones de manera inteligente y eficiente, de forma que se pueda innovar más rápido. El middleware actúa como un puente entre tecnologías, herramientas y bases de datos diversas para que pueda integrarlas sin dificultad en un único sistema (Amazon., 2024).

authMiddleware de Clerk: se utilizó para proteger las rutas de la aplicación, asegurando que solo los usuarios autenticados puedan acceder a ciertas partes, mientras que algunas rutas (como /api/uploadthing) se mantienen públicas.

La Figura 20 presenta un archivo de configuración de middleware de autenticación utilizando authMiddleware de Clerk en un entorno Next.js.

Figura 20
Middleware



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Frontend

Shadcn/Ui

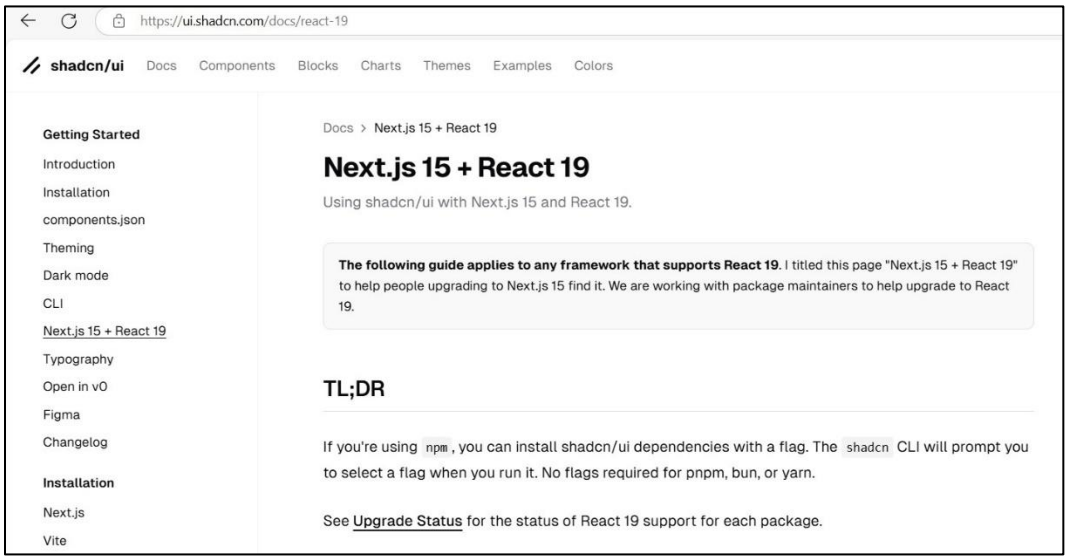
Es una biblioteca de componentes UI originalmente diseñada para web, construida sobre Tailwind CSS y Radix UI (Dribba, 2024).

Su objetivo es simplificar el proceso de desarrollo ofreciendo componentes React accesibles, personalizables y listos para usar, que permiten a los desarrolladores crear interfaces atractivas y funcionales de manera rápida (Dribba, 2024).

En este caso, se utilizó *shadcn/ui* para construir la aplicación haciendo uso de componentes React y Next.js.

La Figura 21 muestra la documentación oficial de shadcn/ui para su uso con Next.js 15 y React 19.

Figura 21
Shadcn/ui



Fuente. Tomada de shadcn/ui (Shadcn/ui, 2024).

React

ReactJS es una de las librerías más populares de JavaScript para el desarrollo de aplicaciones móviles y web. Creada por Facebook, React contiene una colección de fragmentos de código JavaScript reutilizables utilizados para crear interfaces de usuario (UI) llamadas componentes (Hostinger, 2024).

Next.js

Next.js es un *framework* de JavaScript diseñado para crear aplicaciones web basadas en

React, que ofrece soporte para aplicaciones web estáticas y representadas en el lado del servidor.

Next.js, creado teniendo en cuenta las mejores prácticas, permite crear aplicaciones web "universales" de forma coherente y con una configuración mínima. Estas aplicaciones web "universales" representadas en el servidor, también denominadas "isomórficas", comparten código entre el cliente y el servidor (Microsoft, 2024).

En la aplicación, se utilizó Next.js de la siguiente manera:

Configuración Del Servidor. Next.js se encargó automáticamente de la configuración del servidor, por lo que no fue necesario configurar un servidor Express por separado. Al ejecutar el comando npm run dev, Next.js iniciaba un servidor que escuchaba en el puerto 3000.

Rutas. Next.js empleó un sistema de enrutamiento basado en el sistema de archivos. Las páginas se crearon en la carpeta pages, y Next.js gestionó las rutas automáticamente para cada archivo, lo que simplificó la creación de rutas.

Middleware. Aunque Next.js permite el uso de middleware, en este caso se utilizó el middleware propio de Next.js para manejar la autenticación y otras funcionalidades de la aplicación.

Integración Con Socket.Io. Para integrar Socket.io en la aplicación, se empleó directamente con Next.js. Esto se logró creando un servidor HTTP dentro de la configuración de Next.js.

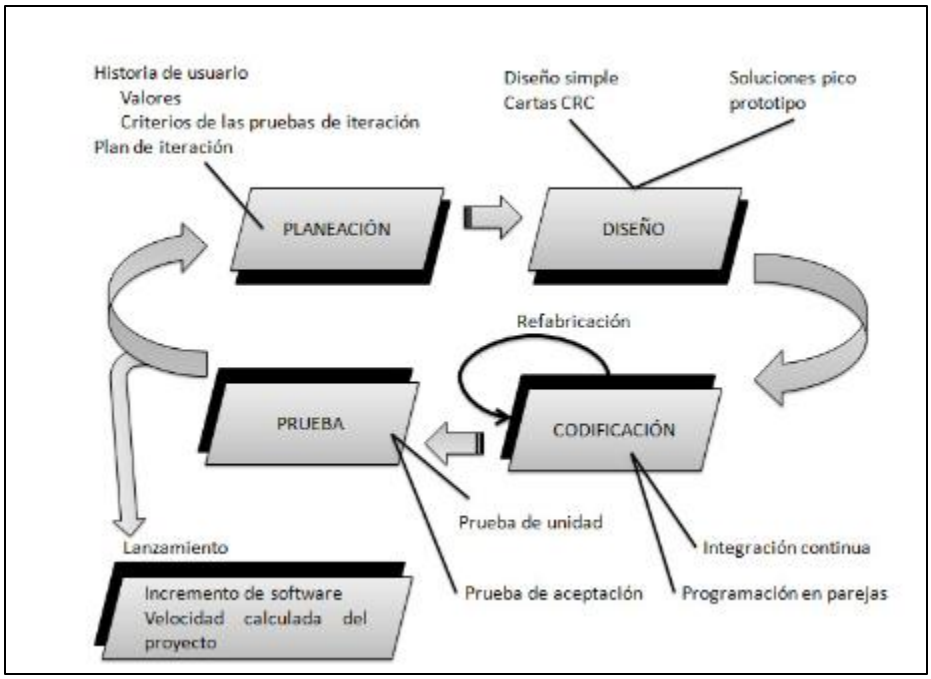
Metodología

Programación Extrema O XP

La Programación Extrema o XP *(Extreme Programming)* pertenece a la familia de las metodologías agiles. XP propone cuatro prácticas esenciales (Meneses Guevara, Rosado Gómez, & Quintero Duarte, 2012).

Como ilustra la Figura 22, las etapas de desarrollo de XP se estructuran de manera que permiten una evolución iterativa y constante del producto, facilitando la adaptación a los cambios y la mejora continua.

Figura 22
Etapas de desarrollo de XP



Fuente. Tomada de Revista UFPS (Meneses Guevara, Rosado Gómez, & Quintero Duarte, 2012).

Planeación

Esta actividad definió las características principales y la funcionalidad requerida. Estas características se transformaron en requisitos del negocio, los cuales se especificaron mediante Historias de Usuario (Meneses Guevara, Rosado Gómez, & Quintero Duarte, 2012).

Historias De Usuario

Registro De Usuario. Como nuevo usuario, quiero registrarme en la aplicación para poder crear una cuenta y acceder a las funcionalidades de chat.

Inicio De Sesión. Como usuario registrado, quiero iniciar sesión en la aplicación para poder acceder a mi cuenta y mis conversaciones.

Enviar Mensajes. Como usuario, quiero enviar mensajes a otros usuarios para poder comunicarme en tiempo real.

Recibir Mensajes. Como usuario, quiero recibir mensajes de otros usuarios para poder ver las respuestas y mantener la conversación.

Crear Salas De Chat. Como usuario, quiero crear salas de chat para poder conversar con múltiples usuarios al mismo tiempo.

Unirse A Salas De Chat. Como usuario, quiero unirme a salas de chat existentes para poder participar en conversaciones grupales.

Perfil De Usuario. Como usuario, quiero ver y editar mi perfil para poder actualizar mi información personal y foto de perfil.

Historial De Conversaciones. Como usuario, quiero ver el historial de mis conversaciones para poder revisar mensajes anteriores.

Eliminar Mensajes. Como usuario, quiero eliminar mensajes que he enviado para poder mantener mi conversación organizada.

Buscar Usuarios. Como usuario, quiero buscar otros usuarios para poder encontrar y agregar amigos a mi lista de contactos.

Gestión De Preferencias De Seguridad. Como usuario, quiero gestionar las preferencias de seguridad de mi cuenta, para asegurarme de que mi información personal esté protegida.

Diseño

Etapa en donde fueron evaluadas las historias de usuario por el equipo del proyecto para dividirlas en tareas. Cada tarea representaba una característica distinta del sistema y se diseñó una prueba de unidad para verificar cada tarea. Estas tareas se representaron por medio de las tarjetas CRC (Clase-Responsabilidad-Colaborador) (Meneses Guevara, Rosado Gómez, & Quintero Duarte, 2012).

Las tarjetas CRC identificaron y organizaron las clases bajo el paradigma orientado a objetos (lo que incluyó la asignación de responsabilidades). Cada tarjeta contenía el nombre de la clase (que representaba una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con las que se relacionaba o que colaboraban con ella (Meneses Guevara, Rosado Gómez, & Quintero Duarte, 2012).

Tarjetas CRC: Cada tarjeta ilustra las responsabilidades esenciales de la clase y sus respectivas colaboraciones, proporcionando una visión detallada del comportamiento esperado dentro del sistema.

Tabla 2
Tarjeta CRC Usuario

Usuario	
Responsabilidades	Colaboradores
Autenticarse en la aplicación.	Mensaje
Enviar y recibir mensajes.	Chat
Mantener el estado de conexión.	Servicio de Autenticación

Nota. Tarjeta CRC del Usuario describe la clase Usuario con sus responsabilidades.
Fuente. Autoría propia.

Tabla 3
Tarjeta CRC Mensaje

Mensaje	
Responsabilidades	Colaboradores
Almacenar el contenido del mensaje.	Usuario
Marcar el mensaje como leído o no leído.	Chat
Proporcionar la hora de envío.	

Nota. Tarjeta CRC Mensaje: Define responsabilidades de envío y recepción de mensajes.
Fuente. Autoría propia.

Tabla 4
Tarjeta CRC Chat

Chat	
Responsabilidades	Colaboradores
Gestionar la lista de mensajes.	Usuario
Proporcionar la interfaz para enviar y recibir mensajes.	Mensaje
Mantener el historial de mensajes.	

Nota. Tarjeta CRC Chat: Gestiona conversaciones entre usuarios.
Fuente. Autoría propia.

Tabla 5
Tarjeta CRC Servicio de autenticación

Servicio de autenticación	
Responsabilidades	Colaboradores
Manejar el registro y la autenticación de usuarios.	Usuario
Proteger las rutas de acceso a la aplicación.	Base de datos
Gestionar la sesión del usuario.	

Nota. Tarjeta CRC Servicio de autenticación: Valida credenciales y gestiona accesos; colabora con Usuario y Base de datos.
Fuente. Autoría propia.

Tabla 6
Tarjeta CRC Base de datos

Base de datos	
Responsabilidades	Colaboradores
Almacenar datos de usuarios y mensajes. Proporcionar acceso a los datos a través de consultas. Manejar la persistencia de datos.	Servicio de autenticación Mensaje

Nota. Tarjeta CRC Base de datos: Maneja almacenamiento y consultas; colabora con Servicio de autenticación y Mensaje.
Fuente. Autoría propia.

Desarrollo

Configuración Del Entorno. Primero, se configuró el entorno de desarrollo. Esto incluye la instalación de las dependencias necesarias y el inicio del servidor de desarrollo. Los comandos utilizados son: npm install, npx run dev.

Ide (Integrated Development Environment). Considero que IntelliJ IDEA es un IDE más apto para mi proyecto de aplicación de chat en tiempo real en lugar de VS Code debido a varias razones. En primer lugar, IntelliJ IDEA ofrece características avanzadas como asistencia inteligente de código, depuración y herramientas de prueba que pueden ayudarte a desarrollar código de alta calidad de manera más eficiente.

También proporciona una profunda integración con varios *framework* y tecnologías, incluyendo *Next.js*, que puede ayudar a aprovechar al máximo sus características.

Por el contrario, VS Code es un IDE más ligero que está diseñado para la simplicidad y facilidad de uso. Aunque ofrece una amplia gama de extensiones y *plugin*, puede que no proporcione el mismo nivel de funciones avanzadas e integración que IntelliJ IDEA.

Además, IntelliJ IDEA ofrece soporte superior para *JavaScript* y *TypeScript*, que son los

principales lenguajes utilizados en el proyecto de aplicación de chat en tiempo real. Proporciona herramientas avanzadas de finalización de código, detección de errores y refactorización que pueden ayudarte a escribir un código más limpio y fácil de mantener.

En general, IntelliJ IDEA ofrece herramientas integradas para probar, depurar y perfilar código, lo que puede ayudar a identificar y solucionar problemas más rápidamente. También proporciona integración con sistemas de control de versiones como Git, que pueden ayudar a gestionar tu código base de forma más eficaz.

Estructura Del Código

Frontend

React (v17.0.2). Utilizado para construir la interfaz de usuario dinámica y modular de la aplicación.

Next.js. *Framework* elegido para el desarrollo del frontend, que facilita la renderización del lado del servidor y la generación estática de páginas.

Tailwind *CSS*. *Framework* de CSS empleado para la estilización eficiente y personalizada de la interfaz. La comunicación con el *backend* se realiza a través de solicitudes, permitiendo el envío de mensajes y archivos de forma fluida.

El archivo *package.json* contiene metadatos sobre el proyecto y define los paquetes y scripts necesarios para su funcionamiento.

La Figura 23 muestra el archivo *package.json* de la aplicación, donde se definen scripts y dependencias esenciales para el desarrollo.

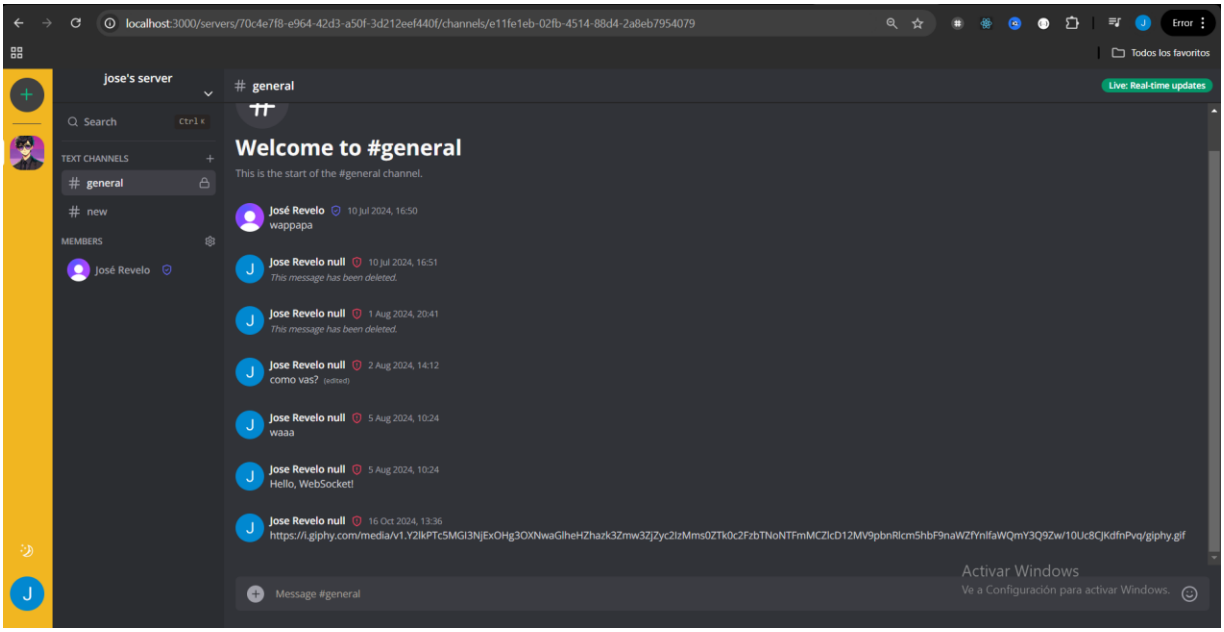
Figura 23
Package.json

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 24 muestra la interfaz de usuario de la aplicación, diseñada para ofrecer una

experiencia intuitiva y visualmente atractiva.

Figura 24
Interfaz de usuario



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Desarrollo Iterativo

El desarrollo del sistema o aplicación se realizó de manera incremental y gradual, trabajando en pequeños bloques o ciclos llamados iteraciones. Cada iteración se enfocó en completar una funcionalidad específica o en mejorar la que ya se había desarrollado, lo que permitió realizar ajustes y validar el progreso antes de pasar a la siguiente etapa (Meneses Guevara, Rosado Gómez, & Quintero Duarte, 2012).

Pruebas Automatizadas. Se implementaron pruebas automatizadas para asegurar que cada funcionalidad funciona correctamente. Esto incluye pruebas unitarias y de integración (Meneses Guevara, Rosado Gómez, & Quintero Duarte, 2012).

Integración Continua. Se utilizó un sistema de integración continua *(GitHub Actions)* para asegurar que el código se integre y despliegue frecuentemente. Esto permite detectar errores rápidamente y mantener la calidad del código.

Revisiones De Código. Las revisiones de código son una parte importante de la etapa de codificación. Se realizan revisiones regulares para asegurar que el código cumpla con los estándares de calidad y que no haya errores.

Documentación. Finalmente, se documentó el código para que otros desarrolladores pudieran entenderlo fácilmente. Esto incluyó comentarios en el código y documentación sobre cómo usar las diferentes funcionalidades de la aplicación (Meneses Guevara, Rosado Gómez, & Quintero Duarte, 2012).

Pruebas

Pruebas Unitarias

Jest. Jest es un framework flexible y sencillo de utilizar. Además de sus funciones básicas de comprobación de JavaScript, ofrece configuraciones y complementos para admitir la comprobación de aplicaciones basadas en Babel, webpack, Vite, Parcel o TypeScript. (Kinsta, 2025)

Configuración Archivo Tsconfig. Jest.Json. La Figura 25 muestra el archivo tsconfig. jest.json, configurado para habilitar las pruebas unitarias con Jest en un entorno TypeScript.

Figura 25
tsconfig.jest,json

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Configuración Archivo Jest.Config.Ts. La Figura 26 presenta la configuración de Jest en TypeScript, donde se definen aspectos clave como el entorno de pruebas (node), la raíz del proyecto, alias para rutas de importación y la transformación de archivos .ts y .tsx.

Figura 26
jest.config.ts

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Componente Current Profile. Ruta de acceso: lib/tests/currentProfile.test.ts

La Tabla 7 presenta dos casos de prueba diseñados para validar el comportamiento de la función currentProfile.

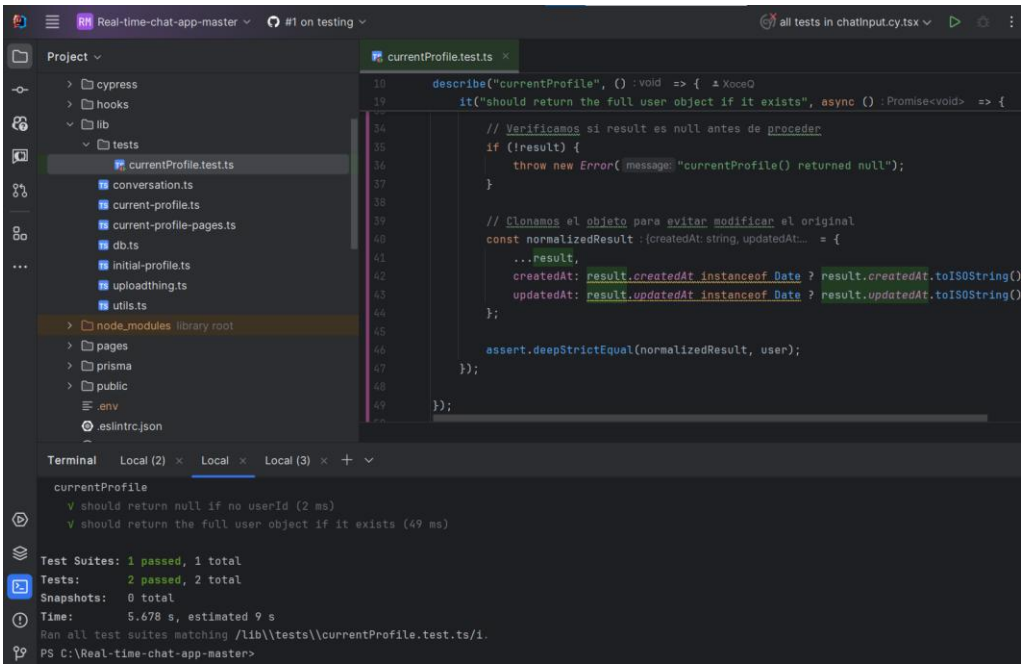
Tabla 7
Tabla TC-001-002

Id del test	Nombre del test	Descripción	Entrada simulada	Resultado esperado	Verificación
TC-001	Debe retornar null si no hay userId	Verifica que la función currentProfile retorne null cuando no hay un userId disponible.	auth() simulado con userId: null	La función currentProfile() debe devolver null.	Se verifica con assert.strictEqual(result, null)
TC-002	Debe retornar el objeto completo del usuario si existe	Verifica que la función currentProfile retorne el objeto completo del usuario si existe un userId válido.	auth() simulado con un objeto de usuario que contiene los campos: createdAt, email, id, imageUrl, name, updatedAt, userId.	La función currentProfile() debe devolver un objeto que coincide con el objeto de usuario simulado.	Se verifica con assert.deepStrictEqual(result, user)

Nota. La tabla resume los casos de prueba TC-001 y TC-002 para validar el comportamiento de la función currentProfile () con y sin un userId simulado.
Fuente. Autoría propia.

La figura 27 ilustra cada prueba especifica su identificación, descripción, entrada simulada, resultado esperado y método de verificación.

Figura 27
TC-1-2



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba API Handler – Messages Método POST. Ruta de acceso:

pages/api/socket/direct messages/tests/messageHandlerPost.test.ts

La Tabla 8 detalla una serie de casos de prueba enfocados en validar el comportamiento del *handler* ante diferentes condiciones de entrada en solicitudes HTTP.

Tabla 8
TC-003-009

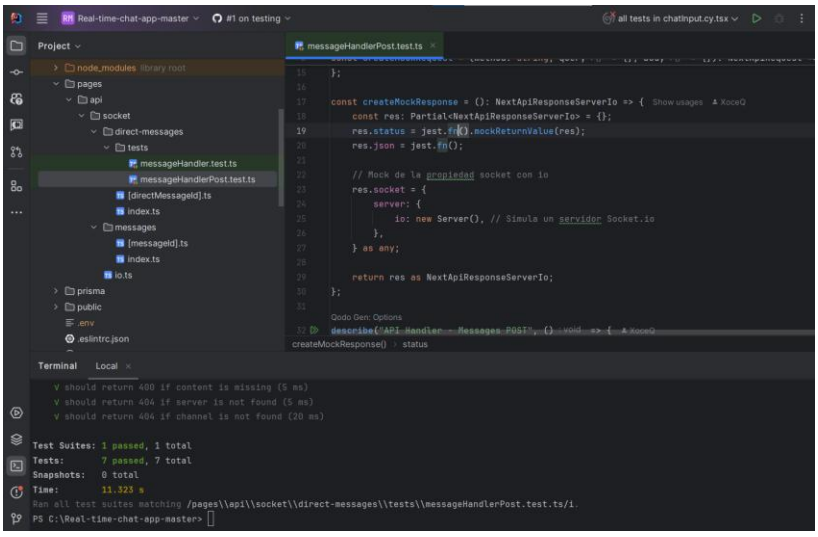
Id del test	Nombre del test	Descripción	Método HTTP	Resultado esperado	Precondiciones	Dependencias
TC - 003	Método no permitido	Verifica que se retorne 405 cuando se usa un método distinto de POST	GET	405 método No autorizado	Ninguna	Handler
TC-004	Usuario no autenticado	Verifica que se retorne 401 si el usuario no está autenticado	POST	401 No autorizado	No autorizado	Handler
TC-005	Falta serverId	Verifica que se retorne 400 si no se envía serverId en la solicitud	POST	400 Solicitud errónea	Usuario no autenticado	Handler
TC-006	Falta channelId	Verifica que se retorne 400 si no se envía channelId en la solicitud	POST	400 solicitud errónea	serverId presente	Handler
TC-007	Falta content	Verifica que se retorne 400 si no se envía	POST	400 solicitud errónea	serverId y channelId presentes	Handler

		content en la solicitud				
TC-008	Servidor no encontrado	Verifica que se retorne 404 si el serverId no existe	POST	404 No encontrado	serverId, channelId y content presentes	Handler
TC-009	Canal no encontrado	Verifica que se retorne 404 si el channelId no existe	POST	404 No encontrado	serverId, channelId y content presentes	Handler

Nota. La tabla presenta los casos de prueba TC-003 a TC-009 para validar errores HTTP comunes en solicitudes al handler, como métodos no permitidos, falta de autenticación o parámetros, y recursos no encontrados.
Fuente. Autoría propia.

La Figura 28 muestra la ejecución de pruebas unitarias sobre el archivo messageHandlerPost.test.ts, encargado de verificar el correcto funcionamiento del manejador de mensajes en la ruta correspondiente.

Figura 28
TC-3-9



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba API Messages Método DELETE/PATCH. Ruta de acceso:

pages/api/socket/direct-messages/tests/messageHandler.test.ts

La Tabla 9 detalla los casos de prueba asociados con las operaciones de eliminación (DELETE) y actualización (PATCH) de mensajes en el sistema de mensajería.

Tabla 9
TC-010-014

Id del test	Nombre del test	Descripción	Método HTTP	Resultado esperado	Precondiciones	Dependencias
TC -010	debe devolver 401 si el usuario no está autenticado	Verifica que se devuelve un error 401 si el usuario no está autenticado.	DELETE /PATCH	El sistema debe devolver un código de estado 401 Unauthorized.	El usuario no tiene un userId (usuario no autenticado).	Auth, handler
TC-011	debe devolver 404 si no se encuentra la conversación	Verifica que se devuelve un error 404 si la conversación no se encuentra en la base de datos.	DELETE /PATCH	El sistema debe devolver un código de estado 404 Not Found.	La conversación solicitada no existe.	auth, currentProfile Pages, db
TC-012	debe devolver 404 si el mensaje no se encuentra o se borra	Verifica que se devuelve un error 404 si el mensaje no	DEETE /PATCH	El sistema debe devolver un código de estado 404 Not	El mensaje solicitado no existe o ya está eliminado.	auth, currentProfile Pages, db

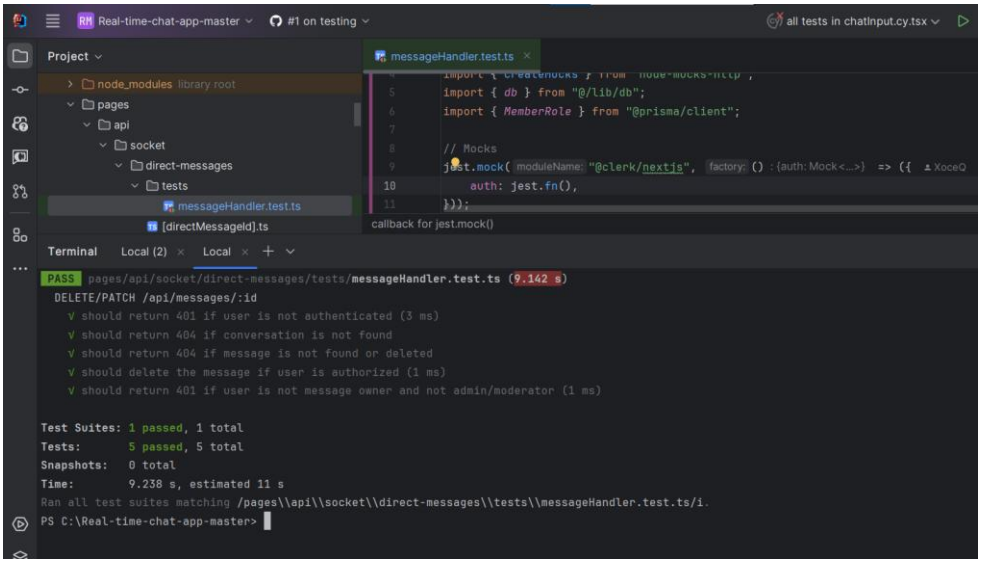
		se encuentra o ha sido eliminado.		Found.		
TC-013	debe borrar el mensaje si el usuario está autorizado	Verifica que el mensaje se elimina si el usuario tiene los permisos adecuados.	DELETE	El mensaje debe ser eliminado con éxito y debe devolver el estado de mensaje eliminado.	El usuario tiene permisos para eliminar el mensaje.	auth, currentProfile Pages, db
TC-014	debe devolver 401 si el usuario no es el propietario del mensaje y no es administrador/moderador	Verifica que se devuelve un error 401 si el usuario no es el propietario del mensaje ni tiene privilegios de administrador/moderador.	DELETE	El sistema debe devolver un código de estado 401 Unauthorized.	El usuario no es el propietario del mensaje y no tiene privilegios de administrador/ moderador.	auth, currentProfile Pages, db

Nota. La tabla muestra los casos de prueba TC-010 a TC-014 para validar la autorización, existencia y eliminación de mensajes o conversaciones mediante métodos DELETE/PATCH. *Fuente.* Autoría propia.

La Figura 29 muestra la ejecución exitosa de la suite de pruebas messageHandler.test.ts

correspondiente a las rutas DELETE/PATCH /api/messages/:id.

Figura 29
TC-10-14



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Socket.Io API Handler. La Tabla 10 presenta los casos de prueba diseñados para verificar el comportamiento del sistema en relación con la inicialización del servidor Socket.io.

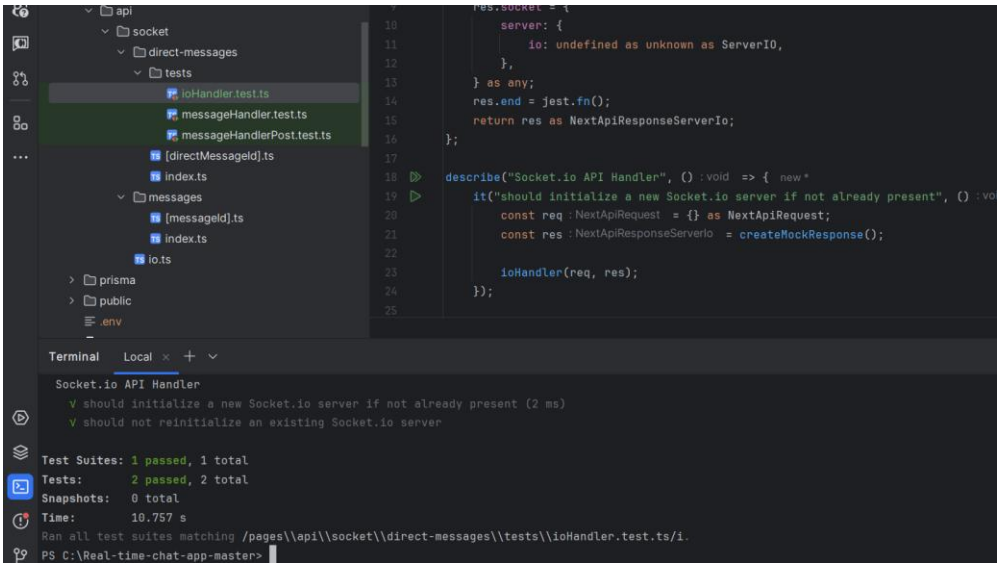
Tabla 10
TC-015-016

Id del test	Nombre del test	Descripción	Resultado esperado	Precondiciones	Dependencias
TC -015	Inicialización del servidor Socket.io.	Verifica si se inicializa un nuevo servidor Socket.io cuando no existe previamente.	Se debe crear una instancia de ServerIO y llamar a res.end().	El servidor no debe tener una instancia previa de Socket.io.	HTTP, Next, Socket.io, NextApiResponseServerIo, ioHandler.
TC-016	Reutilización del servidor Socket.io.	Verifica que si ya existe un servidor Socket.io, no se vuelve a inicializar.	Se debe mantener la misma instancia de ServerIO y llamar a res.end().	El servidor ya debe tener una instancia de Socket.io.	HTTP, Next, Socket.io, NextApiResponseServerIo, ioHandler.

Nota. La tabla detalla los casos de prueba TC-015 y TC-016 para validar la inicialización y reutilización del servidor Socket.io en el entorno Next.js.
Fuente. Autoría propia.

La Figura 30 muestra los resultados de la ejecución de las pruebas correspondientes al archivo ioHandler.test.ts, el cual valida el correcto comportamiento del manejador de la API de Socket.io.

Figura 30
TC-15-16



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Pruebas E2e

Cypress. Cypress es un framework de pruebas de extremo a extremo (E2E) basado en JavaScript para aplicaciones web. Se utiliza principalmente para automatizar pruebas en navegadores y verificar que una aplicación funciona como se espera.

(Cypress, 2025)

Instalar. Instala Cypress como dependencia de desarrollo:

npm install cypress --save-dev.

Verificar La Instalación. Una vez instalado, puedes abrir Cypress ejecutando:

npx cypress open

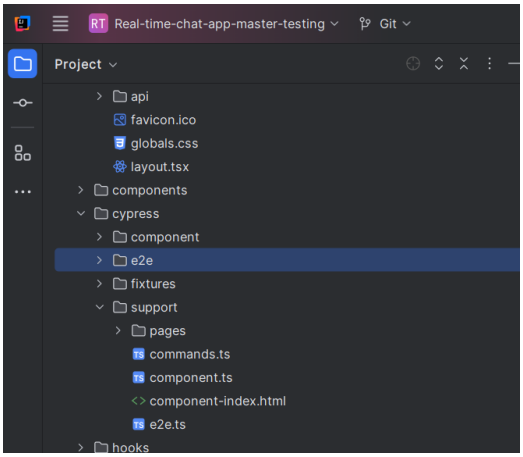
Esto abrirá la interfaz de Cypress donde se puede seleccionar las pruebas a ejecutar.

