

Índice

1	Antecedentes	4
2	Objetivos	6
2.1	Objetivo general	6
2.2	Objetivos específicos	6
3	Justificación	6
3.1	Justificación técnica	6
3.2	Justificación económica	6
4	Limitaciones	7
5	Marco Teórico	7
5.1	Factibilidad	7
5.1.1	Factibilidad Técnica	7
5.1.2	Factibilidad Económica	7
5.1.3	Factibilidad Legal	8
5.2	Robustez	8
5.3	Capacitación	9
5.3.1	Aspectos de la Capacitación:	9
5.4	Soporte	9
5.4.1	Componentes del Soporte:	10
5.5	Observabilidad	10
5.5.1	Componentes de la Observabilidad:	10
5.6	Monitoreo	10
5.6.1	Componentes del Monitoreo:	11
5.7	Requerimientos	11
5.7.1	Requerimientos Funcionales	11
5.7.2	Requerimientos No Funcionales	11
5.8	Diagrama de cuadrantes de requerimientos	11
5.8.1	Cuadrante 1: Requerimientos de Alta Importancia y Alta Dificultad (Desafiantes y Críticos):	12
5.8.2	Cuadrante 2: Requerimientos de Alta Importancia y Baja Dificultad (Rápidos pero Cruciales):	12
5.8.3	Cuadrante 3: Requerimientos de Baja Importancia y Baja Dificultad (Rutinarios y Menos Críticos):	12
5.8.4	Cuadrante 4: Requerimientos de Baja Importancia y Alta Dificultad (Evitar o Reevaluar):	13
5.9	Lista de actividades	13
5.10	Diagrama de Gantt	13
5.11	Ficha de Requerimientos	13
5.12	Scrum	13
5.12.1	Roles en Scrum:	13
5.12.2	Eventos de Scrum:	14
5.13	Artefactos de Scrum:	14
5.14	Mockups	14
5.15	Prototipo	14
5.15.1	Tipos de prototipos:	15

6	Requerimientos Funcionales	15
7	Requerimientos No Funcionales	16
8	Diagrama de cuadrantes de requerimientos	17
8.1	Tipo de programación a emplear	17
9	Módulos	17
9.1	Módulo 1	19
9.2	Módulo 2	20
10	Diagrama de Gantt	22
10.1	Diagrama de Gantt modulo 1	22
10.2	Diagrama de Gantt modulo 2	23
10.3	Fichas de requerimientos	25
11	Factibilidad	25
11.1	Factibilidad económica	25
11.2	Factibilidad legal	26
11.3	Factibilidad técnica	27
12	Robustez	27
12.1	Seguridad	27
12.2	Entrada	27
12.3	Adaptividad	28
12.4	Manejo de errores	28
13	Capacitación	28
13.1	Iniciar Sesión y Registro de Usuarios	28
13.2	Funciones Básicas de Chat	28
13.2.1	Configuración de Perfiles y Preferencias	28
13.3	Funciones Avanzadas	28
13.4	Entorno de Desarrollo	28
13.5	Comunicación cliente servidor	28
13.6	Seguridad y Encriptación	29
13.7	Documentación y Mantenimiento	29
14	Soporte	29
14.1	Centro de Ayuda en Línea	29
14.2	Asistencia Técnica por Correo Electrónico	29
14.3	Chat en Vivo	29
14.4	Reuniones Periódicas	29
15	Diagramas	30
15.1	Diagrama de clases	30
15.2	Diagrama de casos de uso	30
15.3	Diagrama de arquitectura por capas	32
15.4	Diagrama de despliegue	32
15.5	Diagrama de estados	33
15.6	Diagrama de actividades	33
15.7	Diagrama de secuencia	34

15.8 Diagrama de colaboración	35
16 Versiones	37

Chad5

RICARDO ROJAS CARVAJAL (Encargado)

SIMON EDUARDO ABASTO MARTINIS

JHON BRAYAN YAVIRA LEON

JONATHAN PERALTA FLORES

COSMI CLEMENTE FLORES

1 de febrero, 2024

1. Antecedentes

A fines de la década de los 80 surge la primera red de chat en internet llamada internet relay chat desarrollada por Jarkko Oikarinen en Finlandia, este chat se caracterizaba por un protocolo en tiempo real basado en texto que permitía debates entre 2 o mas personas en distintas salas o servidores. En este sistemas solo podías apreciar a las personas en sala en tiempo real y no hacía ver si estaban conectados, desconectados o ausentes, no podías realizar llamadas, video llamadas, emojis , o reaccionar a los mensajes.

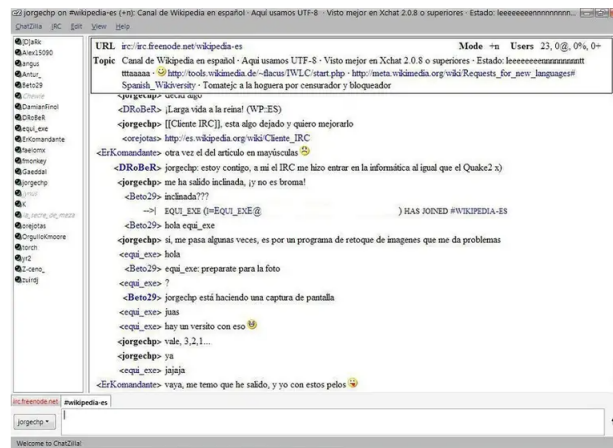


Figura 1: El primer chat en la red Relay chat

ICq, otra plataforma, nace en el año 2002 que en español significa “YO TE BUSCO” donde cada usuario tenía su número de identificación al momento de crear su cuenta llamado “número universal de internet”. Y fue el primero en su tipo en enviar mensajería instantánea a otros usuarios conectados a la red de ICQ. Y se caracteriza por que también te permitía enviar archivos, videoconferencias y charlas de voz pero al igual que “internet relay chat” tiene muchas deficiencias en el tema de seguridad y registro de usuario ya que estas plataformas no cuentan con lo necesario para que la experiencia del usuario sea mas eficiente.



