UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD DE CIENCIAS Y TECNOLOGÍAS COMPUTACIONALES

Título: “Componente de mensajería instantánea en la nube para la plataforma de telecomunicaciones Platel”

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Claudia Suárez González

Tutores:

Ing. Faviannys Gamez Abad

Lic. Rafael Batista Sánchez

MSc. Lorenzo Mario Quintero Ortiz

La Habana, diciembre de 2023

“Año 65 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Claudia Suárez González, soy el autor del Trabajo de Diploma “Componente de mensajería instantánea en la nube para la plataforma de telecomunicaciones Platel” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo a su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración de autoría en La Habana a los 5 días del mes de diciembre del año 2023.

_____

Claudia Suárez González

Autor

_____	_____	_____
Lic. Rafael Batista Sánchez	**Ing. Faviannys Gamez Abad**	**MSc. Lorenzo Mario Quintero Ortiz**
Tutor	**Tutor**	**Tutor**


"La independencia no es una bandera, o un himno, o un escudo. La independencia no es cuestión de símbolos. La independencia depende del desarrollo, la independencia depende de la tecnología, depende de la ciencia en el mundo de hoy"
Fidel Castro Ruz

DEDICATORIA

Este trabajo de diploma va dedicado a mis padres que siempre me han apoyado hasta cuando me negaba a seguir adelante. Su amor, paciencia, comprensión y principalmente su sacrificio fueron los pilares que me sostuvieron para lograr culminar satisfactoriamente la última parte de mi vida como estudiante.

AGRADECIMIENTOS

Primeramente, les agradezco a Dios y la Virgen que siempre me acompañan.

A mi papá y mi mamá por todo el sacrificio durante estos largos 5 años y su amor incondicional, por sus palabras de aliento dándome fuerzas cuando creí que era imposible cumplir esta meta, por cuidarme siempre, aunque de muy lejos; no me alcanzan las palabras para poder agradecerles todo lo que han hecho por mí.

A mi hermana por cuidar de ellos mientras yo no estoy, por soportarme en mis días malos y nunca dejarme sola.

A mis padrinos que siempre se han preocupado y me han apoyado mostrando en todos estos años interés en mis resultados y confianza en mí.

A mi mejor amiga por siempre estar para mí sin importar qué, por brindarme su hombro y escucharme las horas que necesitaba hablar.

A mi compañero de vida Danys que me ha ayudado tanto en estos años juntos.

A mis amigos Addiel y Antony por ayudarme en mis estudios.

A todos los familiares, amistades y profesores que han marcado este camino tan difícil que es la universidad.

Muchas gracias.

RESUMEN

El presente trabajo titulado “Componente de mensajería instantánea en la nube para la plataforma de telecomunicaciones Platel” tiene como objetivo desarrollar un componente de mensajería instantánea que permita el intercambio de mensajes de textos, audios y multimedia entre los usuarios registrados en tiempo real y de forma segura ya que la plataforma de telecomunicaciones Platel se encuentra carente de estas funcionalidades. Para ello se utiliza el protocolo XMMP y se incluye una interfaz de usuario, un sistema de chat en tiempo real, así como la capacidad de enviar, recibir archivos adjuntos y mensajes de audio. Para su desarrollo se utiliza la metodología AUPvUCI asociada al lenguaje de modelado UML, Visual Paradigm como herramienta CASE y Visual Studio Code como entorno de desarrollo utilizando Python 3.11 y JavaScript como lenguaje de programación, Django 4.2 como framework y como Gestor de Bases de Datos PostgresSQL 14. Se implementan las funciones básicas y se realizan las pruebas correspondientes para la solución propuesta.

Palabras clave:

Mensajería instantánea, protocolo XMPP, sistema de chat.

ABSTRACT

The present work entitled "Instant messaging component in the cloud for the Platel telecommunications platform" aims to develop an instant messaging component that allows the exchange of text, audio and multimedia messages between registered users in real time and safe since the Platel telecommunications platform lacks these functionalities. This uses the XMMP protocol and includes a user interface, a real-time chat system, as well as the ability to send and receive attachments and audio messages. For its development, the AUPvUCI methodology associated with the UML modeling language is used, Visual Paradigm as a CASE tool and Visual Studio Code as a development environment using Python 3.11 and JavaScript as a programming language, Django 4.2 as a framework and as a PostgresSQL Database Manager. 14. The basic functions are implemented and the corresponding tests are carried out for the proposed solution.

Keywords:

Instant messaging, XMPP protocol, chat system.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) son el conjunto de tecnologías que permiten la comunicación, el procesamiento y la gestión de información a través de dispositivos digitales; avanzan cada día siendo un instrumento valioso para la vida cotidiana de la sociedad actual. Estas tecnologías han revolucionado la manera en que las organizaciones y las empresas operan, permitiendo una mayor eficacia, profesionalidad y colaboración.

Uno de los avances más importantes en el mundo empresarial ha sido la llamada Industria 4.0 que se basa en la automatización, la conectividad y la digitalización de los procesos de producción (Industria 4.0, implicaciones, certezas y dudas en el mundo laboral., 2022); lo que permite la recopilación y análisis de datos en tiempo real integrando las tecnologías de comunicación en el proceso de producción para lograr una mayor eficacia y flexibilidad en la industria.

En la Industria 4.0 se encuentran las Comunicaciones Unificadas siendo un parte importante que integra las diferentes formas de comunicación como la telefonía, la mensajería instantánea y las videoconferencias en una sola plataforma. Esto permite una colaboración más eficiente y efectiva entre los trabajadores en diferentes ubicaciones geográficas. Utilizando las tecnologías VoIP[1] en llamadas telefónicas y videoconferencias a través de internet. Gracias al constante avance de la tecnología y las telecomunicaciones permite que sea posible que se utilicen protocolos de redes inalámbricas en los dispositivos autónomos comunicándose entre sí en el área personal y empresarial.

En esta nueva era tecnológica, las empresas de telecomunicaciones buscan mejorar su capacidad de comunicación para enfrentar los desafíos de la industria moderna y mejorar su competitividad en el mercado utilizando FreeSWITCH como herramienta de software libre que permite integrar diferentes formas de comunicación en una sola plataforma. Además, con la capacidad de integrarse a otras aplicaciones empresariales, permite a las empresas automatizar los procesos de comunicación y mejorar la eficiencia de las operaciones comerciales reduciendo los costos al hacer uso de la tecnología VoIP en

[1] Voz Sobre Protocolo de Internet es una tecnología que permite realizar y recibir llamadas de voz a través de la red.

llamadas y videoconferencias. Convirtiéndose en una herramienta valiosa para las empresas.

Como la Empresa de Tecnologías de la Información para la Defensa (XETID) que implementa soluciones informáticas, la autonomía y las comunicaciones. Esta empresa garantiza que las organizaciones y entidades estén comunicadas de forma segura e inmediata con varias soluciones de comunicaciones unificadas, integradas a sus sistemas tecnológicos. Basándose en la tecnología de código libre FreeSWITCH y desarrollo propio XETID crea la plataforma de telecomunicaciones Platel para conectar en tiempo real dos o más personas en distintos lugares permitido en un entorno colaborativo la comunicación segura.

La plataforma de telecomunicaciones Platel en tiempo de pandemia por COVID-19 fue la alternativa más viable para mantener los niveles de producción a distancia, aumentando la demanda en las empresas, gobiernos y entidades presupuestadas. Con el despliegue de los módulos Platel-PBX, Platel-Videoconferencia, Platel-Call Center las organizaciones se han ahorrado tiempo, dinero y combustible, facilitando el teletrabajo[2] causando un impacto favorable como solución en la coyuntura que se atravesaba.

Platel-Comunicaciones Unificadas es la plataforma integral de comunicaciones que ofrece diversas funcionalidades unificando llamadas telefónicas, videoconferencias, mensajería unificada[3], seguridad, supervisión y notificaciones en una misma interfaz de usuario única donde se puede acceder y compartir información en tiempo real permitiendo el trabajo colaborativo. Emplea protocolos que son capaces de interoperar en otros sistemas empresariales optimizando la administración de los servicios de comunicación al utilizar una arquitectura de software segura y flexible que permite configurar sus funciones e interfaces a las necesidades de cada cliente. En la actualidad Platel-CU tiene deficiencia con la funcionalidad de comunicación entre los usuarios autenticados en la plataforma, teniendo que utilizar aplicaciones ajenas para establecer la comunicación implicando la pérdida de tiempo, así como la privacidad necesaria para esta institución.

[2] Trabajo que una persona realiza para una empresa desde un lugar alejado de la sede de esta (habitualmente su propio domicilio), por medio de un sistema de telecomunicación.
[3] Servicio que pone a disposición del cliente todos los mensajes en un formato al que se puede acceder en cualquier momento, sin importar la vía de acceso al mensaje.

Dada esta **situación problemática** se plantea el siguiente **problema a resolver:** ¿Cómo realizar el intercambio de mensajes de textos, audios, documentos y multimedia; para la comunicación interna entre un usuario o varios registrado en la plataforma Platel?

Definiendo como **objeto de estudio**: sistemas web de comunicación y el **campo de acción** centrado en sistemas web de mensajería instantánea. Definiendo como **objetivo general** desarrollar un módulo de mensajería instantánea para la plataforma de telecomunicaciones Platel que permite el intercambio de mensajes de textos, audios y archivos adjuntos entre los usuarios registrados.

Detallando en los siguientes **objetivos específicos**:

1. Elaborar el estado del arte acerca de las tendencias tecnológicas actuales y estrategias utilizadas para la mensajería instantánea en la nube, definiendo las tecnologías y herramientas a usar durante el desarrollo del módulo.
2. Realizar el modelado del diseño el componente de mensajería instantánea para la plataforma Platel.
3. Implementar el diseño del módulo de mensajería instantánea.
4. Validar el correcto funcionamiento mediante las pruebas necesarias.

Para cumplir con el objetivo propuesto y una mayor organización en el desarrollo del presente trabajo se plantean las siguientes **tareas de investigación**:

1. Realización de un estudio del arte relacionado a la mensajería instantánea en la nube.
2. Identificación de soluciones similares existentes en el ámbito nacional como internacional.
3. Verificación de las herramientas, tecnologías, lenguajes y metodología.
4. Identificación de los artefactos que genera la metodología seleccionada.
5. Descripción de los artefactos que genera la metodología seleccionada.
6. Implementación del componente con los requisitos propuestos.
7. Validación del correcto funcionamiento mediante las pruebas necesarias.

Los **métodos de investigación científica** utilizados en el desarrollo de la investigación son los siguientes**:**

<u>Métodos teóricos:</u>

- **Analítico-sistémico:** es una técnica de análisis que involucra la descomposición de un objeto o fenómeno en sus componentes o relaciones para una mejor comprensión del mismo. Este método permite al analista descubrir características,

funciones, conceptos y relaciones importantes facilitando el estudio de la investigación sobre los sistemas de mensajería instantánea en la nube.

- **Histórico-lógico:** se enfoca en el análisis de la trayectoria y evolución del fenómeno a lo largo del tiempo para descubrir las leyes generales que rigen su funcionamiento y desarrollo. Este método permite una comprensión profunda del fenómeno y cómo ha sido influenciado por factores sociales, tecnológicos y económicos a lo largo del tiempo los sistemas de mensajería instantánea en la nube.

Métodos empíricos:

- **Observación:** una herramienta fundamental en la investigación científica y se utiliza para recopilar información de manera consciente y orientada hacia un objetivo determinado. Se utiliza para percibir directamente la realidad y actualidad de los sistemas de mensajería instantánea en el mundo para entender mejor el comportamiento de los usuarios y cómo interactúan con el sistema en la vida real.
- **Entrevista:** se enfoca en recopilar información necesaria de las personas especializadas en el tema del propio negocio. Es principalmente útil cuando la descripción y bibliografía dada por el proyecto no proporciona suficiente información.

Con esta investigación se pretende obtener un componente que permita dotar a la plataforma de telecomunicaciones Platel de un mecanismo de mensajería instantánea en la nube, garantizando una adecuada velocidad de respuesta y seguridad para dicha plataforma como **posible resultado**.

Estructura del trabajo por capítulos:

Capítulo I: Fundamentación teórica

En este capítulo se realiza una revisión del estado del arte del tema de investigación y se describen los principales conceptos y referentes teórico-metodológicos relacionados con el objeto de estudio. Se verifican y seleccionas las herramientas, tecnologías y metodología de desarrollo de software para la solución propuesta.

Capítulo II: Diseño de la propuesta de solución

En este capítulo con el fin de comprender el contexto del negocio se realiza el análisis y diseño de la solución propuesta al problema científico. Se identifican los requisitos funcionales y no funcionales de la solución y se generan los artefactos correspondientes a

la metodología de desarrollo seleccionada. Además, se especifican los detalles de la implementación de la solución y se planifica su desarrollo.

Capítulo III: Pruebas o validación de la solución propuesta

En este capítulo se implementa y valida la propuesta de solución ejecutando casos de pruebas en los niveles de unidad, integración, sistema y aceptación. Evaluando la calidad del software y la especificación de los requisitos y el diseño realizado.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

En este capítulo se hace un estudio de los conceptos relacionados al problema para su mejor entendimiento y analizar con mayor flexibilidad las posibles soluciones. Se describen las tecnologías utilizadas para la confección del componente a desarrollar. Evaluando las características del sistema y analizando la arquitectura y metodología que se utiliza en éste.

1.1 Conceptos asociados a la investigación.

La comunicación entre las personas en la era digital actual ha sido una gran revolución, brindando posibilidades más eficientes y convenientes. Sin embargo, para comprender este fenómeno y su impacto en la sociedad, es crucial explorar los conceptos claves relacionados con esta tecnología en contrastante evolución.