Configurar Cypress. Crea la estructura de directorios de Cypress: Si no se generaron automáticamente, crear las carpetas necesarias:

La Figura 31 muestra la estructura del proyecto orientado a pruebas E2E (end-to-end)

utilizando la herramienta Cypress.

Figura 31
Carpeta Support



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Configurar El Archivo Cypress.Config.Js. La Figura 32 presenta la configuración principal de Cypress en el archivo cypress.config.ts. Se muestran dos secciones: una dedicada a pruebas end-to-end (E2E) y otra a pruebas de componentes.

Figura 32
Cypress.config.js

```
import { defineConfig } from "cypress";

export default defineConfig({
  e2e: {
    baseUrl: "http://localhost:3000", // Cambia la URL si tu aplicación corre en otro puerto
    setupNodeEvents(on, config) {
      // Puedes agregar aquí los listeners de eventos si los necesitas
    },
    specPattern: "cypress/e2e/**/*.spec.js", // Ubicación de tus archivos de pruebas
    supportFile: false, // Opción para no usar un archivo de soporte, ya que no lo estamos configurando
  },

  component: {
    devServer: {
      framework: "next", // Framework de tu aplicación (Next.js)
      bundler: "webpack", // Empaquetador de tu aplicación
    },
    specPattern: "cypress/component/**/*.cy.tsx", // Ubicación de tus archivos de pruebas de componentes
  },
});
```


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

El archivo cypress.config.js contiene la URL base de la aplicación:

Iniciar La Aplicación

Ejecuta el servidor con el comando npm run dev.

Ejecutar Pruebas

Abre Cypress y corre los tests con npx cypress open.

Archivo Tsconfig. Json. Este archivo configura TypeScript para trabajar con Cypress, Next.js y un entorno de pruebas, asegurando que el código esté validado correctamente, que las bibliotecas adecuadas estén disponibles y que las pruebas de Cypress puedan escribirse de manera eficiente.

La Figura 33 muestra una sección del archivo tsconfig. json, que define las opciones del compilador TypeScript utilizadas en el proyecto.

Figura 33
Tsconfig.json

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba De Inicio De Sesión De Usuario Y Redirección A Github

Ruta de acceso:

cypress/e2e/githubLogin.spec.js

cypress/support/pages/githubAuthFlow.js

La Tabla 11 describe paso a paso el flujo de prueba para validar el proceso de inicio de sesión en una aplicación utilizando Cypress.

Tabla 11
E2e Github sesion

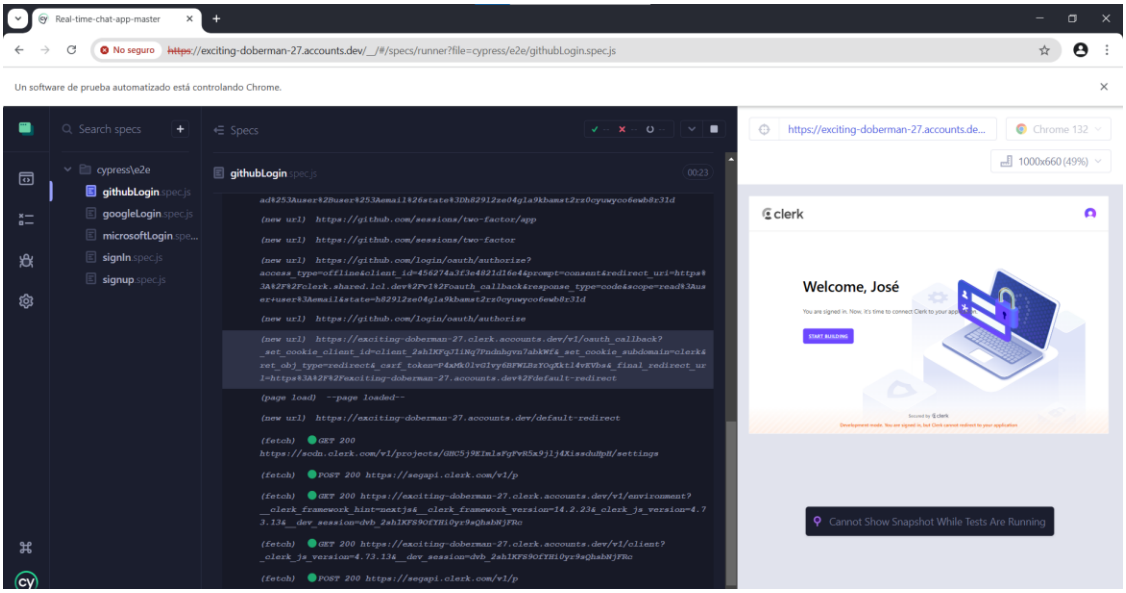
Pasos	Descripción	Acción esperada	Resultado esperado
1	Interceptar solicitud de inicio de sesión	Simulación de respuesta de la API con código 200 y un session_id falso	La API devuelve un session_id falso
2	Visitar la página de inicio de sesión	Acceder a https://exciting-doberman-27.accounts.dev/sign-in	Se muestra la página de inicio de sesión
3	Simular el flujo de inicio de sesión con GitHub	Clic en el botón "Sign in with GitHub	Se redirige a github.com/login/oauth/authorize
4	Interceptar la solicitud de autorización de GitHub	Esperar la redirección a GitHub	La URL debe incluir github.com/login/oauth/authorize
5	Introducir credenciales de GitHub	Ingresar email y contraseña	Campos de login completados correctamente
6	Clic en "Continuar" en GitHub	Confirmar autenticación en GitHub	El proceso continuo
7	Simular autenticación con Microsoft	Clic en "Sign in with Microsoft	Se redirige a la autenticación de Microsoft
8	Interceptar solicitud de inicio de sesión en la API	Esperar respuesta de POST /v1/client/sign_ins	Respuesta con session_id falso

9	Establecer cookie de sesión	Configurar cookie __session con session_id falso	Cookie creada correctamente
10	Redirigir a la aplicación	Visitar http://localhost:3000/	URL de la aplicación cargada correctamente
11	Verificar mensaje de bienvenida	Buscar texto "Welcome back" en la página	El mensaje "Welcome back" es visible.

Nota. La tabla describe los pasos del flujo de prueba de inicio de sesión mediante GitHub y Microsoft, validando la autenticación, redirecciones y establecimiento de sesión con una cookie simulada.
Fuente. Autoría propia.

La Figura 34 muestra la ejecución en tiempo real de una prueba automatizada utilizando Cypress, específicamente el archivo githubLogin.spec.js.

Figura 34
Github login



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba De Inicio De Sesión Con Cuenta Google

Ruta de acceso:

cypress/e2e/googleLogin.spec.js

cypress/support/pages/googleAuthFlow.js

Esta tabla describe los pasos detallados para simular y validar el flujo de autenticación de un usuario a través de Google, utilizando pruebas automatizadas con Cypress.

Tabla 12
E2e Google sesión

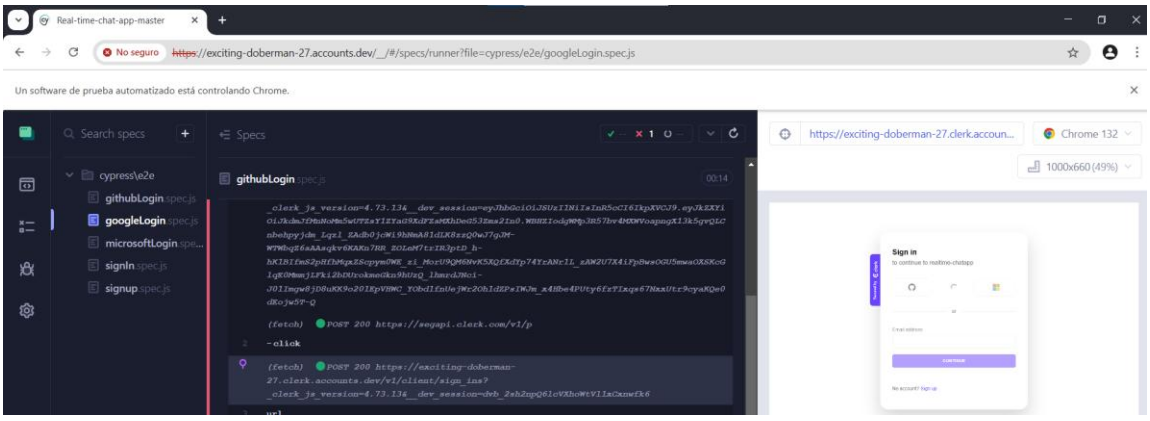
Pasos	Descripción	Acción esperada	Resultado esperado
1	El usuario navega a la página de autenticación	El usuario accede a la URL de inicio de sesión	La página de inicio de sesión se carga correctamente
2	Simular una sesión exitosa interceptando la solicitud POST	La aplicación envía la solicitud de inicio de sesión	La respuesta devuelve un session_id falso
3	El usuario selecciona la opción de autenticación con Google	La página redirige a la autenticación de Google	La URL de Google se muestra correctamente
4	Se valida que la URL incluye "https://accounts.google.com/o/oauth2/auth/oauthchooseaccount"	La URL debe contener los parámetros de autenticación esperados	La URL coincide con el formato esperado
5	El usuario elige la cuenta de Google para autenticarse	La aplicación permite la selección de la cuenta	Se muestra la pantalla de ingreso de credenciales
6	Se ingresa un correo válido en el campo de email	El usuario introduce su dirección de correo	El campo de email acepta la entrada
7	Se confirma el correo electrónico ingresado	El usuario presiona el botón "Continuar"	La página avanza a la siguiente etapa
8	Se introduce la contraseña de la cuenta de Google	El usuario escribe su contraseña en el campo correspondiente	El campo de contraseña es visible y acepta la entrada
9	Se confirma la contraseña ingresada	El usuario presiona el botón "Continuar"	La autenticación se procesa

10	Se espera la respuesta de la solicitud de inicio de sesión	La aplicación debe manejar correctamente la respuesta de autenticación	La sesión se establece correctamente
11	Se establece una cookie de sesión simulada	La aplicación configura una cookie de sesión	La cookie se almacena en el navegador
12	Se navega automáticamente a la página principal tras la autenticación	La aplicación redirige al usuario a la página principal	La URL debe contener "http://localhost:3000/"
13	Se valida que el mensaje "Welcome back" está visible en la interfaz	La aplicación muestra el mensaje de bienvenida	El usuario ve el mensaje en pantalla

Nota. La tabla presenta los pasos para simular el flujo completo de autenticación con Google, incluyendo validaciones de URL, entrada de credenciales, establecimiento de sesión y redirección final.
Fuente. Autoría propia.

La figura 35 muestra la ejecución del archivo googleLogin.spec.js en Cypress, donde se simula el flujo de autenticación a través de Google.

Figura 35
Google Login



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba De Inicio De Sesión Microsoft

Ruta de acceso:

cypress/e2e/microsoftLogin.spec.js

cypress/support/pages/microsoftAuthFlow.js

La Tabla 13 presenta un resumen del proceso automatizado de autenticación mediante Microsoft utilizando Cypress.

Tabla 13
E2e Microsoft Sesión

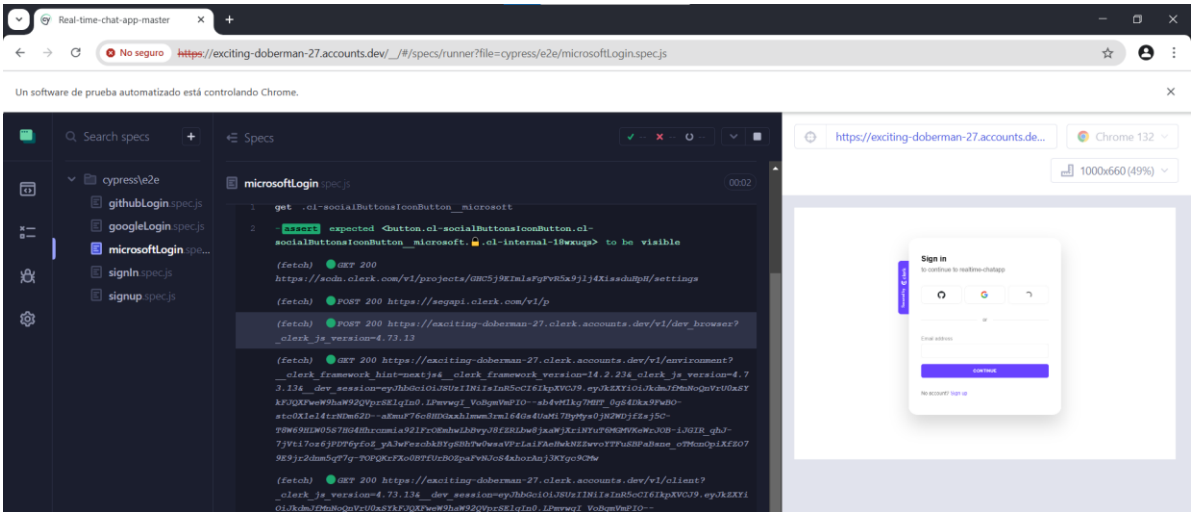
Pasos	Descripción	Acción esperada	Resultado esperado
1	Interceptar la solicitud de inicio de sesión.	Cypress intercepta la solicitud POST al endpoint de inicio de sesión	La solicitud es interceptada y Cypress devuelve una respuesta simulada
2	Visitar la página de inicio de sesión.	Cypress navega a la URL de inicio de sesión	La página de inicio de sesión se carga correctamente
3	Hacer clic en el botón de inicio de sesión con Microsoft.	Cypress simula el clic en el botón de Microsoft	Se redirige a la página de autenticación de Microsoft
4	Ingresar el correo electrónico.	Cypress introduce la dirección de correo en el campo de email	El email es ingresado correctamente en el campo
5	Clic en continuar.	Cypress simula el clic en el botón de continuar	Se muestra el campo de contraseña
6	Ingresar la contraseña.	Cypress introduce la contraseña en el campo de password	La contraseña es ingresada correctamente en el campo
7	Clic en continuar.	Cypress simula el clic en el botón de continuar	Se envía la solicitud de autenticación
8	Esperar la respuesta de la autenticación.	Cypress espera la respuesta interceptada de la solicitud de inicio de sesión	La respuesta simulada Es recibida con un ID de sesión

9	Configurar la cookie de sesión.	Cypress almacena una cookie con el ID de sesión	La cookie es almacenada exitosamente
10	Redirigir a la página de inicio de la aplicación.	Cypress navega a la URL de la aplicación	La aplicación se carga correctamente

Nota. La tabla describe los pasos automatizados con Cypress para simular el proceso completo de inicio de sesión utilizando autenticación con Microsoft, desde la interceptación de la solicitud hasta la redirección final con sesión activa.
Fuente. Autoría propia.

La Figura 36 muestra la ejecución de una prueba automatizada utilizando Cypress para validar el flujo de autenticación mediante una cuenta de Microsoft.

Figura 36
Microsoft Login



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Inicio De Sesión Con Correo

Ruta de acceso:

cypress/e2e/signIn.spec.js

cypress/support/pages/signInPage.js

La Tabla 14 describe el procedimiento para validar, mediante pruebas automatizadas con Cypress, el flujo de autenticación mediante correo electrónico en la aplicación.

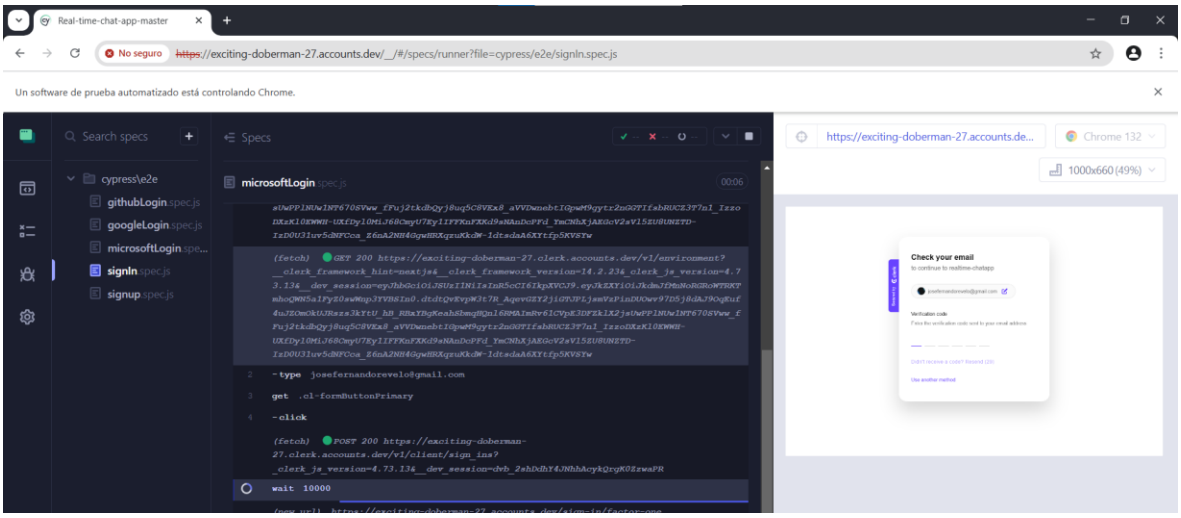
Tabla 14
E2e Correo Sesión

Pasos	Descripción	Acción esperada	Resultado esperado
1	Interceptar solicitud de inicio de sesión	Simular una respuesta de inicio de sesión exitosa con un ID de sesión falso	La solicitud POST a **/v1/client/sign_ins es interceptada y devuelve session_id: 'fake-session-id'
2	Visitar la página de inicio de sesión	Cargar la página de autenticación	Se muestra la interfaz de inicio de sesión
3	Ingresar correo electrónico	El usuario ingresa su dirección de correo	El campo de correo se completa correctamente
4	Hacer clic en "Continuar"	Se envía la solicitud de autenticación	Se procesa el correo y se espera la solicitud del código de verificación
5	Esperar la llegada del código de verificación	Simular un tiempo de espera de 10 segundos	Se espera correctamente antes de verificar la existencia del campo de código
6	Verificar la presencia del campo "Código de verificación"	Revisar si el campo de código de verificación aparece en la interfaz	El campo de código de verificación está visible
7	Finalizar prueba	Registrar un mensaje de log indicando que la prueba ha sido completada	El test finaliza exitosamente y se confirma que el código de verificación ha sido solicitado

Nota. Prueba automatizada que simula el inicio de sesión con código, interceptando respuestas y validando la UI sin autenticación real.
Fuente. Autoría propia.

La Figura 37 muestra la ejecución de una prueba automatizada utilizando Cypress para validar el flujo de autenticación mediante una cuenta de correo electrónico.

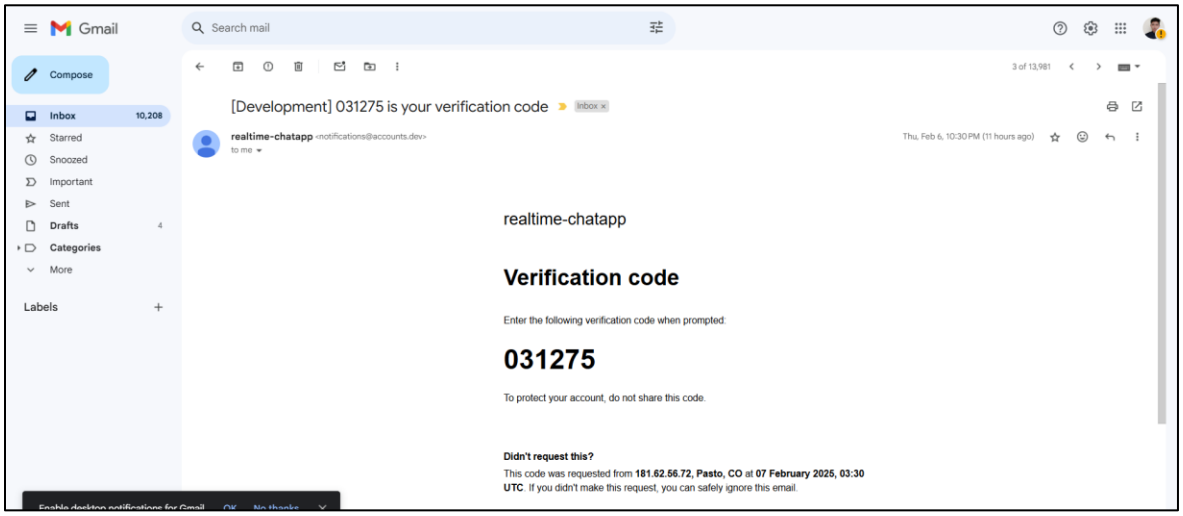
Figura 37
Inicio Sesión correo



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 38 muestra el código de verificación generado y enviado a la cuenta de correo electrónico.

Figura 38
Código de verificación



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba De Registro De Cuenta

Ruta de acceso:

cypress/e2e/signup.spec.js

cypress/support/pages/registrationForm.js

La siguiente tabla describe detalladamente el proceso de prueba de flujo de autenticación y registro en una aplicación web, incluyendo cada paso que un usuario típico podría realizar al interactuar con el sistema.

Tabla 15
Registro de cuenta

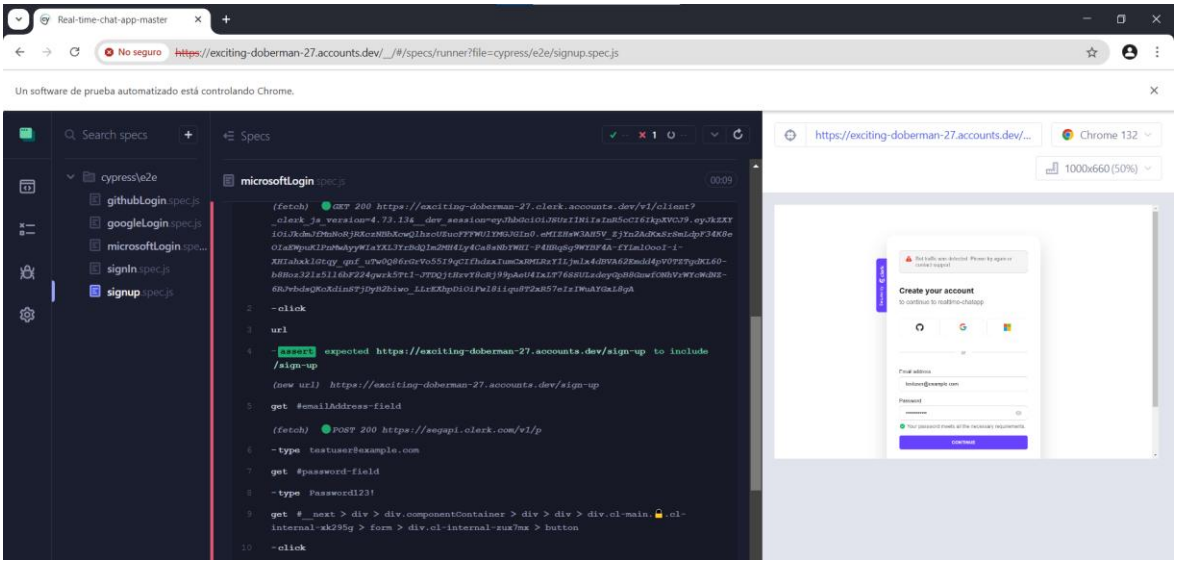
Pasos	Descripción	Acción esperada	Resultado esperado
1	Interceptar solicitud de inicio de sesión	Simular una respuesta de inicio de sesión exitosa con un ID de sesión falso	La solicitud POST a **/v1/client/sign_ins es interceptada y devuelve session_id: 'fake-session-id'
2	Visitar la página de inicio de sesión	Cargar la página de autenticación	Se muestra la interfaz de inicio de sesión
3	Ingresar correo electrónico	El usuario ingresa su dirección de correo	El campo de correo se completa correctamente
4	Hacer clic en "Continuar"	Se envía la solicitud de autenticación	Se procesa el correo y se espera la solicitud del código de verificación
5	Esperar la llegada del código de verificación	Simular un tiempo de espera de 10 segundos	Se espera correctamente antes de verificar la existencia del campo de código
6	Verificar la presencia del campo "Código de verificación"	Revisar si el campo de código de verificación aparece en la interfaz	El campo de código de verificación está visible
7	Finalizar prueba	Registrar un mensaje de log indicando que la prueba ha sido completada	El test finaliza exitosamente y se confirma que el código de verificación ha sido solicitado
8	Hacer clic en el enlace de registro	El usuario hace clic en el enlace para ir a la página de registro	La página de registro se carga correctamente
9	Ingresar correo electrónico y	El usuario introduce su	Los campos de entrada

	contraseña	correo y contraseña válidos	aceptan correctamente los valores
10	Hacer clic en "Continuar" en el formulario de registro	Se envía la información del usuario para el registro	Se procesa la solicitud y se redirige a la página de bienvenida

Nota. Esta secuencia de pasos verifica todo el flujo de autenticación, desde la interceptación de la solicitud de inicio de sesión hasta el registro exitoso de un nuevo usuario.
Fuente. Autoría propia.

La Figura 39 muestra la ejecución de una prueba automatizada utilizando Cypress para validar el flujo de registro de cuenta.

Figura 39
Registro cuenta



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Pruebas De Integración

Prueba Componente Chat Input

Ruta de acceso: cypress/component/chat/chat-input.cy.tsx

La siguiente tabla presenta una serie de pruebas diseñadas para validar el correcto funcionamiento del componente ChatInput dentro de una aplicación web.

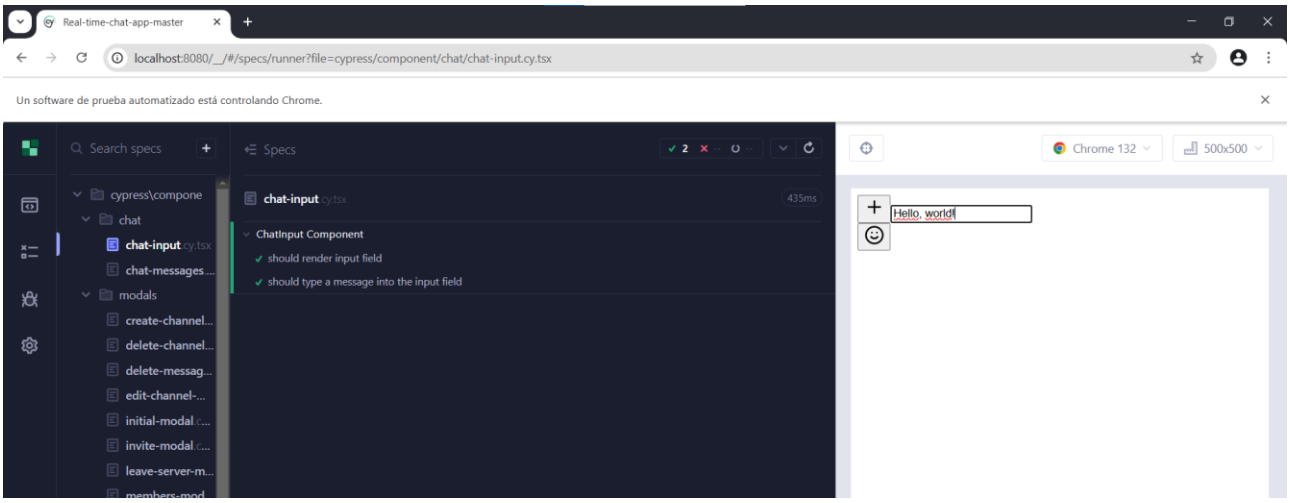
Tabla 16
TI-001-002

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-001	Renderizado del campo de entrada	Verificar que el componente ChatInput muestra el campo de entrada.	La aplicación debe estar en ejecución con el ChatInput montado en el contexto de la app.	1. Renderizar el componente ChatInput. 2. Verificar la visibilidad del campo de entrada.	El campo de entrada debe ser visible en la interfaz.	Aprobado
TI-002	Entrada de texto en el campo	Verificar que el usuario puede escribir un mensaje en el campo de entrada.	La aplicación debe estar en ejecución con el ChatInput montado en el contexto de la app.	1. Renderizar el componente ChatInput. 2. ¡Escribir “Hello, world!" en el campo de entrada. 3. Verificar que el valor del campo de entrada coincida con el texto ingresado.	¡El campo de entrada debe contener el mensaje “Hello, world!".	Aprobado

Nota. Las pruebas TI-001 y TI-002 se realizaron en un entorno controlado con la aplicación en ejecución y el componente ChatInput debidamente montado.
Fuente. Autoría propia.

La Figura 40 muestra la ejecución de una prueba automatizada utilizando Cypress para validar el ingreso correcto de texto en la entrada del chat.

Figura 40
Chat input



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Chat Messages

Ruta de acceso: cypress/component/chat/chat-messages.cy.tsx

Esta tabla documenta una prueba específica dirigida a verificar el comportamiento del componente ChatMessages cuando se encuentra en estado de carga, es decir, "pending".

Tabla 17
TI-003

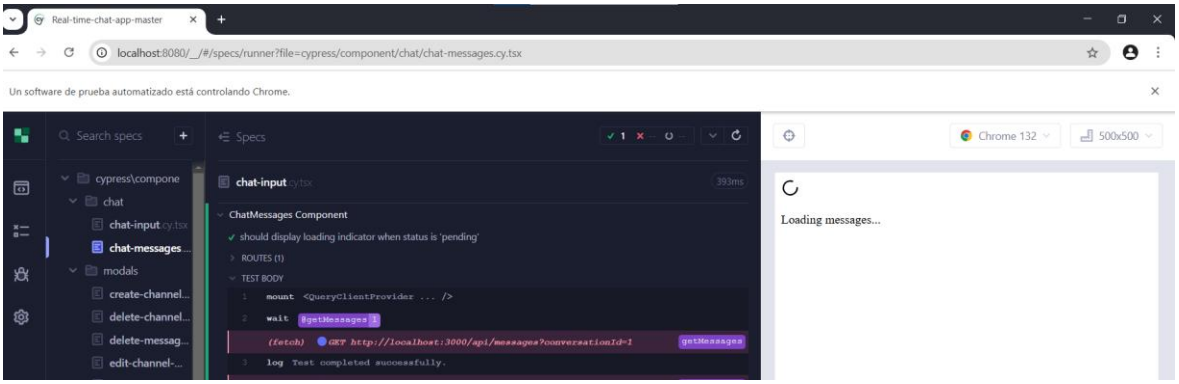
ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
I-003	Indicador de carga en estado "pending"	Verificar que el componente ChatMessages muestra un indicador de carga cuando el estado es pending.	La aplicación debe estar en ejecución y debe poder interceptar la solicitud GET a /api/messages?conversationId	1. Interceptar la solicitud GET a /api/messages?conversationId=1. 2. Montar el component	Componente debe mostrar un indicador de carga mientras la solicitud está en estado pending.	Aprobado

	=1.	e ChatMessages. 3. Esperar a que la solicitud sea interceptada. 4. Verificar si el indicador de carga se muestra correctamente.

Nota. La prueba TI-003 se realizó interceptando de manera controlada la solicitud a /api/messages?conversationId=1 para simular el estado pending.
Fuente. Autoría propia.

La figura 41 muestra el resultado del test del componente ChatMessages cuando se encuentra en estado de carga, es decir, "pending".

Figura 41
Chat messages



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modals: Prueba Componente Create Channel Modal

Ruta de acceso: cypress/component/modals/create-channel-modal.cy.tsx

La siguiente tabla documenta un conjunto de pruebas enfocadas en validar el correcto funcionamiento del componente CreateChannelModal, el cual permite a los usuarios crear un nuevo canal dentro de la aplicación.

Tabla 18
TI-004-007

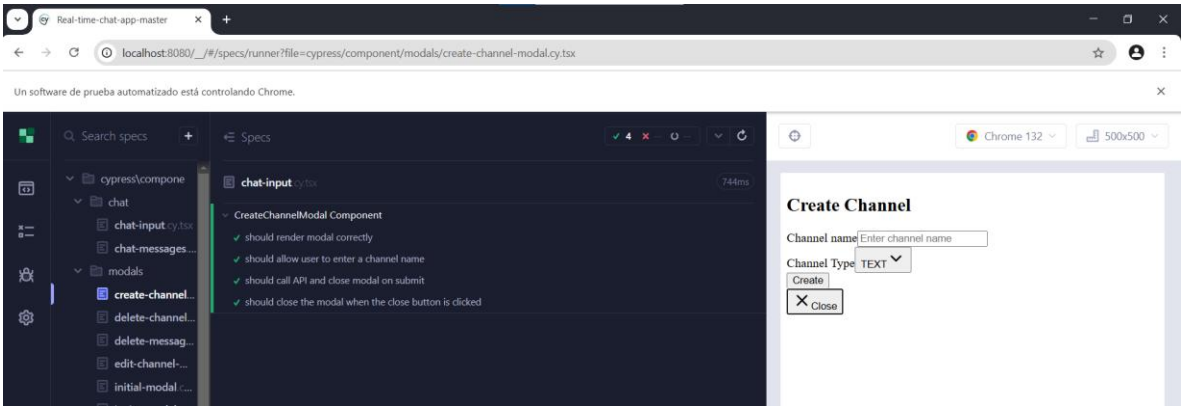
ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-004	Renderizado del modal	Verificar que el modal CreateChannelModal se renderiza correctamente.	La aplicación debe estar en ejecución y el estado del modal debe estar configurado como isOpen: true.	1. Montar el componente CreateChannelModal. 2. Verificar que el texto "Create Channel" es visible.	El modal debe mostrarse con el texto "Create Channel".	Aprobado
TI-005	Ingreso del nombre del canal	Verificar que el usuario puede ingresar un nombre en el campo de entrada.	La aplicación debe estar en ejecución y el modal debe	1. Montar el componente CreateChannelModal. 2. Escribir "My Channel" en el campo de entrada.	El campo de entrada debe aceptar el texto y mostrar el valor	Aprobado

			estar abierto.	3. Verificar que el valor ingresado es "My Channel".	ingresado.	
TI-006	Llamada a la API y cierre del modal	Verificar que se realiza una solicitud a la API y que el modal se cierra al enviar el formulario.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente CreateChannel Modal. 2. Escribir "New Channel" en el campo de entrada. 3. Hacer clic en el botón "Create". 4. Verificar que se llama a axios.post. 5. Verificar que el modal se cierra.	La API debe ser llamada y el modal debe cerrarse correctamente.	Aprobado
TI-007	Cierre del modal con botón de cierre	Verificar que el modal se cierra al hacer clic en el botón de cierre.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente CreateChannel Modal. 2. Hacer clic en el botón de cierre (.absolute.right-4.top-4.rounded-sm).	El modal debe cerrarse al hacer clic en el botón de cierre.	Aprobado

Nota. Las pruebas TI-004 a TI-007 están enfocadas en validar el comportamiento del componente CreateChannelModal en distintas situaciones de uso.
Fuente. Autoría propia.

La Figura 42 muestra la interfaz visual del componente CreateChannelModal, el cual permite a los usuarios crear un nuevo canal dentro de la aplicación.

Figura 42
Create channel modal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Delete Channel Modal

Ruta de acceso: cypress/component/modals/delete-channel-modal.cy.tsx

La siguiente tabla presenta una serie de pruebas dirigidas a validar el comportamiento del componente DeleteChannelModal, responsable de gestionar la eliminación de canales en la aplicación.