Figura 2: ICQ, Yo Te Busco

Y es donde surgió el famoso MSN MESSENGER que fue lanzado por Microsoft y si querías ser parte de este servicio de mensajería era necesario tener una cuenta de Hotmail que era su filtro de seguridad que aun así no era suficiente ya que un usuario podía tener varias cuentas a la vez. Si tenías la opción de crear avatares ,estados , subir archivos e enviar ,realizar llamadas o vídeo llamadas hasta incluso zumbidos cuando los usuarios estaban conectados o desconectados como una especie de notificaciones en el 2004 fue relanzado y reemplazado por Windows Live Messenger y que hoy en día es Skype.



Figura 3: Windows Live Messenger

SKYPE.- Es un software propietario distribuido por Microsoft que permite comunicaciones de textos, voz y llamadas utilizando internet “VoIP” donde el código y protocolo permanecen cerrados pero se pueden descargar gratuitamente desde el sitio web y se puede acceder con las mismas credenciales de Hotmail/Outlook. Skype también se caracteriza por implementar el servicio de llamadas convencionales ya que puedes llamar sin la necesidad de que la otra persona tenga o no la aplicación a cualquier parte del mundo y que a su vez cuenta con un buzón de voz. Fue uno de los pioneros en realizar llamadas o vídeo llamadas grupales

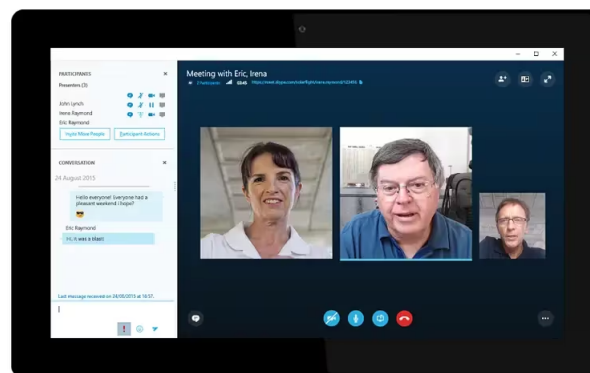


Figura 4: Skype

Después del boom de los servicios de mensajería instantánea, las videollamadas e incluso los foros en Internet, llegaron los smartphones y las redes sociales para revolucionarlo todo, y transformar la manera de comunicarse para siempre. Lo que era ICQ o Messenger, hoy está en WhatsApp y en menor medida Facebook Messenger. Las video llamadas hoy se realizan de manera mucho más sencilla y de manera gratuita a través de Facetime, con los mismos contactos y usuarios de Whatsapp, además del mencionado Skype.

2. Objetivos

2.1. Objetivo general

Desarrollar e implementar un robusto sistema de comunicación multiusuario para redes locales, con el propósito de fortalecer los procesos internos de las empresas, proporcionando una interacción ágil, segura y eficiente entre los miembros de la organización, con el fin último de mejorar la productividad, la cohesión organizativa y la capacidad de respuesta ante los desafíos y oportunidades del entorno empresarial actual.

2.2. Objetivos específicos

1. Diseñar la arquitectura del sistema, definiendo la estructura cliente-servidor para gestionar las conexiones de múltiples usuarios.
2. Desarrollar una interfaz de usuario intuitiva y accesible que permita una fácil navegación y gestión de conversaciones para cada usuario.
3. Implementar medidas de seguridad robustas para proteger contra posibles vulnerabilidades, como cifrado de datos y prevención de ataques.
4. Implementar un mecanismo de autenticación seguro para garantizar la identidad de los usuarios y proteger la privacidad de las conversaciones.
5. Incorporar herramientas de análisis que permitan a los administradores realizar un seguimiento del uso del chat, identificar patrones de comunicación y obtener información valiosa para mejorar la productividad y la eficiencia organizativa.
6. Ofrecer soporte técnico y capacitación a los usuarios para maximizar su experiencia de uso y garantizar la satisfacción del cliente.
7. Realizar pruebas exhaustivas para asegurarse de que el software funciona correctamente y cumple con las expectativas de los usuarios.

3. Justificación

3.1. Justificación técnica

La privacidad de la información es un factor clave en el ámbito de las empresas pioneras, por lo tanto, para evitar el filtrado de información se realizará un chat con servidor local.

3.2. Justificación económica

Dado el caso en que la información vital de un proyecto termine en manos de otra empresa, esto puede afectar gravemente el éxito producto o servicio que se brinda, teniendo como resultado una gran probabilidad de que la competencia salga con mejores soluciones, por lo tanto, se implementará como forma de comunicación digital dentro de la empresa el chat multiusuario con servidor local.