- *Industria 4.0:* una nueva forma de industrialización en donde se interrelacionan los sistemas operacionales de fabricación, los procesos productivos y las Tecnologías de Información y Comunicación (TIC), debidamente fundamentadas en el Internet de las cosas, orientada a un nuevo modelo de negocio mucho más tecnológico, con manejo de información actualizada y en tiempo real (Industria 4.0, implicaciones, certezas y dudas en el mundo laboral., 2022).
- FreeSWITCH: es una plataforma de telefonía de código abierto a nivel de operador implementada como un agente de usuario consecutivo. Gracias a este diseño, puede realizar una gran cantidad de tareas diferentes, desde una PBX hasta un conmutador de tránsito, conversión TTS (texto a voz), host de conferencias de audio y video e incluso un teléfono VoIP y más (Inc., 2023).
- *Comunicaciones unificadas:* son redes IP de última generación que permiten la integración de todos los componentes separados de comunicación en una experiencia de usuario homogénea, eficiente y productiva (Durango, et al., 2018).
- *Redes inalámbricas:* son redes que utilizan ondas de radio para conectar los dispositivos, sin la necesidad de utilizar cables de ningún tipo (Redes Inalámbricas, 2017).
- XMMP: Extensible Messaging and Presence Protocol es un protocolo abierto que se creó para ser usado en sistemas de mensajería instantánea originalmente, está basado en XML (Foundation, 1999).

- *Mensajería instantánea:* es un servicio online de comunicación que permite a dos o más personas enviar y recibir mensajes de texto en tiempo real (Florentín, 2022).
- *Aplicación web:* son programas informáticos (software[4]) construidos con tecnologías web como HTML, CSS, JavaScript y PHP, entre otras. Estas permiten a los usuarios a interactuar a través de un servidor de navegador web a la vez que facilitan a los desarrolladores poder crear y administrar contenido en línea. Las aplicaciones web permiten a los usuarios compartir información, colaborar en proyectos y realizar tareas desde cualquier lugar desde una conexión a Internet o Intranet[5] (Arquitectura).

Con el análisis de los conceptos anteriores se logra entender las definiciones de las tecnologías y su evolución en la sociedad con la presente investigación. En la revisión de estos conceptos se comprende el funcionamiento de las tecnologías de comunicación más específicamente de la mensajería instantánea en sistemas basados en la web para plataformas de comunicaciones unificadas como es Platel.

Platel

Desarrollada sobre software libre, es una plataforma integral de comunicaciones que ofrece diversos servicios como voz, datos, video, mensajería unificada, seguridad, supervisión y notificaciones. Para ello, emplea protocolos estándares e interfaces flexibles, con la capacidad de interoperar con otros sistemas empresariales.

Esta herramienta cuenta con una arquitectura de software segura, robusta y flexible, que permite configurar sus funciones y servicios de acuerdo con las características y necesidades de cada cliente, para optimizar la administración de los servicios de comunicación. Es importante destacar que esta soporta una amplia gama de tecnologías IP, interfaces para la conexión a la red telefónica pública conmutada y a la internet.

En su núcleo cuenta con elementos estándares de comunicación IP y no IP, sobre el cual evolucionan servicios de valor agregado que valorizan la solución y la adaptan a cualquier escenario. Ofrece mensajería unificada, VoIP, IVR, videoconferencia, centro de contacto, presencia integrada, agenda y calendario integrado.

[4] Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora. https://dle.rae.es
[5] Red digital de uso interno en una organización. https://dle.rae.es

La plataforma puede adecuarse a las necesidades de cada organización. La experiencia en la aplicación de esta solución por diversas organizaciones ha permitido desarrollar tres modalidades generales de servicio, un paquete básico, uno estándar y un paquete empresarial.

1.2 Análisis de soluciones similares

Hoy en día existen diversos sistemas de mensajería instantánea que se utilizan en varios sistemas operativos con diferentes fines. Se escogieron algunos de estos sistemas de mensajería instantánea tanto los que operan en aplicaciones web, desktop y Android para el análisis y comprensión de soluciones similares a la propuesta:

Todus es una plataforma de mensajería instantánea hecha en Cuba desarrollada por jóvenes ingenieros de la Universidad de Ciencias Informáticas (UCI). A través de esta aplicación disponible para los dispositivos con sistema operativo Android, puedes comunicarte con tus amigos y familiares en tiempo real, enviar mensajes, imágenes, notas de voz y mucho más, de forma sencilla e intuitiva. La plataforma ofrece seguridad y confianza, garantizando que tus conversaciones siempre serán privadas, mediante un servicio de alta disponibilidad y rapidez (Informaticas, 2023). No se encuentran más detalles de la aplicación en el sitio oficial.

Prosody es una solución de mensajería instantánea de código abierto y altamente escalable que utiliza el protocolo XMPP. Es fácil de usar, se integra con otros servicios web y plataformas de mensajería, es altamente personalizable y ofrece una amplia variedad de características de seguridad y privacidad. Utiliza el lenguaje de programación Lua, lenguaje de scripting ligero y de alto rendimiento. Es buena elección por su simplicidad, eficiencia y capacidad para integrarse fácilmente. También utiliza C y C++ para optimizar el rendimiento en partes críticas del código (IM, 2008).

WhatsApp es una aplicación de mensajería instantánea de propiedad de Facebook puede utilizarse como aplicación web, desktop y aplicación móvil. Se lanzó en 2009 y ha crecido hasta convertirse en una de las aplicaciones de mensajería instantánea más populares del mundo, con más de 2 mil millones de usuarios activos mensuales. Aunque es una aplicación de mensajería instantánea muy popular, también ha sido objeto de críticas por cuestiones de privacidad y seguridad, lo que ha llevado a algunos usuarios a buscar alternativas de mensajería instantánea más seguras. WhatsApp emplea una combinación de lenguajes de programación como Erlang, C, C++, Java, Objetive-C, Swift y JavaScript junto con diversos protocolos y tecnologías como XMPP, MQTT, WebRTC; para desarrollar su plataforma de

mensajería instantánea (Inc., 2009). No se encuentra documentación pública del funcionamiento por detrás (backend).

WhatsApp ofrece una amplia variedad de características, incluyendo:

- *Mensajería instantánea.*
- *Llamadas y videollamadas.*
- *Grupos.*
- *Integración con otros servicios.*
- *Funciones de seguridad.*
- *Compatibilidad con múltiples plataformas.*

Telegram es una aplicación de mensajería instantánea de código abierto disponible para dispositivos móviles, así como para computadoras como aplicación web o de escritorio, que ofrece una amplia variedad de características, incluyendo mensajería instantánea, chats secretos, grupos, canales, bots, opciones de personalización y multiplataforma. También se ha ganado una gran reputación por su enfoque en la privacidad y la seguridad, lo que lo convierte en una opción popular entre aquellos que buscan una alternativa más privada y segura a otras aplicaciones de mensajería instantánea. Proporciona una API abierta y documentada que permite a los desarrolladores crear aplicaciones y bots. Su backend desarrollado principalmente en Erlang, está diseñado para ser escalable y tolerante a fallos por lo que permite manejar un gran número de usuarios, utiliza su propio protocolo MTProto para garantizar la seguridad y privacidad de las comunicaciones. Los mensajes y archivos se almacenan en la nube por lo que los usuarios pueden acceder a ellos desde múltiples dispositivos. También utiliza los lenguajes de programación C/C++, Python, Java, Kotlin, Swift, JavaScript, HTML/CSS, Rust y Go (LLP, 2013).

Pidgin anteriormente llamado Gaim, es el cliente universal de mensajería instantánea, multiplataforma, capaz de conectarse a múltiples redes (multiprotocolo) y cuentas (multicuenta) de manera simultánea utilizando la biblioteca Libpurple para la comunicación y conexión con los diversos protocolos de mensajería. Ofrece una interfaz de usuario donde se puede ver y gestionar los contactos y conversaciones en múltiples redes de mensajería en una sola ventana (Abreo, 2010). Tiene diversas características como:

- *Conversaciones mostradas en pestañas.*
- *Posibilidad de conectarse a varias redes simultáneamente.*
- *Registro de conversaciones.*

- *Permite el reemplazo de los nombres de los contactos de la lista.*
- *Posibilidad de transparencia para las ventanas de contactos y de conversación mediante un plugin.*
- *Transferencia de archivos.*
- *Corrector ortográfico en español.*

Converse.js es una aplicación web de mensajería instantánea de código abierto que utiliza el protocolo XMPP para la comunicación entre el cliente y el servidor lo que garantiza una comunicación en tiempo real entre los usuarios. Esta desarrollado principalmente con JavaScript y es fácil de integrar en sitios web y aplicaciones existentes, ofrece una interfaz de usuario moderna y personalizable, y es compatible con una amplia variedad de plataformas. Además, ofrece características de seguridad y privacidad avanzadas, como la encriptación de extremo a extremo (Brand, 2013). No hay mucha documentación sobre este ejemplo de solución.

Valoración de los resultados obtenidos

Después del intenso estudio de las soluciones similares encontradas se puede concluir en que ninguna es viable para la propuesta de solución. Todas estas soluciones son sistemas de terceros en algunos casos software libre como Converse.js, Prosody; aunque no es el caso de WhatsApp, Telegram y Pidgin. Se tiene el riesgo de no presentar la privacidad y seguridad que la empresa necesita y filtrarse información confidencial por una puerta trasera. Sin embargo, con el análisis de cada una de las soluciones estudiadas se pudo comprender mejor el funcionamiento de los sistemas de mensajería instantánea sirviendo como base para el desarrollo de la solución propuesta al problema a resolver.

1.3 Tecnologías y herramientas

1.3.1 Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es un lenguaje de modelado visual utilizado en el desarrollo de software para representar, diseñar y documentar sistemas orientados a objetos. Fue creado por Grady Booch, James Rumbaugh e Ivar Jacobson en la década de 1990 y actualmente es un estándar de facto para el modelado de sistemas de software. Consiste en una serie de diagramas que representan diferentes aspectos de un sistema de software que permiten describirlos de manera más

precisa y completa facilitando la comprensión entre los miembros del equipo de desarrollo de software (Rumbaug, y otros, 2007).

1.3.2 Herramienta CASE

Visual Paradigm 17.1 es una herramienta de modelado y desarrollo de software que admite diversas metodologías y lenguajes de programación. La herramienta proporciona una plataforma de desarrollo para sistemas de TI[6] con una amplia gama de funcionalidades para el modelado, diseño, validación y generación de código fuente, lo que permite a los desarrolladores crear y mantener sistemas de software complejos de manera eficiente. se integra fácilmente con otras herramientas CASE y la mayoría de los IDE líderes, lo que facilita su interoperabilidad y su uso en todo el proceso de desarrollo de software, desde el modelado hasta la implementación (Paradigm, 2020).

1.3.3 Lenguaje de desarrollo

Python es un lenguaje de programación de alto nivel y de propósito general que se utiliza en una amplia variedad de aplicaciones, desde el desarrollo de aplicaciones web hasta la inteligencia artificial y el análisis de datos. Es conocido por su facilidad de uso y su sintaxis clara y legible, lo que lo hace popular entre los principiantes y los programadores experimentados (Matthes, 2023).

Algunas de las ventajas de Python incluyen:

- Sintaxis clara y legible.
- Amplia biblioteca estándar.
- Interpretado, lo que significa que el código se ejecuta línea por línea en lugar de compilarse antes de la ejecución.
- Tipado dinámico, lo que significa que los tipos de datos se determinan en tiempo de ejecución en lugar de en tiempo de compilación.

Para el desarrollo de la solución se utiliza Python 3.11 como lenguaje de programación.

JavaScript: es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe

6 Herramienta esencial para la gestión y el procesamiento de la información digital en una amplia gama de contextos y aplicaciones.

Acrobat JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional). Fue creado originalmente para agregar interactividad a las páginas web y se ha convertido en uno de los lenguajes más populares en el ámbito del desarrollo web.

Se utiliza la JavaScript 3.7.1 como lenguaje de programación para el desarrollo de la solución propuesta.

1.3.4 Marco de trabajo

Django es un framework[7] de desarrollo web de alto nivel escrito en Python que se utiliza para desarrollar aplicaciones web complejas y escalables. Django es conocido por su enfoque en la eficiencia y la productividad, lo que lo hace popular entre los desarrolladores web.

Algunas de sus ventajas son:

- Arquitectura modelo-vista-plantilla (MVT).
- ORM (Mapeo Objeto-Relacional)[8].
- Administrador de Django.
- Compatible con múltiples bases de datos.
- Seguridad.
- Biblioteca estándar.

Se utiliza Django 4.2 como framework del lenguaje de programación Python para desarrollar la solución propuesta.

Bootstrap: es un framework de desarrollo front-end que se basa en HTML, CSS y JavaScript. Proporciona una colección de estilos predefinidos, componentes y scripts listos para usar, lo que permite crear rápidamente interfaces web atractivas y responsivas. Facilita el desarrollo de interfaces web. Ofrece una amplia gama de componentes reutilizables, como botones, barras de navegación, tarjetas, formularios, entre otros. Esto permite ahorrar

[7] Marco de trabajo
[8] Permite a los desarrolladores trabajar con bases de datos relacionales de manera más fácil y eficiente, sin tener que escribir SQL directamente.

tiempo y esfuerzo al desarrollar interfaces web. Cuenta con una documentación detallada y ejemplos prácticos, lo que facilita su aprendizaje y uso.

Se utiliza Bootstrap 5.3.2 como framework de estilos para el desarrollo de la propuesta solución.

1.3.5 Sistema Gestor de Base de Datos

PostgreSQL es un sistema de gestión de bases de datos relacionales de código abierto y gratuito. Es conocido por ser uno de los sistemas de bases de datos más avanzados y completos disponibles en la actualidad. Tiene su propio lenguaje de programación llamado PL/pgSQL. Se trabaja en PostgresSQL 14.