Tabla 19
TI-008-011

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-008	Renderización del modal	Verificar que el modal DeleteChannelModal se renderiza correctamente.	La aplicación debe estar en ejecución y el estado del	1. Montar el componente DeleteChannelModal. 2. Verificar que el texto	El modal debe mostrarse con el título y el nombre del canal.	Aprobado

			modal debe estar configurado como isOpen: true.	"Delete Channel" es visible. 3. Verificar que el nombre del canal aparece en el modal.		
TI-009	Confirmar eliminación del canal	Verificar que al hacer clic en "Confirm" se llama a la API y se redirige al usuario.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente DeleteChannelModal. 2. Interceptar la solicitud DELETE a /api/channels/*. 3. Hacer clic en el botón "Confirm". 4. Verificar que se llama a la API DELETE. 5. Verificar que el router redirige a	La API debe ser llamada y el usuario debe ser redirigido.	Aprobado

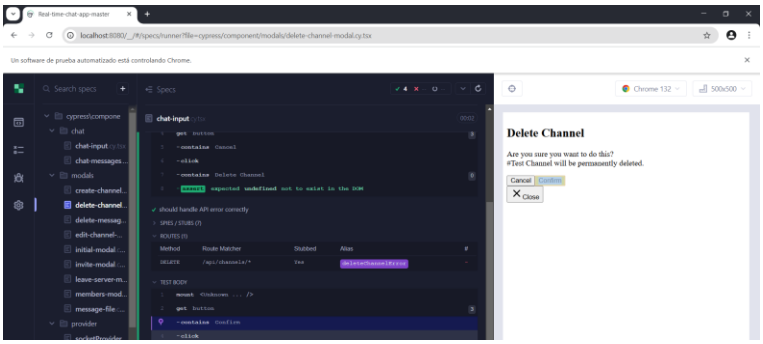
				/servers/1.		
TI-010	Cancelar eliminación del canal	Verificar que el modal se cierra al hacer clic en "Cancel".	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente DeleteChannelModal. 2. Hacer clic en el botón "Cancel". 3. Verificar que el modal desaparece.	El modal debe cerrarse correctamente.	Aprobado
TI-011	Manejo de error en la API	Verificar que se maneja correctamente un error al eliminar un canal.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente DeleteChannelModal. 2. Simular un error 500 en la API DELETE. 3. Hacer clic en "Confirm". 4. Verificar que el error es manejado correctame	El error debe manejarse y mostrarse un mensaje adecuado.	Aprobado

nte.

Nota. Las pruebas TI-008 a TI-011 verifican el correcto funcionamiento del modal. DeleteChannelModal, desde su renderización inicial hasta la interacción del usuario en los casos de confirmación, cancelación y manejo de errores.
Fuente. Autoría propia.

La Figura 43 ilustra el componente DeleteChannelModal, una interfaz que permite al usuario confirmar o cancelar la eliminación de un canal específico.

Figura 43
Delete channel modal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Delete Message Modal

Ruta de acceso: cypress/component/modals/delete-message-modal.cy.tsx

La Tabla 20 detalla los casos de prueba diseñados para validar el comportamiento del componente DeleteMessageModal, el cual permite a los usuarios confirmar o cancelar la eliminación de un mensaje específico dentro de la aplicación.

Tabla 20
TI-012-015

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-012	Renderización del modal	Verificar que el modal DeleteMessageModal se renderiza	La aplicación debe estar en ejecución y	1. Montar el componente DeleteMessageModal. 2. Verificar que	El modal debe mostrarse con el título y el	Aprobado

		correctamente.	el estado del modal debe estar configurado como isOpen: true.	el texto "Delete Message" es visible. 3. Verificar que el mensaje de confirmación aparece.	mensaje de confirmación.	
TI-013	Cancelar eliminación del mensaje	Verificar que al hacer clic en "Cancel" el modal se cierra.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente DeleteMessageModal. 2. Hacer clic en el botón "Cancel". 3. Verificar que la función onClose ha sido llamada.	El modal debe cerrarse correctamente al hacer clic en "Cancel".	Aprobado
TI-014	Confirmar eliminación del mensaje	Verificar que se realiza una solicitud DELETE cuando se confirma la eliminación.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente DeleteMessageModal. 2. Interceptar la solicitud DELETE a https://api.example.com/messages/1. 3. Hacer clic en el botón "Confirm". 4. Verificar que se envía la	La API debe ser llamada correctamente con la solicitud DELETE.	Aprobado

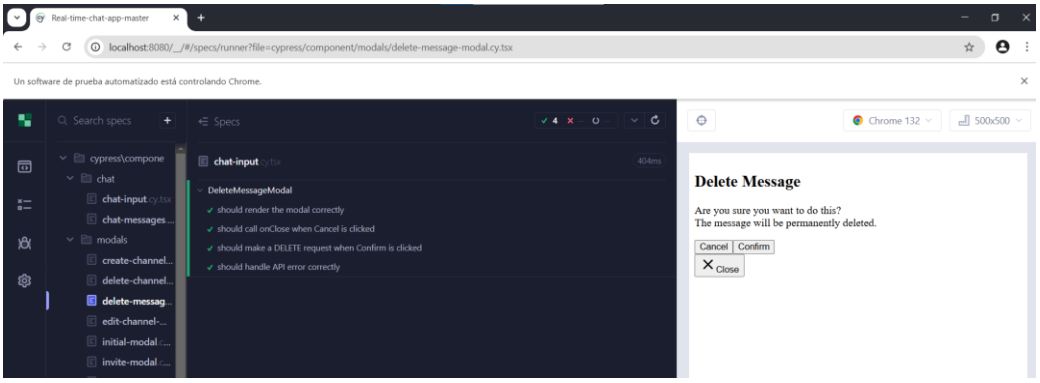
				solicitud DELETE.		
TI-015	Manejo de error en la API	Verificar que se maneja correctamente un error al eliminar un mensaje.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente DeleteMessageModal. 2. Simular un error 500 en la API DELETE. 3. Hacer clic en "Confirm". 4. Verificar que el error es manejado correctamente.	El error debe manejarse y mostrarse un mensaje adecuado.	Aprobado

Nota. Las pruebas TI-012 a TI-015 validan el correcto comportamiento del modal. DeleteMessageModal frente a diferentes acciones del usuario y estados del sistema.
Fuente. Autoría propia.

La Figura 44 muestra la interfaz del componente DeleteMessageModal, utilizado para confirmar la eliminación de un mensaje dentro del sistema de mensajería de la aplicación.

Figura 44

Delete message modal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Edit Channel Modal

Ruta de acceso: cypress/component/modals/edit-channel-modal.cy.tsx

La Tabla 21 presenta los casos de prueba correspondientes al componente

EditChannelModal, el cual permite a los usuarios modificar las propiedades de un canal existente, como su nombre o tipo.

Tabla 21
TI-016-019

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-016	Renderización del modal	Verificar que el modal EditChannel Modal se renderiza correctamente.	La aplicación debe estar en ejecución y el estado del modal debe estar configurado como isOpen: true.	1. Montar el componente EditChannel Modal. 2. Verificar que el texto "Edit Channel" es visible. 3. Verificar que el campo de nombre contiene "Test Channel". 4. Verificar que el tipo de canal está seleccionado correctamente.	El modal debe mostrarse con el título y los valores correctos.	Aprobado
TI-017	Guardar cambios en el canal	Verificar que se realiza una solicitud PATCH cuando se guardan los	La aplicación debe estar en ejecución y el modal	1. Montar el componente EditChannel Modal. 2. Interceptar la solicitud	La API debe recibir correctamente la solicitud PATCH.	Aprobado

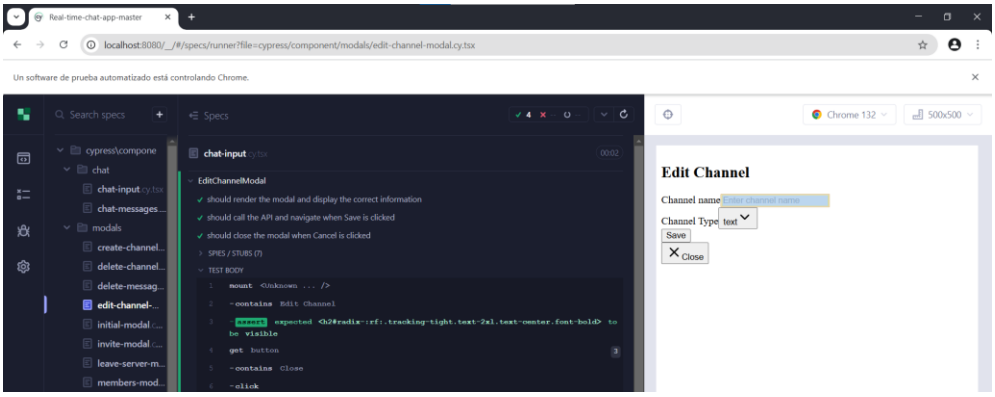
		cambios.	debe estar abierto.	PATCH a /api/channels /*?serverId= *. 3. Modificar el nombre del canal. 4. Hacer clic en el botón "Save".		
TI-018	Cerrar modal al cancelar	Verificar que al hacer clic en "Cancel" el modal se cierra.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente EditChannel Modal. 2. Hacer clic en el botón "Cancel". 3. Verificar que el modal se cierra correctament e.	El modal debe cerrarse correctamen te al hacer clic en "Cancel".	Aprobado
TI-019	Manejo de error en la API	Verificar que se maneja correctamente un error al actualizar un canal.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente EditChannel Modal. 2. Simular un error 500 en la API PATCH. 3. Modificar el nombre del canal. 4. Hacer clic	El error debe manejarse y mostrarse un mensaje adecuado.	Aprobado

	en "Save". 5. Verificar que el error se maneja correctamente.

Nota. Las pruebas TI-016 a TI-019 evalúan el comportamiento del modal EditChannelModal en distintos escenarios de uso.
Fuente. Autoría propia.

La Figura 45 muestra el componente visual EditChannelModal, el cual proporciona una interfaz para que los usuarios puedan modificar los detalles de un canal existente dentro de la aplicación.

Figura 45
Edit channel modal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Initial Modal

Ruta de acceso: cypress/component/modals/initial-modal.cy.tsx

La Tabla 22 presenta los casos de prueba correspondientes al componente InitialModal, encargado de guiar al usuario en la creación inicial de un servidor dentro de la aplicación.

Tabla 22
TI-020-021

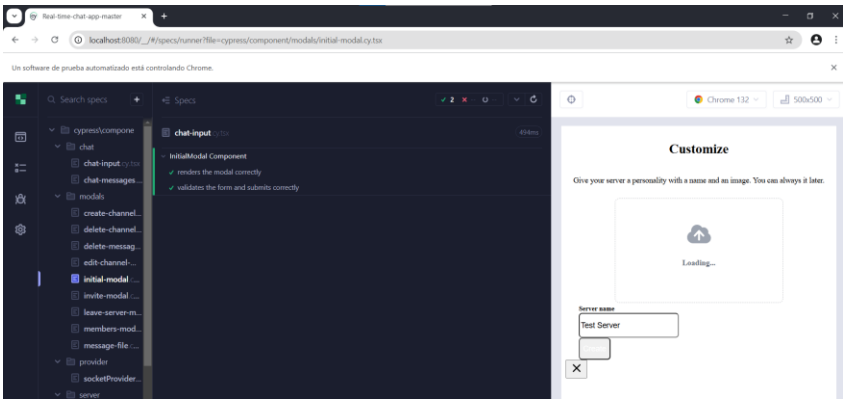
ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado

TI-020	Renderización del modal	Verificar que el modal InitialModal se renderiza correctamente. La aplicación debe estar en	La aplicación debe estar en ejecución.	1. Montar el componente InitialModal. 2. Verificar que el modal está presente. 3. Verificar que el texto "Customize" es visible.	El modal debe mostrarse correctamente con el texto esperado.	Aprobado
TI-021	Validación y envío del formulario	Verificar que el formulario valida la entrada y envía una solicitud POST correctamente.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente InitialModal. 2. Ingresar "Test Server" en el campo de entrada. 3. Hacer clic en el botón "Create". 4. Verificar que se hace la solicitud POST a /api/servers.	La API debe recibir correctamente la solicitud POST.	Aprobado

Nota. Las pruebas TI-020 y TI-021 se enfocan en la validación del comportamiento del componente InitialModal.
Fuente. Autoría propia.

La Figura 46 muestra la interfaz gráfica del componente InitialModal, la cual se presenta al usuario durante el proceso de creación de un nuevo servidor.

Figura 46
Initial modal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Invite Modal

Ruta de acceso: cypress/component/modals/invite-modal.cy.tsx

La Tabla 23 presenta los casos de prueba realizados al componente InviteModal, el cual permite a los usuarios compartir enlaces de invitación a un servidor.

Tabla 23
TI-022-025

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-022	Renderización del modal	Verificar que el modal InviteModal se renderiza correctamente.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente InviteModal. 2. Verificar que el modal se muestra correctamente.	El modal debe mostrarse sin errores.	Aprobado
TI-023	Visualización del	Verificar que el	La aplicació	1. Montar el componente	El input debe mostrar el	Aprobado

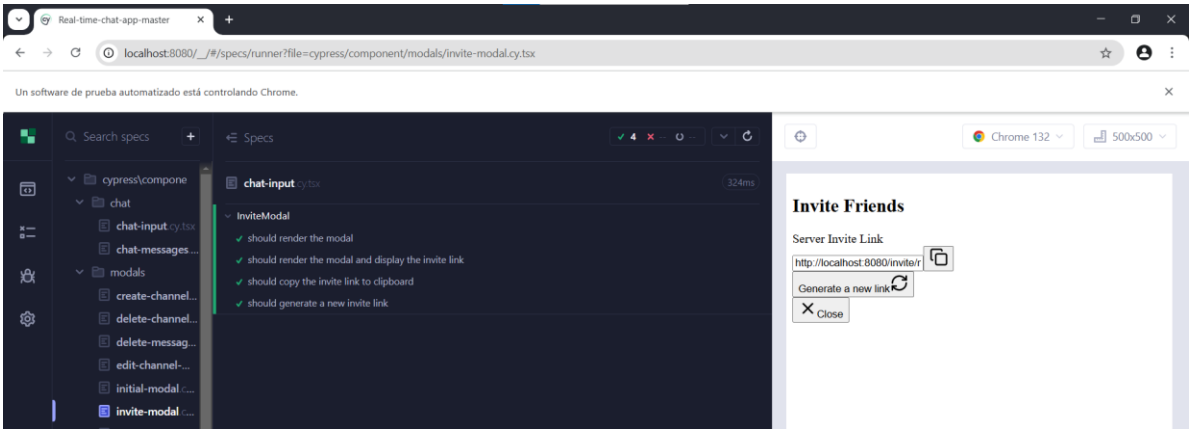
	enlace de invitación	enlace de invitación se muestra correctamente en el input.	n debe estar en ejecución y el modal debe estar abierto.	InviteModal. 2. Verificar que el input contiene la URL esperada.	enlace de invitación "http://localhost:8080/invite/test-invite-code".	
TI-024	Copiar enlace de invitación	Verificar que el enlace de invitación se copia correctamente al portapapeles.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente InviteModal. 2. Hacer clic en el icono de copiar. 3. Verificar que el enlace se copia al portapapeles.	El enlace debe copiarse correctamente al portapapeles.	Aprobado
TI-025	Generar nuevo enlace de invitación	Verificar que el sistema genera un nuevo enlace de invitación cuando se solicita.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente InviteModal. 2. Hacer clic en el botón "Generate a new link". 3. Verificar que el input muestra un nuevo enlace de invitación.	El input debe mostrar el nuevo enlace "http://localhost:8080/invite/new-invite-code".	Aprobado

Nota. Estas pruebas aseguran la correcta funcionalidad del componente InviteModal.
Fuente. Autoría propia.

La Figura 47 muestra la interfaz del modal de invitación (InviteModal), el cual permite a

los usuarios generar y copiar enlaces para invitar a otros miembros a un servidor específico.

Figura 47
Invite modal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Leve Server Modal

Ruta de acceso: cypress/component/modals/leave-server-modal.cy.tsx

La siguiente tabla detalla los casos de prueba implementados para verificar el correcto funcionamiento del componente LeaveServerModal, que permite al usuario salir de un servidor.

Tabla 24
TI-026-029

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-026	Renderización del modal	Verificar que el modal LeaveServer Modal se renderiza correctamente.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente LeaveServerModal. 2. Verificar que el modal muestra el mensaje "Leave Server".	El modal debe mostrarse correctamente sin errores.	Aprobado
TI-027	Cerrar el	Verificar que	La	1. Montar el	El modal	Aprobado

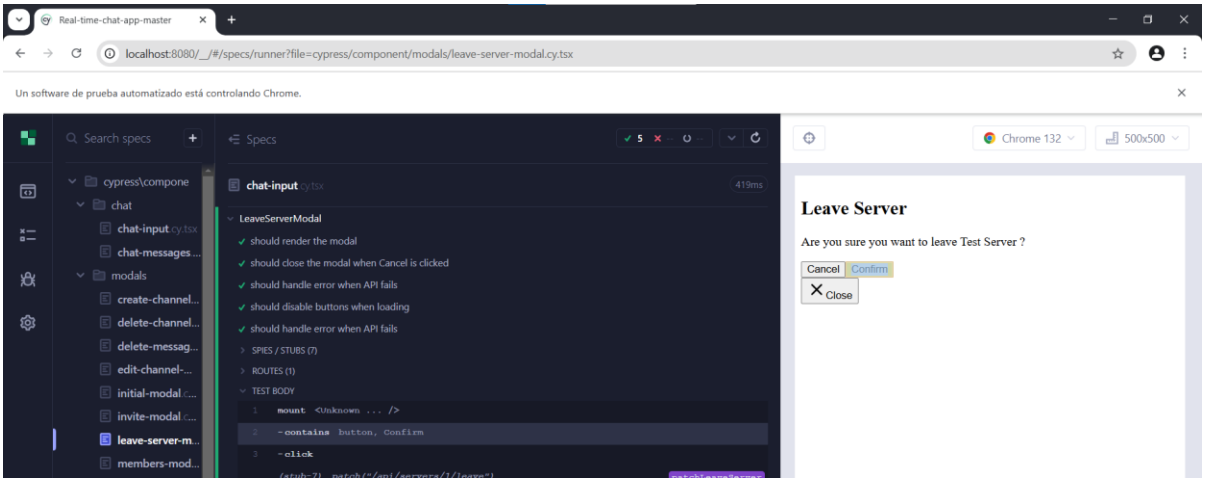
	modal al hacer clic en "Cancel"	el modal se cierra cuando el usuario hace clic en "Cancel".	aplicación debe estar en ejecución y el modal debe estar abierto.	componente LeaveServerModal. 2. Hacer clic en el botón "Cancel". 3. Verificar que la página no se recarga ni navega.	debe cerrarse sin realizar cambios.	
TI-028	Manejar error cuando la API falla	Simular un error en la API y verificar que se maneja correctamente.	La API debe responder con un error 500.	1. Montar el componente LeaveServerModal. 2. Simular un error en la llamada a la API. 3. Hacer clic en "Confirm".	Debe mostrarse un mensaje de error adecuado al usuario.	Aprobado
TI-029	Deshabilitar botones cuando está cargando	Verificar que los botones están deshabilitados mientras la API procesa la solicitud.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente LeaveServerModal. 2. Hacer clic en "Confirm". 3. Verificar que los botones están deshabilitados.	Los botones deben estar deshabilitados hasta que la API responda.	Aprobado

Nota. Estas pruebas verifican el correcto funcionamiento del componente LeaveServerModal.
Fuente. Autoría propia.

La Figura 48 muestra la interfaz visual del componente LeaveServerModal, utilizada para

confirmar la acción de abandonar un servidor.

Figura 48
Leave server modal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Members Modal

Ruta de acceso: cypress/component/modals/members-modal.cy.tsx

La Tabla 25 recopila los casos de prueba diseñados para verificar el correcto funcionamiento del componente MembersModal, el cual permite gestionar a los miembros de un servidor.

Tabla 25
TI-030-033

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-030	Renderización del modal	Verificar que el modal MembersModal se renderiza correctamente.	La aplicación debe estar en ejecución y el modal	1. Montar el componente MembersModal. 2. Verificar que el modal muestra el	El modal debe mostrarse correctamente sin	Aprobado

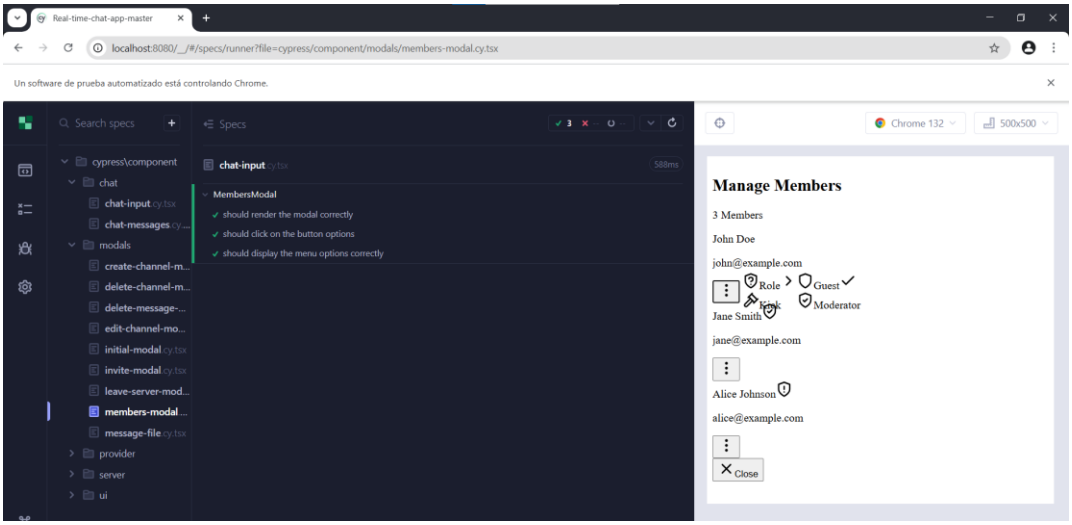
			debe estar abierto.	mensaje "Manage Members".	errores.	
TI-031	Hacer clic en el botón de opciones	Verificar que el botón de opciones es visible y se puede hacer clic en él.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente MembersModal. 2. Verificar que el botón de opciones (button[type="button"]) es visible.	El botón debe estar visible y listo para la interacción.	Aprobado
TI-032	Mostrar el menú de opciones correctamente	Verificar que el menú de opciones se muestra cuando se hace clic en el botón de más opciones.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente MembersModal. 2. Hacer clic en el ícono MoreVertical. 3. Verificar que se abre el menú de opciones.	El menú de opciones debe mostrarse correctamente al hacer clic.	Aprobado
TI-033	Abrir submenú de opciones	Verificar que se puede abrir el submenú de opciones correctam	La aplicación debe estar en ejecución y el modal debe	1. Montar el componente MembersModal. 2. Hacer clic en el ícono MoreVertical. 3. Hacer clic	El submenú debe abrirse correctamente mostrando	Aprobado

	ente.	estar abierto.	en el ícono de ChevronRight en el menú.	opciones adicionales.	

Nota. Estas pruebas se enfocan en verificar la correcta renderización y funcionalidad del componente MembersModal.
Fuente. Autoría propia.

La Figura 49 muestra la interfaz gráfica del componente MembersModal, el cual permite al usuario visualizar y gestionar los miembros asociados a un servidor.

Figura 49
Members modal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Message File Modal

Ruta de acceso: cypress/component/modals/message-file.cy.tsx

La Tabla 26 presenta los casos de prueba diseñados para validar el correcto funcionamiento del componente MessageFileModal, encargado de gestionar la carga de archivos como adjuntos en los mensajes.

Tabla 26
TI-034-037

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado

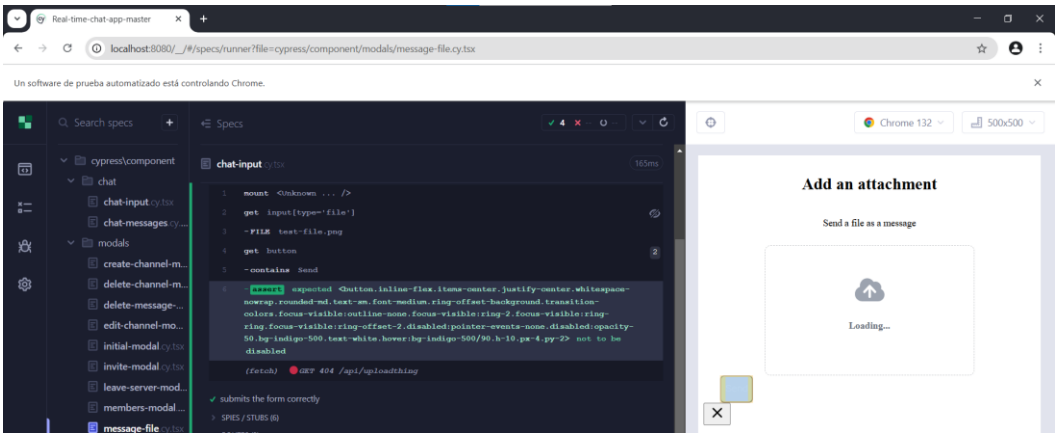
TI-034	Renderización del modal	Verificar que el modal MessageFileModal se renderiza correctamente.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente MessageFileModal. 2. Verificar que el título "Add an attachment" es visible.	El modal debe mostrarse correctamente sin errores.	Aprobado
TI-035	Verificar la descripción del modal	Verificar que la descripción dentro del modal es correcta.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente MessageFileModal. 2. Verificar que el texto "Send a file as a message" es visible.	La descripción del modal debe mostrarse correctamente.	Aprobado
TI-036	Subir archivo y habilitar botón de envío	Verificar que al subir un archivo, el botón "Send" se habilita.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente MessageFileModal. 2. Adjuntar un archivo de imagen en el input <input	El botón "Send" debe habilitarse al adjuntar un archivo válido.	Aprobado

				type='file'>. 3. Verificar que el botón "Send" está habilitado.		
TI-037	Enviar formulario correctamente	Verificar que la solicitud de envío del archivo se realice correctamente.	La aplicación debe estar en ejecución y el modal debe estar abierto.	1. Montar el componente MessageFileModal. 2. Adjuntar un archivo. 3. Interceptar la solicitud POST /api/uploadthing. 4. Verificar que la solicitud se envía con éxito.	El archivo debe enviarse correctamente sin errores.	Aprobado

Nota. Estas pruebas se centran en garantizar que el componente MessageFileModal funcione correctamente desde su renderización hasta el envío exitoso de archivos.
Fuente. Autoría propia.

La Figura 50 muestra el modal MessageFileModal, utilizado para adjuntar archivos a un mensaje dentro de la aplicación.

Figura 50
Message file modal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Provider: Prueba Socketprovider Con Socketindicator

Ruta de acceso: cypress/component/provider/socketProvider.cy.tsx

La Tabla 27 resume las pruebas realizadas para validar el comportamiento del componente SocketIndicator, el cual informa al usuario sobre el estado actual de la conexión WebSocket dentro de la aplicación.

Tabla 27
TI-038-039

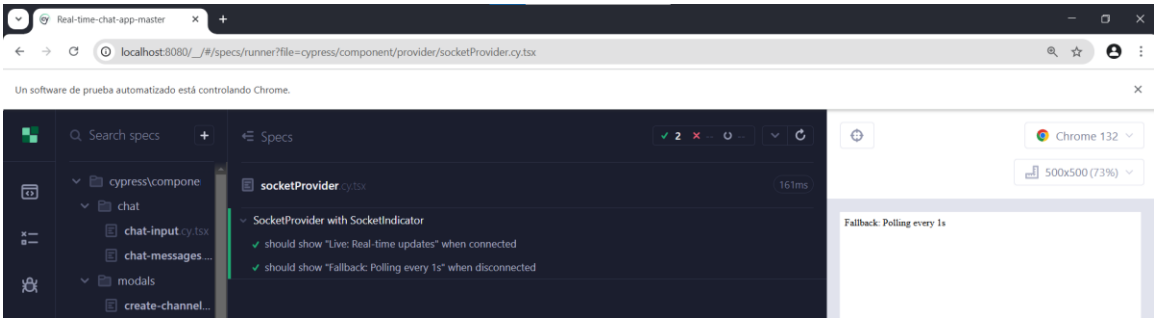
ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-038	Conexión exitosa al WebSocket	Verificar que el indicador muestra el estado "Live: Real-time updates" cuando el	La aplicación debe estar en ejecución con el servidor WebSocket habilitado.	1. Interceptar la solicitud GET /api/socket/io* para simular una conexión exitosa. 2. Montar el componente SocketProvider	1 indicador debe mostrar "Live: Real-time updates".	Aprobado

		WebSocket está conectado.		con SocketIndicator. 3. Esperar a que la conexión se establezca.		
TI-039	Estado de fallback al desconectarse del WebSocket	Verificar que el indicador muestra "Fallback: Polling every 1s" cuando no hay conexión WebSocket.	La aplicación debe estar en ejecución sin conexión activa a WebSocket.	1. Montar el componente SocketProvider con SocketIndicator. 2. Verificar que el badge amarillo	l indicador debe mostrar "Fallback: Polling every 1s".	Aprobado

Nota. Estas pruebas validan el comportamiento visual del componente SocketIndicator en función del estado de la conexión WebSocket.
Fuente. Autoría propia.

La Figura 51 muestra el componente Socket Indicator, el cual informa visualmente al usuario sobre el estado de la conexión en tiempo real mediante WebSockets.

Figura 51
Socket Indicator



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Server: Prueba Componente Server Channel

Ruta de acceso: cypress/component/server/server-channel.cy.tsx

La Tabla 28 presenta los casos de prueba diseñados para verificar el correcto funcionamiento del componente ThemeToggle, encargado de cambiar el tema visual de la aplicación entre los modos claro y oscuro.

Tabla 28
TI-040-041

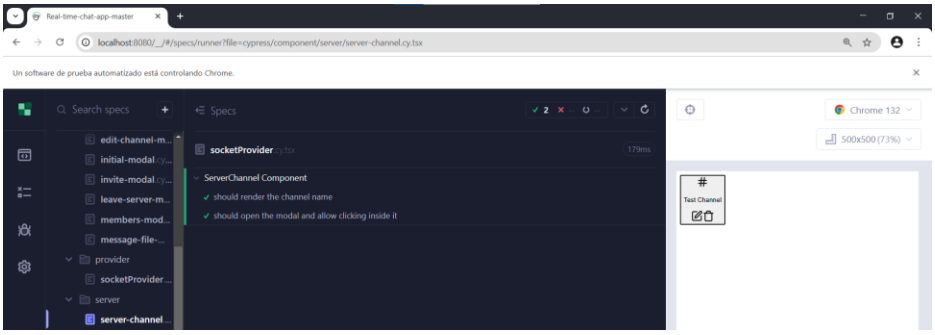
ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-040	Renderización del nombre del canal	Verificar que el nombre del canal se renderiza correctamente.	La aplicación debe estar en ejecución y el canal debe existir.	1. Montar el componente ServerChannel con los datos simulados. 2. Verificar que el nombre "Test Channel" es visible en la interfaz.	El nombre "Test Channel" debe mostrarse en pantalla.	Aprobado
TI-041	Apertura del modal al hacer clic en el botón	Verificar que al hacer clic en el botón, se abre el modal asociado al canal.	La aplicación debe estar en ejecución y el botón del canal debe ser accesible.	1. Montar el componente ServerChannel. 2. Hacer clic en el botón dentro del componente. 3. Verificar si el modal se muestra correctamente	El modal debe abrirse correctamente.	Aprobado

.

Nota. La tabla valida que el componente ServerChannel renderice correctamente el nombre del canal y abra el modal correspondiente al hacer clic en el botón.
Fuente. Autoría propia.

La Figura 52 ilustra el componente ServerChannel, responsable de representar visualmente los canales disponibles dentro de un servidor.

Figura 52
Server channel component



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Server Member

Ruta de acceso: cypress/component/server/server-member.cy.tsx

La Tabla 29 presenta las pruebas funcionales realizadas al componente ServerMember, el cual es responsable de mostrar la información de los miembros de un servidor, como el nombre, el avatar y los íconos de rol asignados.

Tabla 29
TI-042-043

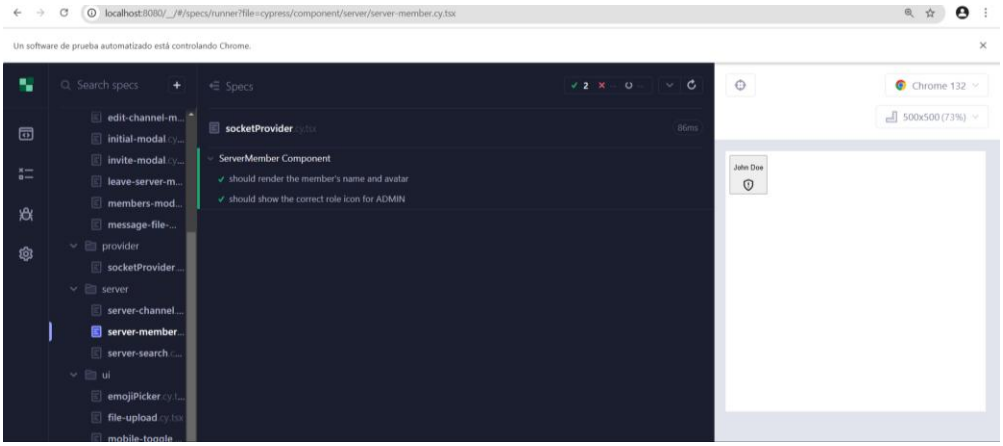
ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-042	Renderización del nombre y avatar del miembro	Verificar que el nombre y el avatar del miembro se renderizan correctamente	La aplicación debe estar en ejecución y el usuario debe ser un miembro del	1. Montar el componente ServerMember con datos simulados. 2. Verificar	El nombre "John Doe" y su avatar deben	Aprobado

		.	servidor.	que el nombre "John Doe" es visible en la interfaz.	mostrarse en pantalla.	
TI-043	Mostrar el icono de rol correcto para ADMIN	Verificar que el icono de rol correcto se muestra cuando el miembro es ADMIN.	El usuario debe tener el rol de ADMIN en el servidor.	1. Montar el componente ServerMember. 2. Verificar que el nombre "John Doe" es visible. 3. Verificar que el icono de escudo de ADMIN (lucide-shield-alert) está presente con los estilos correctos.	El icono de ADMIN debe mostrarse junto al nombre del usuario.	Aprobado

Nota. La tabla verifica la correcta renderización del nombre, avatar e ícono de rol del miembro en el componente ServerMember bajo distintas condiciones.
Fuente. Autoría propia.

La Figura 53 muestra la interfaz visual del componente ServerMember, encargado de representar a cada miembro dentro de un servidor.

Figura 53
Server member component



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Componente Server Search

La Tabla 30 detalla los casos de prueba aplicados al componente ServerSearch, el cual permite a los usuarios realizar búsquedas rápidas dentro del servidor.