4. Limitaciones

1. El proyecto estará implementado en un servidor local.
2. Capacidad de Usuarios Conectados: El sistema debe manejar hasta 1000 personas en el chat en tiempo real. Esta capacidad puede ser una limitación en términos de rendimiento y recursos del servidor.
3. Tamaño de Archivos Multimedia: Los usuarios pueden enviar y recibir archivos multimedia de hasta 50 megas. Esto puede generar limitaciones en términos de ancho de banda y capacidad de almacenamiento.
4. Duración para Editar Mensajes: Los usuarios solo pueden editar mensajes en un lapso de un minuto. Esto puede ser una limitación en términos de flexibilidad para corregir o modificar mensajes después de este período.
5. Tecnología Python 3.9.2: El sistema está limitado a la versión de Python 3.9.2 en adelante. Esto puede afectar la compatibilidad con ciertas bibliotecas o funcionalidades más recientes.
6. Compatibilidad con Sistemas Operativos: El sistema debe ser compatible con Linux (Distribución Debian y derivados), Windows 10 en adelante, y MacOS. Esta restricción puede generar desafíos en el desarrollo y la prueba en múltiples plataformas.
7. Adaptabilidad a Dispositivos: El sistema debe adaptarse a distintos dispositivos (responsive). Esto puede generar limitaciones en términos de diseño y funcionalidades específicas para diferentes dispositivos.

5. Marco Teórico

5.1. Factibilidad

La factibilidad dentro de un proyecto de desarrollo de software es un aspecto crucial que determina su viabilidad en términos legales, económicos, técnicos y operativos. Este análisis exhaustivo permite evaluar si el proyecto es realizable, es beneficioso y si puede cumplir con los objetivos y requerimientos establecidos. La factibilidad abarca varios aspectos fundamentales que deben ser considerados antes de comprometer recursos y esfuerzos en el desarrollo de un software. Este análisis exhaustivo proporciona una base sólida para la toma de decisiones y ayuda a garantizar el éxito del proyecto a largo plazo.

5.1.1. Factibilidad Técnica

La factibilidad técnica es el análisis que se centra en evaluar si es posible desarrollar y poner en funcionamiento el software utilizando las tecnologías, herramientas y recursos disponibles. Se debe analizar la infraestructura tecnológica necesaria, la compatibilidad con sistemas existentes, la capacidad de escalabilidad y rendimiento del software, y cualquier otro requisito técnico relevante. Es crucial identificar posibles desafíos técnicos y riesgos asociados con el desarrollo del software, y desarrollar estrategias para mitigarlos. La factibilidad técnica garantiza que el proyecto pueda ser implementado de manera efectiva y que el software cumpla con los estándares de calidad y rendimiento esperados.

5.1.2. Factibilidad Económica

La factibilidad económica evalúa la viabilidad financiera del proyecto de software. Esto implica analizar los costos asociados con el desarrollo, implementación, mantenimiento y operación del software, así como también estimar los beneficios esperados y el retorno de la inversión (ROI). Se deben tener en cuenta aspectos como el presupuesto disponible, los gastos de desarrollo de software, los costos de infraestructura, los honorarios del personal y cualquier otro costo operativo. Es esencial realizar un análisis detallado de costos y beneficios para

determinar si el proyecto es financieramente viable y para tomar decisiones informadas sobre la asignación de recursos

5.1.3. Factibilidad Legal

La factibilidad legal implica examinar si el proyecto de software cumple con las leyes, regulaciones y normativas vigentes en la jurisdicción donde será utilizado. Se deben considerar aspectos como la protección de la propiedad intelectual, la privacidad de los datos de los usuarios, el cumplimiento de normativas específicas de la industria y cualquier otro requisito legal aplicable. Es fundamental asegurar que el software no infrinja derechos de autor, patentes u otras formas de propiedad intelectual, y que cumpla con los estándares de seguridad y privacidad establecidos por las autoridades competentes.

5.2. Robustez

La robustez de un software es un atributo fundamental que garantiza su capacidad para mantener un rendimiento adecuado y una funcionalidad sólida bajo diversas condiciones y situaciones adversas. En el contexto del desarrollo de software, la robustez se refiere a la resistencia del sistema ante fallos, errores, sobrecargas de trabajo y otros eventos inesperados que podrían comprometer su estabilidad y disponibilidad. Se puede medir haciendo uso de diferentes test o pruebas:

1. **Pruebas de Carga:** Evalúan cómo responde el sistema cuando se somete a una carga significativa, como un gran número de usuarios concurrentes, transacciones o volumen de datos.
2. **Pruebas de Estrés:** Examinan la capacidad del sistema para manejar situaciones extremas, como picos repentinos de tráfico, fallos de hardware, o eventos inesperados que pueden causar sobrecarga en el sistema.
3. **Registro de Errores:** Se registra y analiza cómo el sistema maneja los errores, excepciones y condiciones excepcionales. Se evalúa la capacidad del sistema para detectar, registrar y gestionar errores de manera adecuada.
4. **Resiliencia ante Fallas:** Se prueba la capacidad del sistema para recuperarse de errores y fallos sin interrumpir por completo el funcionamiento del sistema. Esto puede incluir la recuperación automática, la reanudación de operaciones críticas y la minimización del impacto en el usuario final.
5. **Pruebas de Aceptación del Usuario:** Los usuarios finales realizan pruebas para evaluar la facilidad de uso, la usabilidad y la capacidad de respuesta del sistema en diferentes escenarios de uso.
6. **Registro de Errores:** Se registra y analiza cómo el sistema maneja los errores, excepciones y condiciones excepcionales. Se evalúa la capacidad del sistema para detectar, registrar y gestionar errores de manera adecuada.
7. **Pruebas de Regresión:** Se realizan pruebas para verificar que las nuevas características o correcciones no introduzcan nuevos errores o problemas en áreas previamente funcionales del sistema.
8. **Análisis de Riesgos:** Se identifican y evalúan los posibles riesgos y vulnerabilidades del sistema, como problemas de seguridad, fallos de integridad de datos o amenazas potenciales.
9. **Pruebas de Seguridad:** Se realizan pruebas de penetración y análisis de seguridad para identificar y corregir posibles vulnerabilidades y brechas de seguridad en el sistema.
10. **Monitorización en Tiempo Real:** Se implementan herramientas de monitorización para supervisar el rendimiento del sistema, identificar cuellos de botella y detectar problemas antes de que afecten a los usuarios finales.