1.3.6 Entorno de Desarrollo Integrado

Visual Studio Code (VS Code) 1.82.2 es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Es conocido por su versatilidad, facilidad de uso y extensibilidad, lo que lo hace popular entre los desarrolladores de software en todo el mundo.

A continuación, se presentan algunas de las características notables de Visual Studio Code:

- Extensibilidad.
- Integración con Git.
- Depuración.
- Integración con el terminal.
- Personalización.
- Multiplataforma.

1.4 Metodología de desarrollo de software

Metodología AUPvUCI: Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI apoyándose en el Modelo CMMI-DEV como guía para aplicar buenas prácticas y aumentar la calidad del software (Sánchez, 2015).

La metodología AUP propone 4 fases de estas se decide para el ciclo de vida de los proyectos de la UCI las siguientes:

1. **Inicio:** se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto (Sánchez, 2015).
2. **Ejecución:** se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto (Sánchez, 2015).
3. **Cierre:** se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (Sánchez, 2015).

También, esta metodología propone 7 disciplinas de las cuales la variación AUPvUCI define un igual número, pero a un nivel más atómico; estas son: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación, Pruebas de aceptación, Procesos CMMI-DEV (Procesos de gestión y soporte).

Para el modelado del sistema en los proyectos AUPvUCI cuenta con 4 escenarios:

Escenario No 1: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los Casos de Uso del Negocio (CUN) muestran como los procesos son llevados a cabo por personas y los activos de la organización (Sánchez, 2015).

Escenario No 2: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información (Sánchez, 2015).

Escenario No 3: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad.

Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados (Sánchez, 2015).

Escenario No 4: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una Historia de Usuario (HU) no debe poseer demasiada información (Sánchez, 2015).

Según las características del proyecto se selecciona la metodología AUPvUCI para guiar el proceso de desarrollo del componente para la solución propuesta al problema. Al ser una metodología híbrida permite utilizar su escenario ágil con un modelo incremental e iterativo garantizando la calidad de todas las etapas del proyecto. Como el proceso del negocio a informatizar está bien definido con una buena comunicación con el cliente y no es muy extenso se escoge el escenario No. 4 de la metodología para el desarrollo del proyecto.

Conclusiones del capítulo

En este capítulo se tras realizar una investigación de la literatura fomentando las bases teóricas guiada por el análisis de los conceptos relacionados al tema para comprender a fondo la comunicación entre el personal de una empresa mediante mensajería instantánea; permitiendo caracterizar cada uno de los conceptos estudiados, logrando con su descripción identificando los elementos principales de estos. Se analizaron sistemas que se han estado usando, así como los sistemas homólogos que se podrían utilizar para la mejor comunicación en la plataforma Platel y otras plataformas de Cuba, aprendiendo de las particularidades que presentan y obteniendo conocimiento previo de las funcionalidades que tienen estos. Además de analizar sus ventajas y desventajas por lo que ninguno se utiliza en la solución al problema. Teniendo en cuenta todo lo investigado se analiza el ambiente de desarrollo conveniente para el desarrollo del sistema según las características deseadas por la plataforma Platel, estableciendo la metodología, las herramientas y lenguajes.

CAPÍTULO II: DISEÑO DE LA SOLUCION PROPUESTA

Introducción

En este capítulo se refleja la descripción de la solución propuesta al problema a resolver. Así como el diseño de la misma y los artefactos correspondientes para realizar la implementación

2.1 Propuesta de solución

En este trabajo se propone realizar un módulo basado en la web de mensajería instantánea para la plataforma Platel. Son los usuarios registrados en dicha plataforma que van hacer uso del sistema propuesto como vía de comunicación interna. Esto permite mejorar la comunicación entre los usuarios de Platel y sin tener que usar aplicaciones de terceros que pueden incurrir en la privacidad y seguridad de la empresa Xetid.

El módulo tiene de inicio la autenticación mediante un token por tanto solo los usuarios de la empresa que pueden registrarse en la plataforma Platel aceden al sistema de mensajería instantánea. Cada usuario registrado puede mandar y recibir mensajes de texto, así como multimedia, audios y documentos de cualquier otro usuario registrado en la plataforma. El sistema permite crear grupos de chat con los contactos del usuario; recibir notificaciones a la entrada de algún mensaje. También bloquear contactos y personalizar el perfil de usuario en los ajustes. El sistema brinda a cada usuario la condición de privacidad y seguridad para su información personal en el chat.

2.2 Definición de requisitos, análisis y diseño del sistema

Los requisitos funcionales (RF) son enunciados acerca de servicios que el sistema debe proveer, de cómo debe reaccionar el sistema a entradas particulares y de cómo debe comportarse el sistema en situaciones específicas. En algunos casos, los requerimientos funcionales también explican lo que no debe hacer el sistema (Sommerville, 2011).

De cada RF se especifica su descripción y prioridad; tomando como parámetros de prioridad Media, Alta y Muy Alta según el nivel de prioridad para el desarrollo del software obtenidos en la entrevista con el cliente:

Muy Alta: Funcionalidades que son sumamente primordiales en la aplicación, que por su función no deben faltar, por lo que, la aplicación depende de estas funciones para su correcto funcionamiento.

Alta: Funcionalidades que no son muy necesarias, pero otras dependen de ellas para poder ejecutarse.

Media: Funcionalidades que su existencia en la aplicación no es primordial.

2.2.1 Requisitos funcionales

Tabla 1: Requisitos funcionales. Fuente: Elaboración propia.

No	Requisitos	Descripción	Prioridad
RF1	Autenticar usuario.	El sistema debe permitir al usuario autenticarse en la plataforma atendiendo su rol correspondiente, validando los datos siguientes: usuario y contraseña.	Muy Alta
RF2	Modificar perfil.	El sistema debe permitir al usuario modificar su perfil en el módulo de chat, introduciendo los siguientes datos: 1. Nombre. 2. Correo. 3. Información que el usuario desee.	Media
RF3	Configurar ajustes de las notificaciones.	El sistema debe permitir al usuario configurar en los ajustes si desea recibir o no las notificaciones.	Media
RF4	Configurar ajustes de privacidad.	El sistema debe permitir al usuario configurar en los ajustes su privacidad.	Alta
RF5	Listar contacto.	El sistema debe mostrar un listado con todos los contactos guardados del usuario.	Alta
RF6	Eliminar contacto.	El sistema debe permitir al usuario eliminar algún contacto.	Media
RF7	Buscar contacto.	El sistema debe permitir al usuario buscar por el nombre a algún contacto.	Media
RF8	Mostrar lista de contacto dinámico.	El sistema debe mostrar la lista de los contactos dinámicamente según el último que envió algún mensaje.	Alta
RF9	Mostrar información de un contacto.	El sistema debe mostrar la información del perfil de algún contacto.	Media
RF10	Bloquear un contacto.	El sistema debe permitir al usuario bloquear algún contacto.	Alta
RF11	Crear grupo.	El sistema debe permitir al usuario crear algún grupo, introduciendo la siguiente información: 1. Nombre del grupo. 2. Descripción del grupo. 3. Contacto a añadir al grupo.	Alta
RF12	Modificar grupo.	El sistema debe permitir al usuario modificar algún grupo, introduciendo la siguiente información: 1. Nombre del grupo. 2. Descripción del grupo. 3. Contacto a añadir al grupo.	Media

RF13	Listar grupo.	El sistema debe mostrar la lista de todos los grupos en el que se encuentra el usuario.	Alta
RF14	Eliminar grupo.	El sistema debe permitir al usuario eliminar algún grupo seleccionado.	Media
RF15	Enviar mensajes de texto.	El sistema debe permitir al usuario enviar mensajes de texto a algún contacto o algún grupo.	Muy alta
RF16	Enviar imagen.	El sistema debe permitir al usuario enviar imágenes a algún contacto o algún grupo.	Muy alta
RF17	Enviar video.	El sistema debe permitir al usuario enviar videos a algún contacto o algún grupo.	Muy alta
RF18	Enviar documentos.	El sistema debe permitir al usuario enviar documentos a algún contacto o algún grupo.	Muy alta
RF19	Enviar mensajes de voz.	El sistema debe permitir al usuario enviar mensajes de voz a algún contacto o algún grupo.	Muy alta
RF20	Enviar emoticonos.	El sistema debe permitir al usuario enviar emoticonos a algún contacto o algún grupo.	Muy alta
RF21	Eliminar mensaje enviado.	El sistema debe permitir al usuario eliminar mensaje enviado de algún contacto o de algún grupo.	Media
RF22	Eliminar historial de mensaje de un chat.	El sistema debe permitir al usuario eliminar historial de chat de algún contacto o de algún grupo.	Media
RF23	Buscar palabra en un chat.	El sistema debe permitir al usuario buscar palabra en el chat de algún contacto o de algún grupo.	Media
RF24	Buscar archivos en un chat.	El sistema debe permitir al usuario buscar archivos en el chat de algún contacto o de algún grupo.	Media

2.2.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) Son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requerimientos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema (Sommerville, 2011). Son aquellos que no describen funciones ni información a guardar, sino que especifican criterios en el diseño, implementación, calidad y cualidades que el sistema debe tener.

Interfaz:

RNF1. El sistema debe incluir el nombre y logo de Platel.

RNF2. El sistema contendrá un diseño gráfico que no debe ser complejo, con colores suaves en la gama de colores azul, blanco y rojo que no afecten la visión de los usuarios y totalmente libre de anuncios que puedan distraer a las personas que usan el sistema.

RNF3. El sistema poseerá un diseño manteniendo una letra visible y legible, preferiblemente Arial 12.

Seguridad:

RNF4. Disponer de un mecanismo de autenticación de usuarios seguro.

Eficiencia:

RNF5. El tiempo de respuesta del sistema no debe exceder de 15 segundos.

Usabilidad:

RNF6. El sistema debe garantizar un fácil manejo de la aplicación por parte del personal que no tenga experiencia en informática.

Hardware:

RNF7. El sistema requiere de una tarjeta de red con conectores RJ-45 o que permita la conexión inalámbrica para la centralización de su información.

Software:

RNF8. El sistema deberá funcionar sobre cualquier distribución de sistema operativo Linux y/o superior a Windows 7.

2.2.3 Personas relacionadas con el sistema

Se definen como actores del sistema las personas que están relacionadas con el mismo e interactúan obteniendo resultados de los procesos:

Tabla 2: Actores del sistema. Fuente: Elaboración propia.

Actor	Descripción
Usuario del sistema	Rol que realiza las actividades en el chat.
Administrador del sistema	Rol que realiza la gestión del módulo.

2.2.4 Historias de Usuario

Las historias de usuario (HU) son una herramienta que se utiliza en la ingeniería de requisitos para la encapsulación de estos en las metodologías agiles como AUPvUCI, captando un enfoque centrado del usuario y una entrega de valor iterativa. Describe breve y concisa una funcionalidad o requisito del sistema desde un punto de vista del usuario enfocándose en el que se necesita lograr en el sistema.

Ventajas de utilizar historias de usuario:

- Centradas al usuario.
- Comunicación efectiva.
- Son independientes.
- Facilitan la planificación e implementación.

Estructura de las historias de usuario:

- Número: Número asignado a la HU.
- Nombre: Nombre de la HU.
- Programador responsable: Programador del equipo de desarrollo que la implementara.
- Prioridad: Se divide en tres opciones Media, Alta y Muy Alta.
- Descripción: Descripción de la funcionalidad a realizar por la historia de usuario.
- Observaciones: Restricciones asociadas a la historia.
- Prototipo de interfaz gráfica elemental: Aquí se coloca una ilustración con la interfaz gráfica de esta historia de usuario.

A continuación, se muestran las HU pertenecientes a dos (2) requisitos funcionales, las restantes se incluyen en los anexos:

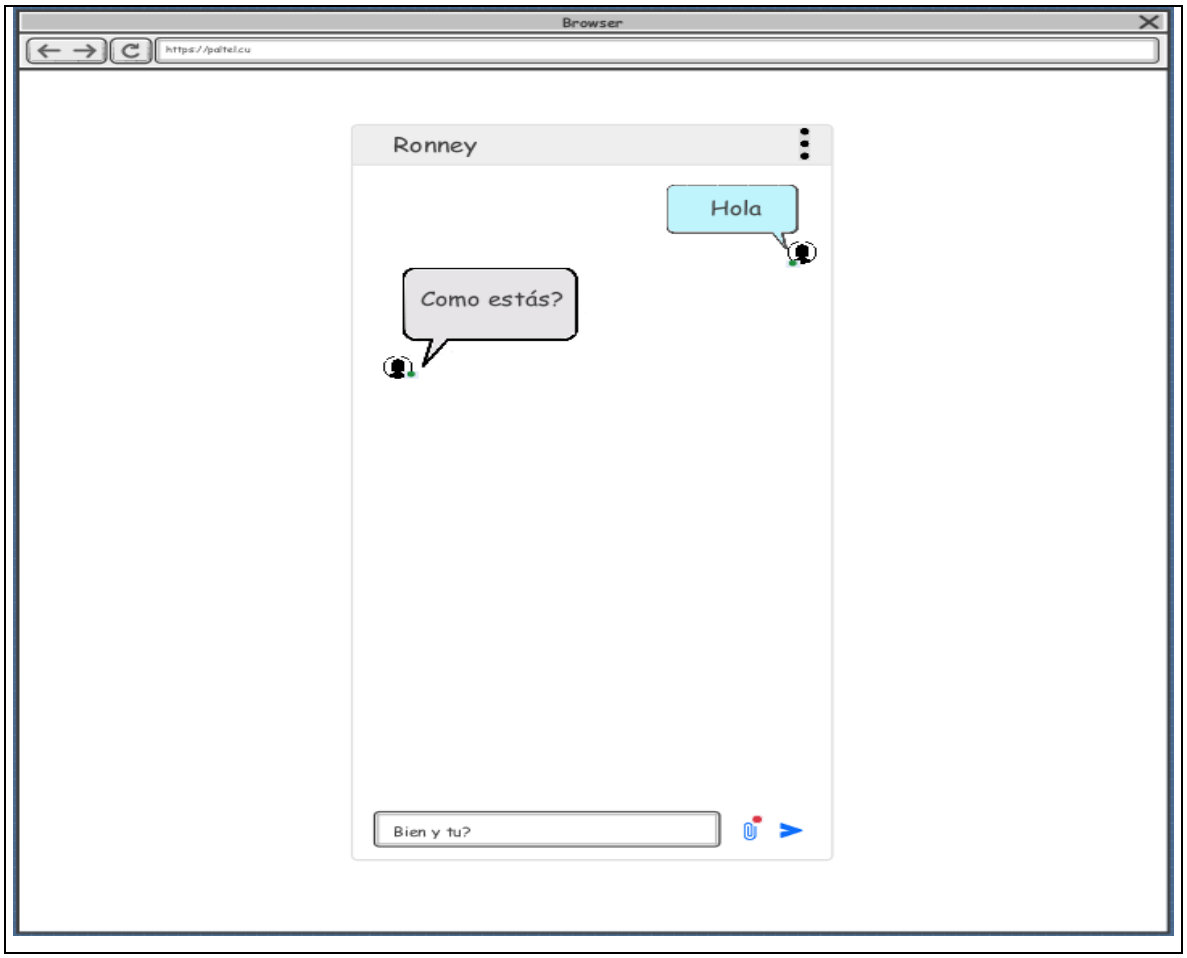
Tabla 3: Historia de usuario-Autenticar usuario. Fuente: Elaboración propia.