Tabla 30
TI-044-048

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-044	Renderizar botón de búsqueda	Verificar que el botón de búsqueda está visible en la interfaz.	La aplicación debe estar en ejecución.	1. Montar el componente ServerSearch. 2. Verificar que el botón "Search" es visible en la interfaz.	El botón "Search" debe mostrarse correctamente.	Aprobado
TI-045	Abrir el diálogo de búsqueda al hacer clic	Verificar que el cuadro de búsqueda aparece al	La aplicación debe estar en ejecución.	1. Montar el componente ServerSearch.	El cuadro de búsqueda debe abrirse.	Aprobado

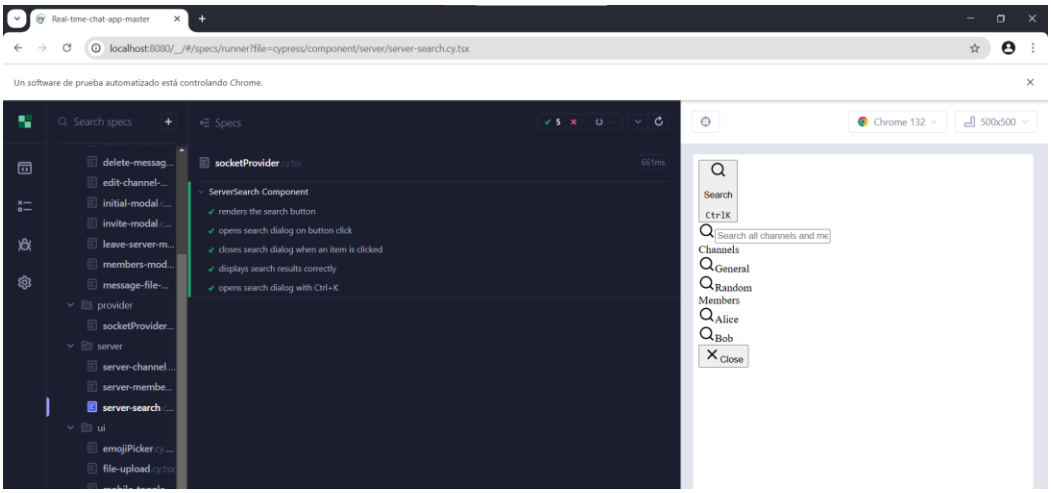
		hacer clic en el botón.		2. Hacer clic en el botón "Search". 3. Verificar que el diálogo de búsqueda se abre.		
TI-046	Cerrar el diálogo al seleccionar un elemento	Verificar que el cuadro de búsqueda se cierre tras seleccionar un elemento.	La aplicación debe estar en ejecución y el diálogo debe estar abierto.	1. Montar el componente ServerSearch. 2. Hacer clic en el botón "Search". 3. Hacer clic en "General". 4. Verificar que el diálogo se cierra.	El cuadro de búsqueda debe cerrarse tras seleccionar un ítem.	Aprobado
TI-047	Mostrar correctamente los resultados de búsqueda	Verificar que los resultados de búsqueda aparecen correctamente en el diálogo.	La aplicación debe estar en ejecución y el diálogo debe estar abierto.	1. Montar el componente ServerSearch. 2. Hacer clic en el botón "Search". 3. Verificar que las secciones "Channels" y	Los resultados de búsqueda deben mostrarse correctamente.	Aprobado

				"Members" se muestran. 4. Verificar que los elementos "General", "Random", "Alice" y "Bob" son visibles.		
TI-048	Abrir el cuadro de búsqueda con Ctrl+K	Verificar que el diálogo de búsqueda se abra con la combinación de teclas.	La aplicación debe estar en ejecución.	1. Montar el componente ServerSearch. 2. Presionar Ctrl+K en el teclado. 3. Verificar que el diálogo de búsqueda se abre.	El cuadro de búsqueda debe abrirse al presionar Ctrl+K.	Aprobado

Nota. Estas pruebas validan la funcionalidad y experiencia de usuario del componente ServerSearch.
Fuente. Autoría propia.
Ruta de acceso: cypress/component/server/server-search.cy.tsx

La Figura 54 presenta el componente ServerSearch, diseñado para facilitar la navegación y búsqueda dentro de los servidores.

Figura 54
Server search component



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Ui: Prueba Emoji Component

La Tabla 31 presenta los casos de prueba realizados sobre el componente EmojiPicker, encargado de permitir a los usuarios insertar emojis en sus mensajes mediante una interfaz interactiva.

Tabla 31
TI-049-051

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-049	Renderizar el selector de emojis	Verificar que el botón para abrir el selector de emojis se muestra correctamente.	La aplicación debe estar en ejecución.	1. Montar el componente EmojiPicker. 2. Verificar que el botón del emoji (icono Smile) es	El botón del emoji debe mostrarse correctamente.	Aprobado

				visible.		
TI-050	Abrir el selector de emojis al hacer clic	Verificar que el selector de emojis se abre al hacer clic en el botón de emojis.	La aplicación debe estar en ejecución.	1. Montar el componente EmojiPicker. 2. Hacer clic en el botón del emoji. 3. Verificar que el popover de selección de emojis aparece.	El selector de emojis debe abrirse correctamente.	Aprobado
TI-051	Adaptación al tema oscuro	Verificar que el selector de emojis se adapta al tema oscuro.	La aplicación debe estar en ejecución con el tema dark.	1. Montar el componente EmojiPicker dentro de ThemeProvider con el tema en "dark". 2. Abrir el selector de emojis. 3. Verificar que el selector respeta la clase "dark".	El selector de emojis debe estar en modo oscuro.	Aprobado

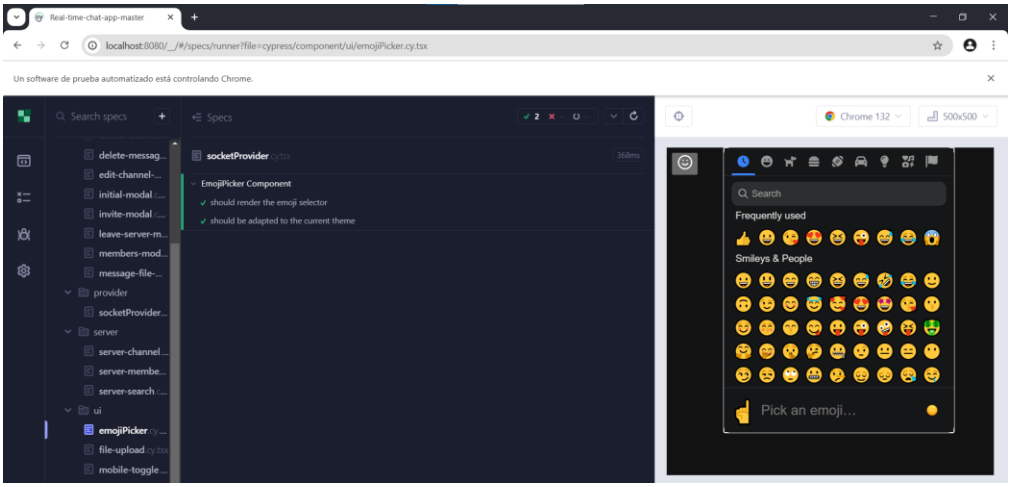
Nota. Estas pruebas aseguran la correcta visualización y funcionalidad del componente EmojiPicker.

Fuente. Autoría propia.

Ruta de acceso: cypress/component/ui/emojiPicker.cy.tsx

La Figura 55 muestra el componente Emoji Picker, una herramienta interactiva diseñada para facilitar la selección de emojis dentro de formularios o áreas de texto enriquecido.

Figura 55
Emoji picker component



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba File Upload Component

Ruta de acceso: cypress/component/ui/file-upload.cy.tsx

La Figura 56 presenta el archivo de configuración principal de un proyecto Next.js: next.config.js.

Figura 56
Configuración next.config.js

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Tabla 32 detalla una serie de casos de prueba funcionales aplicados al componente

FileUpload.

Tabla 32
TI-052-054

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-052	Renderizar la zona de carga vacía	Verificar que la zona de carga se muestra cuando no hay valor en el FileUpload.	La aplicación debe estar en ejecución.	1. Interceptar la solicitud GET a /api/uploadthing. 2. Montar el componente FileUpload con un valor vacío. 3. Esperar la respuesta interceptada. 4. Verificar que la función onChange no se ha llamado.	La zona de carga debe mostrarse sin errores.	Aprobado
TI-053	Renderizar una	Verificar que el	La aplicación debe estar en	1. Montar FileUploa	La imagen debe	Aprobado

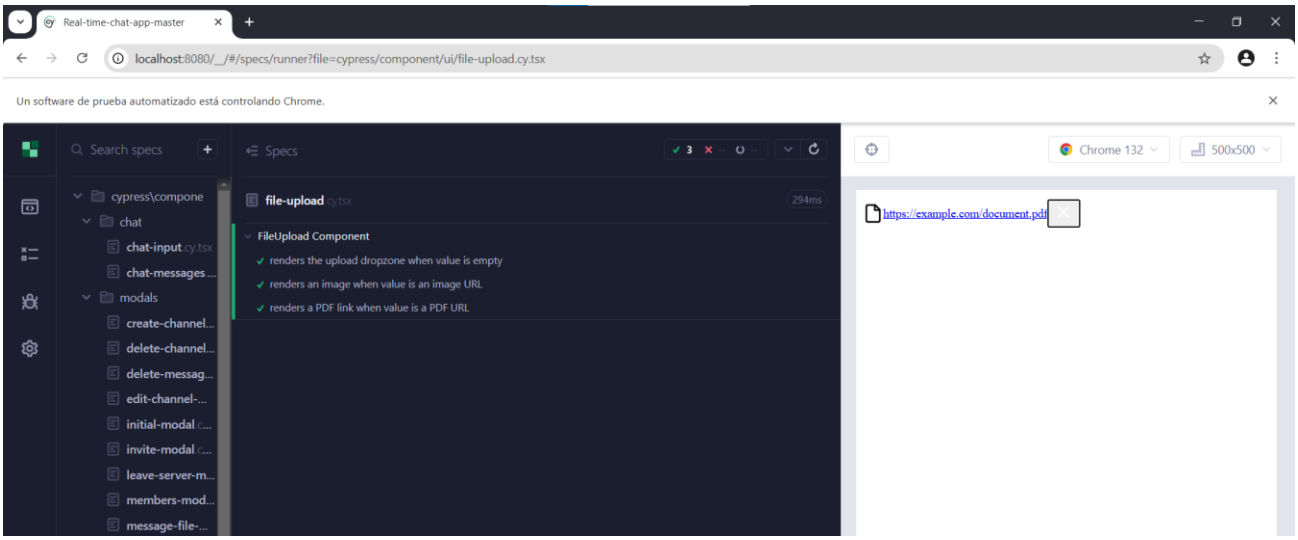
	imagen si el valor es una URL	componente muestra la imagen correctamente si se proporciona una URL de imagen.	ejecución.	d con una URL de imagen como valor. 2. Verificar que la etiqueta <img> tiene el atributo src con la URL de la imagen optimizada.	mostrarse correctamente.	
TI-054	Renderizar un enlace de PDF si el valor es una URL de PDF	Verificar que el componente muestra un enlace al archivo PDF cuando se proporciona una URL de PDF.	La aplicación debe estar en ejecución.	1. Montar FileUpload con una URL de PDF. 2. Verificar que el enlace al PDF está presente. 3. Hacer clic en el botón de eliminar. 4. Verificar	El enlace al PDF debe mostrarse y eliminarse correctamente.	Aprobado

	que la función onChange se llamó con una cadena vacía.

Nota. Estas pruebas garantizan el comportamiento adecuado del componente FileUpload en distintos escenarios.
Fuente. Autoría propia.

La Figura 57 muestra el componente File Upload, una interfaz diseñada para permitir a los usuarios seleccionar y cargar archivos desde su dispositivo local hacia la aplicación.

Figura 57
File upload component



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba Mobile Toggle Component

Ruta de acceso: cypress/component/ui/mobile-toggle.cy.tsx

Esta tabla documenta los casos de prueba funcionales relacionados con el componente MobileToggle.

Tabla 33
TI-055-057

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-055	Renderizar el botón de alternancia (toggle)	Verificar que el botón de alternancia se renderiza correctamente.	La aplicación debe estar en ejecución.	1. Montar el componente MobileToggle con un serverId. 2. Verificar que el botón con la clase md:hidden existe. 3. Asegurar que el botón contiene un icono lucide-menu.	El botón de alternancia debe mostrarse con su ícono correspondiente.	Aprobado
TI-056	Abrir la hoja (Sheet) al hacer clic en el botón	Verificar que la hoja lateral se abre correctamente al hacer clic en el botón de alternancia.	La aplicación debe estar en ejecución.	1. Montar el componente MobileToggle con un serverId. 2. Hacer clic en el botón de alternancia. 3. Verificar que la hoja con data-state="open" está presente en el DOM.	La hoja lateral debe abrirse correctamente.	Aprobado
TI-057	Renderizar los sidebars	Verificar que los component	La aplicación debe	1. Montar MobileToggle con un serverId.	Los sidebars deben	Aprobado

	dentro de SheetContent	es Navigation Sidebar y ServerSidebar se renderizan dentro de la hoja.	estar en ejecución.	2. Ajustar la vista para simular un dispositivo móvil con cy.viewport('iphone-6'). 3. Hacer clic en el botón de alternancia. 4. Verificar que los sidebars (NavigationSidebar y ServerSidebar) están dentro de la hoja abierta.	mostrarse dentro de la hoja lateral.

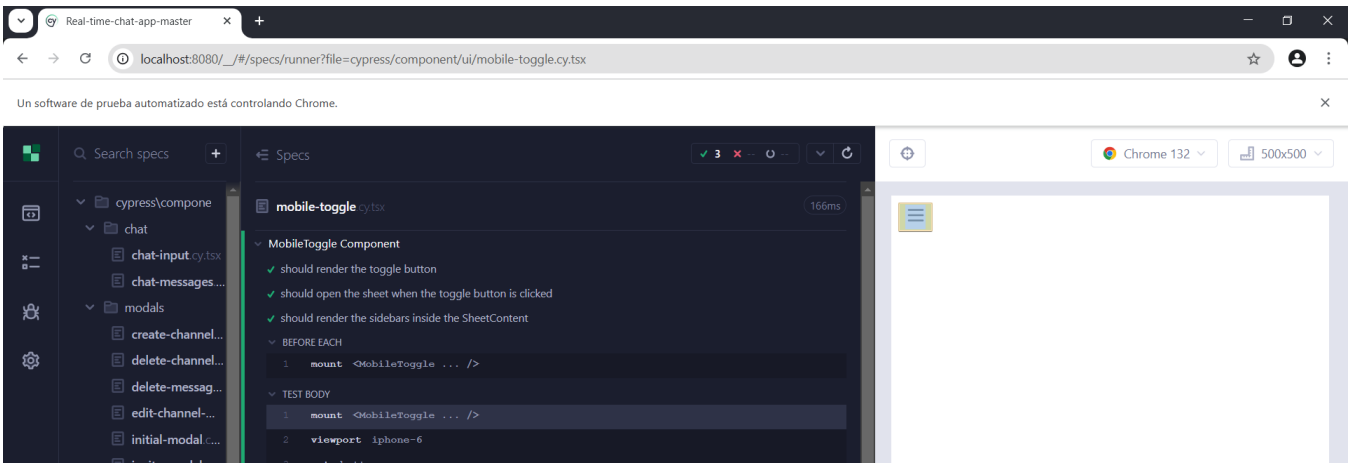
Nota. Estas pruebas aseguran la funcionalidad responsive del componente MobileToggle.

La Figura 58 ilustra el funcionamiento del componente Mobile Toggle, una interfaz diseñada para mejorar la experiencia de navegación en dispositivos móviles.

Fuente. Autoría propia.

Figura 58

Mobile toggle component



Nota. Captura de pantalla propia.

Fuente. Autoría propia.

Prueba Switch Component

Ruta de acceso: cypress/component/ui/mode-toggle.cy.tsx

Esta tabla presenta los casos de prueba funcionales diseñados para validar el comportamiento del componente ModeToggle, encargado de gestionar la selección del tema visual de la aplicación (claro, oscuro o automático según el sistema).

Tabla 34
TI-058-062

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-058	Renderizar el botón de alternancia (ModeToggle)	Verificar que el botón de alternancia se renderiza correctamente.	La aplicación debe estar en ejecución.	1. Montar el componente ModeToggle. 2. Verificar que el botón existe. 3. Asegurar que contiene los íconos de sol y luna.	El botón de alternancia debe mostrarse con sus íconos respectivos.	Aprobado
TI-059	Mostrar opciones de tema en el menú desplegable	Verificar que al hacer clic en el botón, se despliega el menú con opciones.	La aplicación debe estar en ejecución.	1. Montar el componente ModeToggle. 2. Hacer clic en el botón.	Las opciones del tema deben mostrarse correctamente.	Aprobado

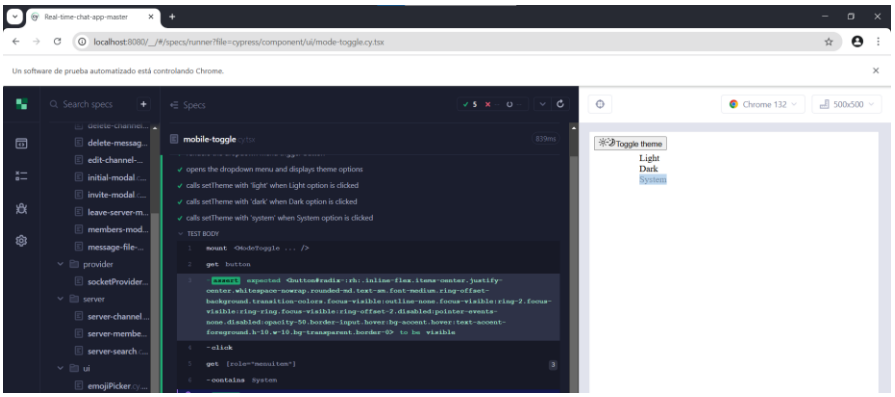
				3. Verificar que aparecen las opciones Light, Dark y System.		
TI-060	Cambiar el tema a "Light"	Verificar que al seleccionar Light, la función setTheme es llamada.	La aplicación debe estar en ejecución.	1. Montar ModeToggle. 2. Hacer clic en el botón. 3. Seleccionar Light. 4. Verificar que setTheme se ha llamado con light.	setTheme debe ser llamado con el valor light.	Aprobado
TI-061	Cambiar el tema a "Dark"	Verificar que al seleccionar Dark, la función setTheme es llamada.	La aplicación debe estar en ejecución.	1. Montar ModeToggle. 2. Hacer clic en el botón. 3. Seleccionar Dark. 4. Verificar que setTheme	setTheme debe ser llamado con el valor dark.	Aprobado

				se ha llamado con dark.		
TI-062	Cambiar el tema a "System"	Verificar que al seleccionar System, la función setTheme es llamada.	La aplicación debe estar en ejecución.	1. Montar ModeToggle. 2. Hacer clic en el botón. 3. Seleccionar System. 4. Verificar que setTheme se ha llamado con system.	setTheme debe ser llamado con el valor system.	Aprobado

Nota. Estas pruebas validan que el componente ModeToggle funcione correctamente para permitir a los usuarios cambiar entre temas visuales (Light, Dark, System).
Fuente. Autoría propia.

La Figura 59 representa el componente Mode Toggle, una funcionalidad clave que permite a los usuarios cambiar entre los diferentes modos de tema visual disponibles en la aplicación: claro (Light), oscuro (Dark) y automático (System).

Figura 59
Mode toggle



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prueba User Avatar Component

Ruta de acceso: cypress/component/ui/user-avatar.cy.tsx

Esta tabla documenta los casos de prueba funcionales aplicados al componente UserAvatar, responsable de mostrar la imagen de perfil del usuario dentro de la aplicación.

Tabla 35
TI-063-066

ID de la prueba	Nombre de la prueba	Descripción	Precondiciones	Pasos	Resultados esperados	Estado
TI-063	Renderizar el avatar con una imagen proporcionada	Verificar que el componente muestra la imagen cuando se pasa una URL.	La aplicación debe estar en ejecución.	1. Montar UserAvatar con una URL de imagen. 2. Verificar que la imagen tiene el atributo src.	La imagen debe renderizarse correctamente con la URL proporcionada.	Aprobado
TI-064	Aplicar clases adicionales	Verificar que el	La aplicación	1. Montar UserAvatar con una	La clase adicional	Aprobado

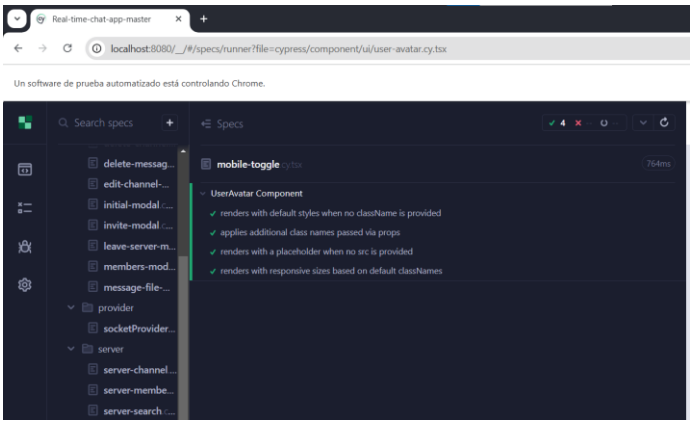
	proporcionadas vía props	componente aplica clases personalizadas correctamente.	debe estar en ejecución.	clase adicional. 2. Verificar que la clase se ha aplicado correctamente	debe estar presente en el avatar.	
TI-065	Renderizar un placeholder cuando no se proporciona una imagen	Verificar que si src no se proporciona, no se renderiza ninguna imagen.	La aplicación debe estar en ejecución.	1. Montar UserAvatar sin src. 2. Verificar que no hay imágenes en el DOM.	No debe haber imágenes en el DOM, solo el contenedor del avatar.	Aprobado
TI-066	Renderizar con tamaños responsivos	Verificar que el componente usa tamaños responsivos según las clases CSS.	La aplicación debe estar en ejecución.	1. Montar UserAvatar. 2. Verificar que las clases de tamaños responsivos (.h-7.w-7.md\:h-10.md\:w-10) están presentes.	El avatar debe tener clases CSS para tamaño responsivo.	Aprobado

Nota. Estas pruebas aseguran que el componente UserAvatar funcione correctamente en diferentes escenarios.
Fuente. Autoría propia.

La Figura 60 presenta el componente User Avatar, diseñado para mostrar la imagen de perfil de un usuario dentro de la interfaz de la aplicación.

Figura 60

User avatar component



Nota. Captura de pantalla propia.

Fuente. Autoría propia.

Test De Carga

K6

Grafana k6 es una herramienta de pruebas de carga de código abierto, fácil de usar para desarrolladores y extensible. k6 le permite prevenir problemas de rendimiento y mejorar la fiabilidad de forma proactiva.

Con k6, puede probar la fiabilidad y el rendimiento de la aplicación e infraestructura.

Comandos:

Install k6.

choco install k6 -y.

crear archivo ws-test.js.

k6 run ws-test.js.

La Tabla 36 describe una prueba de carga enfocada en evaluar el rendimiento y la estabilidad del servidor WebSocket utilizado en la aplicación.

Tabla 36

Tabla test de carga

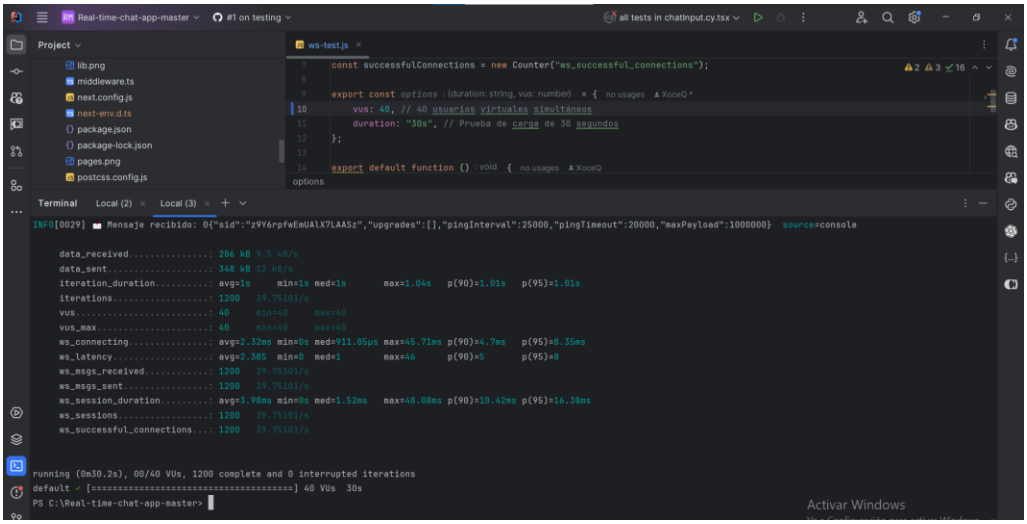
Parámetro	Descripción
Herramienta	K6
Objetivo	Evaluar el rendimiento y estabilidad del servidor WebSocket
Usuarios Virtuales (Bush)	10 usuarios simultáneos
Duración	30 segundos
Endpoint	ws://localhost:3000/api/socket/io/?EIO=4&transport=websocket
Métricas Registradas	- ws_latency: Latencia de conexión (ms) - ws_successful_connections: Conexiones exitosas
Acciones de la prueba	1. Intentar conectar al WebSocket 2. Medir latencia 3. Enviar mensaje de prueba 4. Recibir respuesta 5. Contabilizar conexiones exitosas

Nota. Esta prueba tiene como objetivo evaluar el rendimiento y la estabilidad del servidor WebSocket utilizando la herramienta K6.

Fuente. Autoría propia.

La Figura 61 ilustra visualmente los resultados obtenidos durante la prueba de carga realizada sobre el servidor WebSocket.

Figura 61
Test de carga



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

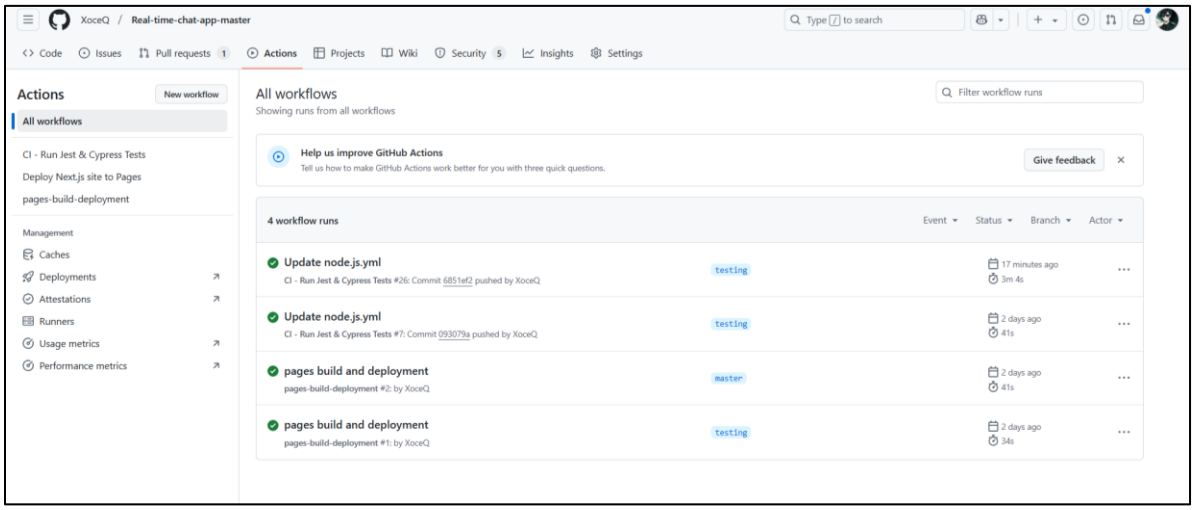
Github Actions Ci: Integración Continua EN Github

GitHub Actions es una plataforma de automatización integrada en GitHub que permite ejecutar flujos de trabajo (workflows) para realizar tareas como pruebas, despliegues y compilaciones.

CI (Continuous Integration, Integración Continua) en GitHub Actions significa que cada vez que se realiza un cambio en el código (por ejemplo, en un push o un pull request), se ejecutan pruebas automáticamente para asegurarse de que el código sigue funcionando correctamente. (Github, 2025)

La Figura 62 muestra la implementación de workflows en GitHub Actions, una herramienta de integración y entrega continua (CI/CD) nativa de GitHub.

Figura 62
Workflows



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Despliegue Frontend

Vercel

Vercel es una plataforma unificada en la nube que permite a los desarrolladores desplegar, gestionar y escalar sus aplicaciones y sitios web. (Aplyca, 2025)

comandos:

prisma generate && npm run build.

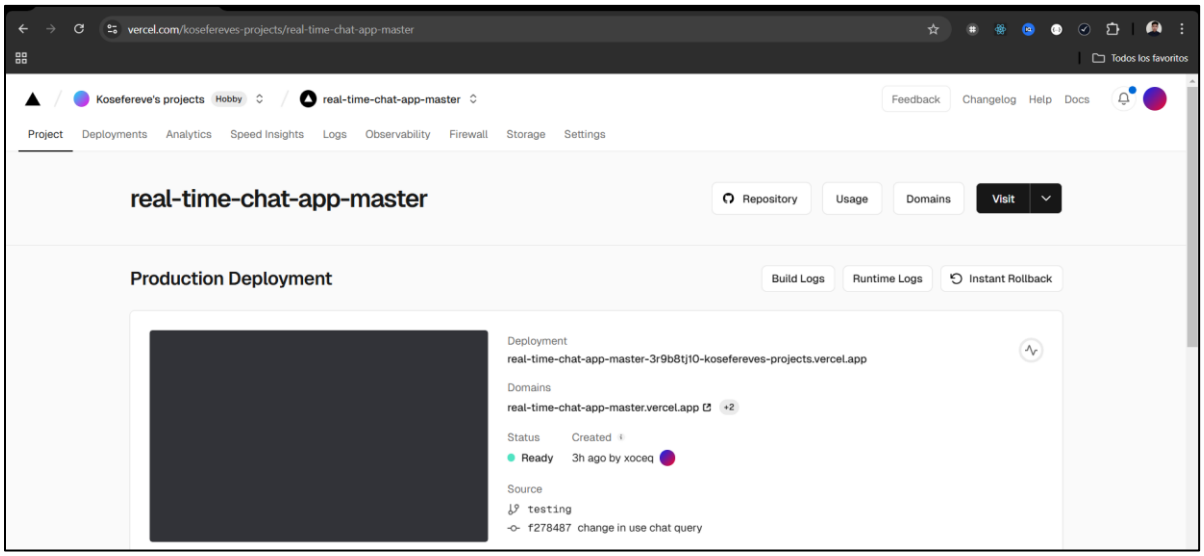
Realiza la generación de Prisma Client antes de la compilación de Next.js.

npm install.

Instala paquetes en un proyecto.

La Figura 63 presenta el proceso de despliegue del frontend de la aplicación utilizando Vercel, una plataforma de alojamiento optimizada para aplicaciones web modernas.

Figura 63
Despliegue Frontend



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Despliegue Backend

Render

Render es una plataforma en la nube que permite crear, implementar y escalar aplicaciones. También se puede usar para alojar sitios, bases de datos, API y otros servicios. (Render, 2025)

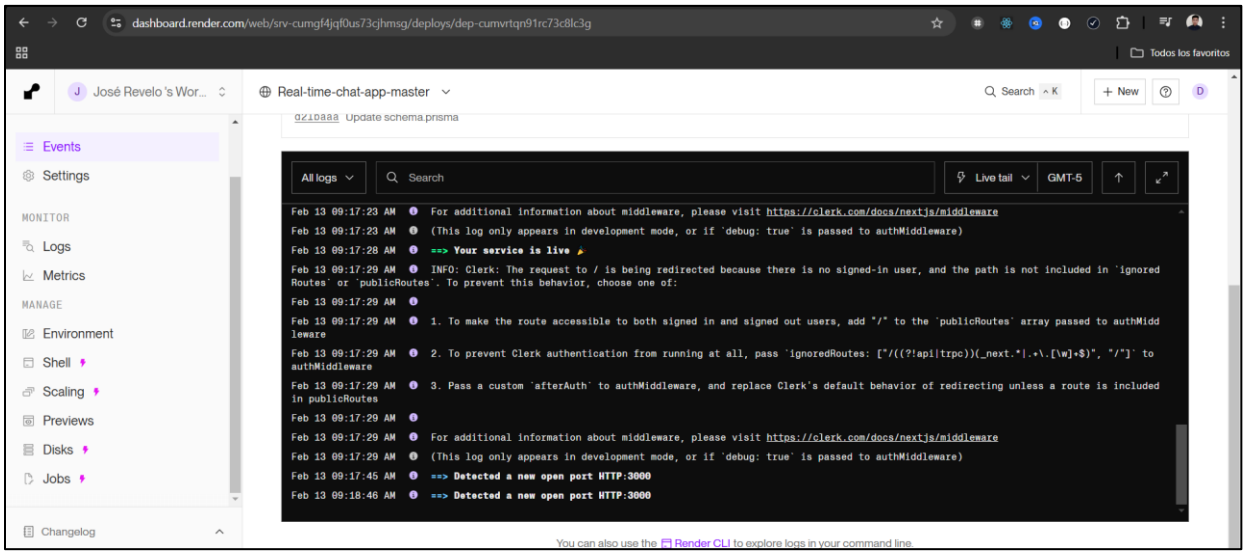
Comandos:

npm install; npm run build.

npm start.

La Figura 64 ilustra el proceso de despliegue del backend mediante la plataforma Render, una solución moderna de alojamiento en la nube que permite implementar aplicaciones web de manera rápida y sencilla.

Figura 64
Despliegue Backend



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Características Y Funcionalidades

Mensajería Instantánea En Tiempo Real

Permite enviar y recibir mensajes de forma inmediata entre usuarios conectados.

Creación De Salas De Chat

Los usuarios pueden crear nuevas salas de chat o unirse a salas ya existentes para mantener conversaciones grupales.

Intercambio De Archivos

La aplicación soporta el envío de imágenes (JPEG, PNG, GIF) y archivos PDF dentro de las conversaciones.

Límites De Carga

Se establecen límites de tamaño para los archivos subidos: 1 MB para imágenes y 10 MB para archivos PDF. Consular manual de usuario (véase Apéndice A).

Comunicación En Tiempo Real

Utiliza WebSockets para garantizar la transmisión de datos en tiempo real entre el servidor y los clientes.

Variables De Entorno

El archivo. env del proyecto almacena las variables de entorno necesarias para el funcionamiento adecuado de la aplicación. Consultar manual de administrador (véase Apéndice B).

Roles

La aplicación maneja diferentes roles para los usuarios, como:

Administrador

puede gestionar miembros, crear salas de chat, eliminar canales, eliminar servidores.

Moderador

puede eliminar mensajes, eliminar canales, gestionar miembros en una sala de chat específica.

Usuario

puede enviar mensajes, unirse a salas de chat.