11. **Diagnóstico de Problemas:** Se establecen procedimientos y herramientas para diagnosticar y resolver problemas de manera rápida y eficiente cuando surgen.

Resumiendo, la robustez es un atributo crítico que garantiza la estabilidad, confiabilidad y seguridad del software en todas las condiciones operativas. Al priorizar la robustez en el proceso de desarrollo de software, se puede garantizar una experiencia de usuario fluida y segura, incluso en entornos desafiantes y cambiantes.

5.3. Capacitación

La capacitación en el desarrollo de software se refiere al proceso mediante el cual los profesionales del área adquieren conocimientos, habilidades y competencias necesarias para realizar eficazmente sus tareas y responsabilidades en proyectos de software. Este proceso abarca una variedad de temas, desde la comprensión de lenguajes de programación y tecnologías específicas, hasta la adopción de metodologías de desarrollo y prácticas de ingeniería de software.

5.3.1. Aspectos de la Capacitación:

1. **Tecnologías y Herramientas:** Comprender el uso adecuado de las herramientas de desarrollo, frameworks, librerías y plataformas relevantes para la construcción de software.
2. **Lenguajes de Programación:** Dominar los lenguajes de programación pertinentes para el desarrollo de aplicaciones, incluidos sus paradigmas, sintaxis y mejores prácticas de codificación.
3. **Metodologías de Desarrollo:** Familiarizarse con metodologías ágiles, como Scrum y Kanban, así como también con modelos de desarrollo tradicionales como el modelo en cascada.
4. **Pruebas y Control de Calidad:** Aprender técnicas de pruebas de software, incluyendo pruebas unitarias, de integración, funcionales y de aceptación, así como también herramientas de automatización de pruebas.
5. **Gestión de Proyectos:** Adquirir habilidades en gestión de proyectos de software, incluyendo la planificación, seguimiento y control de actividades, gestión de riesgos y comunicación con los stakeholders.
6. **Seguridad y Cumplimiento:** Entender los principios de seguridad informática y las prácticas recomendadas para garantizar la protección de datos y el cumplimiento de regulaciones.
7. **Desarrollo de Habilidades Interpersonales:** Desarrollar habilidades de comunicación, trabajo en equipo, resolución de conflictos y liderazgo, que son fundamentales para el éxito en proyectos colaborativos de software.

La capacitación en desarrollo de software puede ser impartida a través de una variedad de medios, incluyendo cursos presenciales, programas de formación en línea, tutoriales, libros y seminarios. La combinación de teoría y práctica es crucial para asegurar la comprensión profunda y la aplicación efectiva de los conceptos aprendidos.

5.4. Soporte

Se refiere a la asistencia técnica y el mantenimiento proporcionado antes, durante y después de la implementación del software, con el objetivo de resolver problemas, mejorar la calidad y asegurar la experiencia del usuario.

5.4.1. Componentes del Soporte:

1. **Atención al Cliente:** Ofrecer canales de comunicación accesibles, como correo electrónico, chat en vivo o líneas telefónicas, para que los usuarios puedan reportar problemas y recibir asistencia de manera oportuna.
2. **Base de Conocimientos:** Mantener una base de conocimientos actualizada con preguntas frecuentes, tutoriales y guías de solución de problemas para que los usuarios puedan resolver consultas por sí mismos.
3. **Actualizaciones y Parches:** Publicar regularmente actualizaciones de software que incluyan correcciones de errores, nuevas características y mejoras de seguridad para garantizar la estabilidad y seguridad del sistema.
4. **Capacitación del Usuario:** Ofrecer recursos de capacitación y material educativo para ayudar a los usuarios a aprovechar al máximo el software y resolver problemas comunes de manera autónoma.
5. **Seguimiento y Retroalimentación:** Realizar un seguimiento pro activo de las consultas de los usuarios y recopilar retroalimentación sobre su experiencia para identificar áreas de mejora y oportunidades de desarrollo futuro.

5.5. Observabilidad

Se refiere a la capacidad de comprender, analizar y diagnosticar el comportamiento interno de un sistema en tiempo real, a través de la recopilación y análisis de datos operativos, registros y métricas.

5.5.1. Componentes de la Observabilidad:

1. **Registro de Eventos (Logs):** La generación y almacenamiento de registros de eventos permite rastrear la actividad del sistema, registrar errores, advertencias y eventos importantes que pueden ayudar a diagnosticar problemas y realizar un seguimiento del comportamiento del software.
2. **Métricas de Rendimiento:** La recopilación de métricas de rendimiento, como el tiempo de respuesta, la utilización de recursos y la disponibilidad del sistema, proporciona una visión cuantitativa del funcionamiento del software y ayuda a identificar cuellos de botella y áreas de mejora.
3. **Tracing:** El tracing permite seguir la ruta de ejecución de una solicitud a través de múltiples componentes y servicios, facilitando la identificación de puntos de fallo y la optimización del rendimiento en sistemas distribuidos y microservicios.
4. **Dashboards y Visualizaciones:** La presentación de datos operativos a través de dashboards y visualizaciones ofrece una representación gráfica y comprensible del estado del sistema, permitiendo a los equipos de desarrollo detectar tendencias, anomalías y patrones de comportamiento.