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Autenticar usuario.
Programador: Claudia Suárez	**Prioridad:** Muy alta

Descripción: Esta funcionalidad permite al usuario registrado en la plataforma autenticarse en el sistema.
Observaciones: Se introducen los siguientes datos: usuario y contraseña.
Prototipo de interfaz: 

Tabla 4: Historia de usuario-Enviar mensaje. Fuente: Elaboración propia.

Historia de Usuario	
Número: 15	**Nombre Historia de Usuario:** Enviar mensaje de texto.
Programador: Claudia Suárez	**Prioridad:** Muy Alta
Descripción: Esta funcionalidad permite al usuario enviar mensajes de texto a algún contacto o algún grupo.	
Observaciones: Se debe escribir el mensaje a enviar en la entrada de texto.	
Prototipo de interfaz:	



2.3 Diseño e implementación del sistema

2.3.1 Arquitectura de software

La arquitectura de software se refiere a la estructura y organización de un sistema de software. Es la representación de alto nivel que describe cómo los diferentes componentes del sistema interactúan entre sí, cómo se comunican y cómo se organizan para lograr los objetivos del sistema. Proporciona una visión global del sistema, estableciendo las bases para su diseño, implementación y mantenimiento. Define los principios, patrones y decisiones clave que guían el desarrollo del software.

La arquitectura de software es una representación abstracta del sistema que muestra los componentes principales, su comportamiento y su interacción para cumplir con los objetivos del sistema. Es una visión general que se enfoca en la coordinación y comunicación entre los componentes, evitando los detalles técnicos específicos (Arquitectura).

La plataforma Platel se basa en una arquitectura híbrida que combina dos enfoques principales: la arquitectura en capas y el patrón arquitectónico Modelo-Vista-Plantilla (MVP), que es el utilizado por el lenguaje de programación Python y el framework Django con el que está implementado el sistema.

La arquitectura en capa se utiliza para gestionar las tareas y los componentes bien definidos en la plataforma que llevan a cabo las comunicaciones unificadas. Utiliza cinco capas definiendo detalladamente cada una por su complejidad y estructura de la plataforma; la primera capa se lleva a cabo el Front-end donde están los distintos tipos de aplicaciones por las que se pueden acceder a la plataforma Platel: un portal web, softphone, APK y teléfono físico con protocolo ZIP interactuando con los servicios de Back-end que responden a cada una de las aplicaciones relacionado a componentes para lograr la comunicación en tiempo real mediante la web. Siguiendo a la tercera capa de Buses e Interfaces hasta llegar a la capa de Almacenamiento que implementa la telefonía almacenando todos los mensajes de voz que son capas de apoyo a todos los componentes y servicios que se ejecutan en la capa superior. El resto de las capas son comunes y transversales que se encargan de las notificaciones, monitorio, trazas y logs e identidades de los usuarios de manera que a todos los servicios se acceden de forma segura y se verifica en todo momento que se está realizando en la plataforma. Todas estas capas se ejecutan en una infraestructura de despliegue basada en kubemetes y Máquinas virtuales. Esto permite una mayor flexibilidad y escalabilidad en el manejo de las operaciones y procesos en tiempo real.

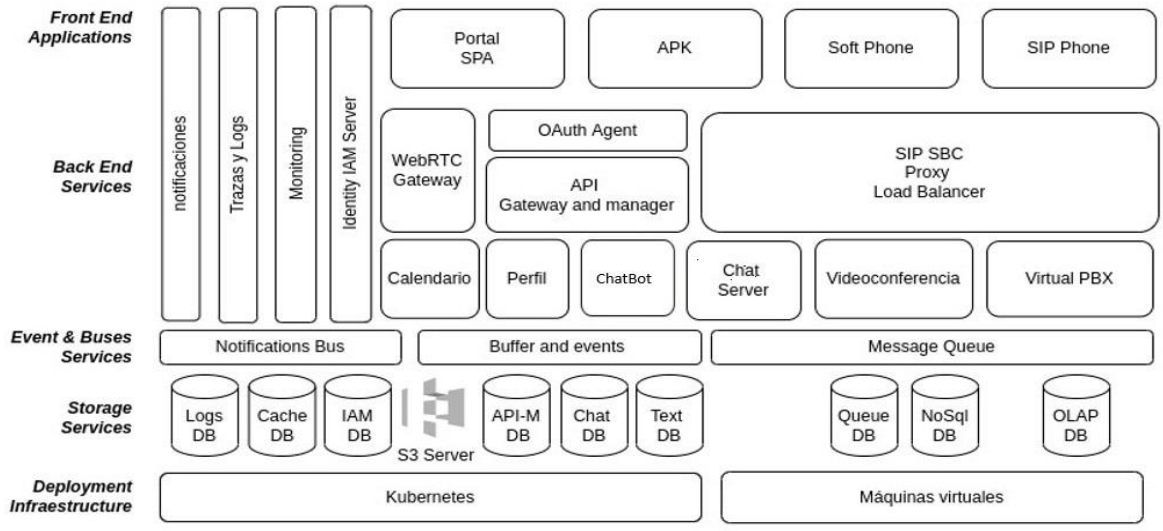


Ilustración 1: Arquitectura de software Platel. *Fuente: Platel*

Además, la plataforma Platel utiliza el patrón arquitectónico MVP para la organización y estructuración del código. El patrón MVP separa claramente las responsabilidades de los diferentes componentes de la plataforma. El Modelo (Model) representa los datos y la lógica de negocio subyacente. La Vista (View) es responsable de la presentación de la interfaz de usuario y la interacción con el usuario. Plantilla (Template) actúa como intermediario entre el Modelo y la Vista, gestionando la lógica de presentación y comunicación entre ambos (Fowler).

El módulo de mensajería instantánea para ser integrado también se programa en Python con framework Django utilizando el patrón arquitectónico Modelo-Vista-Plantilla.

* **Modelo-Vista-Plantilla**

Los patrones arquitectónicos son soluciones probadas y reutilizables para problemas comunes de diseño. La arquitectura puede utilizar diferentes patrones arquitectónicos, como el **Modelo-Vista-Plantilla** que es el utilizado por el framework de desarrollo Django el cual utilizamos para implementar el sistema.

Se compone de tres componentes principales:

- **Modelo**: Este componente se encarga de manejar los datos, la lógica y las reglas del negocio.
- **Vista**: Es la representación visual de los datos del Modelo. En otras palabras, es el componente que determina cómo se muestran los datos al usuario.
- **Plantilla**: Es la capa que decide cómo se presentarán los datos en la Vista. Se encarga de la renderización de los datos y proporciona flexibilidad para cambiar cómo se muestran los datos sin necesidad de modificar la lógica de la aplicación en sí.

El uso del patrón MVP nos ofrece varias ventajas:

- **Separación de responsabilidades**: Cada componente tiene un papel definido, lo que facilita el mantenimiento y la ampliación de la aplicación.
- **Reutilización de código:** Debido a la separación de roles, es posible reutilizar el código en diferentes partes de la aplicación sin necesidad de duplicarlo.
- **Pruebas más fáciles**: Como los componentes son independientes, se pueden probar por separado.

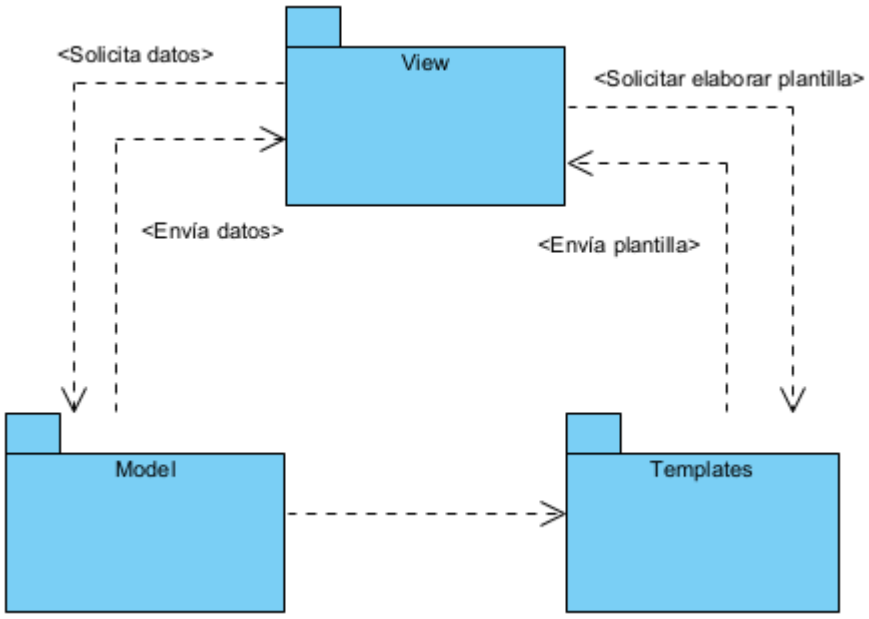


Ilustración 2: Patrón arquitectónico Modelo-Vista-Plantilla. *Fuente: Elaboración propia.*

2.3.2 Diagramas de clase del diseño

El Diagrama de Clases del Diseño (DCD) es la estructura estática que se usa para mostrar las relaciones entre clases y objetos que se programa. Establecen una conexión entre los requerimientos del cliente y la implementación del sistema. Proporcionando una representación visual que ayuda a la comprensión de la organización e interacción de las clases para cumplir con sus funcionalidades.

En el siguiente DCD se muestra la estructura y funcionamiento de los requisitos relacionados con el Perfil de usuario que conforman un CRUD completo, se tienen: Crear Perfil, Modificar Perfil, Mostrar Perfil y Eliminar Perfil. En la siguiente ilustración se muestra como la ClientPage que es la parte del front-end en la aplicación la que hace una petición a la ServerPage con los datos introducidos a través de un Formulario necesario para crear o modificar el Perfil de usuario. Una vez que la ServerPage recibe la petición, busca la vista que se encarga de gestionar todo el proceso del funcionamiento de Gestionar Perfil de usuario.

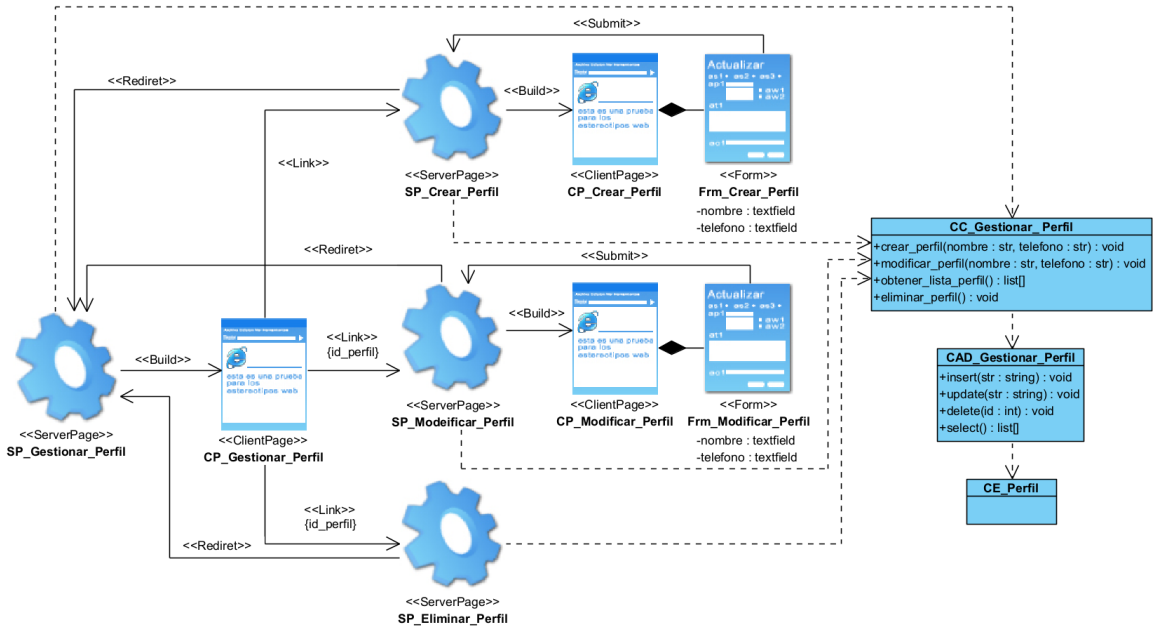


Ilustración 3: Diagrama de Clases de Diseño. Fuente: Elaboración propia.