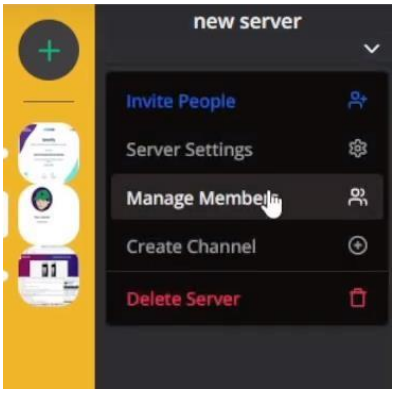
NOTA: Si un miembro crea un nuevo servidor pasa a tener el rol Administrador del mismo.

Opción "Manejar Miembros". Solo disponible para Rol Administrador.

Opción "Kick". Utilizada para eliminar un miembro creado. Solamente disponible para Rol Administrador.

En la figura 65 muestra el botón "Manejar miembros", el cual permite al usuario acceder a la sección destinada a la gestión de los miembros registrados en el sistema.

Figura 65
Botón "manejar miembros"

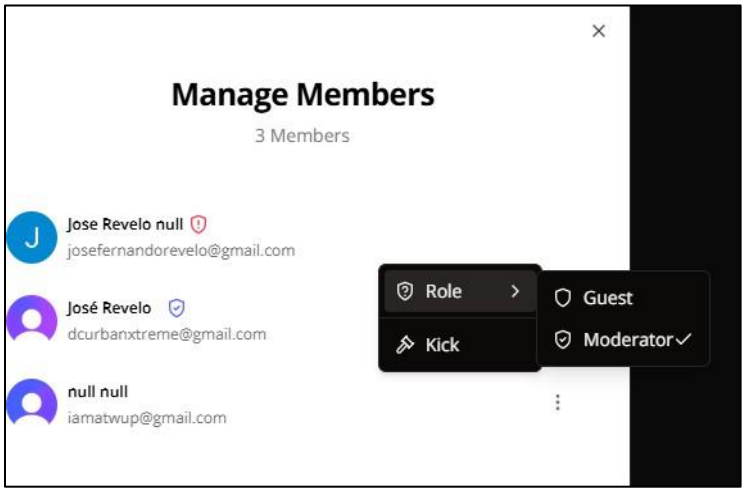


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 66 presenta la vista correspondiente a la funcionalidad "Manejar miembros".

En esta interfaz, el usuario puede gestionar de manera eficiente los miembros del sistema.

Figura 66
Vista "manejar miembros"



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Configuración De La Base De Datos

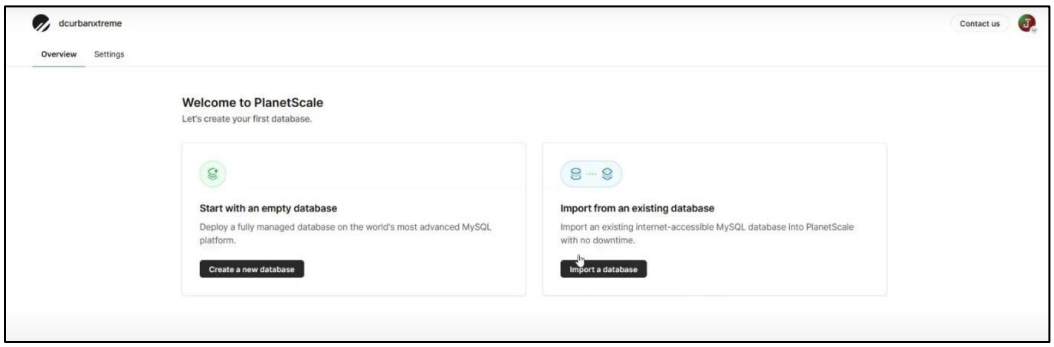
Planetscale

PlanetScale se utilizó en la aplicación para gestionar la base de datos de manera escalable

y eficiente.

La Figura 67 muestra la interfaz principal luego de haber creado exitosamente una cuenta en PlanetScale, una plataforma de bases de datos en la nube basada en MySQL.

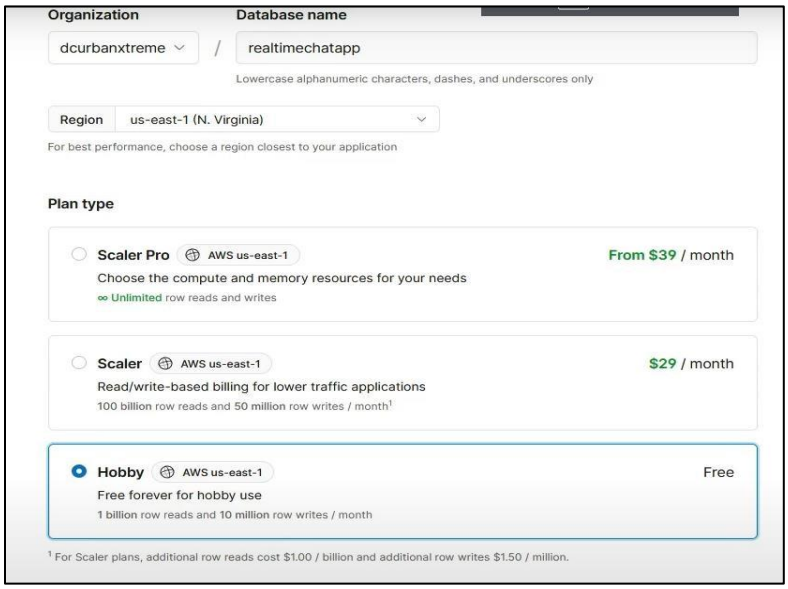
Figura 67
Cuenta creada en PlanetScale



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 68 muestra el tipo de cuenta seleccionada al registrarse en la plataforma PlanetScale.

Figura 68
Tipo de cuenta usada en PlanetScale

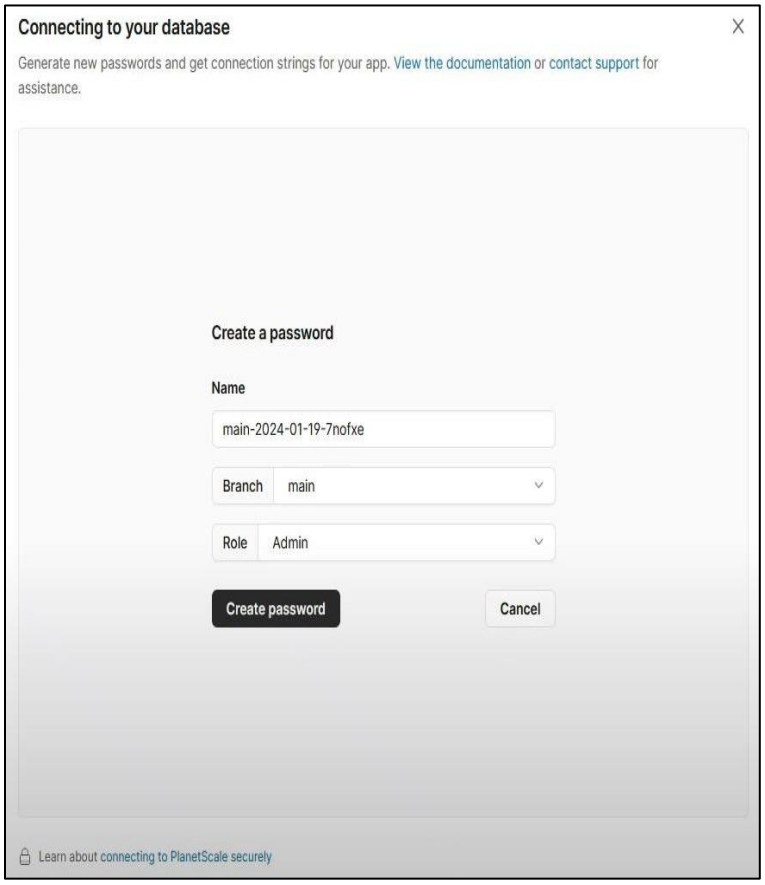


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 69 ilustra el proceso de creación de una nueva base de datos dentro de la

plataforma PlanetScale.

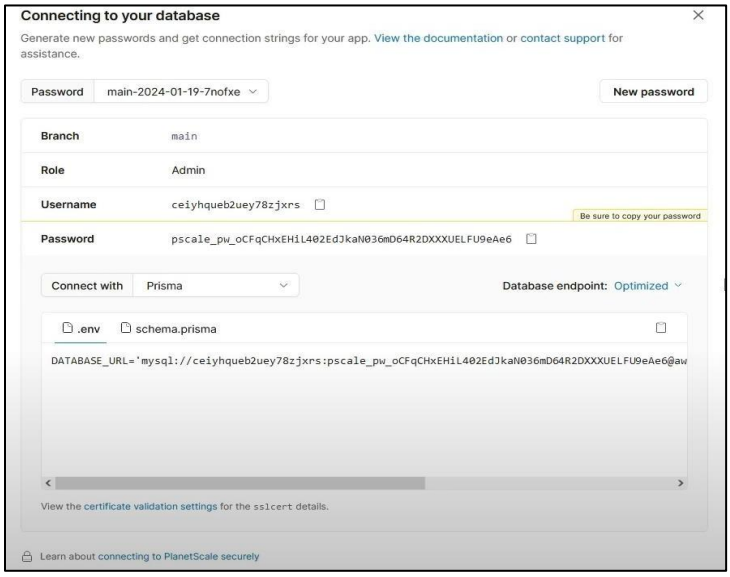
Figura 69
Creación de base de datos



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 70 muestra la URL de conexión generada automáticamente por PlanetScale una vez creada la base de datos.

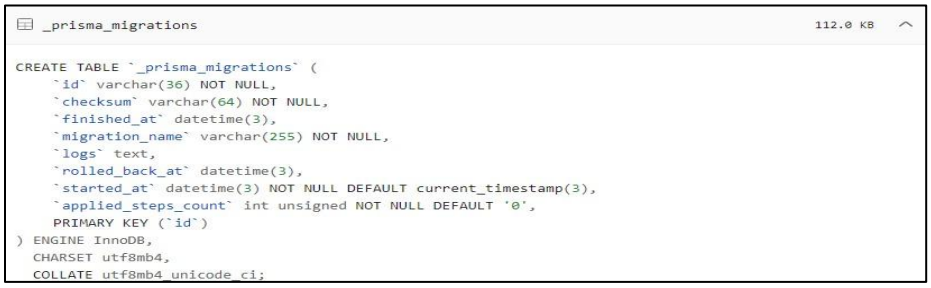
Figura 70
URL Database generada



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 71 presenta la tabla “migraciones”, una estructura generada automáticamente por el sistema de migraciones de Laravel.

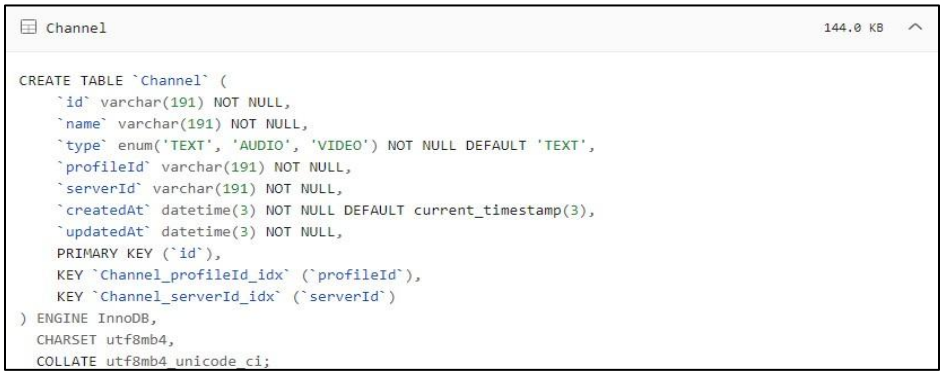
Figura 71
Tabla “migraciones”



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 72 muestra la estructura de la tabla “Canal” dentro de la base de datos del sistema.

Figura 72
Tabla "Canal"



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

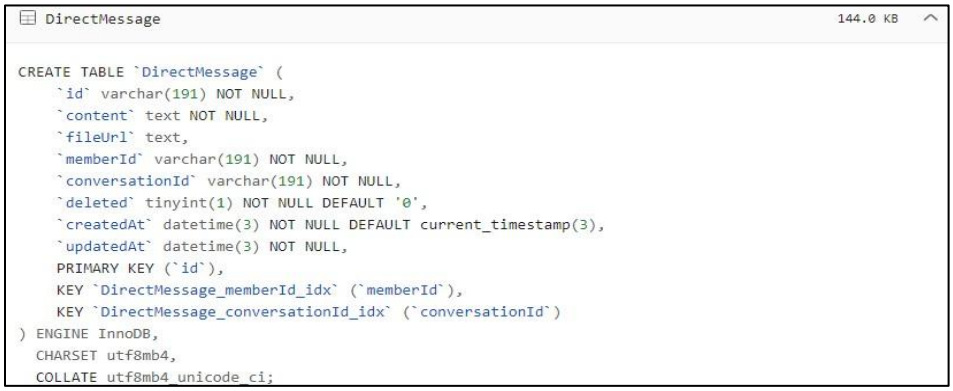
La Figura 73 presenta la estructura de la tabla "Conversación", diseñada para almacenar los registros de las interacciones entre usuarios.

Figura 73
Tabla "Conversación"

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 74 muestra la estructura de la tabla "Mensajes Directos", destinada a almacenar los mensajes enviados de forma privada entre usuarios dentro del sistema.

Figura 74
Tabla "Mensajes Directos"



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 75 presenta la estructura de la tabla “Miembro”, encargada de almacenar la información básica de los usuarios registrados en el sistema.

Figura 75
Tabla “Miembro”

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

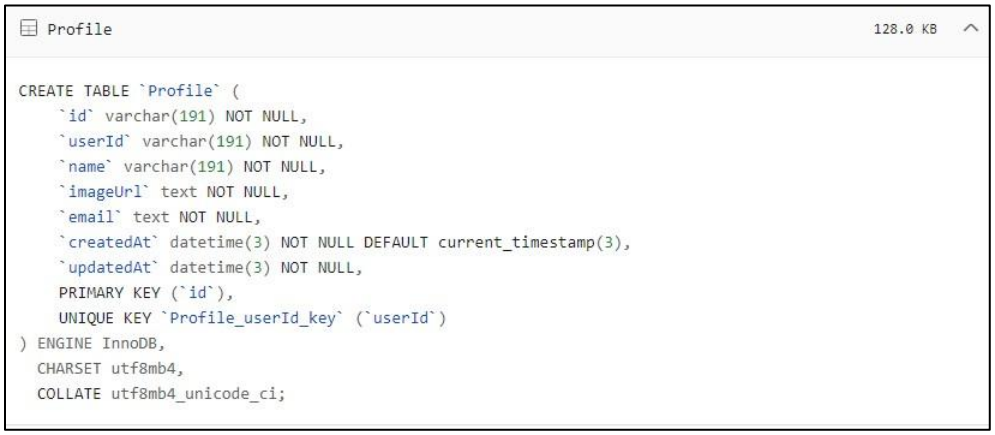
La Figura 76 muestra la estructura de la tabla “Mensaje”, la cual se encarga de almacenar los mensajes enviados dentro de las conversaciones del sistema.

Figura 76
Tabla "Mensaje"

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 77 presenta la estructura de la tabla “Perfil”, la cual contiene información complementaria asociada a cada miembro del sistema.

Figura 77
Tabla “Perfil”



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 78 muestra la estructura de la tabla “Servidor”, utilizada para almacenar información sobre los servidores creados dentro del sistema.

Figura 78
Tabla "Servidor"

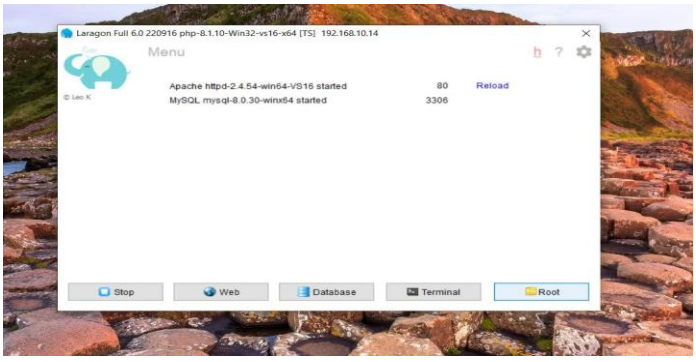
Server 144.0 KB

```
CREATE TABLE `Server` (
    `id` varchar(191) NOT NULL,
    `name` varchar(191) NOT NULL,
    `imageUrl` text NOT NULL,
    `inviteCode` varchar(191) NOT NULL,
    `profileId` varchar(191) NOT NULL,
    `createdAt` datetime(3) NOT NULL DEFAULT current_timestamp(3),
    `updatedAt` datetime(3) NOT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `Server_inviteCode_key` (`inviteCode`),
    KEY `Server_profileId_idx` (`profileId`)
) ENGINE InnoDB,
  CHARSET utf8mb4,
```

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

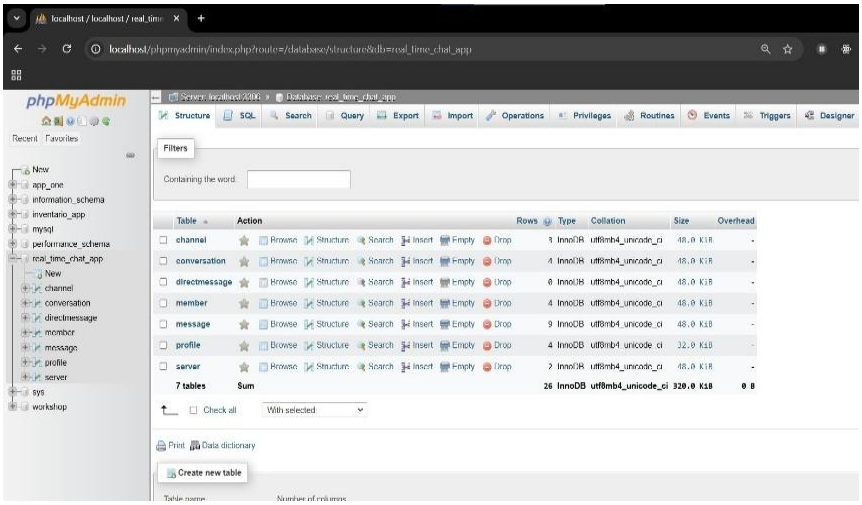
Además, se utilizaron *phpMyAdmin* y *Laragon* para gestionar el almacenamiento localmente debido a que el plan "Hobby" en PlanetScale ya expiró, configurando los archivos prisma y. env para establecer las conexiones y credenciales necesarias.

Figura 79
Laragon en ejecución



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Figura 80
PhpMyAdmin en ejecución



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

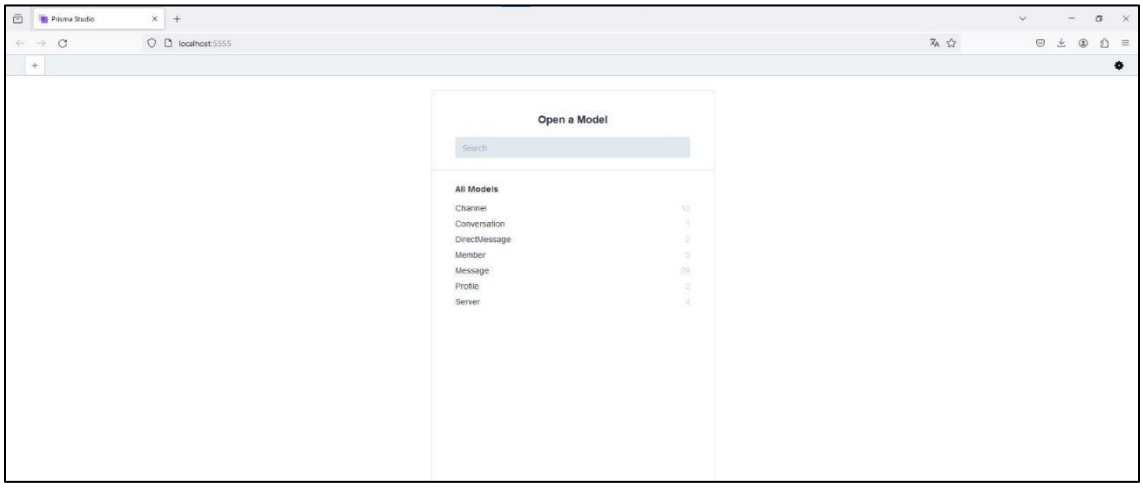
Configuración Servidor

Prisma Studio es un editor visual de los datos de la base de datos que se encuentra utilizando (Prisma, 2024).

Al ejecutar el comando “npx prisma studio” en IntelliJ IDEA, se inició un servidor local al que se puede acceder desde un navegador web en http://localhost:5555. Desde allí, es posible ver e interactuar con la base de datos, incluyendo la adición, actualización y eliminación de datos.

La Figura 81 muestra Prisma Studio en funcionamiento, una interfaz gráfica proporcionada por Prisma para visualizar y gestionar los datos de la base de datos de forma intuitiva.

Figura 81
Prisma Studio en ejecución



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Apis Utilizadas

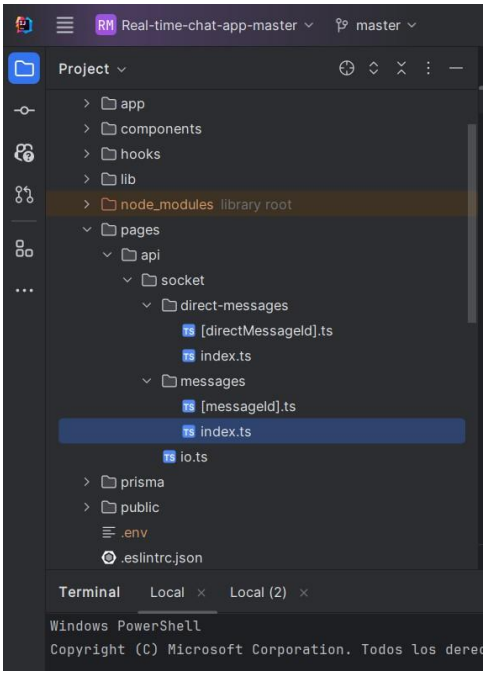
Next.Js API Routes

Las rutas API de Next.js se encuentran en la carpeta pages/api del proyecto.

Funcionalidad: Estas rutas permitieron crear endpoints que pueden manejar solicitudes HTTP, como GET, POST, PUT, DELETE.

La Figura 82 presenta un ejemplo del uso de API Routes en Next.js, una funcionalidad que permite crear puntos de acceso (endpoints) del lado del servidor dentro del mismo proyecto.

Figura 82
Next.js API Routes



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

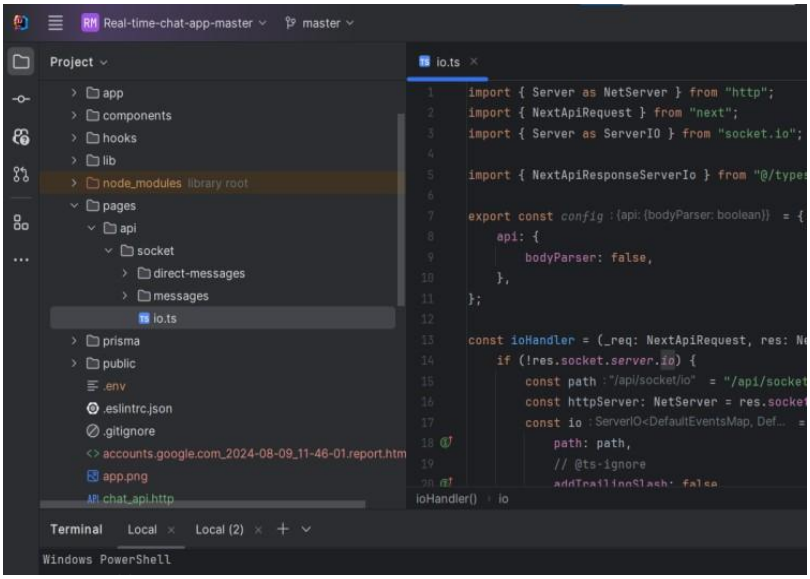
Websocket API

La conexión se estableció en el archivo como pages/api/io.ts

Funcionalidad: Se utilizó para establecer una conexión en tiempo real entre el cliente y el servidor, permitiendo la comunicación instantánea.

La Figura 83 muestra el archivo encargado de configurar la API de WebSocket dentro del proyecto.

Figura 83
WebSocket API file



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

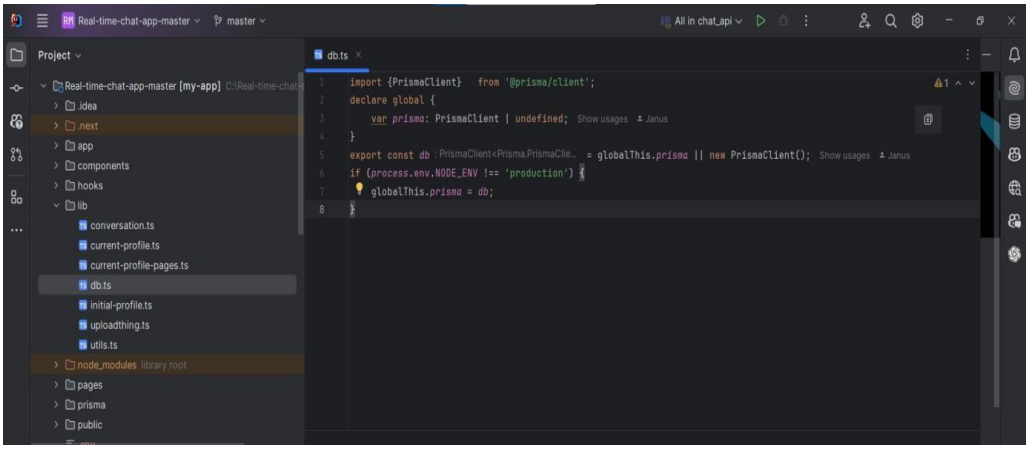
Prisma API

En el archivo lib/db.js, se puede ver la configuración de Prisma y cómo se realizan las consultas a la base de datos.

Funcionalidad: Prisma se utilizó para interactuar con la base de datos MySQL, facilitando la creación de consultas y la manipulación de datos.

La Figura 84 muestra un ejemplo de una API construida con Prisma, utilizada para interactuar con la base de datos desde el Backend del sistema.

Figura 84
Prisma API



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Productos Esperados

Implementación De Real-Time Chat App En Atención Al Cliente Para La Resolución De Consultas Y Problemas En Tiempo Real.

la Real-Time Chat App tiene como objetivo principal proporcionar un sistema de atención al cliente en tiempo real que garantice accesibilidad, fiabilidad y consistencia en todas las interacciones. Para ello, la aplicación asegura que la información proporcionada sea precisa y fácilmente comprensible, mejorando la experiencia del usuario y optimizando la gestión de consultas y reclamaciones.

Alineación Con La ISO 18295-1. Para cumplir con los requisitos establecidos en la UNE-EN ISO 18295-1, la Real-Time Chat App implementa diversas estrategias y tecnologías que optimizan el servicio de atención al cliente:

Accesibilidad Y Fiabilidad. La plataforma está disponible 24/7 y utiliza WebSockets para garantizar conexiones en tiempo real, permitiendo una interacción fluida y sin interrupciones.

Gestión Consistente En Todos Los Canales. Se unifican las interacciones mediante un chat en tiempo real, con posibilidad de integración con otros canales, asegurando una comunicación homogénea.

Exactitud Y Claridad De La Información. La aplicación implementa respuestas automatizadas, combinadas con asistencia humana escalonada, para ofrecer información precisa y reducir tiempos de espera.

Experiencia Del Cliente Final. Se utilizan métricas de satisfacción del usuario y análisis de tiempos de respuesta para una mejora continua del servicio.

Gestión De Quejas Y Reclamaciones. Se registra y analiza cada consulta no resuelta para identificar patrones y optimizar la resolución de problemas.

Protección Del Cliente Final. La plataforma cumple con normativas de privacidad y protección de datos, garantizando la seguridad y confidencialidad de la información.

Liderazgo Y Medición Del Desempeño. Se implementan KPIs como tiempo medio de respuesta, tasa de resolución y nivel de satisfacción, asegurando un monitoreo constante del rendimiento.

Asignación De Recursos Humanos. La automatización de respuestas permite escalar consultas a agentes humanos solo cuando sea necesario, optimizando la carga de trabajo y reduciendo tiempos de atención.

Gestión Segura De La Información. Se emplean protocolos de cifrado y seguridad, protegiendo la información transmitida entre usuarios y servidores.

Infraestructura Para La Prestación Del Servicio. La aplicación está alojada en servidores con alta disponibilidad y escalabilidad, garantizando un servicio estable incluso en momentos de alta demanda.

Relación Con La Organización Cliente. Se generan reportes y análisis de rendimiento, asegurando el cumplimiento de los acuerdos de servicio establecidos con las empresas usuarias. (AENOR, 2025)

Quién puede ser el Admin:

El dueño de la empresa o el gerente de atención al cliente.

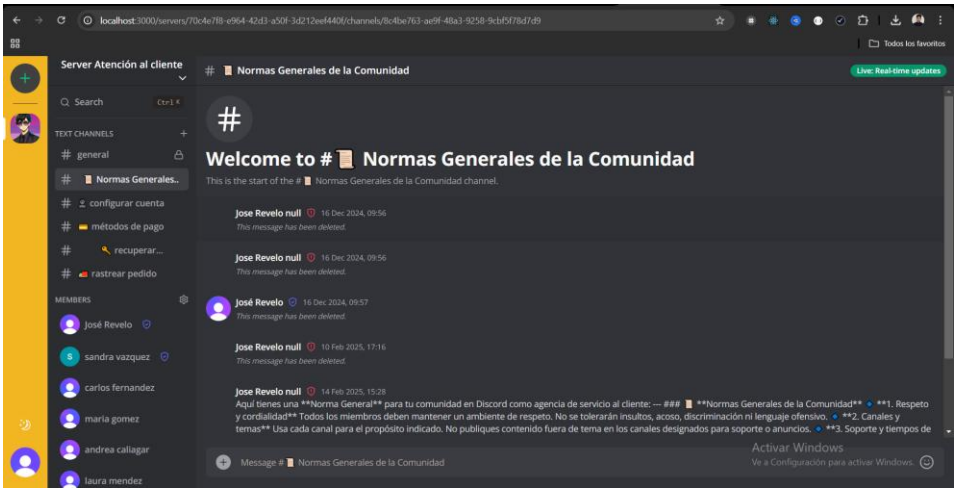
Moderadores:

Supervisores de atención al cliente.

Agentes con más experiencia en soporte técnico.

La Figura 85 muestra la sala titulada “Normas Generales”, la cual forma parte del entorno de comunicación del sistema.

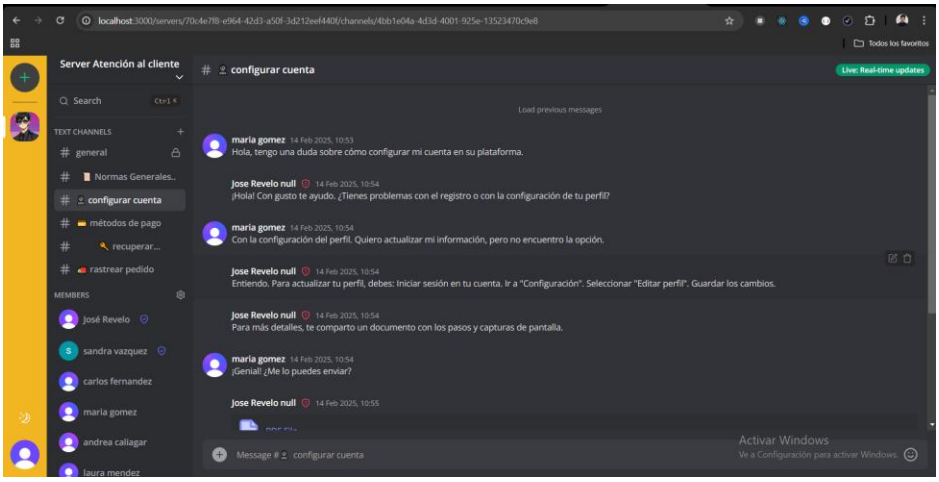
Figura 85
Sala normas generales



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 86 muestra la sala “Configuración de Cuenta”, un espacio dentro de la plataforma destinado a que los usuarios gestionen aspectos relacionados con su perfil personal.

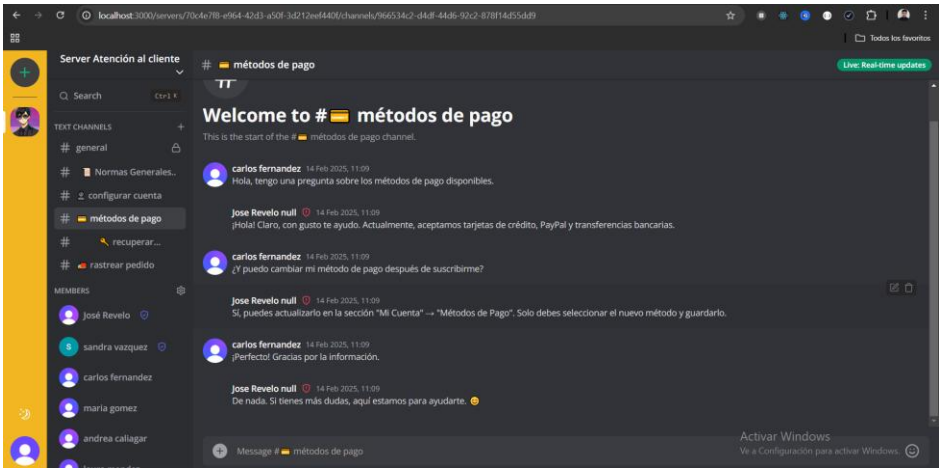
Figura 86
Sala configuración cuenta



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 87 presenta la sala “Métodos de Pago”, un espacio dentro del sistema donde los usuarios pueden gestionar y visualizar las opciones de pago disponibles o vinculadas a su cuenta.

Figura 87
Sala métodos de pago



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

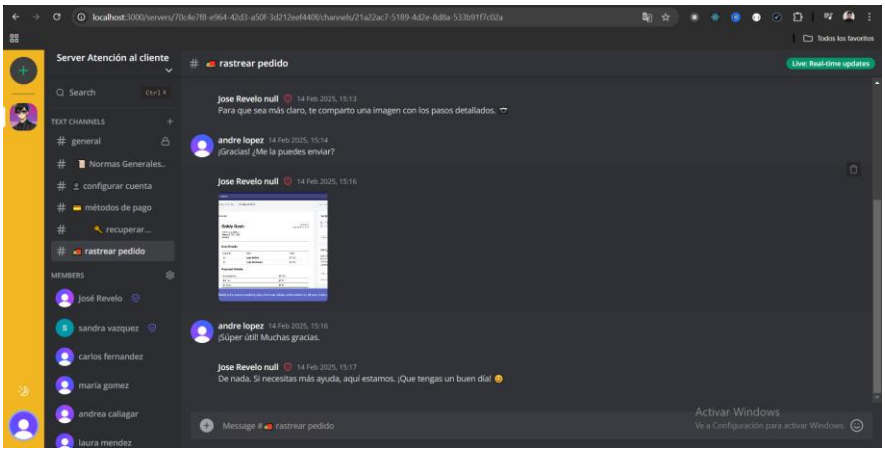
La Figura 88 muestra la sala “Recuperar Contraseña”, diseñada para que los usuarios puedan restablecer el acceso a sus cuentas en caso de haber olvidado su clave.