5.6. Monitoreo

El monitoreo es un proceso esencial que implica la supervisión continua y la recopilación de datos relacionados con el rendimiento, la disponibilidad, la integridad y otros aspectos clave del sistema. Este proceso proporciona información valiosa que ayuda a los equipos de desarrollo a identificar problemas, optimizar el rendimiento y garantizar la estabilidad y la eficiencia del software en producción.

5.6.1. Componentes del Monitoreo:

1. **Métricas de Rendimiento:** Incluyen indicadores como el tiempo de respuesta de la aplicación, la utilización de recursos (CPU, memoria, almacenamiento), la tasa de errores y la disponibilidad del sistema. Estas métricas proporcionan una visión cuantitativa del funcionamiento del software y ayudan a identificar áreas de mejora.
2. **Alertas y Notificaciones:** Las alertas se activan cuando se alcanzan ciertos umbrales o se detectan condiciones anómalas en el sistema. Estas notificaciones permiten a los equipos de operaciones y desarrollo responder rápidamente a problemas críticos y tomar medidas correctivas de manera oportuna.
3. **Registros (Logs):** El monitoreo de registros implica la recopilación y el análisis de registros de eventos generados por el sistema. Estos registros proporcionan información detallada sobre las actividades del software y facilitan la resolución de problemas y la auditoría de seguridad.
4. **Seguimiento de Transacciones:** El seguimiento de transacciones permite rastrear el flujo de datos y las interacciones entre los diferentes componentes del sistema, lo que facilita la identificación de cuellos de botella y la optimización del rendimiento en entornos distribuidos y micro servicios.

5.7. Requerimientos

Los requerimientos en el desarrollo de software son la base sobre la cual se define, diseña, desarrolla y prueba un sistema de software. Constituyen la especificación formal de las funcionalidades, características y restricciones que debe cumplir el software para satisfacer las necesidades del cliente y los usuarios finales.

5.7.1. Requerimientos Funcionales

Los requerimientos funcionales en el desarrollo de software describen las acciones específicas que el sistema debe realizar y las funcionalidades que debe proporcionar para cumplir con las necesidades del usuario y los objetivos del negocio. Estos requerimientos definen el comportamiento y las características operativas del software, delineando las interacciones entre el sistema y sus usuarios.

5.7.2. Requerimientos No Funcionales

Los requerimientos no funcionales en el desarrollo de software son atributos de calidad que describen características del sistema que no están directamente relacionadas con sus funcionalidades específicas, sino con su rendimiento, seguridad, usabilidad, fiabilidad y otros aspectos que afectan la experiencia del usuario y la operación del sistema en su conjunto.

5.8. Diagrama de cuadrantes de requerimientos

El diagrama de cuadrantes de requerimientos es una herramienta visual que ayuda a clasificar y priorizar los requerimientos de un proyecto de software en función de su importancia y dificultad de implementación. Este diagrama se compone típicamente de cuatro cuadrantes, cada uno representando un área específica en función de dos ejes: el eje X representa la dificultad de implementación y el eje Y representa la importancia del requerimiento.