2.3.3 Patrones de diseño

Los patrones de diseño son descripciones de clases y objetos relacionados que están particularizados en resolver problemas de diseño general en un determinado contexto. Un patrón de diseño nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. Identifica las clases e instancias participantes, sus roles y responsabilidades centrándose en un problema concreto; describiendo cuando aplicarlo y si tiene sentido hacerlo teniendo en cuenta otras restricciones del diseño (Gamma, y otros, 2003).

* **Patrones GRASP**

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 2003). A continuación, se muestran los utilizados en la implementación del sistema propuesto:

- **Experto:** Experto en información nos dice que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo o ejecutarlo, de este modo obtendremos un diseño con mayor cohesión y cuya información se mantiene encapsulada, es decir, disminuye el acoplamiento (SOLID y GRASP, 2012). Se evidencia el patrón

de diseño Experto en la clase Message que conoce la información de cómo debe ser un mensaje en el sistema, se muestra en la siguiente ilustración:

```python
class Message(models.Model):
    sender = models.ForeignKey(User, related_name='sender', on_delete=models.DO_NOTHING)
    receiver = models.ForeignKey(User, related_name='receiver', on_delete=models.DO_NOTHING)
    message = models.TextField(max_length=512)

    SENT = "S"
    DELIVERED = "D"
    READ = "R"
    MESSAGE_STATUSES = {
        (SENT, "Enviado"),
        (DELIVERED, "Entregado"),
        (READ, "Visto")
    }
    status = models.CharField(max_length=1, choices=MESSAGE_STATUSES, default=SENT)
    date = models.DateTimeField( verbose_name: 'sent_date', default=timezone.now)

    class Meta:
        ordering = ['date']
```

Ilustración 4: Representación del patrón Experto. Fuente: Elaboración propia.

- **Creador:** El patrón creador ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulamiento y reutilización (SOLID y GRASP, 2012). Se evidencia en la clase new_message con el método de la creación de un nuevo mensaje a enviar, se muestra en la ilustración siguiente:

```python
def new_message(request):
    if not request.user.is_authenticated:
        return HttpResponseNotAllowed(('POST',))
    else:
        try:
            text = request.POST['message']
            contact = request.POST['contact']
        except KeyError:
            return HttpResponseBadRequest()
        else:
            sender = User.objects.get(id=request.user.id)
            receiver = User.objects.get(id=int(contact))
            message = Message(sender=sender, receiver=receiver, message=text)
            message.save()
            return HttpResponse(status=200)
```

Ilustración 5: Representación del patrón Creador. Fuente: Elaboración propia.

- **Alta cohesión:** Permite que la información que almacena una clase debe de ser coherente y debe estar, en la medida de lo posible, relacionada con la clase. El grado de cohesión mide la coherencia de una clase, esto es, lo coherente que es la información que almacena una clase con las responsabilidades y relaciones que ésta tiene con otras clases (SOLID y GRASP, 2012).

- **Bajo acoplamiento:** Es la idea de tener las clases lo menos ligadas entre sí que se pueda, de tal forma que, en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. También hay varios tipos de acoplamiento (SOLID y GRASP, 2012). Se evidencia en la clase Contactos como se muestra en la ilustración siguiente:

```python
class Contactos(ListView):
    context_object_name = 'contactos'
    template_name = 'contactos.html'

    def get_queryset(self):
        return Contact.objects.filter(user_id=self.request.user)

    def get(self, *args, **kwargs):
        if not self.request.user.is_authenticated:
            return redirect(reverse('login'))
        return super(Contactos, self).get(self, *args: *args, **kwargs)
```

Ilustración 6: Representación del patrón Bajo acoplamiento. *Fuente: Elaboración propia.*

- **Controlador:** El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocio debe estar separada de la capa de presentación, lo que aumenta la reutilización de código y permite a la vez tener un mayor control (SOLID y GRASP, 2012).

2.3.4 Modelo de datos

Un modelo de base de datos es la estructura lógica que adopta la base de base datos, incluyendo las relaciones entre las tablas y definiendo las operaciones que se pueden realizar con los datos.

El modelo de datos del sistema propuesto está representado por el diagrama Entidad-Relación siguiente:

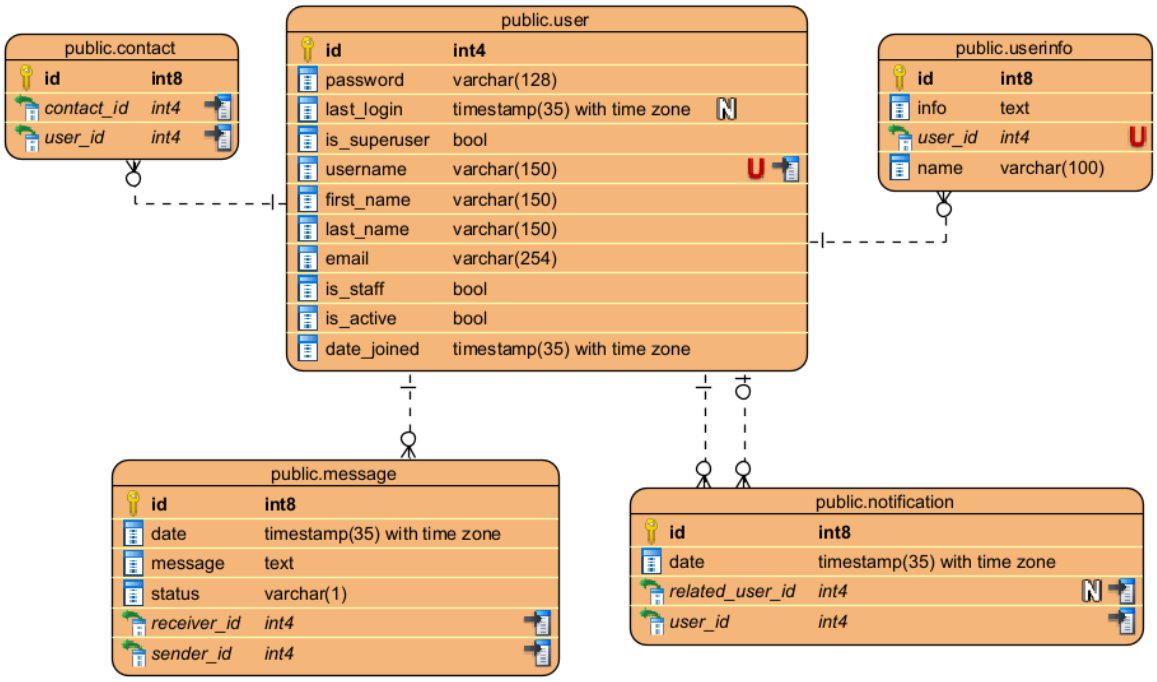


Ilustración 7: Diagrama Entidad-Relación. Fuente: Elaboración propia.

El diagrama representa el modelo de datos del módulo de mensajería instantánea, donde se representa la información en un total de cinco tablas (user, userinfo, contact, message, notification) y la relación que existe entre ellas garantizando jerárquicamente la estructura para la gestión de bases de datos del sistema.

Public_user: es la tabla que almacena los usuarios de la plataforma, el estado del usuario, token de autenticación entre más funciones. Esta tabla se relaciona con las restantes tablas para mantener el funcionamiento del sistema.

Public_userinfo: es la tabla que almacena la información del usuario que se registra, nombre, correo y descripción. Solo se relaciona con la tabla public_user.

Public_contact: es la tabla que almacena los contactos del usuario que se registra, el identificador de cada contacto e identificador del usuario. Se relaciona con la tabla public_user.

Public_message: es la table que almacena los mensajes que se envían y reciben, fecha y hora, estado de mensaje y quien recibe y envía el mensaje. Se relaciona con la tabla public_user.

Public_notification: es la tabla que almacena las notificaciones del sistema, la fecha y hora, usuario que recibe y contacto de quien recibe la notificación. Se relaciona con la tabla public_user.

2.3.5 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes, mostrando las dependencias que existen entre ellos. Los componentes físicos incluyen archivos, bibliotecas compartidas, módulos, ejecutables y paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema; estos son utilizados para modelar la vista estática y dinámica de un sistema y; muestran la organización y las dependencias entre un conjunto de componentes (Pressman, 2020).

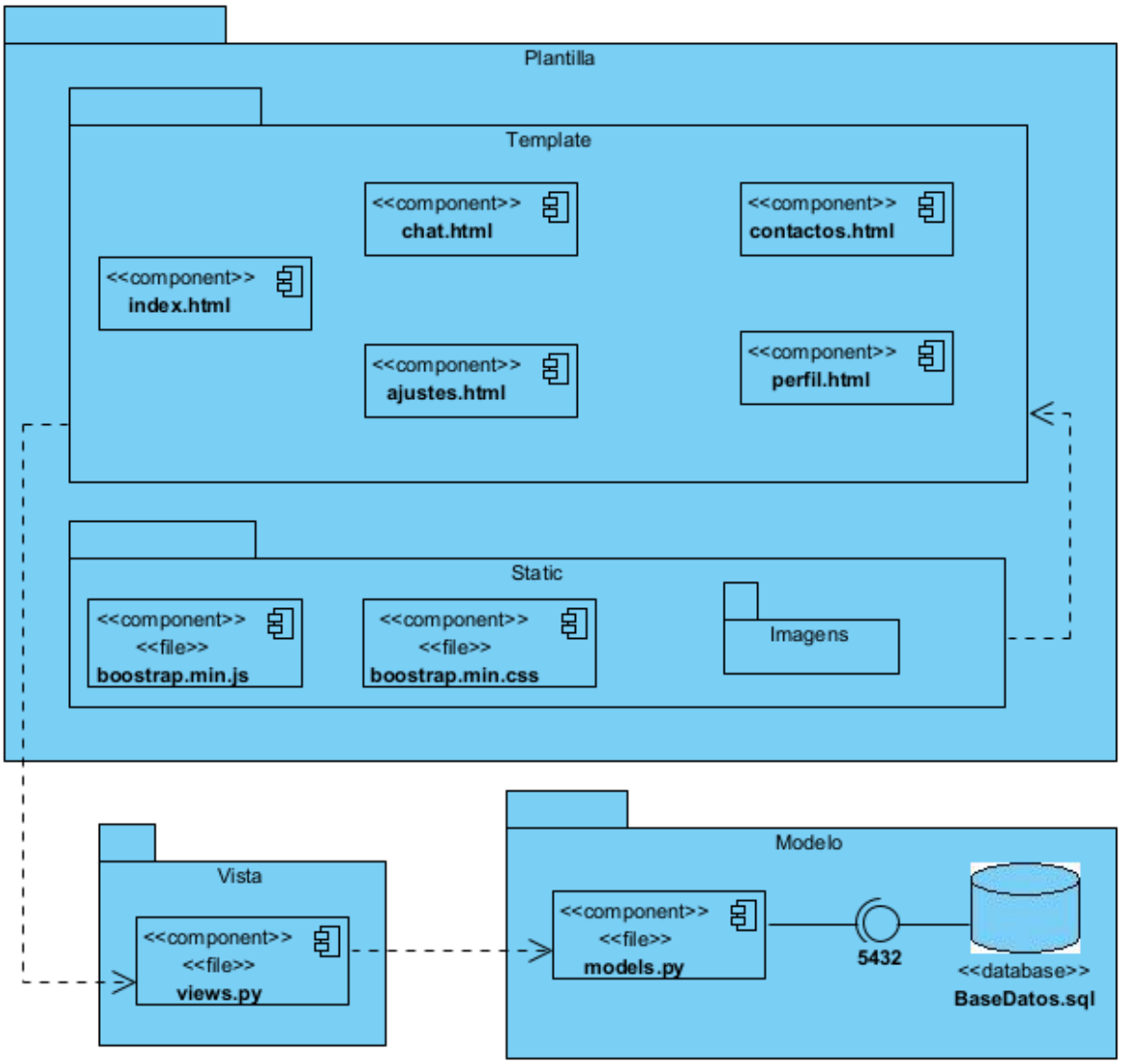


Ilustración 8: Diagrama de componentes. *Fuente: Elaboración propia.*

2.4 Diseño de interfaz de usuario

La interfaz de usuario es el medio que comunica al usuario con el sistema en el que interactúa. Debe presentar la información visual de manera fácil e intuitiva para las personas que no son diestros en los sistemas informáticos para una mejor experiencia con el software que utilice. A continuación, se muestran las principales interfaces del módulo de mensajería instantánea para la plataforma Platel.

La pantalla de autenticación donde cada usuario introducen su contraseña para poder entrar al sistema y utilizar el módulo de mensajería instantánea.