Figura 88
Sala recuperar contraseña

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 89 presenta la sala "Rastrear Pedido", un espacio dentro del sistema que permite a los usuarios consultar el estado actual de sus compras o envíos.

Figura 89
Sala rastrear pedido

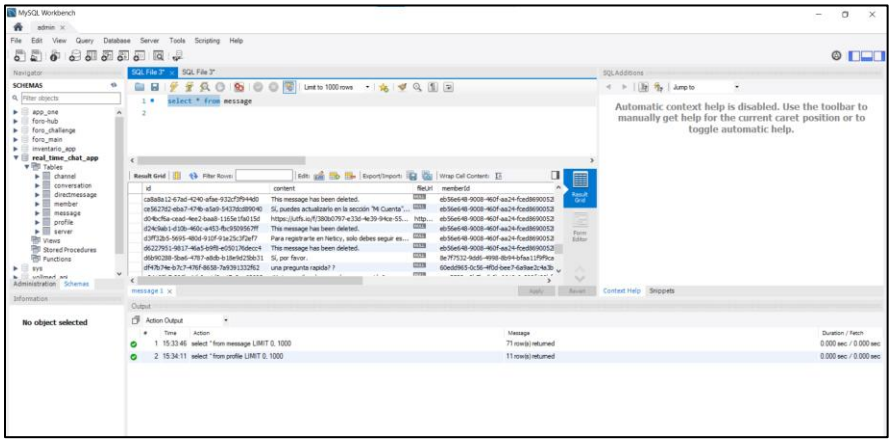


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 90 muestra la estructura de la tabla "Mensajes" correspondiente al servidor de atención al cliente, donde se almacenan las interacciones entre los usuarios y el equipo de soporte.

Figura 90

Tabla mensajes server atención al cliente



Nota. Captura de pantalla propia.

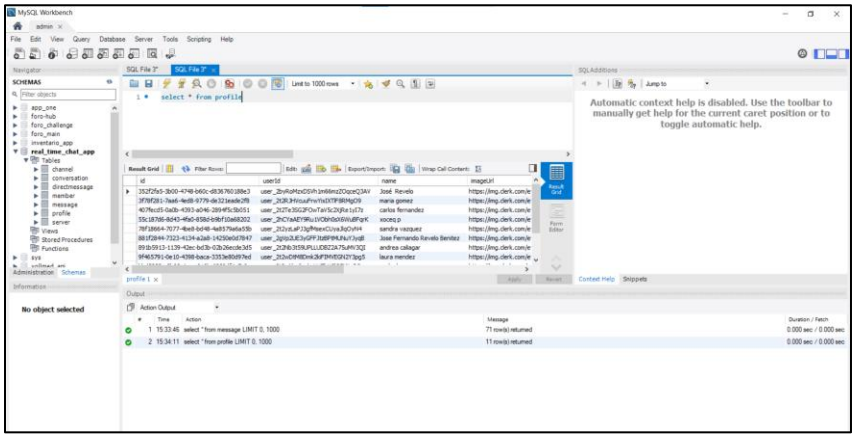
Fuente. Autoría propia.

La Figura 91 muestra la estructura de la tabla “Perfiles” correspondiente al servidor de atención al cliente, encargada de almacenar información detallada de cada usuario dentro del sistema.

Nota: si no se configura el nombre y apellido de usuario al momento de crearlo es resultado es nombre de usuario null.

Figura 91

Tabla perfiles server atención al cliente



Nota. Captura de pantalla propia.

Fuente. Autoría propia.

Implementación De Real-Time Chat App En Un Sistema De Emergencias Médicas

La aplicación Real-Time Chat App, desarrollada con Next.js, React, Socket.io, Prisma, Tailwind y MySQL, tiene como objetivo mejorar la comunicación en situaciones de emergencia médica.

Base del Sistema: El Triage en Emergencias Médicas

El sistema se basa en el Triage, un método de selección y clasificación de pacientes en servicios de urgencias, considerando sus necesidades terapéuticas y los recursos disponibles.

Según la Resolución 5596 del 24 de diciembre de 2015 del Ministerio de Salud y Protección Social, el Triage se divide en cinco categorías:

Triage I. Atención inmediata. Pacientes en riesgo vital que requieren maniobras de reanimación o presentan compromiso ventilatorio, hemodinámico o neurológico.

Triage II. Atención en menos de 30 minutos. Pacientes con riesgo de rápido deterioro o pérdida de miembro/órgano, incluyendo dolor extremo.

Triage III. Atención urgente. Pacientes estables que requieren diagnóstico y tratamiento rápido para evitar complicaciones.

Triage IV. Atención prioritaria. Pacientes sin riesgo inmediato, pero con posibilidad de complicaciones si no reciben tratamiento oportuno.

Triage V. Atención no urgente. Pacientes con problemas agudos o crónicos sin riesgo evidente para la vida o funcionalidad de órganos.

Consideraciones Importantes

El Triage no puede ser utilizado como un mecanismo para negar la atención de urgencias.

La verificación de derechos del paciente debe realizarse después del triage, garantizando una evaluación inmediata al ingreso.

Los prestadores de salud deben informar a los pacientes y acompañantes sobre los tiempos y recursos disponibles para su atención.

Para Triage IV y V, es fundamental que las entidades responsables del pago, junto con sus redes de prestadores, implementen estrategias para mejorar el acceso a consulta externa, especializada y prioritaria, así como a servicios de diagnóstico y apoyo. (minsalud, 2025)

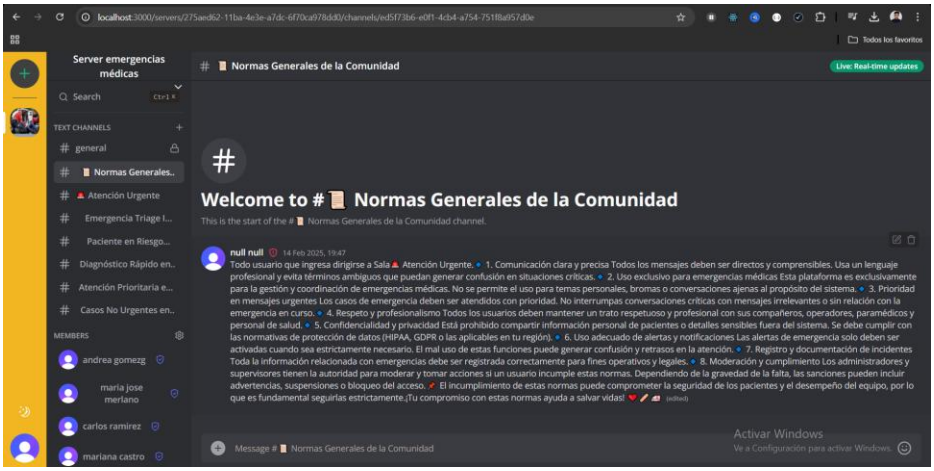
Los pacientes se identifican debido a que no tienen insignia.

Los moderadores cumplen rol de enfermera/o.

El administrador cumple el rol de doctor.

La Figura 92 muestra la sala titulada “Normas Generales”, la cual forma parte del entorno de comunicación del sistema.

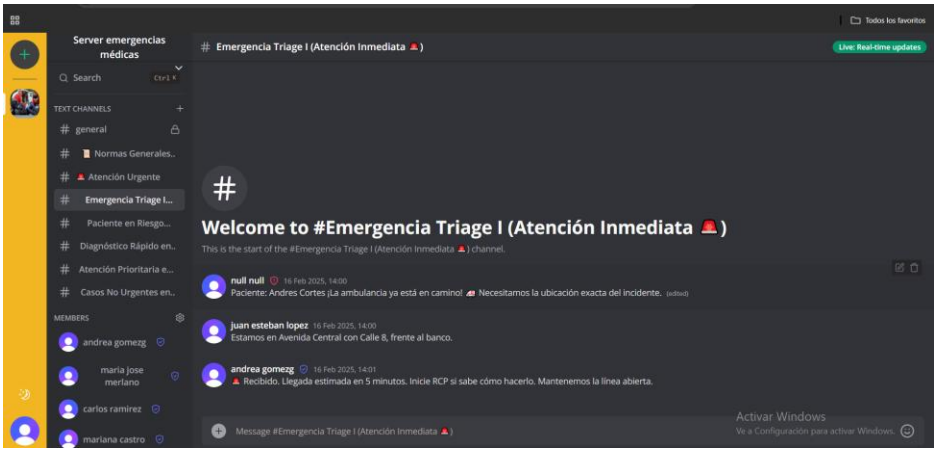
Figura 92
Normas generales



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 93 muestra la sala “Emergencia Triage I (Atención Inmediata 🚨)”, un espacio crítico dentro del sistema destinado a la atención prioritaria de casos urgentes.

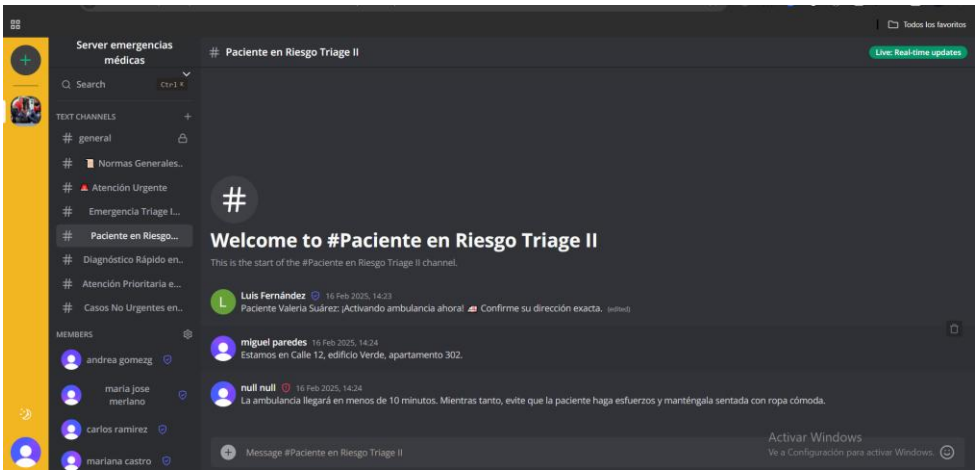
Figura 93
Sala Emergencia Triage I (Atención Inmediata)



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 94 presenta la sala “Paciente en Riesgo Triage II”, un espacio dentro del sistema enfocado en la atención de pacientes con condiciones de riesgo moderado, que requieren una evaluación médica pronta pero no inmediata.

Figura 94
Sala Paciente en Riesgo Triage II

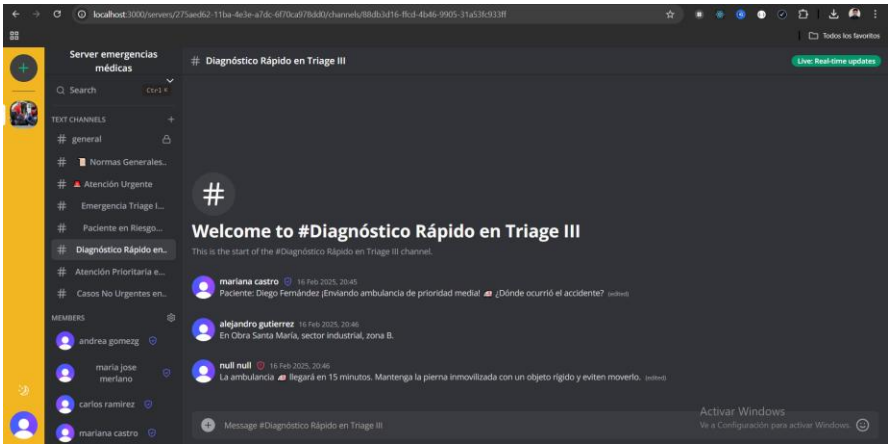


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 95 muestra la sala “Diagnóstico Rápido en Triage III”, un entorno destinado a

la evaluación de pacientes con síntomas leves o condiciones no urgentes, que requieren atención médica sin riesgo inmediato.

Figura 95
Sala Diagnóstico Rápido en Triage III



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 96 representa la sala “Atención Prioritaria en Triage IV”, destinada a pacientes con afecciones crónicas o molestias menores que no representan un riesgo inmediato para la salud, pero que requieren atención médica en un plazo razonable.

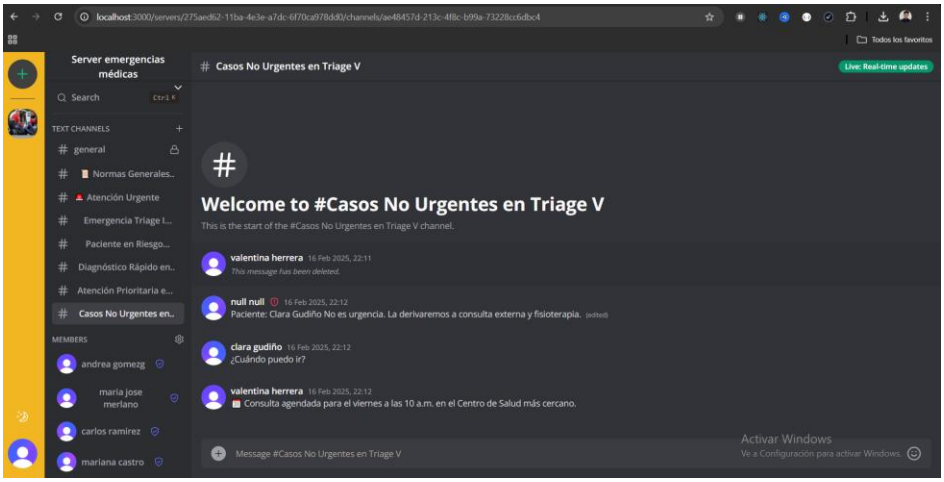
Figura 96
Sala Atención Prioritaria en Triage IV

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 97 muestra la sala “Casos No Urgentes en Triage V”, la cual está destinada a atender a pacientes con síntomas leves o condiciones que no requieren intervención médica

inmediata.

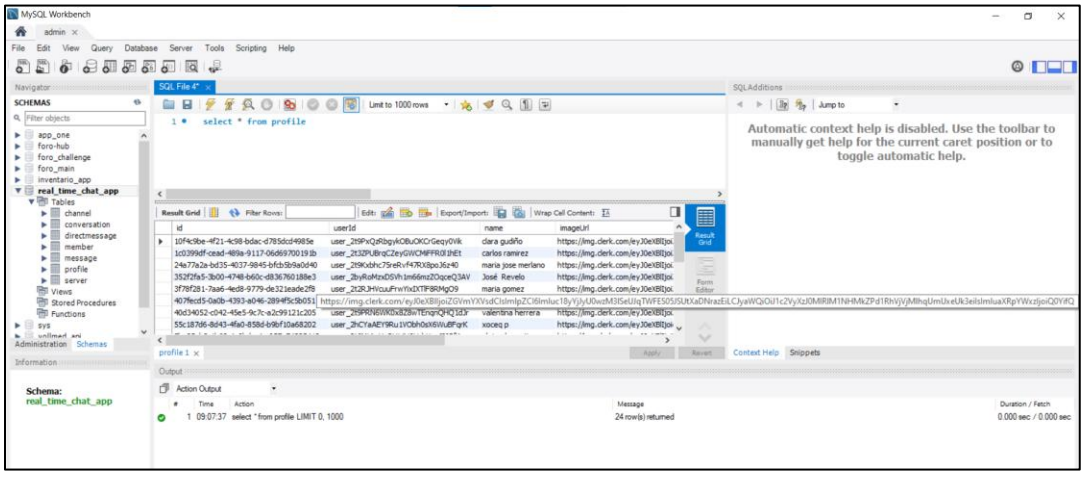
Figura 97
Sala Casos No Urgentes en Triage V



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 98 presenta la estructura de la tabla “Perfiles” correspondiente al servidor de emergencias médicas, donde se almacena información clave sobre cada usuario registrado en el sistema.

Figura 98
Tabla perfiles server emergencias médicas

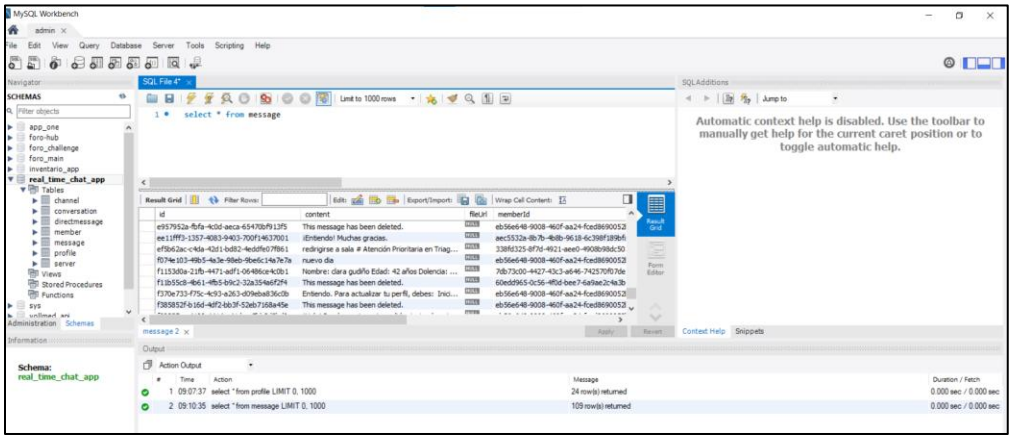


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 99 muestra la estructura de la tabla “Mensajes” del servidor de emergencias

médicas, encargada de registrar todas las comunicaciones realizadas entre los usuarios dentro del sistema.

Figura 99
Tabla mensajes server emergencias médicas



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Plataforma De Soporte Técnico Como Resolución Eficiente De Problemas Técnicos En Tiempo Real

En Colombia, las empresas de soporte técnico que utilicen plataformas para la resolución eficiente de problemas técnicos en tiempo real deben cumplir con diversas normativas y reglamentos que garantizan la calidad del servicio y la protección de los usuarios. Estas regulaciones establecen lineamientos en materia de calidad, seguridad de la información, protección de datos personales y cumplimiento técnico.

Uno de los marcos normativos clave es el **Subsistema Nacional de la Calidad (SNCA)**, que regula las actividades relacionadas con la normalización técnica, reglamentación técnica, acreditación, evaluación de la conformidad y vigilancia del cumplimiento normativo. Su objetivo es garantizar la seguridad, confianza y competitividad en los sectores productivos, incluyendo las empresas de soporte técnico. Cumplir con estos estándares asegura que los servicios prestados sean confiables y cumplan con criterios de calidad establecidos. (Mintic, 2025)

Además, los **Reglamentos Técnicos** son fundamentales en este ámbito, ya que establecen requisitos específicos para los servicios y equipos utilizados en soporte técnico. Un ejemplo relevante es el **Reglamento Técnico para redes internas de telecomunicaciones (RITEL)**, que define las capacidades mínimas de infraestructura necesarias para la correcta operación de redes internas de telecomunicaciones. Estos reglamentos garantizan que los servicios de soporte técnico operen bajo estándares adecuados, minimizando riesgos para los usuarios y asegurando la eficiencia operativa. (Crcom, 2025)

La **protección de datos personales** es otro aspecto crucial para empresas que manejan información de clientes a través de plataformas digitales. La (MinAmbiente, 2025), estableciendo la obligación de garantizar la confidencialidad, integridad y seguridad de la información. Las plataformas de soporte técnico deben implementar políticas de privacidad, encriptación de datos y medidas de seguridad para evitar filtraciones o accesos no autorizados.

En términos de seguridad de la información, el **Decreto 1078 de 2015** recopila las normas relacionadas con el sector de Tecnologías de la Información y las Comunicaciones (TIC). Este decreto establece lineamientos para la gestión de seguridad en sistemas de información, lo que es especialmente relevante para las plataformas de soporte técnico en tiempo real que manejan datos sensibles y requieren estabilidad operativa. (Función Pública, 2025)

Quién puede ser el Admin:

Gerente de Soporte Técnico o IT.

Equipo de gestión de la plataforma.

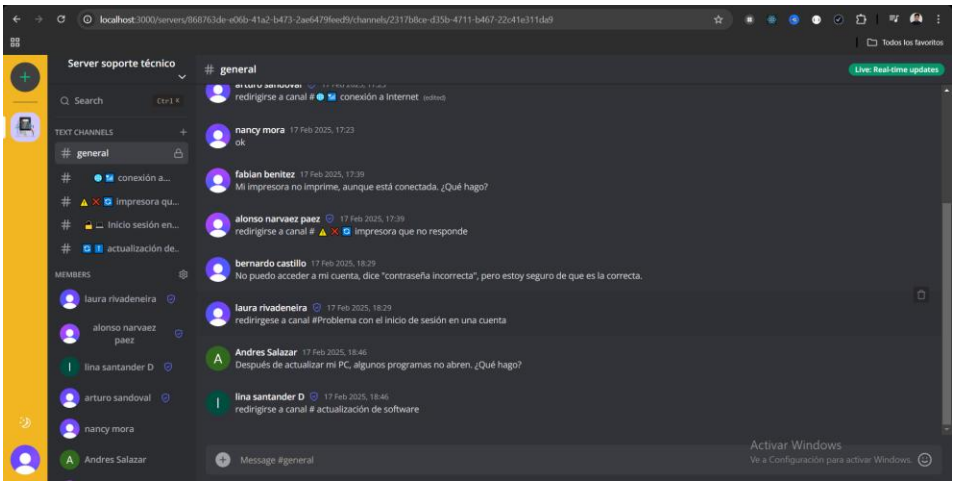
Quiénes pueden ser Moderadores:

Supervisores de Soporte Técnico.

Ingenieros Senior o Especialistas en Infraestructura.

La Figura 100 muestra la sala “General”, un espacio de comunicación abierta dentro del sistema que permite la interacción entre todos los usuarios registrados, independientemente de su rol o nivel de Triage.

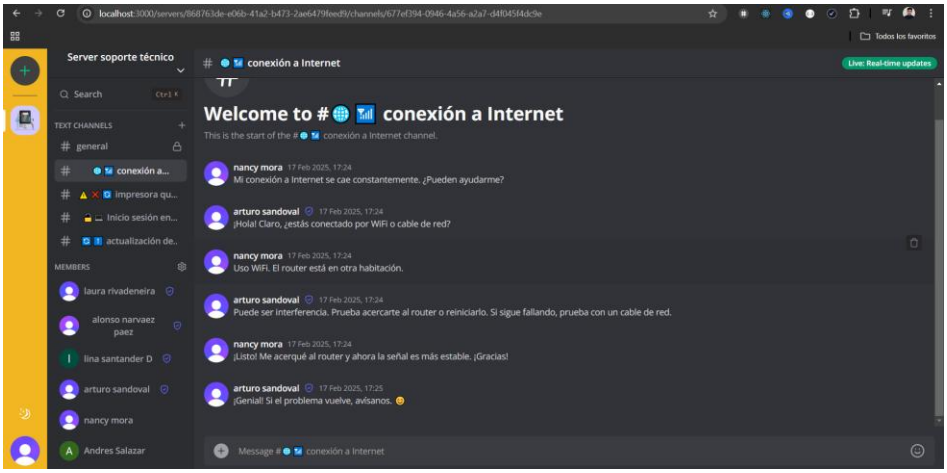
Figura 100
Sala general



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 101 presenta la sala “Conexión a internet”, un espacio dedicado a brindar soporte y orientación a los usuarios que experimentan problemas relacionados con su conectividad.

Figura 101
Sala Conexión a internet



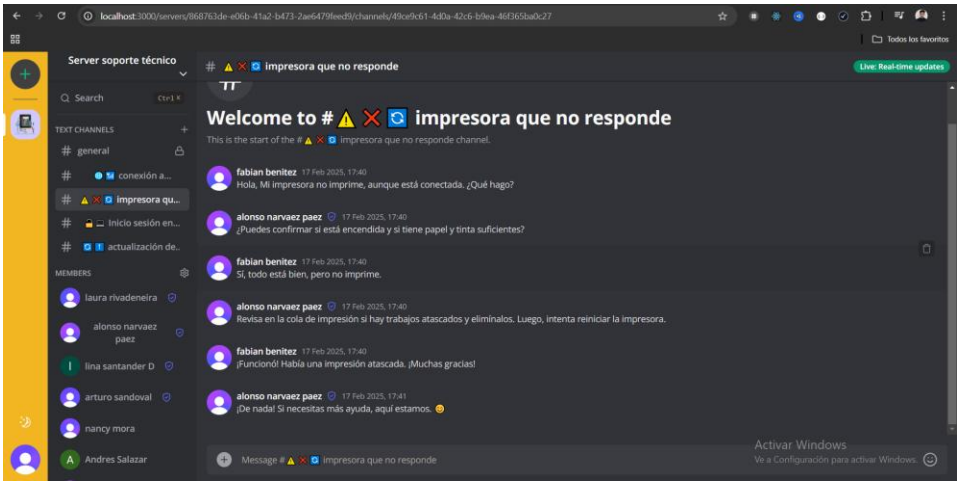
Nota. Captura de pantalla propia.

Fuente. Autoría propia.

La Figura 102 muestra la sala “Impresora que no responde”, un canal de soporte técnico enfocado en la resolución de problemas relacionados con dispositivos de impresión.

Figura 102

Sala impresora que no responde



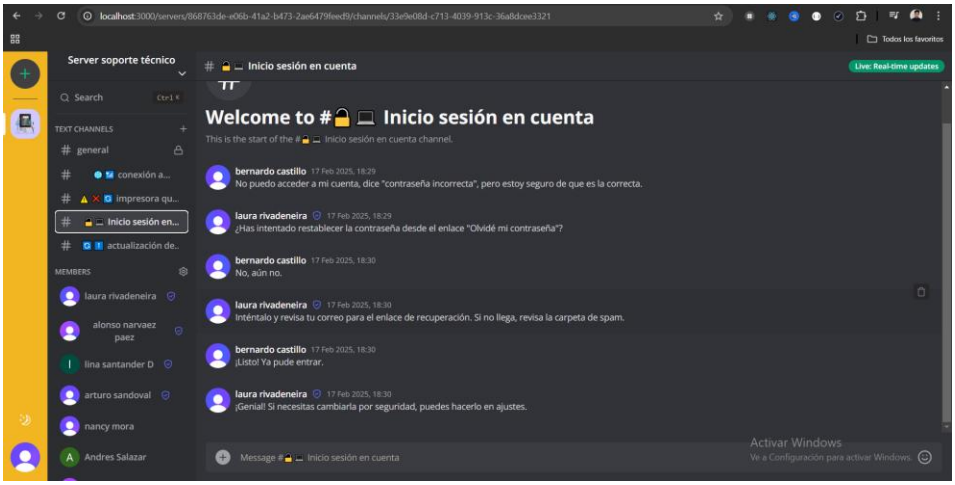
Nota. Captura de pantalla propia.

Fuente. Autoría propia.

La Figura 103 muestra la sala de inicio de sesión en cuenta, la cual permite a los usuarios resolver dudas sobre autenticación dentro del sistema

Figura 103

Inicio de sesión en cuenta



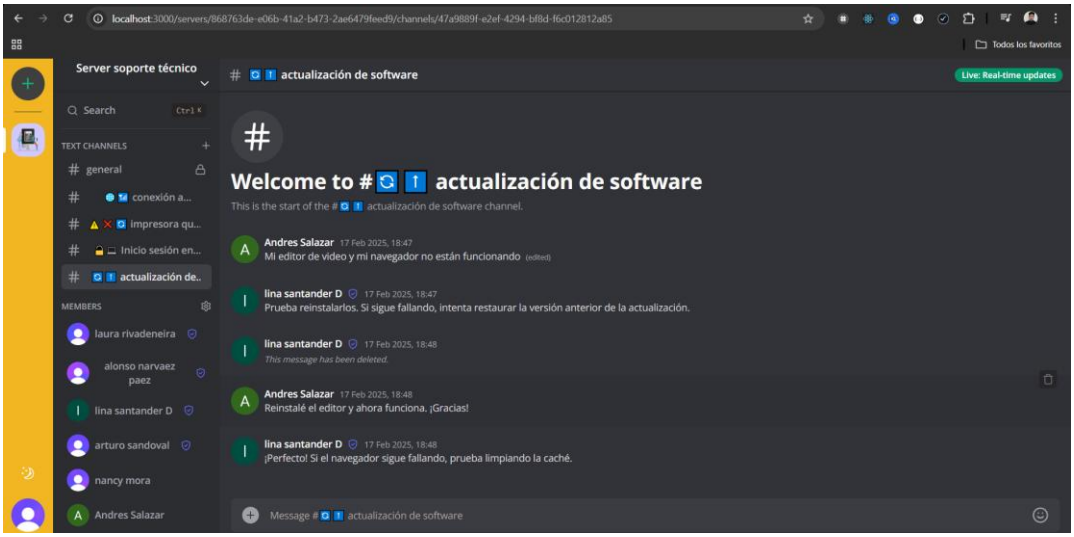
Nota. Captura de pantalla propia.

Fuente. Autoría propia.

La Figura 104 muestra la sección correspondiente a la actualización de software del

sistema, un proceso clave para garantizar el correcto funcionamiento, la seguridad y la incorporación de nuevas funcionalidades.

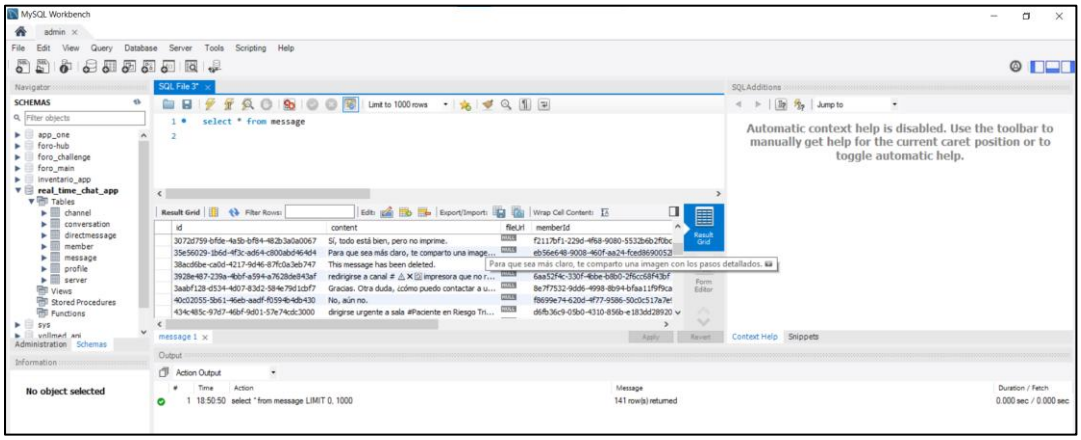
Figura 104
Actualización de software



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 105 muestra la estructura de la tabla de mensajes, elemento clave en la base de datos que almacena cada uno de los mensajes enviados dentro del sistema.

Figura 105
Tabla de mensajes

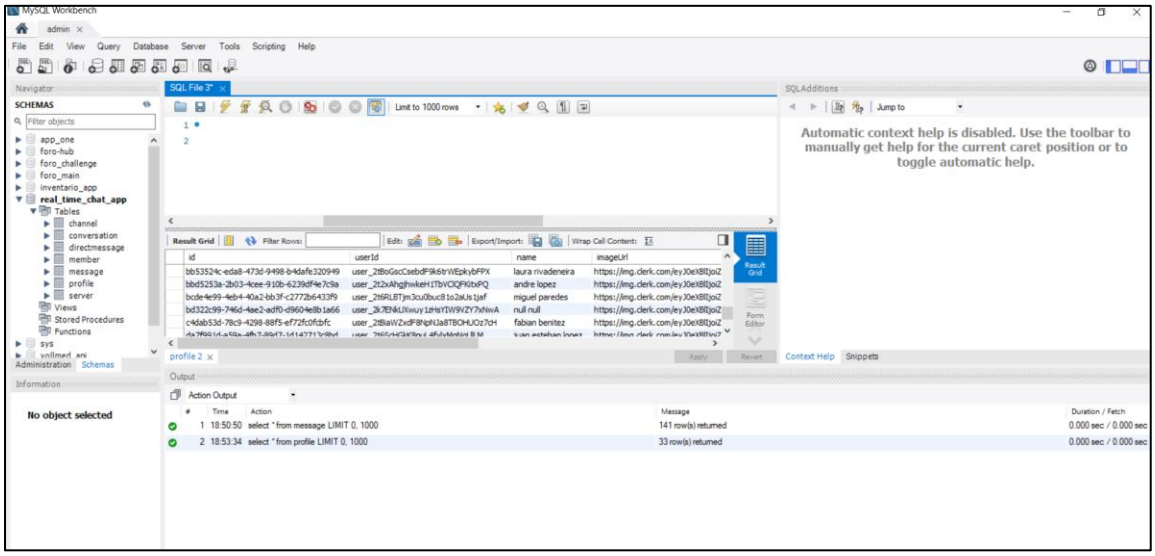


Nota. Captura de pantalla propia.

Fuente. Autoría propia.

La Figura 106 presenta la tabla de perfiles, encargada de almacenar la información personal y de configuración de cada usuario registrado en el sistema.

Figura 106
Tabla perfiles



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Plataforma De Asesoramiento Financiero Personalizado Y En Tiempo Real

En Colombia, la implementación de una plataforma de asesoramiento financiero personalizado y en tiempo real está sujeta a diversas regulaciones que buscan garantizar la transparencia, seguridad y protección de los consumidores financieros. A continuación, se detallan las principales normativas aplicables:

Decreto 2555 De 2010. Este decreto compila y actualiza las normas del sector financiero, asegurador y del mercado de valores en Colombia. Establece las directrices para la prestación de servicios financieros, incluyendo la actividad de asesoría en el mercado de valores. Las disposiciones relacionadas con la asesoría financiera se encuentran en el Libro 40 de la Parte 2 del decreto. Es fundamental que las plataformas de asesoramiento financiero cumplan con los

lineamientos establecidos en este decreto para operar legalmente en el país. (Función Pública, 2025)

Ley 1328 De 2009. Conocida como la Ley de Protección al Consumidor Financiero, esta normativa dicta disposiciones en materia financiera, de seguros y del mercado de valores. Establece los derechos y deberes de los consumidores financieros, así como las obligaciones de las entidades que ofrecen estos servicios. Las plataformas de asesoramiento financiero deben garantizar el cumplimiento de esta ley, asegurando prácticas transparentes y equitativas hacia sus usuarios. (Función Pública, 2025)

Decreto 1297 De 2022. Este decreto modifica el Decreto 2555 de 2010 en lo relacionado con la regulación de las finanzas abiertas en Colombia. Introduce el concepto de finanzas abiertas, permitiendo que las entidades financieras compartan información de los consumidores, con su autorización, a través de interfaces de programación de aplicaciones (API). Las plataformas que ofrecen asesoramiento financiero en tiempo real pueden beneficiarse de este modelo, siempre y cuando cumplan con los estándares tecnológicos y de seguridad establecidos por la Superintendencia Financiera de Colombia (SFC). (Función Pública, 2025)

Protección De Datos Personales. La Ley 1581 de 2012 establece el régimen general de protección de datos personales en Colombia. Las plataformas que manejan información de sus usuarios deben garantizar la confidencialidad, integridad y seguridad de los datos, implementando políticas y procedimientos adecuados para su tratamiento. Es esencial obtener la autorización previa, expresa e informada de los usuarios para el uso de sus datos personales. (MinAmbiente, 2025)

Normativas Fintech. Aunque Colombia no cuenta con una regulación específica para las empresas Fintech, estas deben cumplir con diversas disposiciones legales según la naturaleza de

sus servicios. Entre las leyes aplicables se encuentran la Ley de Comercio Electrónico, la Ley de Habeas Data y la Ley de Protección al Consumidor. Es recomendable que las plataformas de asesoramiento financiero se familiaricen con estas normativas para asegurar su cumplimiento. (Superfinanciera, 2025)

Quién puede ser el Admin:

Gerente de Finanzas o director de la Plataforma de Asesoramiento.