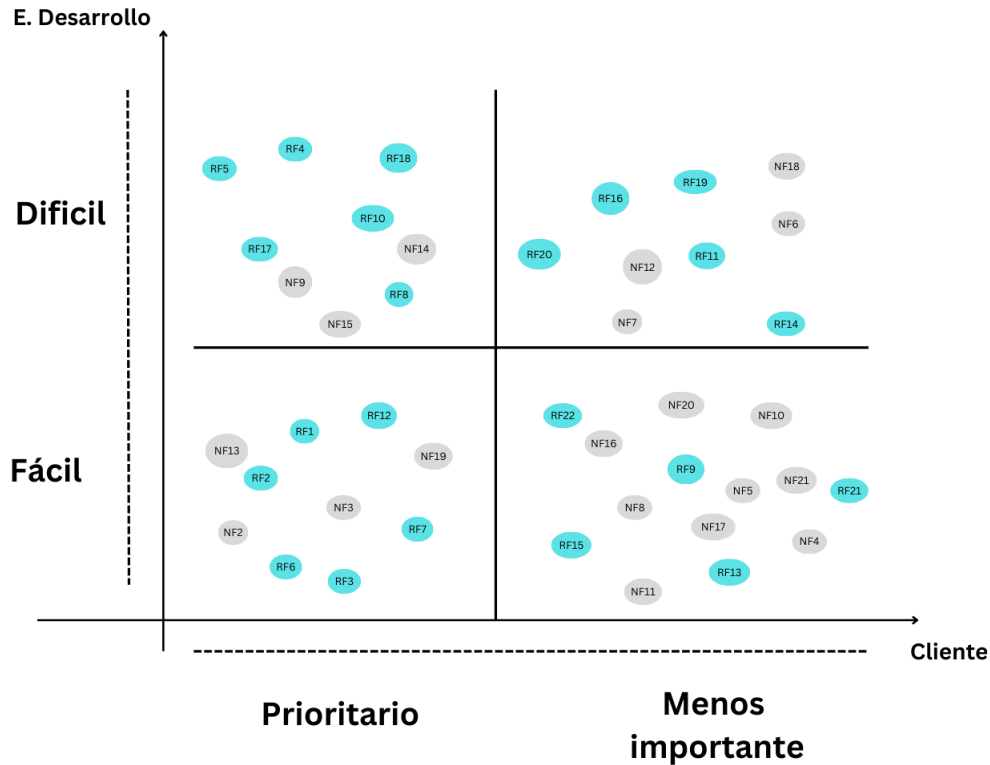


Figura 5: Diagrama de cuadrante de requerimientos

5.8.1. Cuadrante 1: Requerimientos de Alta Importancia y Alta Dificultad (Desafiantes y Críticos):

Incluye los requerimientos que son esenciales para la satisfacción del cliente o la funcionalidad básica del sistema. Estos requerimientos suelen tener un alto impacto en la experiencia del usuario o en los objetivos del negocio. Requieren una atención especial y un enfoque estratégico para su implementación exitosa.

5.8.2. Cuadrante 2: Requerimientos de Alta Importancia y Baja Dificultad (Rápidos pero Cruciales):

Contiene los requerimientos que son fundamentales pero que representan un desafío técnico o de implementación. Estos requerimientos pueden ser críticos para la competitividad del producto o para satisfacer las necesidades específicas del cliente. Generalmente, son relativamente fáciles de implementar y no requieren un esfuerzo técnico significativo.

5.8.3. Cuadrante 3: Requerimientos de Baja Importancia y Baja Dificultad (Rutinos y Menos Críticos):

Agrupar los requerimientos que, aunque son relativamente fáciles de implementar, tienen un impacto menor en la experiencia del usuario o en los objetivos del proyecto. Estos requerimientos pueden abordarse posteriormente en el ciclo de desarrollo o pueden ser considerados como mejoras opcionales.

5.8.4. Cuadrante 4: Requerimientos de Baja Importancia y Alta Dificultad (Evitar o Reevaluar):

Contiene los requerimientos que representan un desafío significativo de implementación pero que tienen un impacto limitado en la satisfacción del cliente o en los objetivos del proyecto. Pueden ser requerimientos técnicos, complejos pero no esenciales para la funcionalidad principal del sistema. Se debe considerar si estos requerimientos realmente agregan valor al producto final o si pueden ser reevaluados o simplificados.

5.9. Lista de actividades

Una lista de actividades de requerimientos es un documento que detalla las tareas específicas que deben llevarse a cabo para implementar cada uno de los requerimientos identificados en el proyecto de desarrollo de software. Estas actividades están agrupadas y organizadas en función de la prioridad de los requerimientos, lo que permite una planificación y ejecución más efectiva del proyecto. Asimismo organiza y prioriza las tareas necesarias para implementar cada uno de los requerimientos funcionales y no funcionales del sistema. Esta lista permite a los equipos de desarrollo enfocarse en las actividades más críticas y relevantes para el éxito del proyecto, asegurando una asignación eficiente de recursos y tiempo.

5.10. Diagrama de Gantt

El Diagrama de Gantt es una herramienta visual utilizada en la gestión de proyectos que permite planificar, programar y controlar las actividades a lo largo del tiempo. En el desarrollo de software, un Diagrama de Gantt es especialmente útil para coordinar las tareas de diseño, desarrollo, pruebas, implementación y mantenimiento de un sistema de software de manera eficiente y organizada. Nos muestra la duración de las actividades planificadas en el tiempo. Cada actividad se representa como una barra horizontal en el gráfico, cuya longitud indica la duración de la actividad. El eje horizontal representa el tiempo, generalmente dividido en semanas o meses, y el eje vertical muestra las distintas actividades del proyecto.

5.11. Ficha de Requerimientos

Una ficha de requerimientos, también conocida como especificación de requerimientos, es un documento detallado que describe los objetivos, funcionalidades y restricciones del sistema de software que se va a desarrollar. Esta ficha sirve como un punto de referencia para todos los miembros del equipo de desarrollo y otras partes interesadas durante el ciclo de vida del proyecto.

5.12. Scrum

Scrum es un marco de trabajo ágil ampliamente utilizado en el desarrollo de software y en proyectos que requieren flexibilidad, adaptabilidad y entrega incremental de resultados. Fue originalmente propuesto por Jeff Sutherland y Ken Schwaber a principios de la década de 1990 y desde entonces ha ganado popularidad en equipos y organizaciones de todo el mundo. Scrum se basa en principios iterativos e incrementales que permiten a los equipos responder de manera efectiva a los cambios en los requisitos del proyecto y a las necesidades del cliente.

5.12.1. Roles en Scrum:

Product Owner (Propietario del Producto): Es responsable de maximizar el valor del producto y del trabajo del equipo de desarrollo. Define y prioriza el backlog del producto, asegurándose de que contenga elementos que aporten el máximo valor al negocio y a los usuarios. **Scrum Master:** Es el facilitador del proceso Scrum. Ayuda al equipo a entender y adoptar los principios y prácticas de Scrum. Elimina los impedimentos que puedan obstaculizar el progreso del equipo y garantiza un entorno en el que el equipo

pueda trabajar de manera eficiente. **Equipo de Desarrollo:** Es un equipo autoorganizado y multifuncional que trabaja para convertir los elementos del backlog del producto en incrementos potencialmente entregables de producto al final de cada sprint.