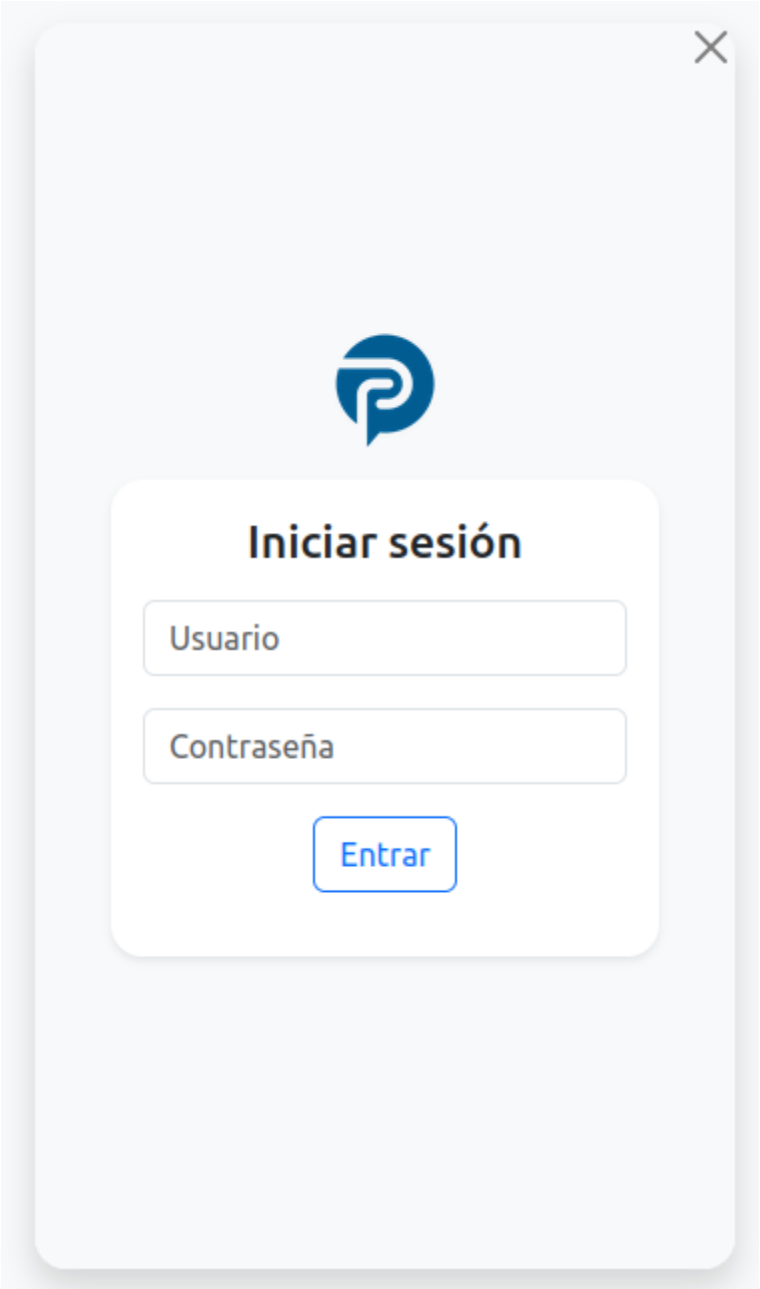


Ilustración 9: Pantalla de Autenticación. *Fuente: Elaboración propia.*

La pantalla de contactos donde el módulo de mensajería muestra un listado de los contactos con los que puedes tener una conversación mediante chat. Los contactos se muestran dinámicamente según el último contacto con el que se chateó. Tiene ajustes donde se configura la privacidad y seguridad del usuario, muestra el perfil y ajustes de notificaciones.

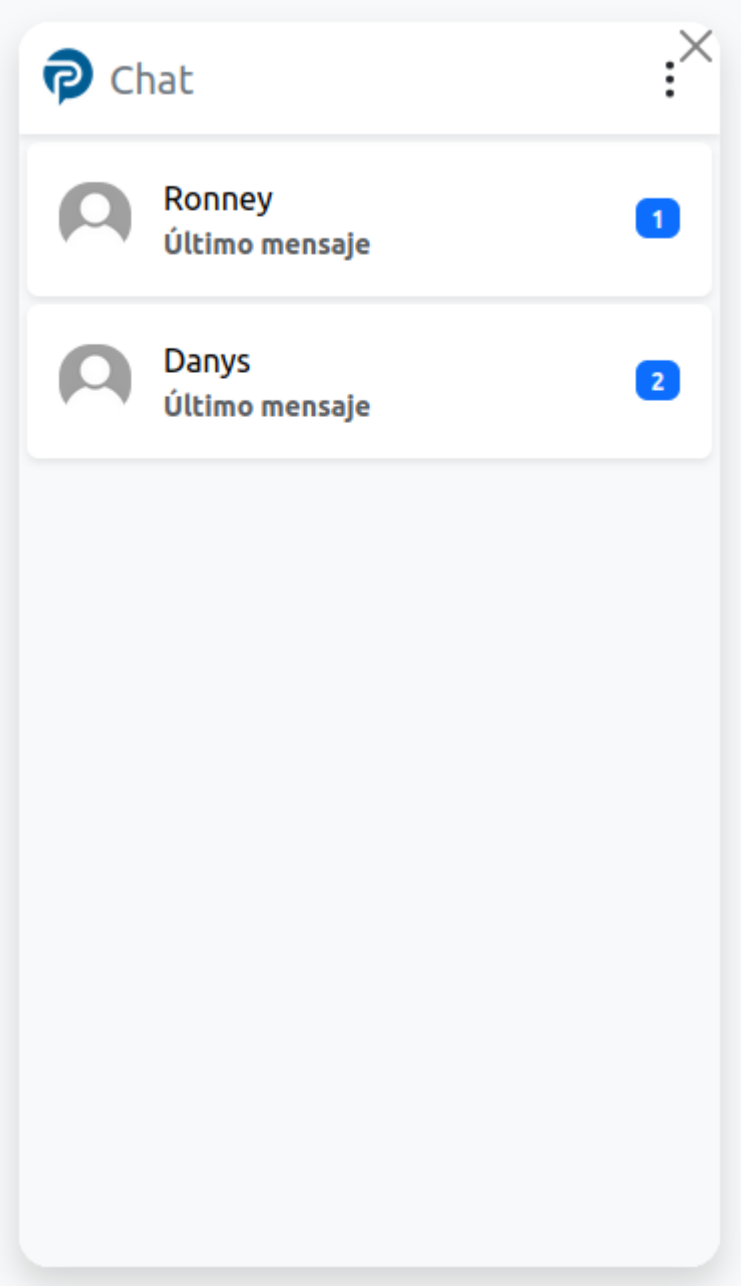


Ilustración 10: Pantalla de Contactos. *Fuente: Elaboración propia.*

La pantalla de conversación donde se chatea con algún contacto y se muestran los mensajes que entran y salen en la conversación. También muestra si se envían archivos adjuntos como multimedia o documentos. Tiene ajustes donde se configura la privacidad y seguridad del usuario.

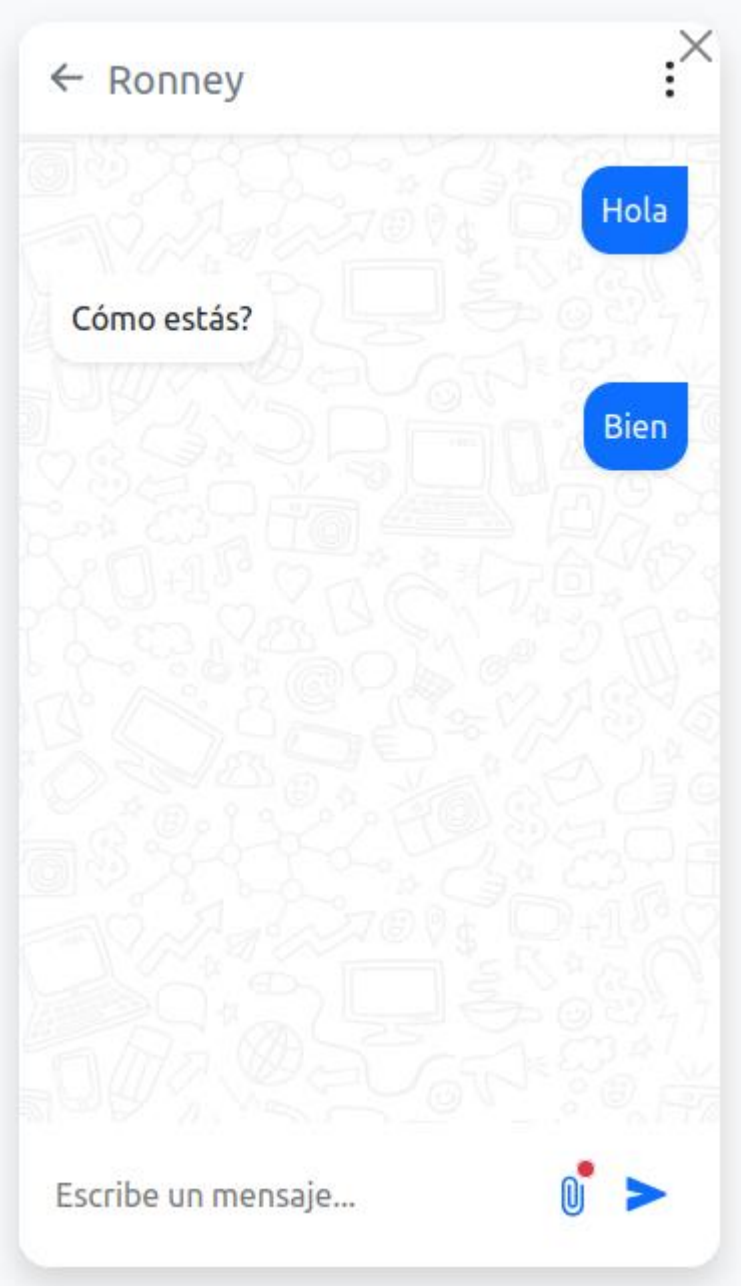


Ilustración 11:Pantalla de Conversación. *Fuente: Elaboración propia.*

Conclusiones del capítulo

En este capítulo se muestra visualmente la lógica de la propuesta solución que se describe en el primer epígrafe. Se logra especificar las necesidades del cliente mediante los requisitos funcionales que se obtienen y los requisitos no funcionales para especificaciones externas necesarias que debe cumplir la solución. Se realizan historias de usuario para una

mejor comprensión de los requisitos funcionales, así como se identificación y muestra de los patrones del diseño estructural que se utilizan el desarrollo del sistema. También se realizan diagrama de clases del diseño y el modelo de datos del sistema propuesto ofreciendo una visión general del sistema de mensajería instantánea mediante los artefactos descritos y generados. Concluyendo con que quedan sentadas las bases para la implementación y validación de la propuesta solución.

CAPÍTULO III: PRUEBAS O VALIDACIÓN

Introducción

En este capítulo se realiza la validación y verificación de software mediante herramientas de pruebas para aportar evidencia que se cumplen los requisitos expuestos en el Capítulo I. trazando una estrategia de pruebas y diseñando los casos de pruebas según su nivel correspondiente.

3.1 Diagrama de despliegue

Un diagrama de despliegue es una representación visual que muestra cómo se configuran las instancias de los componentes y los procesos en los nodos de un entorno de despliegue durante la ejecución en tiempo de ejecución. Proporciona una visión clara de la distribución física o lógica del sistema y cómo se comunican los componentes entre sí.

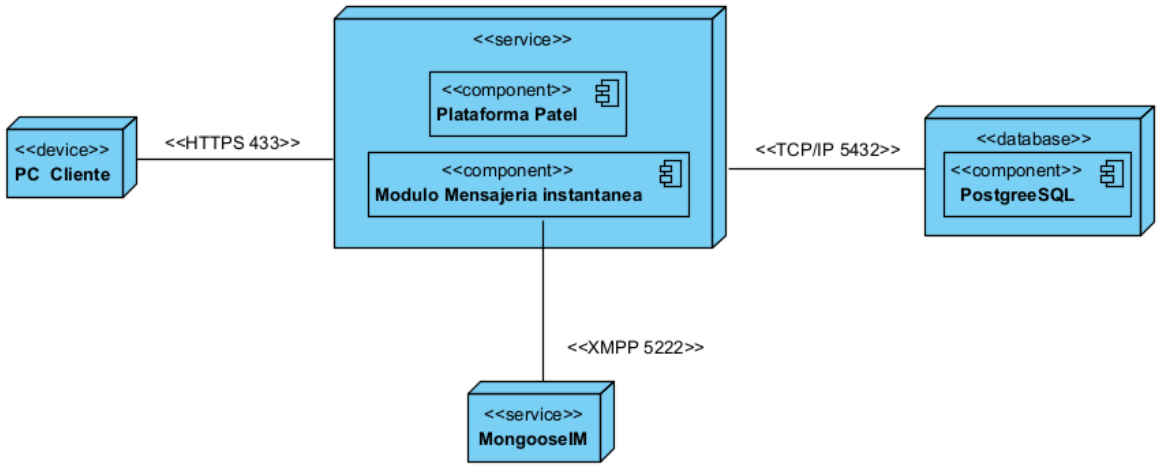


Ilustración 12: Diagrama de despliegue. Fuente: Elaboración propia.

3.1.1 Descripción de los nodos

Servidor de aplicaciones: Es el nodo encargado de tener instalado el sistema al que tendrán acceso los usuarios.

Servidor de Base de Datos: Es el nodo donde se almacenan los datos del sistema mediante el sistema de Base de Datos PostgreSQL.

Servidor MongooseIM: Es el nodo al que se conecta el módulo mediante el protocolo XMPP para enviar y recibir mensajes.

PC Cliente: Es el nodo desde donde el usuario interactúa con el sistema mediante el protocolo HTTPS, utilizando navegadores web.

3.2 Pruebas de software

Las pruebas de software son una parte integral del ciclo de vida del desarrollo de software. Las pruebas son la forma en que puede estar seguro acerca de la funcionalidad, el rendimiento y la experiencia del usuario. Las pruebas de software son una actividad primordial en el proceso de “aseguramiento de la calidad” (Chiu, 2015).

3.2.1 Estrategia de pruebas

Una estrategia de prueba del software integra los métodos de diseño de caso de pruebas del software en una serie bien planeada de pasos que desembocará en la eficaz construcción del mismo. La estrategia proporciona un mapa que describe los pasos que se darán como parte de la prueba, indica cuándo se planean, cuándo se dan estos pasos, además de cuanto esfuerzo, tiempo y recursos consumirán. Por tanto, cualquier estrategia de prueba debe incorporar la planeación de pruebas, el diseño de casos de pruebas, la ejecución de pruebas, la recolección y evaluación de los datos resultantes (Pressman, 2020).