Equipo de gestión técnica y financiera.

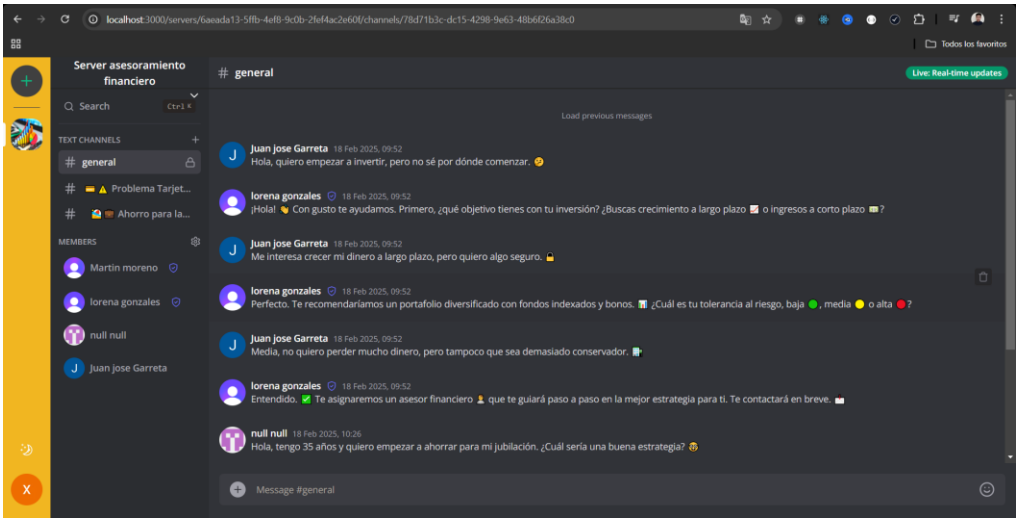
Quiénes pueden ser Moderadores:

Supervisores Financieros.

Especialistas en cumplimiento normativo o riesgos financieros.

La Figura 107 muestra el canal general dentro de la plataforma de asesoramiento financiero personalizado y en tiempo real.

Figura 107
Canal general

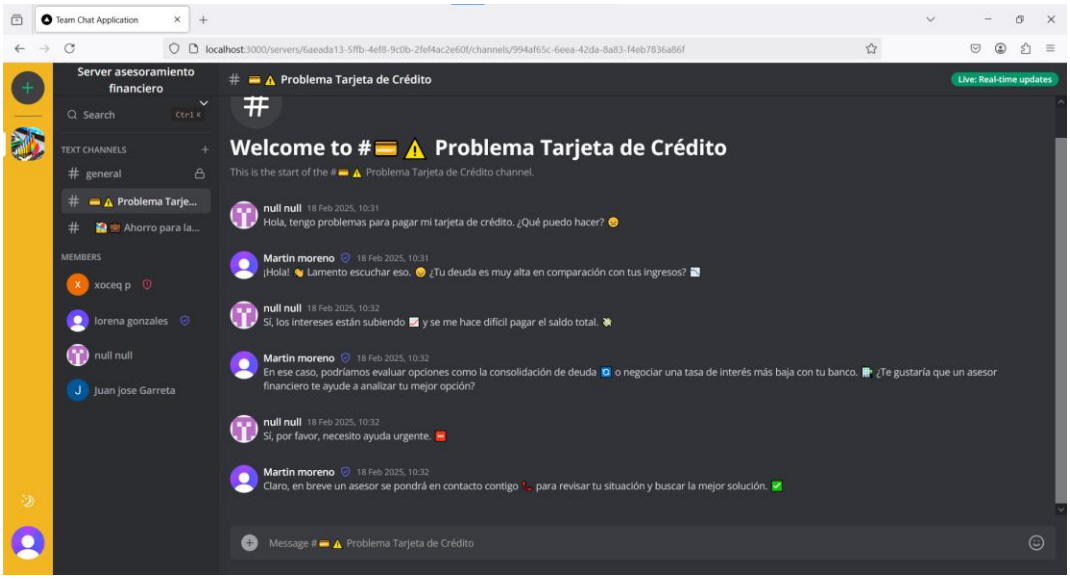


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 108 muestra el canal titulado “Problemas con tarjeta de crédito”, un espacio especializado dentro de la plataforma de asesoramiento financiero personalizado y en tiempo

real.

Figura 108
Canal problemas con tarjeta de crédito

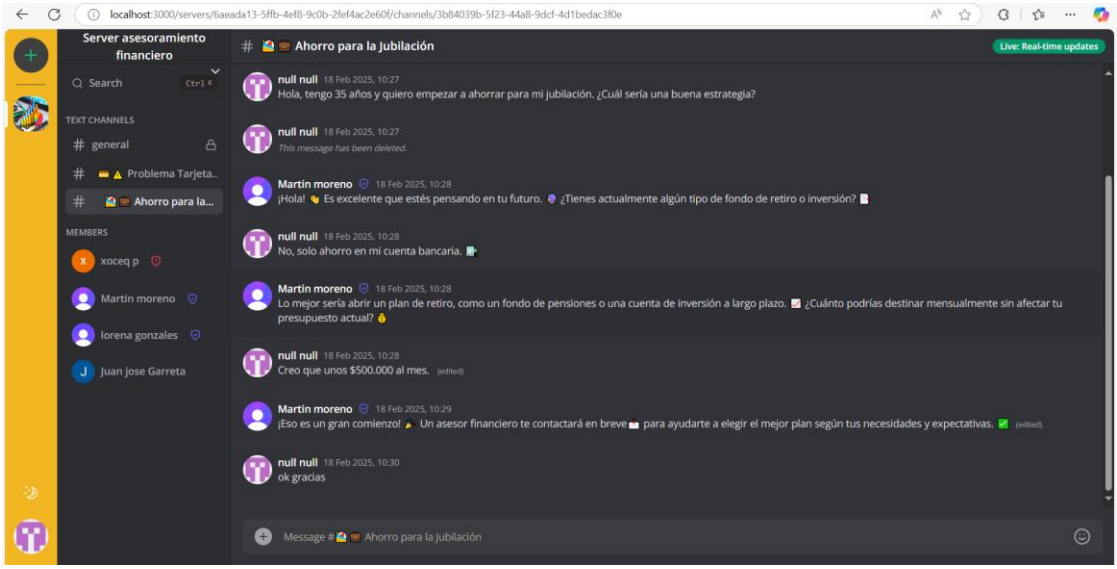


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 109 presenta el canal “Ahorro para la vejez”, un espacio dedicado al intercambio de información, consejos y estrategias relacionadas con la planificación financiera a largo plazo.

Figura 109

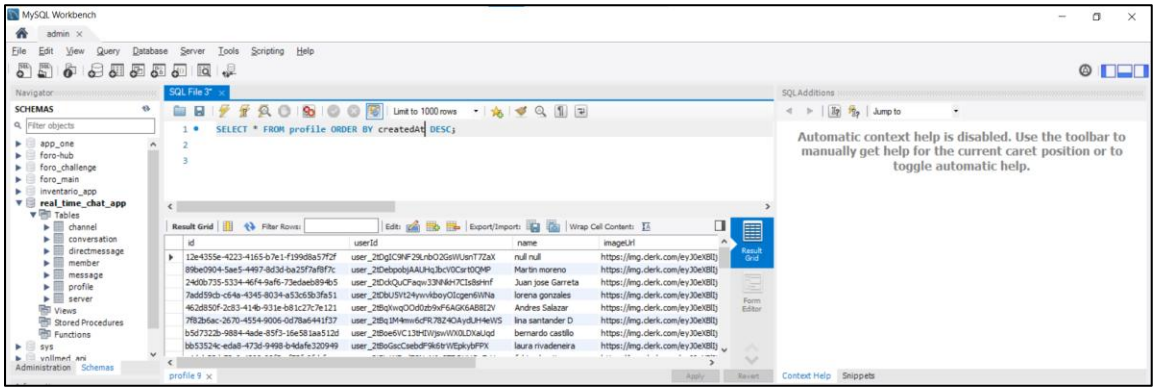
Ahorro para la vejez



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 110 muestra la tabla de perfiles, componente esencial en la base de datos de la plataforma de asesoramiento financiero.

Figura 110
Tabla perfiles

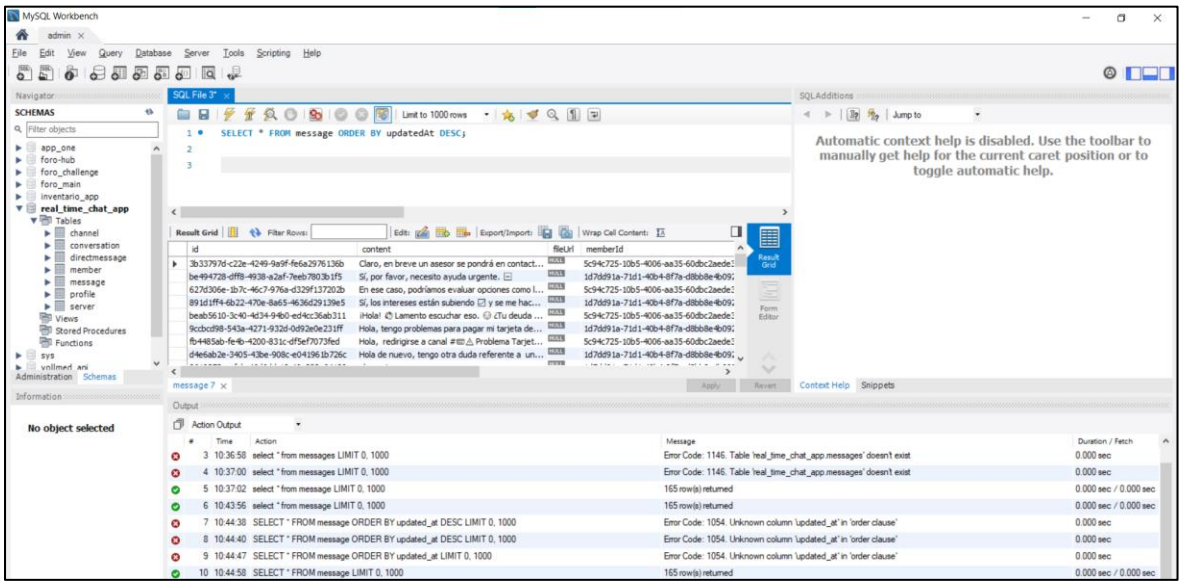


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La Figura 111 muestra la tabla de mensajes, encargada de registrar todas las interacciones escritas que se generan dentro de los canales de la plataforma de asesoramiento financiero.

Figura 111

Tabla mensajes



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Tabla 37
Productos Esperados

RESULTADO/PRODUCTO ESPERADO	INDICADOR	BENEFICIARIO
Implementación de *Real-Time Chat App* en atención al cliente en tiempo real.	• Tiempo medio de respuesta < 2 minutos. • Disponibilidad del servicio 24/7 y tasa de conexiones exitosas. • Número de quejas resueltas en primera interacción. • Cumplimiento con normativas de protección de datos (ISO 18295-1).	• Empresas con centros de contacto que buscan optimizar la atención al cliente. • Clientes que requieren soporte inmediato y confiable. • Empresas que buscan mejorar su reputación y servicio al cliente. • Usuarios y empresas que manejan información sensible.
Sistema de emergencias médicas Comunicación rápida y efectiva en situaciones de emergencia médica	• Tiempo medio de respuesta 1 minuto. • Clasificación eficiente por prioridad según el Triage. • Atención dentro de los tiempos establecidos.	• Servicios de emergencia médica. • Personal médico y paramédico. • Pacientes en estado crítico.
Plataforma de soporte técnico **Resolución eficiente de problemas técnicos en tiempo real.**	• Tiempo medio de respuesta < 2 minutos. • Tasa de resolución de problemas en la primera interacción. • Número de tickets resueltos por técnico. • Cantidad de consultas y registros almacenados en MySQL.	• Clientes que requieren asistencia técnica inmediata. • Empresas y usuarios finales que dependen de servicios digitales. • Equipos de soporte técnico que usan la aplicación. • Empresas que gestionan múltiples incidencias técnicas. • Usuarios concurrentes de la plataforma de soporte.

	• Latencia del chat en tiempo real con Socket.io. • Nivel de satisfacción del usuario en encuestas post-chat.	• Clientes que usan la plataforma para asistencia.
Plataforma de asesoramiento financiero personalizado y en tiempo real	• Tiempo medio de respuesta < 1 min. • Número de consultas atendidas por día. • Tiempo promedio de resolución de consultas < 5 min.	• Clientes que buscan asesoramiento financiero rápido y personalizado. • Usuarios de la plataforma (clientes y asesores financieros). • Empresas financieras que optimizan su atención al cliente. • Asesores financieros y clientes que dependen de la plataforma en tiempo real. • Firmas de asesoría financiera y entidades bancarias.

Nota. La tabla resume soluciones en tiempo real, sus métricas clave y los beneficiarios principales en contextos de atención al cliente, emergencias médicas y soporte técnico.
Fuente. Autoría propia.

Repositorio

Este repositorio contiene el código fuente del proyecto aplicado, donde se desarrollan e implementan sus principales funcionalidades.

https://github.com/XoceQ/Real-time-chat-app-master

Implementación De Aplicación, Consulte Los Videos Adjuntos

En esta sección se presenta la implementación de la aplicación, detallando su despliegue

y funcionamiento. Para una guía más completa, consulte los videos adjuntos en la siguiente lista de reproducción:

https://www.youtube.com/playlist?list=PL6Qsn9JHmcWoDn7XANblpNtythMI8qIG5

Conclusiones

La implementación del servidor central permitió gestionar de manera eficiente la comunicación entre múltiples clientes conectados. Gracias a su diseño optimizado, se logró estabilidad y escalabilidad en la transmisión de datos.

En cuanto a la seguridad, se estableció un sistema de autenticación que protege la información de los usuarios y restringe accesos no autorizados. Esto garantiza un entorno seguro para el almacenamiento y transmisión de datos sensibles.

La comunicación entre clientes y servidor se optimizó mediante protocolos eficientes, reduciendo la latencia y mejorando la confiabilidad del sistema. Esto permite una experiencia de usuario más fluida y sin interrupciones en la conexión.

Se desarrolló una interfaz intuitiva y accesible, priorizando la facilidad de uso y la navegación clara. Este diseño mejora la experiencia del usuario y minimiza la curva de aprendizaje para nuevos participantes en el sistema.

Finalmente, se realizaron pruebas exhaustivas para garantizar la estabilidad, seguridad y rendimiento de la aplicación. Estas pruebas permitieron la identificación y corrección de errores, asegurando un funcionamiento óptimo antes de su implementación final.

Recomendaciones

Para mejorar significativamente la experiencia de los usuarios en la aplicación de chat en tiempo real, es fundamental implementar un sistema de notificaciones eficiente y bien estructurado. Un mecanismo de notificaciones en tiempo real permitirá mantener a los usuarios informados sobre nuevos mensajes, menciones y actividad relevante en las salas de chat, garantizando que no se pierdan interacciones importantes. Para ello, se pueden emplear modales de notificación emergentes, burbujas de aviso y alertas sonoras configurables según las preferencias del usuario. Además, sería ideal incorporar una bandeja de notificaciones persistente, donde los usuarios puedan revisar el historial de alertas y gestionar sus interacciones de manera más organizada.

La integración de funciones de audio y video representa un avance significativo en la comunicación dentro de la aplicación. Incorporar videollamadas grupales y la posibilidad de iniciar conversaciones de audio en un solo clic hará que la experiencia sea más inmersiva y cercana a una interacción presencial. Para garantizar una alta calidad en la transmisión de audio y video, es crucial implementar tecnologías de compresión avanzada y protocolos de transmisión optimizados, como WebRTC, que permiten minimizar el consumo de ancho de banda sin comprometer la calidad.

Asimismo, la opción de compartir pantalla durante las videollamadas añadirá un valor significativo a la aplicación, facilitando presentaciones, soporte técnico en vivo y colaboración en tiempo real. Esta funcionalidad será especialmente útil en entornos empresariales, educativos y de trabajo remoto, donde la comunicación visual juega un papel clave en la productividad y la toma de decisiones.

En definitiva, la combinación de un sistema de notificaciones eficaz y la integración de

audio y video en alta calidad no solo optimizará la experiencia del usuario, sino que también aumentará la versatilidad y competitividad de la aplicación en el mercado de las plataformas de comunicación en tiempo real.

Referencias Bibliográficas

Abramov, D. (2017). Desarrollo de interfaces de usuario con React. Documentación oficial de React.

AENOR. (2025). AENOR. Obtenido de https://revista.aenor.com/337/calidad-internacional-para-los-contact-centres.html

Amazon. (2024). Amazon. Obtenido de https://aws.amazon.com/es/what-is/restful-api/

Amazon. (2024). Amazon. Obtenido de https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/

Amazon. (2024). Amazon. Obtenido de https://aws.amazon.com/es/what-is/middleware/

Amazon. (2024). Amazon. Obtenido de https://aws.amazon.com/es/what-is/sso/

Aplyca. (2025). Aplyca. Obtenido de https://www.aplyca.com/blog/blog-que-es-vercel-desarrollar-previsualizar-enviar

AWS. (2024). What is a Cloud Database? Obtenido de https://aws.amazon.com/es/what-is-aws/

Axarnet. (2024). Axarnet. Obtenido de https://axarnet.es/blog/como-crear-base-de-datos-mysql

Bauman, Z. (2013). Modernidad líquida. Fondo de Cultura Económica.

Bitrix24. (2024). Bitrix24. Obtenido de https://www.bitrix24.co/uses/herramientas-de-gestion-de-archivos.php

Blancarte, O. (2021). Reactive programming. Obtenido de https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor

Bottega. (2020). Devcamp. Obtenido de https://devcamp.es/que-es-backend/

Cadenhead, T. (2015). Socket.IO Cookbook. . Packt Publishing.

Castells, M. (2009). Comunicación y poder. Alianza Editorial.

Choi, D. (2020). Full-Stack React, TypeScript, and Node : Build Cloud-ready Web Applications

Using React 17 with Hooks and GraphQL. . Packt Publishing.

Clerk. (2024). Clerk. Obtenido de https://clerk.com/

Cloudflare. (2024). Cloudflare Inc. Obtenido de https://www.cloudflare.com/es-es/learning/ssl/transport-layer-security-tls/

Crcom. (2025). Crcom. Obtenido de https://www.crcom.gov.co/sites/default/files/webcrc/micrositios/documents/Documento-revision-Reglamento-Tecnico-Redes-Internas-Telecomunicaciones-RITEL.pdf

Cypress. (2025). Cypress. Obtenido de https://docs.cypress.io/app/get-started/why-cypress

Decidesoluciones. (2024). Decide soluciones. Obtenido de https://decidesoluciones.es/arquitectura-de-microservicios/

Declan Williamson, R. O. (2023). WebTransport and WebSockets: An Empirical Analysis of Connection Time, Message Response,and Payload Efficiency. Obtenido de https://ieeexplore- ieee-org.bibliotecavirtual.unad.edu.co/stamp/stamp.jsp?tp=&arnumber=10162060

Developer.mozilla.org. (2024). Developer.mozilla.org. Obtenido de https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing

Dhiwise. (2024). Dhiwise. Obtenido de https://www.dhiwise.com/post/zod-and-react-a-perfect-match-for-robust-validation

DotNetCurry.com. (s.f.). ASP.NET MVC 5: Using a Simple Repository Pattern for Performing Database Operations . Obtenido de https://www.dotnetcurry.com/aspnet-mvc/1155/aspnet-mvc-repository-pattern-perform-database-operations

Dribba. (2024). Dribba. Obtenido de https://www.dribba.com/post/desarrollo-rapido-apps-flutter-shadcnui

Duarte, M. (2020). La importancia de la comunicación en tiempo real en entornos colaborativos.

Easyappcode. (2020). Easyappcode. Obtenido de https://www.easyappcode.com/patron-de-diseno-mvc-que-es-y-como-puedo-utilizarlo

Eitca. (2024). Eitca. Obtenido de https://es.eitca.org/cybersecurity/eitc-is-acc-advanced-classical-cryptography/elliptic-curve-cryptography/elliptic-curve-cryptography-ecc/examination-review-elliptic-curve-cryptography-ecc/what-are-the-steps-involved-in

Ejsmont, A. (2015). Escalabilidad web para ingenieros de startups.

Fette, I. (2011). The WebSocket Protocol. . IETF RFC 6455.

Fielding, R. T. (1999). Hypertext Transfer Protocol -- HTTP/1.1. . RFC 2616. . .

Flanagan, D. (2011). HTML5: The Definitive Guide. . O'Reilly Media.

Flanagan, D. (2011). HTML5: The Definitive Guide. . O'Reilly Media.

Flanagan, D. (2011). JavaScript: The Definitive Guide. O'Reilly Media.

Función Pública. (2025). Función Pública. Obtenido de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=77888

Función Pública. (2025). Función Pública. Obtenido de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=36841

Función Pública. (2025). Función Pública. Obtenido de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=190426

Función Pública. (2025). Función Pública. Obtenido de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=40032

Geeksforgeeks. (2024). Geeksforgeeks. Obtenido de https://www.geeksforgeeks.org/what-is-the-event-driven-programming-paradigm/

Geraint, P. (2014). Desarrollo de aplicaciones web en tiempo real con Socket.io. . Packt

Publishing.

GitHub. (2021). Documentation GitHub para proyectos de código abierto. GitHub.

Github. (2025). Docs.github. Obtenido de https://docs.github.com/es/actions/about-github-actions/understanding-github-actions

Hejlsberg, A. T. (2012). Especificación del lenguaje TypeScript. Microsoft.

Hennessy, J. L. (2017). Computer Architecture: A Quantitative Approach. Morgan Kaufmann. .

Herron, D. (2016). Node.js Web Development . Third Edition. Packt Publishing.

Herron, D. (2016). Node.js Web Development . Third Edition. Packt Publishing.

Hostinger. (2024). Hostinger. Obtenido de https://www.hostinger.co/tutoriales/que-es-react

Hostinger. (2024). Hostinger. Obtenido de https://www.hostinger.co/tutoriales/que-es-react

hostinger. (2024.). hostinger. Obtenido de https://www.hostinger.co/tutoriales/que-es-react

IONOS. (2022). IONOS . Obtenido de https://www.ionos.mx/digitalguide/servidores/know-how/irc/

IONOS. (2023). IONOS. Obtenido de https://www.ionos.mx/digitalguide/servidores/know-how/file-transfer-protocol/

Jackson, S. (2020). Cómo Slack utiliza WebSockets para mejorar la productividad en tiempo real.

JetBrains. (2025). Getting Started with IntelliJ IDEA. Obtenido de JetBrains: https://www.jetbrains.com/help/idea/getting-started.html

Keepcoding. (2024). Keepcoding. Obtenido de https://keepcoding.io/blog/html-fallback/

Keepcoding. (2024). Keepcoding. Obtenido de https://keepcoding.io/blog/que-es-jwt/

Kinsta. (2025). Kinsta. Obtenido de https://kinsta.com/es/blog/jest/

Kumar, R. (2021). Scaling MySQL with Vitess: Overview and Use Cases.

Kurose, J. F. (2017). Computer Networking: A Top-Down Approach. (7th ed.). . Pearson.

Laragon. (2024). Laragon: A Portable, Fast, Lightweight and Free PHP Development Environment. Obtenido de https://laragon.org/

Lubbers, P. G. (2010). Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development. . Apress.

Manz.dev. (s.f.). lenguajejs.com. Obtenido de https://lenguajejs.com/javascript/peticiones-http/xhr/

Mason, H. (2018). Socket.IO para desarrolladores de Node.js: Construcción de aplicaciones en tiempo real. . Packt Publishing.

Mason, H. (2018). Socket.IO para desarrolladores de Node.js: Construcción de aplicaciones en tiempo real. . Packt Publishing.

Meneses Guevara, C. D., Rosado Gómez, A., & Quintero Duarte, A. (2012). DESARROLLO ÁGIL DE SOFTWARE APLICANDO PROGRAMACIÓN EXTREMA. Obtenido de https://revistas.ufps.edu.co/index.php/ingenio/article/view/2003/1959

Microsoft. (2024). Microsoft learn. Obtenido de https://learn.microsoft.com/es-es/windows/dev-environment/javascript/nextjs-on-wsl

Microsoft. (2025). Windows 10: Overview. Obtenido de https://www.microsoft.com/en-us/windows/windows-10-specifications

MinAmbiente. (2025). MinAmbiente. Obtenido de https://www.minambiente.gov.co/politica-de-proteccion-de-datos-personales/#:~:text=Ley%20de%20Protecci%C3%B3n%20de%20Datos,de%20naturaleza%20p%C3%BAblica%20o%20privada.

MinAmbiente. (2025). MinAmbiente. Obtenido de https://www.minambiente.gov.co/politica-de-

proteccion-de-datos-personales/#:~:text=Ley%20de%20Protecci%C3%B3n%20de%20Datos,de%20naturaleza%20p%C3%BAblica%20o%20privada.

minsalud. (2025). Obtenido de https://www.minsalud.gov.co/salud/PServicios/Paginas/triage.aspx

Mintic. (2025). Mintic. Obtenido de https://www.mincit.gov.co/minindustria/estrategia-transversal/regulacion/1-1-subsistema-nacional-de-la-calidad#:~:text=El%20Subsistema%20Nacional%20de%20Calidad,al%20consumidor%20garant%C3%ADas%20e%20informaci%C3%B3n%2C

Nardi, B. A. (2002). The Place of Face-to-Face Communication in Distributed Work. MIT Press.

Neoattack. (2024). Neoattack. Obtenido de https://neoattack.com/neowiki/orm/

Nordpass. (2024). Nordpass. Obtenido de https://nordpass.com/es/blog/password-salt/

Pandasecurity. (2024). Pandasecurity. Obtenido de https://www.pandasecurity.com/es/mediacenter/cifrado-aes-guia/

Phpmyadmin. (2025). Phpmyadmin. Obtenido de https://www.phpmyadmin.net/

PlanetScale. (2025). PlanetScale Database as a Service. Obtenido de https://planetscale.com/

Prisma. (2024). Prisma Data, Inc. . Obtenido de https://www.prisma.io/docs/orm/overview/introduction

Rai, R. (2013). Socket.io Real-time Web Application Development. Packt Publishing.

Rai, R. (2013). Socket.io Real-time Web Application Development. . OAIster.

Rasband, J. (2020). Node.js y Socket.IO: Construcción de aplicaciones en tiempo real. . O'Reilly Media.

Raymond, E. S. (2001). La catedral y el bazar: Reflexiones sobre Linux y el software de código

abierto por un revolucionario accidental. . O'Reilly Media.

Real, M. i. (2024). Make it real. Obtenido de https://guias.makeitreal.camp/docs/setup/entorno-local#:~:text=Un%20entorno%20local%20se%20refiere,entorno%20de%20producci%C3%B3n

Refactoring. (2024). Refactoring. Obtenido de https://refactoring.guru/es/design-patterns/singleton

Refactoring. (2024). Refactoring. Obtenido de https://refactoring.guru/es/design-patterns/observer

Render. (2025). Render. Obtenido de https://render.com/docs/deploys

responsively.app. (2025). Responsively.app. Obtenido de https://responsively.app/

Rosenberg, J. (2019). WebSocket y el futuro de la comunicación en tiempo real en la web. En Programming the Web with Java (pp. 301-320).

Rosenberg, J. (2019). WebSocket y el futuro de la comunicación en tiempo real en la web. En Programming the Web with Java (pp. 301-320). Wiley.

Schmidt, K. (2018). The Challenge of Communication in Distributed Work. Computers in Human Behavior, 21(2)., 117-127.

Schneier, B. (2015). Datos y Goliat: Las batallas ocultas para recopilar sus datos y controlar su mundo. . W. W. Norton & Company.

Shadcn/ui. (2024). Shadcn / ui. Obtenido de https://ui.shadcn.com/docs/react-19

Silberschatz, A. G. (2018). Operating System Concepts.

Simões, C. (2021). ITDO. Obtenido de https://www.itdo.com/blog/que-es-node-js-y-para-que-sirve/

Ssldragon. (2024). Ssldragon. Obtenido de https://www.ssldragon.com/blog/sha1-sha2-sha256-

sha-512/

Stallings, W. (2017). Criptografía y seguridad en redes: Principios y práctica. Pearson.

Stallings, W. (2019). Computer Organization and Architecture: Designing for Performance.

Superfinanciera. (2025). Superfinanciera. Obtenido de https://www.superfinanciera.gov.co/publicaciones/10115018/regulacion-financiera-en-colombia-ha-sido-habilitante-para-las-fintech/

Superviz, & Pearson. (2024). Superviz. Obtenido de https://superviz.com/design-pattern-4-publisher-subscriber-pattern-for-frontend-developers

Universidad del Pireo . (2013). Análisis, diseño e implementación de una aplicación web colaborativa en tiempo real. Obtenido de https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/8331/Papatheodorou_Antonios%20Nikola%20os.pdf?sequence=1&isAllowed=y

Uploadthing. (2024). Obtenido de https://uploadthing.com/

W3C. (2021). WebSocket. Obtenido de https://www.w3.org/

W3C. (2021). WebSocket. Obtenido de https://www.w3.org/

Wikipedia. (2024). Integrated Development Environment. Obtenido de Wikipedia: https://en.wikipedia.org/wiki/Integrated_development_environment

ZDNet. (2021). What is Windows 10? A Complete Guide to Microsoft's Operating System. .

Zúñiga, F. G. (2022). Arsys. Obtenido de https://www.arsys.es/blog/jsonp-api#:~:text=JSONP%20o%20JSON%20con%20padding,en%20la%20mayor%C3%ADa%20de%20sistemas.

Apéndices

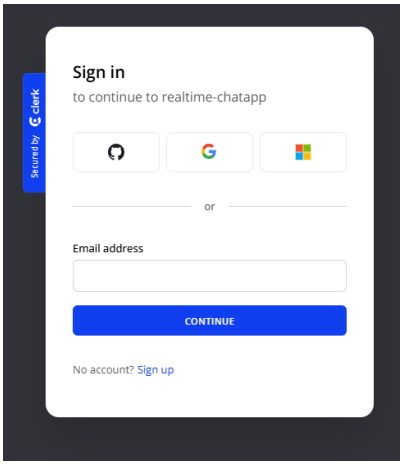
Apéndice A

Manual De Usuario

Vista "Iniciar Sesión"

La Figura 112 muestra la interfaz de inicio de sesión de la aplicación.

Figura 112
Inicio de sesión

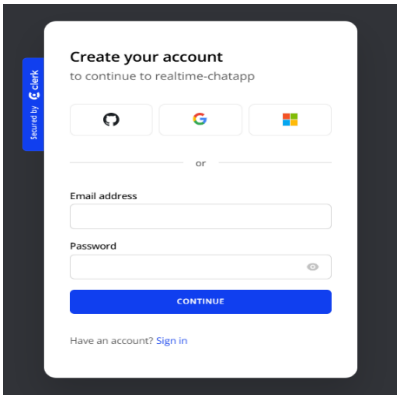


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Vista "Registrarse"

La Figura 113 muestra la pantalla de registro de nuevos usuarios dentro de la aplicación.

Figura 113
Registrarse

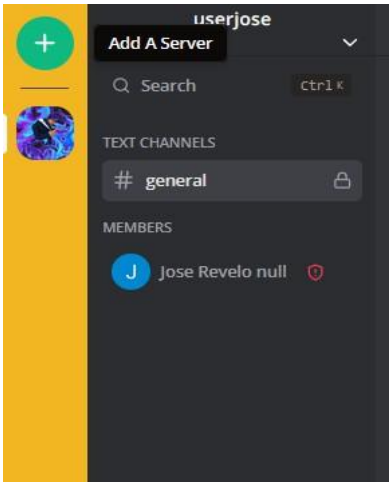


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal "Añadir Servidor"

La Figura 114 muestra la interfaz para la opción "Crear servidor" dentro de la aplicación.

Figura 114
Crear Servidor

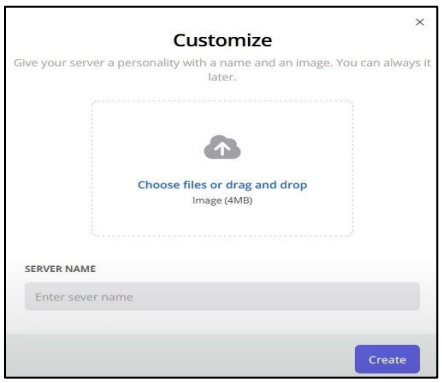


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal "Crear Servidor"

La Figura 115 presenta el modal de creación de servidor, una ventana emergente que guía al usuario a través del proceso de configuración inicial de un nuevo servidor dentro de la aplicación.

Figura 115
Modal crear servidor

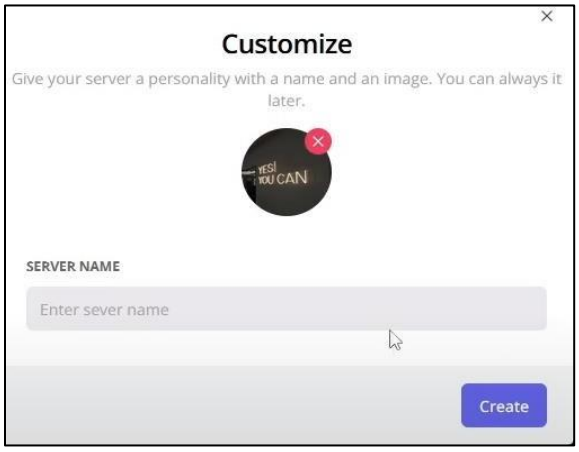


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Servidor Creado

La Figura 116 muestra el proceso de configuración: un servidor creado exitosamente dentro de la plataforma.

Figura 116
Servidor Creado

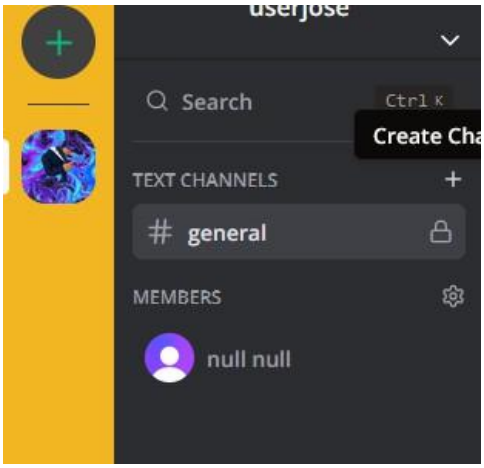


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Botón "Crear Canal"

La Figura 117 muestra la interfaz destinada a la opción "Crear canal" dentro de un servidor previamente creado en la aplicación.