5.12.2. Eventos de Scrum:

Sprint Planning (Planificación del Sprint): Se lleva a cabo al inicio de cada sprint y tiene como objetivo definir el objetivo del sprint y seleccionar las tareas que el equipo se comprometerá a completar durante el sprint.

Daily Scrum (Scrum Diario): Es una reunión diaria de no más de 15 minutos donde el equipo de desarrollo se sincroniza. Cada miembro del equipo responde a tres preguntas: ¿Qué hice ayer? ¿Qué haré hoy? ¿Hay algún impedimento que me esté bloqueando?

Sprint Review (Revisión del Sprint): Al final de cada sprint, se lleva a cabo una reunión de revisión donde el equipo muestra lo que ha construido durante el sprint y recibe retroalimentación del Product Owner y otros stakeholders.

Sprint Retrospective (Retrospectiva del Sprint): También al final de cada sprint, el equipo de Scrum se reúne para inspeccionar su desempeño y buscar formas de mejorar continuamente su proceso de trabajo.

5.13. Artefactos de Scrum:

Product Backlog (Backlog del Producto): Es una lista prioritaria de todas las funcionalidades, mejoras y correcciones de errores que deben realizarse en el producto. Es mantenida y priorizada por el Product Owner.

Sprint Backlog (Backlog del Sprint): Es una lista de elementos del backlog del producto seleccionados para el sprint actual, junto con un plan para entregarlos. Es creado por el equipo de desarrollo durante la planificación del sprint. **Incremento:** Es el resultado del trabajo del equipo de desarrollo durante el sprint. Es un incremento potencialmente entregable de producto que debe cumplir con la definición de "Listo." al final del sprint.

5.14. Mockups

Los mockups son representaciones visuales estáticas de la interfaz de usuario de una aplicación o sitio web. Se utilizan durante el proceso de diseño para mostrar cómo se verá y se sentirá la interfaz antes de comenzar la implementación real. Pueden ser dibujos a mano alzada, bocetos digitales o diseños más detallados creados con herramientas de diseño. A diferencia de los prototipos, los mockups no tienen funcionalidad. No se pueden interactuar con ellos ni realizar pruebas de usuario.

5.15. Prototipo

Un prototipo es una representación funcional y mínimamente viable de una aplicación o sistema. Se crea con el propósito de validar conceptos, probar funcionalidades, y obtener retroalimentación de los usuarios antes de desarrollar la versión final del producto. Los prototipos suelen enfocarse en demostrar un conjunto limitado de funcionalidades clave del producto. No todas las características finales están presentes en el prototipo. El propósito principal de un prototipo es validar ideas y conceptos, así como recopilar comentarios y retroalimentación de los usuarios y stakeholders. Esta retroalimentación se utiliza para iterar y mejorar el diseño del producto. A diferencia de los mockups estáticos, los prototipos pueden tener cierto grado de interactividad. Los usuarios pueden interactuar con los elementos de la interfaz de usuario para experimentar la navegación y las funcionalidades básicas. Los prototipos se construyen rápidamente con el mínimo esfuerzo

de desarrollo. Se utilizan herramientas de prototipado rápido que permiten crear interfaces de usuario de manera ágil y sin necesidad de escribir código complejo.

5.15.1. Tipos de prototipos:

Prototipos de Baja Fidelidad: Son esquemáticos y representan una versión muy básica del producto, a menudo creada con lápiz y papel o herramientas de diseño de baja fidelidad. **Prototipos de Alta Fidelidad:** Son más detallados y representan una versión más cercana al producto final. Pueden incluir colores, imágenes y una interactividad más sofisticada.

6. Requerimientos Funcionales

1. El sistema podrá registrar usuarios privilegiados y no privilegiados.
2. Los usuarios registrados deben poder iniciar sesión en el sistema con sus credenciales.
3. Un usuario administrador podrá restringir cuentas de usuarios no privilegiados.
4. El sistema tendrá un servidor para la conexión de los usuarios.
5. El sistema debe proporcionar un chat en tiempo real que permita enviar y recibir mensajes a 1000 personas.
6. Los usuarios deben poder editar sus perfiles, cambiar contraseñas.
7. El usuario podrá listar los usuarios conectados al chat.
8. El sistema podrá asignar roles a distintos usuarios para la regulación del chat.
9. El sistema deberá controlar que los mensajes enviados no superen los 1000 caracteres.
10. El sistema guardará el historial de mensajes.
11. El sistema podrá enviar emoticones y reacciones.
12. El sistema permitirá establecer un estado personalizado para cada usuario (ejem. En línea, Ausente, No Molestar).
13. El sistema permitirá cambiar el nombre de la sala de chat.
14. El usuario podrá enviar y recibir archivos multimedia no mayor a 50 megas y no permitir el envío de Scripts maliciosos.
15. El usuario podrá buscar mensaje por palabras clave.
16. El usuario podrá editar mensajes en un lapso de un minuto.
17. El usuario podrá reportar comportamiento inadecuado a un administrador.
18. El usuario podrá crear salas de chat y agregar otros usuarios a dicha sala.
19. El sistema podrá censurar archivos de contenido inapropiado.
20. El usuario podrá crear encuestas.
21. El usuario podrá buscar a otros usuarios ya sea por nombre.
22. El usuario podrá fijar mensajes importantes en un chat.