La estrategia de prueba se define según el modelo en V indicando los niveles de pruebas que se aplican y la profundidad en cada nivel.

3.2.2 Niveles de prueba

Para aplicar las pruebas de validación se distinguen los siguientes niveles de pruebas según su objetivo, escenario o nivel de trabajo:

- **Prueba de Sistema:** Se evalúa el sistema desde una perspectiva externa, centrándose en la funcionalidad y los resultados esperados sin considerar la implementación interna.
- **Prueba de Aceptación:** Es la prueba final donde se evalúa si el sistema cumple con las expectativas del cliente.

3.2.3 Casos de pruebas

Pruebas de Sistema

Las pruebas de sistema tienen como objetivo evaluar el funcionamiento del sistema cumple con los requisitos que han sido especificados. Las pruebas del sistema se realizan cuando el producto software está completado; un ciclo de vida iterativo permite probar el sistema

mucho más tempranamente tan pronto como los subconjuntos bien formados de requisitos funcionales se han construido (Pressman, 2020). Entre las técnicas de pruebas que se realizan para evaluar la funcionalidad del sistema están:

- Pruebas funcionales.
- Pruebas de rendimiento.
- Pruebas de seguridad.

Pruebas Funcionales

Tal y como su nombre lo indica las pruebas funcionales se enfocan en validar la correcta implementación de las necesidades del cliente. La funcionalidad puede ser vinculada a los datos de entrada y de salida. Los datos de entrada serán ejecutados y mostrarán un resultado y dicho resultado será comparado con el resultado esperado (comportamiento) (Chiu, 2015), este proceso se muestra en la siguiente ilustración.

Estas pruebas están enfocadas en los requisitos funcionales utilizando el método de Caja Negra.

Método de Caja Negra

El método de Caja Negra se enfoca en hacer pruebas al sistema sin tomar en cuenta su estructura interna. Tiene como objetivo validar que las salidas sean las esperadas, centrándose en encontrar las circunstancias en la que el sistema no se comporta conforme a los requerimientos establecidos (Pressman, 2020).

Pruebas de rendimiento

La prueba de rendimiento se diseña para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado. La prueba del rendimiento ocurre a lo largo de todos los pasos del proceso de prueba. Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme se realizan las pruebas. Sin embargo, no es sino hasta que todos los elementos del sistema están plenamente integrados cuando puede determinarse el verdadero rendimiento de un sistema (Pressman, 2020). Se realiza la prueba de rendimiento con la herramienta Selenium para automatizar las pruebas de carga y estrés al sistema validando la velocidad de respuesta.

Para llevar a cabo esta prueba se utiliza la herramienta Selenium. Esta herramienta se utiliza para automatizar las pruebas al sistema. De manera que se programa y se efectúan diferentes inserciones cuyos parámetros serán aleatorios probando en cada uno el

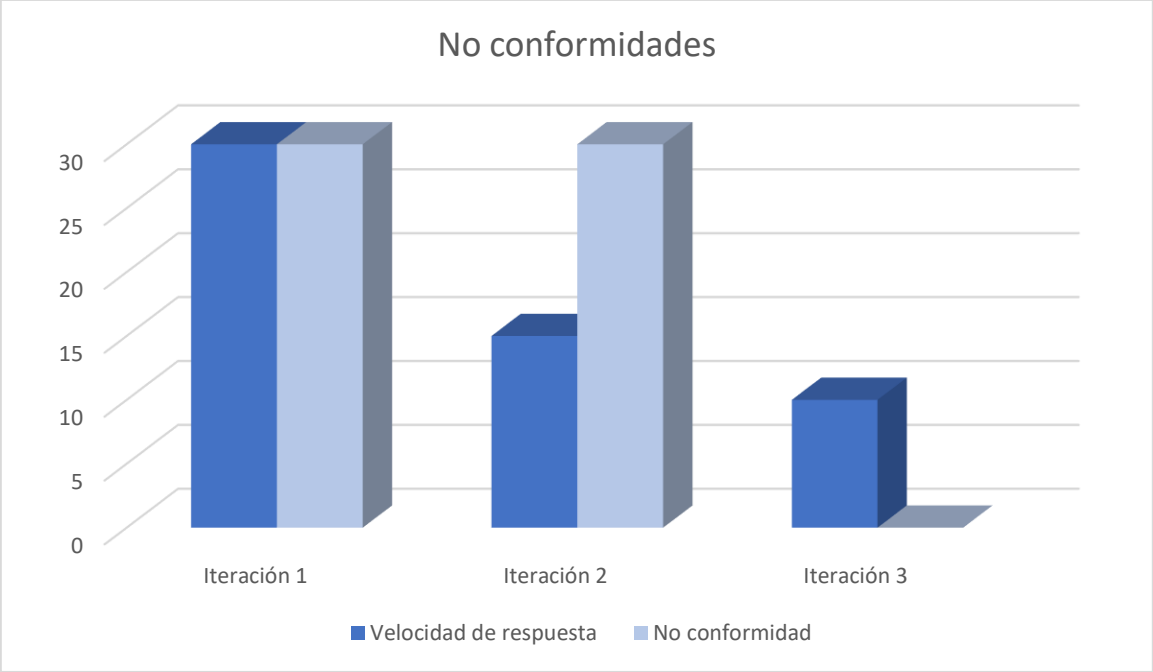
comportamiento del sistema evaluando así el rendimiento del sistema con la velocidad de respuesta del mismo y algunos de los requisitos funcionales. A continuación, se muestra el código que se utiliza con Python.

```python
class SeleniumTest(TestCase):
    def setUp(self):
        options = webdriver.FirefoxOptions()
        options.binary = FirefoxBinary("/snap/firefox/3252/usr/lib/firefox/firefox")
        self.driver = webdriver.Firefox(options)

    # PROBAR QUE EL BOTÓN PARA ABRIR EL CHAT FUNCIONA
    def test_index_button_open_chat(self):
        self.driver.get("http://localhost:8000/")
        self.driver.find_element( by: 'id',  value: 'chat-button').click()
        self.assertTrue(self.driver.find_element( by: 'id',  value: 'chat-frame').is_displayed())

    # PROBAR QUE EL LOGIN FUNCIONA
    def test_login(self):
        self.driver.get("http://localhost:8000/accounts/login")
        self.driver.find_element( by: 'id',  value: 'username').send_keys('claudia')
        self.driver.find_element( by: 'id',  value: 'password').send_keys('admin_123.')
        self.driver.find_element( by: 'id',  value: 'submit').click()
        self.assertIn(self.driver.current_url,  container: "http://localhost:8000/contactos/")

    def tearDown(self):
        self.driver.quit()
```

Ilustración 13: Prueba de Sistema-Selenium. *Fuente: Elaboración propia.*

En la anterior ilustración se evidencia la realización de la prueba con la herramienta Selenium. Después de ejecutar esta prueba se encontraron las siguientes no conformidades: en la iteración 1 se evidencia la no conformidad del cliente en la velocidad de respuesta del sistema; en la iteración 2 el sistema reduce su velocidad de respuesta, pero aún el cliente muestra su no conformidad terminando en una tercera iteración cumpliendo con la necesidad del cliente con 0 no conformidades.

Tabla 5: Resultados de satisfacción Pruebas Sistema. Fuente: Elaboración propia.



Pruebas de Seguridad

La prueba de seguridad pretende validar si los mecanismos de protección del sistema realmente lo protegen de intrusos como se especifica en los requisitos de seguridad. El probador durante la prueba intenta ingresar al sistema como un individuo no autorizado, ingresando credenciales que no están registradas en la base de datos de la plataforma. Se puede intentar atacar el sistema con software diseñados para romper barreras de seguridad o intentar adquirir credenciales de una forma externa. Se puede intentar hacer colapsar el sistema causando errores para penetrar en este durante su recuperación (Pressman, 2020). También para validar la seguridad del sistema mediante el enlace con el protocolo XMPP que comunica el cliente de mensajería instantánea con el servidor MongooseIM.

Se utiliza la librería Test de Python que ayuda para la realización de las pruebas automatizada y permite ejecutar múltiples pruebas en una función con diferentes parámetros verificando las excepciones esperadas y posibles rutas críticas. Se realizaron un total de tres (3) pruebas como se muestra en la Ilustración 14.

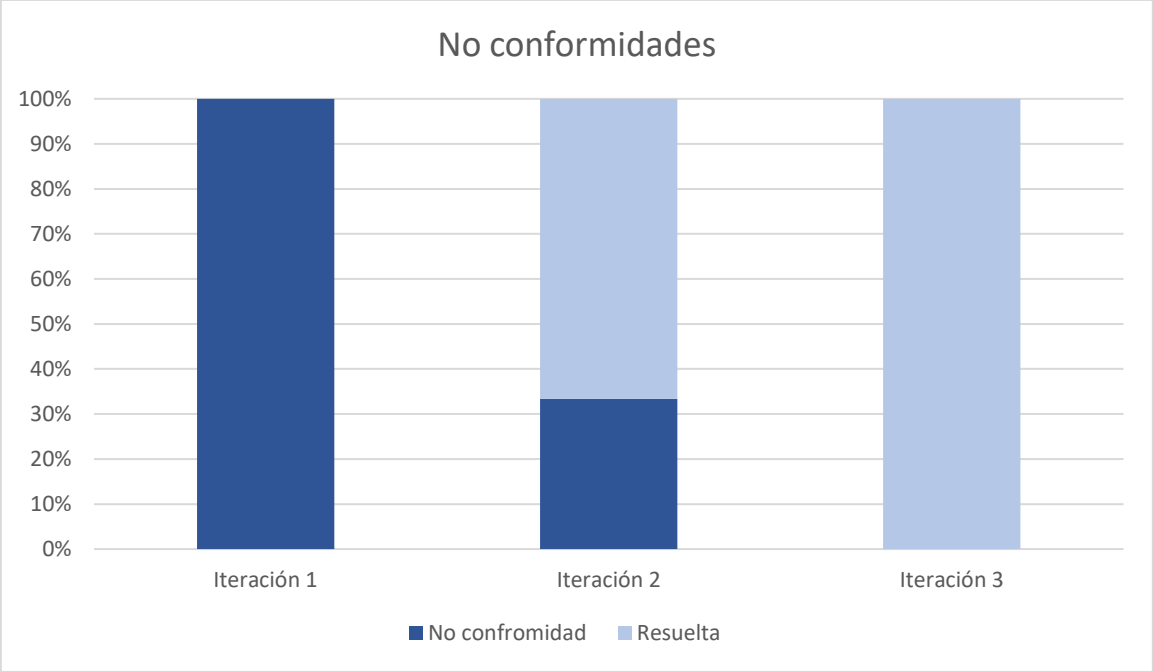
```python
from django.test import TestCase
from django.shortcuts import reverse


class MisTests(TestCase):
    # RETORNA CÓDIGO DE ESTADO 200 SI LA SOLICITUD ES CORRECTA
    def test_envio_de_mensaje(self):
        response = self.client.post(reverse('chat:message'), {'message': 'Hola', 'contact': 3, 'user': 2})
        self.assertEqual(response.status_code, second: 200)

    # RETORNA CÓDIGO DE ERROR 400 SI EL CUERPO DE LA SOLICITUD ESTÁ MAL FORMADA
    def test_envio_bad_request(self):
        response = self.client.post(reverse('chat:message'), {})
        self.assertEqual(response.status_code, second: 400)

    # RETORNA CÓDIGO DE ERROR 405 SI LA SOLICITUD LA REALIZA UN USUARIO NO AUTENTICADO
    def test_envio_not_allowed(self):
        response = self.client.post(reverse('chat:message'), {'message': 'Hola', 'contact': 3, 'user': 2})
        self.assertEqual(response.status_code, second: 405)
```

Ilustración 14: Pruebas de Seguridad. Fuente: Elaboración propia.

- En la primera prueba se introducen correctamente los parámetros de la función y retorna un código de estado 200 indicando que se comunica correctamente.
- En la segunda prueba no se introducen todos los parámetros de la función y retorna un código de error 400 que el cuerpo de la solicitud está mal formado.
- En la tercera prueba se introducen correctamente los parámetros, pero el usuario que establece conexión no está autenticado en el sistema por lo que retorna un código de error 405 de autenticación.

Estas pruebas arrojan en la iteración 1 las siguientes no conformidades; relacionadas con la funcionalidad de enviar mensaje tres no conformidades donde el botón de enviar no funciona correctamente, la funcionalidad de enviar archivos adjuntos no permite seleccionar el archivo del almacenamiento del dispositivo en el que se utiliza el sistema y se no se detalla ningún error en la solicitud de enviar mensaje con un usuario no autenticado en el sistema. En la iteración 2 se solucionan dos de las no conformidades de la iteración anterior, aunque aún se evidencia una no conformidad en la iteración 3 se valida el correcto funcionamiento del sistema mostrando cero no conformidades.

Tabla 6: Resultados de satisfacción Prueba de seguridad. Fuente Elaboración propia.



Pruebas de Aceptación

Cuando se construye un software para un cliente, se realiza una serie de pruebas de aceptación a fin de permitir al cliente validar todos los requerimientos. La prueba de aceptación puede realizarse durante un período de semanas o meses, y mediante ella descubrir errores acumulados que con el tiempo puedan degradar el sistema. Las pruebas de aceptación son pruebas de caja negra, que representan un resultado esperado de determinada transacción con el sistema. La mayoría de los desarrolladores de software usan un proceso llamado prueba alfa y prueba beta para descubrir errores que al parecer solo el usuario final es capaz de encontrar. El cliente registra cada uno de problemas que se encuentran durante las pruebas beta y los informa al desarrollador (Pressman, 2010).