Figura 117
Crear Canal botón

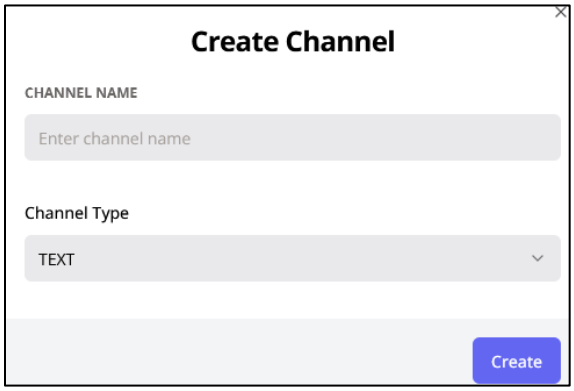


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal "Crear Canal"

La Figura 118 presenta el modal de creación de canal, una ventana emergente que permite a los usuarios configurar rápidamente un nuevo canal dentro de un servidor.

Figura 118
Crear Canal modal

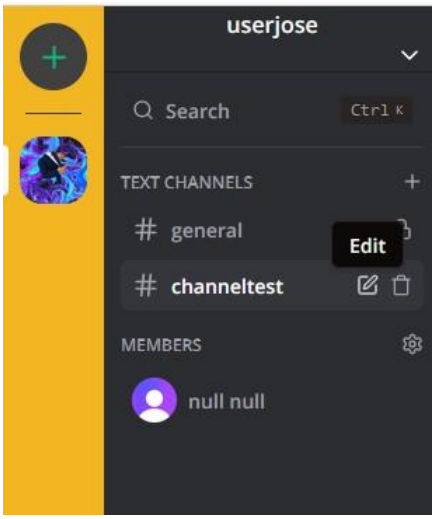


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Botón "Editar Canal"

La Figura 119 muestra la interfaz de edición de canal, donde los usuarios pueden modificar las configuraciones de un canal ya existente dentro del servidor.

Figura 119
Editar Canal



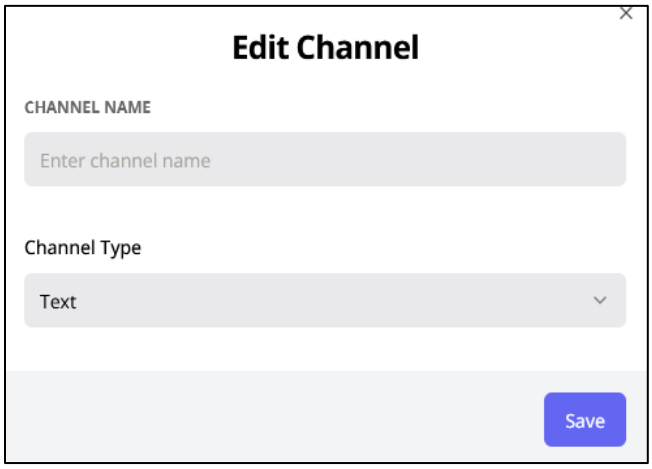
Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal "Editar canal"

La Figura 120 muestra el modal de edición de canal, una ventana emergente que facilita

la modificación rápida de los parámetros de un canal previamente creado.

Figura 120
Modal Editar Canal



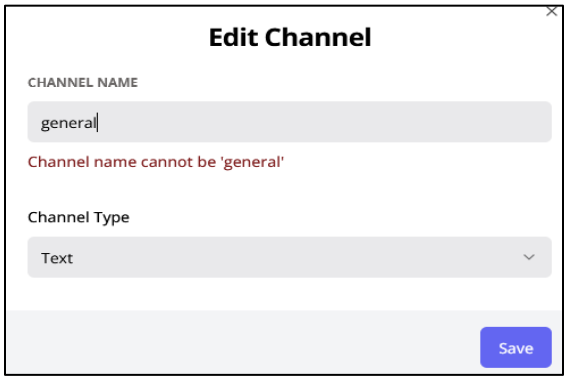
Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Existe una validación que no permite crear un canal con el nombre “general” debido a que es el canal creado por defecto.

Validación “Editar Canal”

La Figura 121 muestra el proceso de validación al editar un canal, el cual se activa una vez el usuario intenta guardar los cambios realizados en el modal de edición.

Figura 121
Validación

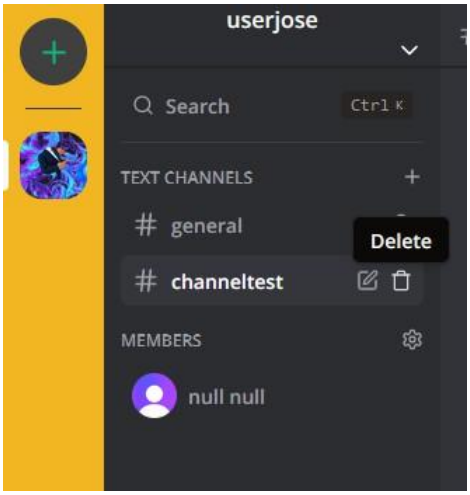


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Botón “Eliminar Canal”

La Figura 122 muestra el botón “Eliminar canal”, una opción que permite al usuario remover permanentemente un canal dentro del servidor.

Figura 122
Eliminar Canal

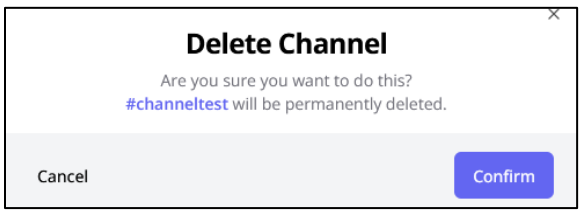


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal “Eliminar Canal”

La Figura 123 muestra el modal de confirmación para eliminar un canal, una ventana emergente que aparece cuando un usuario hace clic en el botón “Eliminar canal”.

Figura 123
Modal Eliminar Canal



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

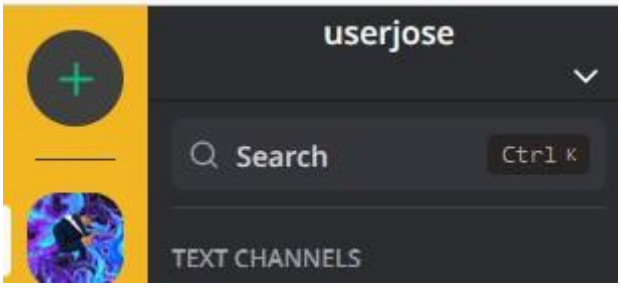
Haciendo uso de teclas Ctrl + k se puede acceder a la barra de búsqueda para buscar nombres de canales y miembros creados en el servidor.

Barra De Búsqueda

La Figura 124 presenta la barra de búsqueda dentro de la aplicación, una herramienta

esencial que permite a los usuarios localizar rápidamente canales, mensajes o miembros dentro del servidor.

Figura 124
Barra de búsqueda

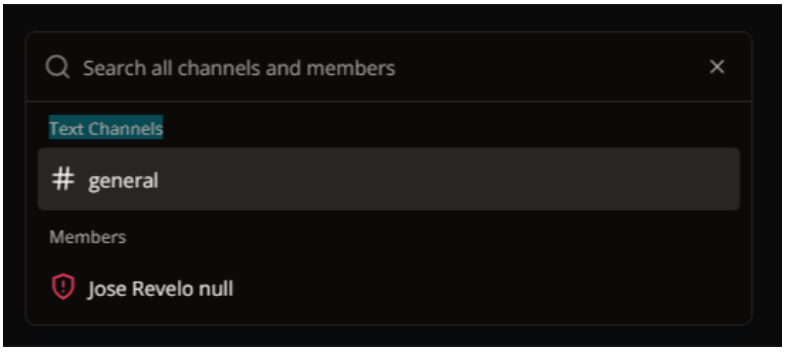


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal "Barra De Búsqueda"

La Figura 125 muestra el modal de la barra de búsqueda, una ventana emergente que aparece al activar la función de búsqueda dentro de la aplicación.

Figura 125
Modal barra de búsqueda

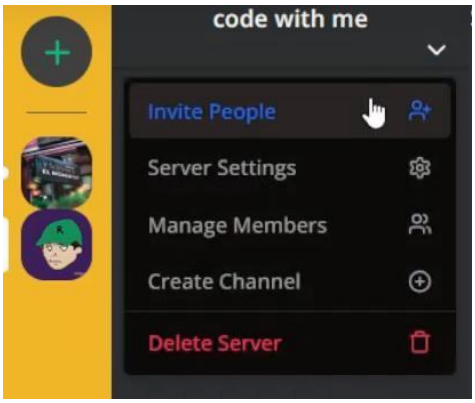


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Opción "Invitar Personas"

Cuando un usuario hace clic en un enlace de invitación, se debe pegar el enlace en una página nueva y se le redirige al servidor creado, que incluye funciones para validar el código de invitación y añadir el usuario al servidor de chat correspondiente.

Figura 126
Invitar personas

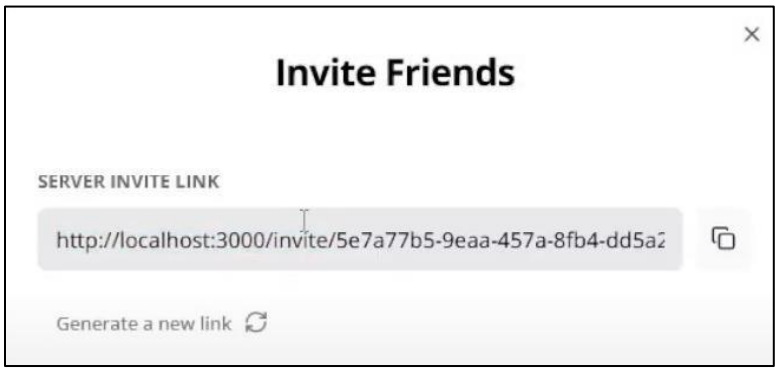


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Enlace De Invitación

La figura 127 muestra un ejemplo visual de una invitación generada automáticamente por el sistema, como parte del flujo de interacción del usuario.

Figura 127
Invitación generada



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Opción "Ajustes Del Servidor"

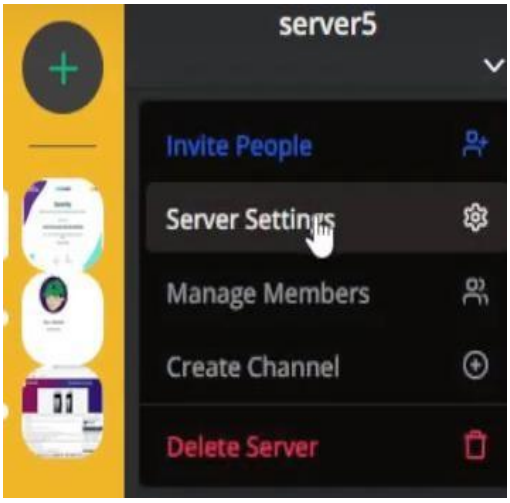
Dentro de las opciones para configurar el servidor se encuentran: Actualizar nombre e imagen del servidor.

Botón "Ajustes Del Servidor"

La figura 128 presenta el botón de Ajustes, una funcionalidad clave dentro de la interfaz

del sistema que permite al usuario acceder a opciones de configuración.

Figura 128
Botón ajustes

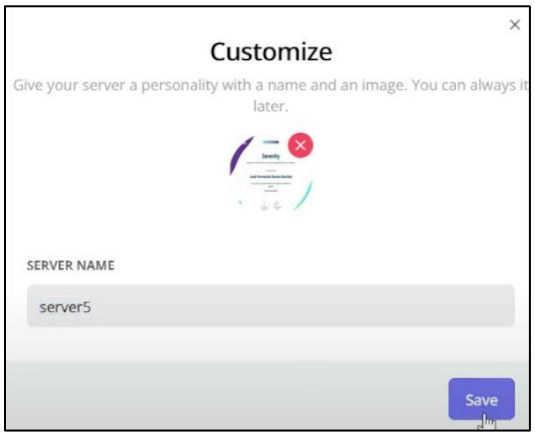


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal "Ajustes Del Servidor"

La figura 129 muestra la sección de Ajustes del servidor, donde se configuran parámetros fundamentales para el funcionamiento del sistema a nivel de infraestructura.

Figura 129
Ajustes del servidor



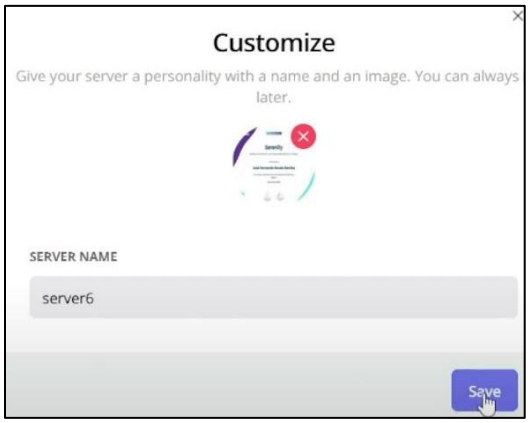
Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Nombre Servidor Editado

La figura 130 ilustra el proceso de edición del nombre del servidor dentro del panel de

configuración.

Figura 130
Nombre servidor editado

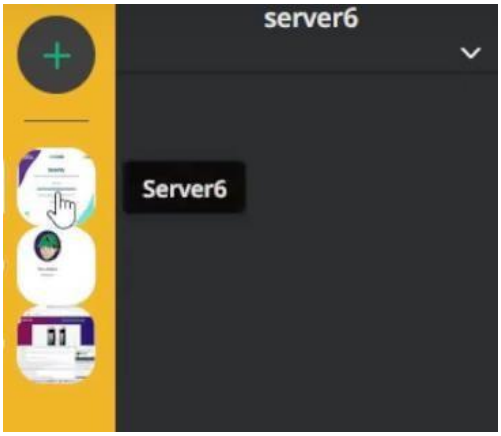


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Nuevo Nombre De Servidor

La figura 131 muestra la interfaz donde se visualiza el nuevo nombre del servidor tras haber sido editado.

Figura 131
Nuevo nombre del servidor



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

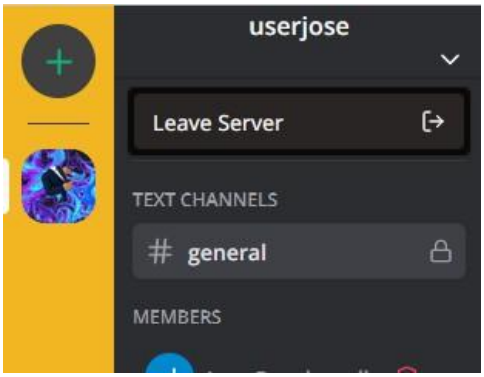
Opción Salir Del Servidor

(Disponible para Rol usuario y moderador)

La figura 132 muestra la opción Salir del servidor, una función que permite al usuario

desconectarse o abandonar de forma segura una instancia del servidor.

Figura 132
Salir del servidor

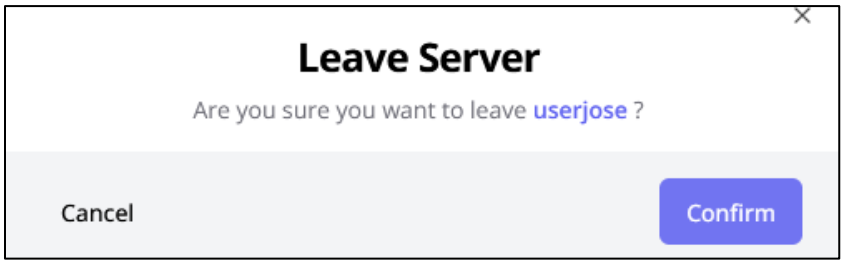


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal "Salir Del Servidor"

La figura 133 presenta el modal de confirmación para salir del servidor, una ventana emergente que solicita al usuario confirmar su decisión antes de abandonar la instancia del servidor.

Figura 133
Modal Salir del servidor



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Cabecera Del Chat

La figura 134 muestra la cabecera del chat, un elemento visual que encabeza la interfaz de conversación y que generalmente incluye información clave como el nombre del canal o contacto, el estado de conexión, y accesos rápidos a opciones adicionales.

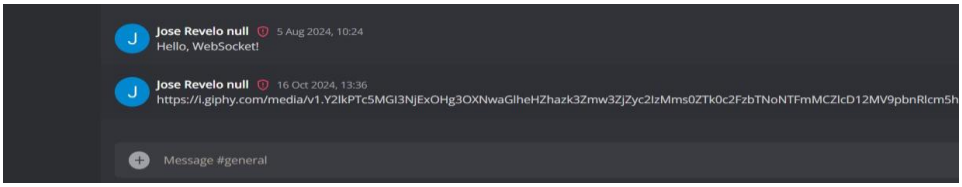
Figura 134
Cabecera del chat

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Entrada De Chat

La figura 135 muestra el área de entrada de chat, donde el usuario puede redactar y enviar mensajes dentro de una conversación.

Figura 135
Entrada de chat



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Opción "Editar Mensaje"

La figura 136 muestra la funcionalidad de editar mensaje dentro del entorno de chat. Esta opción permite al usuario modificar un mensaje ya enviado, corrigiendo errores o actualizando la información sin necesidad de enviar uno nuevo.

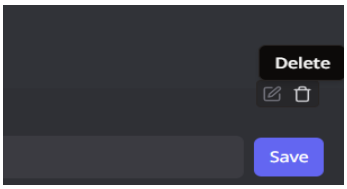
Figura 136
Editar mensaje

Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Opción "Eliminar Mensaje"

La figura 137 presenta la opción de eliminar mensaje, una funcionalidad que permite al usuario remover mensajes previamente enviados dentro de una conversación.

Figura 137
Opción eliminar mensaje

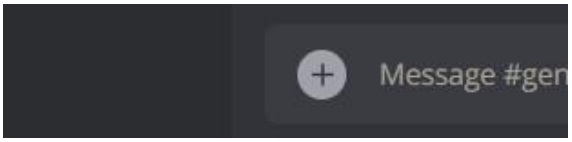


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Botón "Añadir Un Archivo Adjunto"

La figura 138 muestra el botón de archivo adjunto, una herramienta que permite a los usuarios incorporar documentos, imágenes u otros tipos de archivos a una conversación de chat.

Figura 138
Botón archivo adjunto

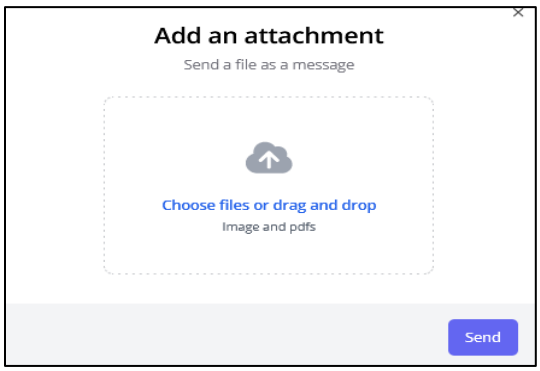


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal "Añadir Un Archivo Adjunto"

La figura 139 muestra el proceso de añadir un archivo adjunto dentro de una conversación.

Figura 139
Añadir un archivo adjunto

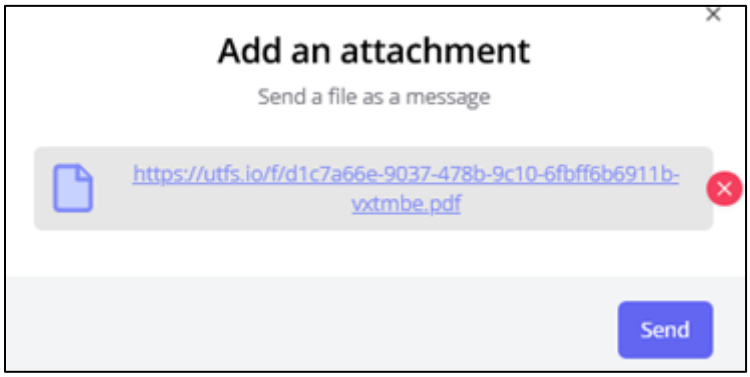


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Archivo Pdf Añadido

La figura 140 muestra un archivo PDF añadido exitosamente a la conversación.

Figura 140
PDF Añadido

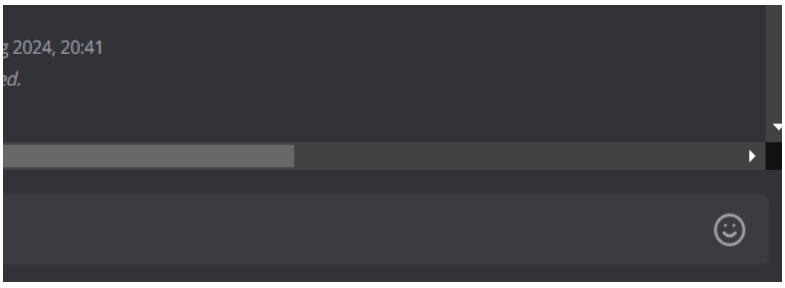


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Barra De Emojis

La figura 141 presenta la barra de emojis, una herramienta integrada en la interfaz de chat que permite al usuario expresar emociones, reacciones o ideas de forma visual y dinámica.

Figura 141
Barra de emojis

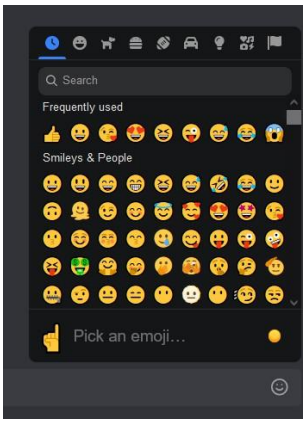


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Barra De Emojis Modo Oscuro

La figura 142 muestra la barra de activación del modo oscuro, una opción dentro de la interfaz que permite al usuario cambiar la apariencia del sistema a una paleta de colores oscuros.

Figura 142
Barra modo oscuro

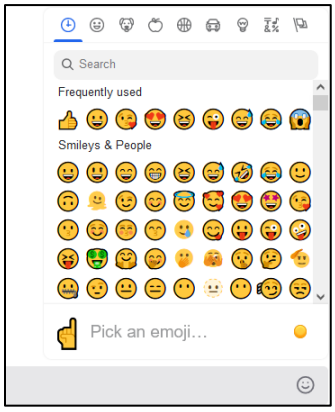


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Barra De Emojis Modo Claro

La figura 143 muestra la barra de activación del modo claro, una opción que permite al usuario establecer una apariencia visual basada en colores claros dentro de la interfaz del sistema.

Figura 143
Barra modo claro

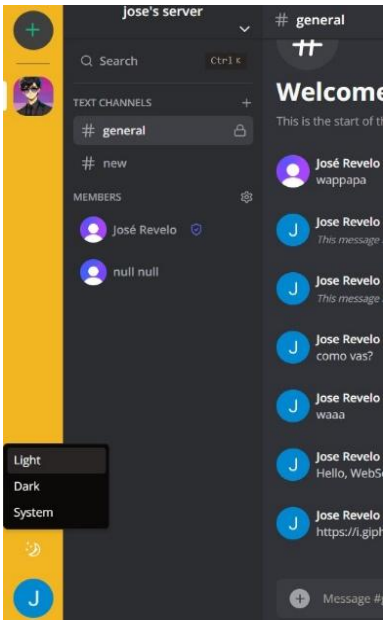


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Switch

La figura 144 presenta un switch o interruptor deslizante, un componente de la interfaz gráfica que permite activar o desactivar funciones específicas de forma rápida y visual.

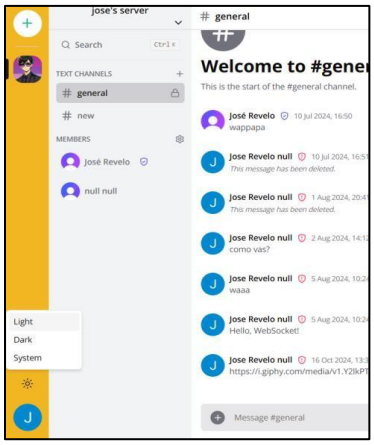
Figura 144
Switch



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

La figura 145 muestra el switch para activar el modo claro, una herramienta dentro de la interfaz que permite al usuario cambiar la apariencia del sistema a una temática con colores claros.

Figura 145
Switch modo claro



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Apéndice B

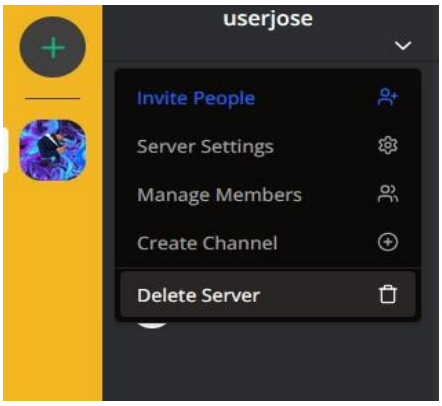
Manual De Administrador

Opción "Eliminar Servidor"

(Solo disponible para Rol Administrador)

La figura 146 presenta el botón eliminar servidor, una opción crítica que permite al usuario borrar de forma permanente un servidor del sistema.

Figura 146
Botón eliminar servidor

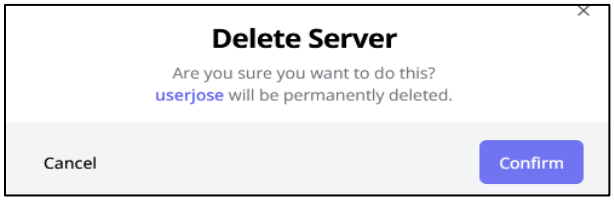


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Modal "Eliminar Servidor"

La figura 147 muestra el modal de confirmación para eliminar servidor, una ventana emergente que solicita al usuario verificar su decisión antes de proceder con la eliminación definitiva del servidor.

Figura 147
Modal eliminar servidor

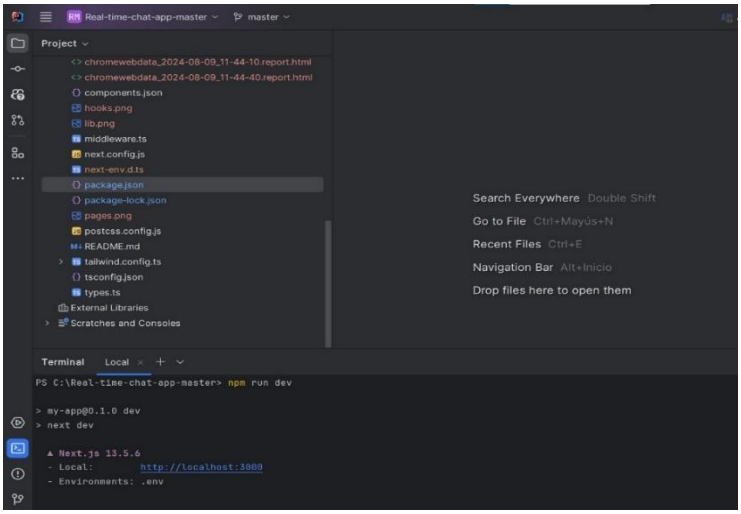


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Servidor En Ejecución

La figura 148 muestra un servidor en ejecución, lo que indica que el servicio se encuentra activo y funcionando correctamente.

Figura 148
Servidor en ejecución



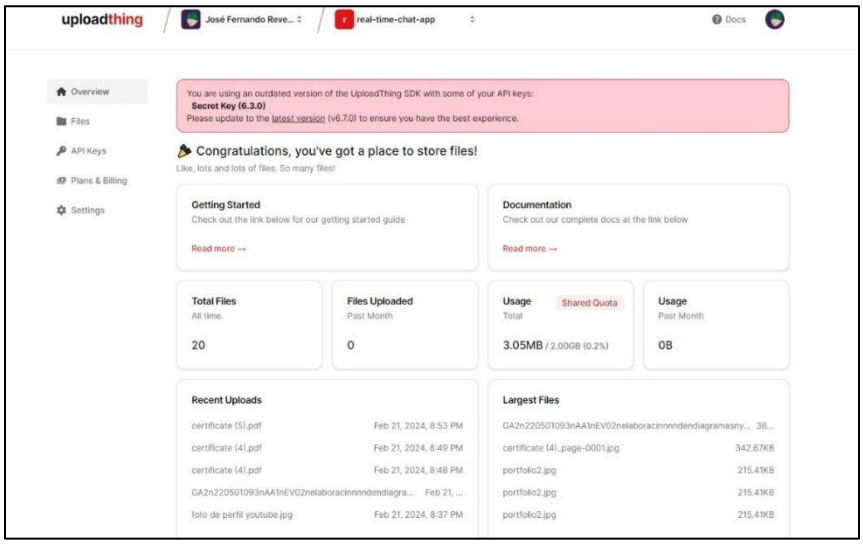
Nota. Captura de pantalla propia.
Fuente. Autoría propia.

UploadThing: Servicio utilizado para gestionar la carga y almacenamiento de archivos e imágenes.

Cuenta Uploadthing Creada

La figura 149 muestra la vista de cuenta en UploadThing, una sección donde el usuario puede gestionar su perfil, configuraciones y credenciales dentro de esta plataforma de carga y gestión de archivos.

Figura 149
Cuenta UploadThing

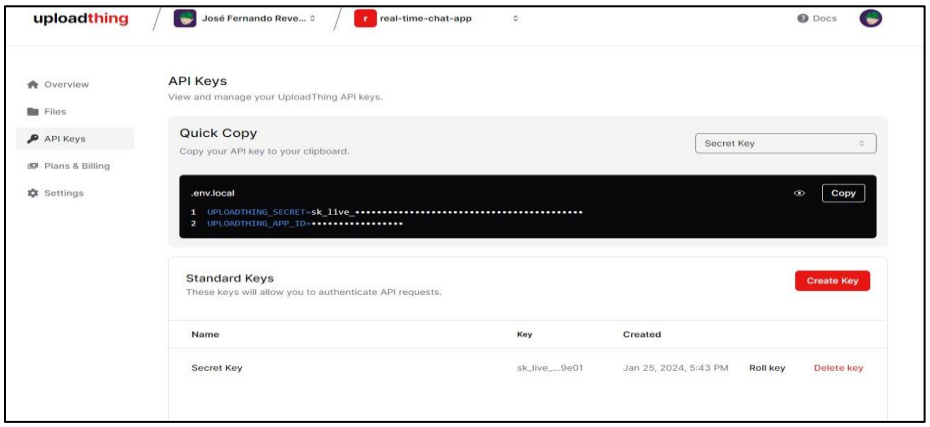


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Secret Key Generada

La figura 150 muestra la clave secreta (Secret key) de UploadThing, un identificador confidencial utilizado para autenticar y autorizar solicitudes desde aplicaciones externas hacia los servicios de UploadThing.

Figura 150
Secret key UploadThing



Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Prisma (v3.15.0): ORM utilizado para interactuar de manera eficiente con la base de

datos MySQL.

WebSockets con socket.io (v4.5.1): Implementado para habilitar la comunicación en tiempo real en la aplicación.

Autenticación

Clerk.com se utilizó en el proyecto de aplicación de chat en tiempo real para la autenticación y gestión de usuarios. Proporcionó un componente SignOutButton y un gancho useAuth que se puede utilizar para acceder al ID del usuario actual. Este ID de usuario se puede utilizar para identificar a los usuarios en el servidor Node.js.

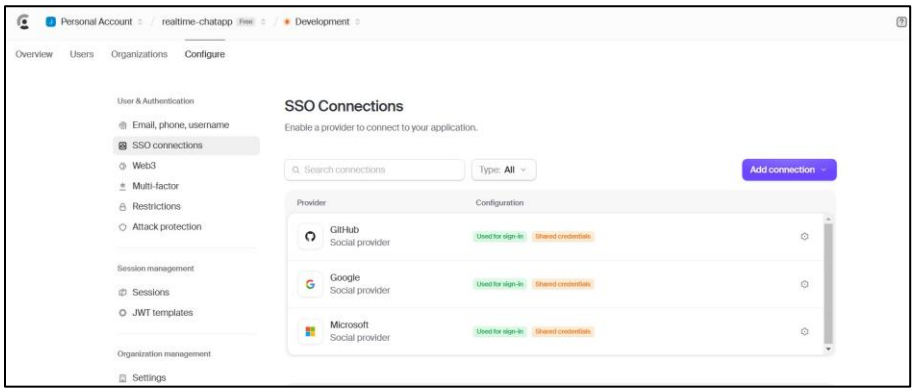
Los componentes SignedIn, SignedOut y RedirectToSignIn se utilizaron para gestionar el estado de autenticación del usuario y redirigir a los usuarios a la página de inicio de sesión si es necesario.

El inicio de sesión único (SSO) es una solución de autenticación que permite a los usuarios iniciar sesión en varias aplicaciones y sitios web con una única autenticación de usuario. Dado que en la actualidad los usuarios acceden con frecuencia a aplicaciones directamente desde sus navegadores, las organizaciones dan prioridad a las estrategias de administración de acceso que mejoran tanto la seguridad como la experiencia del usuario (Amazon., 2024).

Configuración Inicio De Sesión

La figura 151 muestra la sección de configuración de autenticación, donde se definen los parámetros necesarios para controlar el acceso seguro al sistema.

Figura 151
Configuración Autenticación

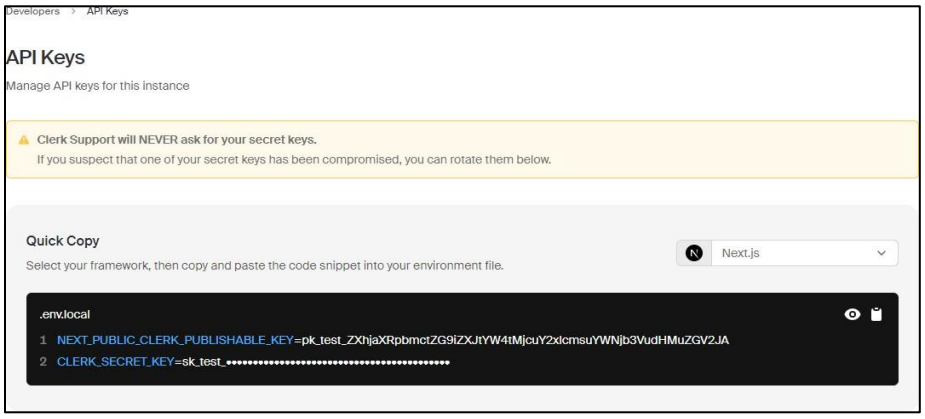


Nota. Captura de pantalla propia.
Fuente. Autoría propia.

Secret Key Generada Clerk Credenciales

La figura 152 muestra la clave secreta (Secret key) de Clerk, una credencial privada utilizada para autenticar y autorizar operaciones seguras entre una aplicación y los servicios de autenticación de Clerk.

Figura 152
Secret key Clerk



Nota. Captura de pantalla propia.
Fuente. Autoría propia.