Para darle fin a la validación del sistema desarrollado se realizan las pruebas de aceptación para determinar los diferentes tipos de errores por el cliente mediante pruebas Alfa y Beta.

- **Pruebas Alfa:** Se desarrollan en conjunto, el desarrollador y los usuarios finales. Con el objetivo de registrar los errores y problemas del uso del software.
- **Pruebas Beta:** Se realiza en el sitio donde se va a desplegar el software. A diferencia de la prueba alfa, la prueba beta es una aplicación del software en su ambiente de despliegue. El objetivo es registrar todos los problemas que se encuentran durante la prueba.

Una vez finalizada estas pruebas se valida el sistema desarrollado para que los usuarios de la plataforma Platel. Como resultado de esto se tienen nuevas no conformidades las cuales fueron resueltas. Al realizar las pruebas Alfa en un ambiente controlado se detectaron pequeños errores en el cumplimiento de los requisitos funcionales especificados por el cliente para el desarrollo del sistema. Al resolver los errores el cliente emite un acta de aceptación del producto emitiendo su total conformidad con la solución desarrollada. Se muestra en el Anexo 3.

Conclusiones parciales

Se facilitó la comprensión del software con la creación del diagrama de despliegue donde muestra la distribución física adecuada para un sistema basado en la web que forman parte de la producción. En la implementación se llevó a cabo según los estándares de codificación que se investigaron obteniendo un código legible y robusto. Las operaciones del sistema se ajustan a las especificaciones según los resultados de las pruebas que se realizaron para validar la solución.

CONCLUSIONES GENERALES

Una vez terminada la investigación se ha dado cumplimiento a los objetivos planteados, desarrollando el sistema de mensajería instantánea basado en la web para la plataforma Platel concluyendo lo siguiente:

- Con el apoyo de diversas tecnologías y documentación se realizó el estudio pertinente que permitió comprender los principales conceptos relacionados al desarrollo de un componente basado en la web de mensajería instantánea.
- Aunque los sistemas homólogos estudiados no aportan una solución al problema sirvieron como base para formar una idea general de la solución desarrollada.
- Los artefactos ingenieriles elaborados al aplicar la metodología AUPvUCI y su posterior implementación proporcionó como resultado la obtención de una aplicación informática que responde a las necesidades del cliente.
- La solución final cumple con los estándares de integración para mantener un estable flujo de información con la plataforma Platel, lo cual fue posible al adoptar patrones y buenas prácticas en la implementación del software.
- Siguiendo una estrategia de pruebas se controló el funcionamiento del componente de mensajería instantánea para la plataforma Platel arrojando resultados satisfactorios, lo cual permitió la validación de todos los requisitos propuestos.

RECOMENDACIONES

Tras haber finalizado la investigación y concluido el desarrollo del sistema propuesto se recomienda a continuación algunas ideas que se pueden implementar en futuras versiones:

- Ampliar las funcionalidades del componente incluyendo al cliente de mensajería instantánea los diferentitos protocolos de mensajería existentes en la actualidad.
- Diversificar las formas de acceder al componente para que sea multiplataforma.

REFERENCIAS BIBLIOGRÁFICAS

Abreo, Alex. 2010. Blog de Alex Abreo. [En línea] Unknown, 13 de abril de 2010. [Citado el: 18 de mayo de 2023.] http://alexabreo.blogspot.com/2010/04/pidgin-el-cliente-de-chat-universal.html.

Arquitectura, Escuela de Postgrado de Ingenieria y. Aplicaciones web: en qué consisten y cuáles son sus ventajas. [En línea] [Citado el: 17 de 11 de 2023.] https://postgradoingenieria.com/que-son-aplicaciones-web/#%C2%BFQue_son_las_aplicaciones_web.

Brand, Jc. 2013. Converse.js. [En línea] 2013. [Citado el: 18 de junio de 2023.] https://conversejs.org/.

Chiu, C. Campos. 2015. Las pruebas en el desarrollo de software. [En línea] 2015. http://www.ptolomeo.unam.mx:8080/xmlui/handle/132.248.52.100/7627.

Chiu, Cindy Campos. 2015. *LAS PRUEBAS EN EL DESARROLLO DE SOFTWARE.* Mexico : s.n., 2015.

Corporation's, Mozilla. 1998-2023. JavaScript. [En línea] 1998-2023. [Citado el: 06 de 11 de 2023.] https://developer.mozilla.org/es/docs/Web/JavaScript.

Durango, Rodrigo del Pozo y Galarza, Juan Manuel. 2018. *Análisis de las alternativas tecnológicas de comunicación unificada para la Facultad de Ciencias Administrativas, Gestión Empresarial e Informática de la Universidad Estatal de Bolívar.* 2018. págs. 942-955.

Ferrer, Jaime Moya. 2010. *Análisis de Herramientas de Gestión de VoIP.* 2010.

Florentín, Bernardo. 2022. Mundo cuentas. [En línea] 19 de 9 de 2022. https://www.mundocuentas.com/mensajeria-instantanea/.

Foundation, Django Software. 2005. Django. [En línea] 2005. [Citado el: 17 de junio de 2023.] https://www.djangoproject.com.

Foundation, Python Software. 2001-2023. Python. [En línea] 2001-2023. [Citado el: 18 de 7 de 2023.] https://www.python.org.

Foundation, XMPP Standards. 1999. XMPP Standards Foundation. [En línea] 1999. [Citado el: 20 de mayo de 2023.] https://xmpp.org/.

Fowler, Martin. martinfowler.com. [En línea] [Citado el: 1 de 10 de 2023.] https://martinfowler.com.

Gamma, Erich y Helm, Richard. 2003. *Patrones de Diseño. Elementos de software orientados a objetos reutilizable.* Madrid : Pearson Educacion SA, 2003.

González, Gabriela. 2014. Hipertextual. [En línea] 29 de 7 de 2014. https://hipertextual.com/2014/07/protocolo-xmpp.

IM, Prosody. 2008. Prosody IM. [En línea] 2008. [Citado el: 18 de junio de 2023.] https://prosody.im/.

Inc., SignalWire. 2023. FreeSWITCH Documentation. [En línea] 2023. [Citado el: 17 de 11 de 2023.] https://developer.signalwire.com/freeswitch/.

Inc., WhatsApp. 2009. Whatsapp. [En línea] 2009. [Citado el: 18 de mayo de 2023.] https://www.whatsapp.com/about.

Industria 4.0, implicaciones, certezas y dudas en el mundo laboral. **Lascano, Alberto Mauricio Pangol. 2022.** 2022, Revista Universidad y Sociedad, págs. 453-465.

Informaticas, Universidad de Ciencias. 2023. Universidad de Ciencias Informaticas. [En línea] 2023. [Citado el: 20 de 11 de 2023.] https://www.uci.cu/investigacion-y-desarrollo/productos/todus-apklis/todus.

Joskowicz, José. 2013. Comunicaciones unificadas. *Cisco.* [En línea] 2013. www.cisco.com.

Kappel, Gerti, y otros. 2006. *Web Engineering, The Discipline of Systematic Development.* s.l. : John Wiley & Sons Ltd, 2006. ISBN: 3-89864-234-8.

Larman, Craig. 2003. *UML y Patrones.* 2003.

LLP, elegram Messenger. 2013. Telegram Messenger. [En línea] 2013. [Citado el: 18 de mayo de 2023.] https://telegram.org.

Matthes, Eric. 2023. *Python Crash Course.* San Francisco : No Starch Press, 2023.

Microsoft. Microsoft. [En línea] [Citado el: 1 de 10 de 2023.] https://learn.microsoft.com/es-es/azure/architecture/guide/architecture-styles/event-driven.

Paradigm, Visual. 2020. *Visual Paradigm User's Guide.* 2020.

Pressman, Roger. 2020. *Software Engineering: A Practitioner's Approach, 9th Edition.* México, D. F. : McGRAW-HILL INTERAMERICANA EDITORES, 2020. ISBN10: 1259872971.

Redes Inalámbricas. **Salazar, Jordi. 2017.** 2017, Techpedia, pág. 40.

Revisión sistemática de Comunicaciones Unificadas de VoIP en redes CAN. **Cedeño Delgado, Silvia Monserrate. 2021.** 2021, Informática y Sistemas: Revista de Tecnologías de la Informática y las Comunicaciones.

Rumbaug, James, Jacobson, Ivar y Booch, Grady. 2007. *El Lenguaje de Modelado Unificado. Manual de Referencia.* Madrid : PEARSON EDUCACIÓN, S.A,, 2007. ISBN: 978-84-7829-087-1.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

SOLID y GRASP. **Carmona, Juan Garcia. 2012.** 2012.

Sommerville, Ian. 2011. *Ingeniería de Software.* Naucalpan de Juárez, Estado de México : Pearson Educación de México, S.A. de C.V., 2011. ISBN: 978-607-32-0603-7.

Team, The Pidgin. 1998. Pidgin. [En línea] 1998. [Citado el: 18 de mayo de 2023.] https://pidgin.im/.

XETID. 2022. Platel plataforma de videoconferencias que mejora la comunicación empresarial. [En línea] 20 de 02 de 2022. [Citado el: 17 de 11 de 2023.] https://www.xetid.cu/es/novedades/platel-plataforma-de-videoconferencias-que-mejora-la-comunicacion-empresarial.

ANEXOS

Anexo 1: Entrevista

1. ¿Cuáles son las principales necesidades de la plataforma que conlleva a crear un sistema de mensajería instantánea?
2. ¿Qué elementos del negocio se debe tener en cuenta para las funcionalidades del sistema?
3. ¿Qué tipo y formato de contenido debe permitir compartir el sistema a través de los mensajes?
4. ¿Los usuarios de la plataforma deben autenticarse al sistema para acceder?
5. ¿Qué elementos debe mostrar el sistema para la información del usuario registrado?

Anexo 2: Historias de Usuario

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Modificar perfil.
Programador: Claudia Suárez	**Prioridad:** Media
Descripción: Esta funcionalidad permite al usuario modificar la información que desea mostrar en su perfil a los demás usuarios.	
Observaciones: Se debe escribir el nombre y correo del usuario.	
Prototipo de interfaz:	

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Ajustes de notificaciones.
Programador: Claudia Suárez	**Prioridad:** Media
Descripción: Esta funcionalidad permite al usuario decidir si desea recibir o no notificaciones de mensajes en el sistema.	
Observaciones: Se debe activar o desactivar la opción en los ajustes del sistema.	
Prototipo de interfaz:	

Historia de Usuario

Número: 4	**Nombre Historia de Usuario:** Ajustes de privacidad.
Programador: Claudia Suárez	**Prioridad:** Alta
Descripción: Esta funcionalidad permite al usuario mantener su privacidad con otros usuarios del sistema.	
Observaciones: Se debe mostrar la información de los demás usuarios que no tienen permitido su comunicación con el usuario autenticado en el sistema.	
Prototipo de interfaz:	

Historia de Usuario	
Número: 5	**Nombre Historia de Usuario:** Listar contacto.
Programador: Claudia Suárez	**Prioridad:** Alta
Descripción: Esta funcionalidad permite al usuario visualizar todos los contactos registrados que tiene en el sistema.	
Observaciones: Se debe mostrar un listado de todos los contactos del usuario.	
Prototipo de interfaz:	

Historia de Usuario	
Número: 6	**Nombre Historia de Usuario:** Eliminar contacto.
Programador: Claudia Suárez	**Prioridad:** Media
Descripción: Esta funcionalidad permite al usuario eliminar algún contacto no deseado en el sistema.	
Observaciones: Se debe seleccionar el contacto que se desea borrar y aceptar la solicitud del sistema.	
Prototipo de interfaz:	

Historia de Usuario	
Número: 7	**Nombre Historia de Usuario:** Buscar contacto.
Programador: Claudia Suárez	**Prioridad:** Media

Descripción: Esta funcionalidad permite al usuario buscar algún contacto que tiene registrado en el sistema.
Observaciones: Se debe escribir el nombre o correo del contacto en la barra de búsqueda.
Prototipo de interfaz:

Historia de Usuario	
Número: 8	**Nombre Historia de Usuario:** Listar contacto dinámico.
Programador: Claudia Suárez	**Prioridad:** Alta
Descripción: Esta funcionalidad permite al usuario mostrar la lista de los contactos registrados en el sistema dinámicamente.	
Observaciones: Se debe mostrar un listado de todos los contactos con un orden de prioridad según el último contacto que envió algún mensaje.	
Prototipo de interfaz:	

Historia de Usuario	
Número: 9	**Nombre Historia de Usuario:** Información de un contacto.
Programador: Claudia Suárez	**Prioridad:** Media
Descripción: Esta funcionalidad permite al usuario ver la información de los contactos registrados en el sistema.	
Observaciones: Se debe visualizar el nombre y correo del contacto seleccionado.	
Prototipo de interfaz:	

Historia de Usuario	
Número: 10	**Nombre Historia de Usuario:** Modificar perfil.
Programador: Claudia Suárez	**Prioridad:** Media

Descripción: Esta funcionalidad permite al usuario modificar la información que desea mostrar en su perfil a los demás usuarios.
Observaciones: Se debe escribir el nombre y correo del usuario.
Prototipo de interfaz:

Anexo 3: Aval del cliente

AVAL DEL CLIENTE

Título: Componente de mensajería instantánea en la nube para la plataforma Platel.

Autor(es): Claudia Suárez González

La Parte Cliente, luego de haber revisado la documentación y artefactos generados, así como el Módulo desarrollado; determina que quedan resueltos los problemas que existían para la creación de un componente de mensajería instantánea para la Plataforma de Telecomunicaciones (PLATEL). El Módulo cumple con los requisitos funcionales y no funcionales pactados en la etapa de Modelación y se encuentra listo para su utilización.

Nombre y Apellidos: Faviannys Gámez Abad

Cargo: Jefe de centro

Firma:

Fecha: 01/12/23