7. Requerimientos No Funcionales

1. El sistema funcionará con tecnología Python 3.9.2 para adelante.
2. El sistema debe ser capaz de manejar un número significativo de usuarios concurrentes y salas de chat sin degradar el rendimiento.
3. La interfaz de usuario debe ser intuitiva y fácil de usar, permitiendo a los usuarios navegar y utilizar las funciones del sistema sin dificultad.
4. El sistema debe estar disponible la mayor parte del tiempo, minimizando el tiempo de inactividad programado y no programado.
5. Asegurar que el sistema sea compatible con Linux (Distribución Debian y derivados, Windows 10 en adelante, MacOS).
6. Proporcionar documentación detallada para desarrolladores y usuarios, facilitando la comprensión y el mantenimiento del sistema.
7. El sistema tendrá una base de datos PostgreSQL.
8. El sistema dispondrá de un modo nocturno.
9. La comunicación entre el cliente y el servidor debe estar encriptada mediante SSL.
10. El sistema deberá adherirse a estándares de red IPv4.
11. El sistema deberá encriptar las contraseñas de los usuarios.
12. El usuario podrá cambiar los fondos de las salas de chat.
13. El sistema deberá adaptarse a distintos dispositivos (responsive).
14. El sistema tendrá un tiempo de carga mínimo (menor a 2 segundos).
15. El sistema deberá contar con un ancho de banda mayor a 150 KB/s.
16. El sistema podrá guardar la contraseña del usuario para iniciar sesión mas rápido.
17. El usuario podrá pasar dictados a texto con SpeechRecognition
18. El sistema podrá reproducir audio del contenido multimedia recibido y enviado en segundo plano.
19. El sistema contará con autenticación de dos pasos mediante correo electrónico.
20. El sistema tendrá un soporte de mensajería en idioma inglés usando Google Translator API.
21. El sistema pondrá estados a los mensajes acorde a si fueron entregados, leídos.

8. Diagrama de cuadrantes de requerimientos

Los diagramas de requerimientos son herramientas visuales que ayudan a representar de manera gráfica los diversos aspectos y relaciones entre los requisitos de un sistema. Ayuda a los equipos de desarrollo a priorizar y planificar mejor sus actividades, centrándose inicialmente en los requisitos esenciales mientras dejan espacio para la mejora continua.

Dividimos el cuadrante en cuatro partes, en el eje horizontal del cliente se tiene dos divisiones las cuales representan la importancia del requerimiento (prioritario, menos importante), en el eje vertical del equipo de desarrollo se tiene de igual forma dos divisiones las cuales representan la dificultad en el desarrollo del software para el cumplimiento del requerimiento.

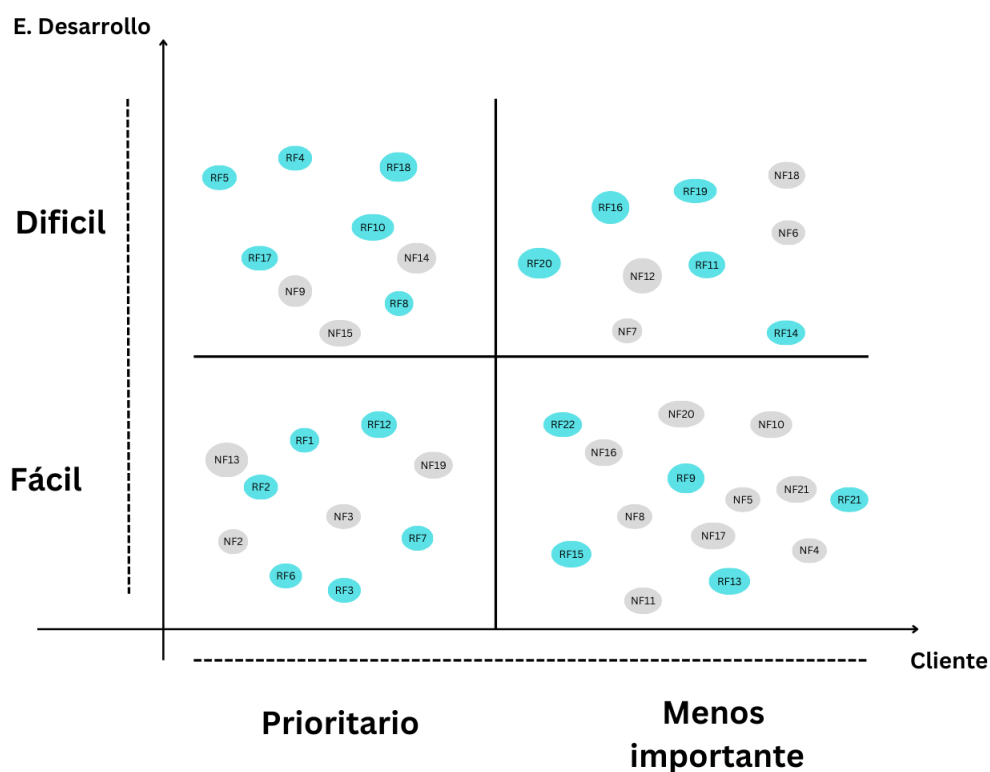


Figura 6: Diagrama de cuadrante de requerimientos

8.1. Tipo de programación a emplear

Para escoger que requerimiento va a que cuadrante primeramente se escogió el tipo de programación que se va a emplear para el proyecto, se empleará la programación deductiva ya que se considera que algunos elementos ya van a estar hechos o se van a hacer en un futuro y en base a que ya funcionará varias cosas se va a trabajar en otras funcionalidades.

9. Módulos

Una vez realizado el diagrama se puede dividir el trabajo del desarrollo del software en módulos, en este caso se va a dividir el proyecto en dos módulos.

La siguiente imagen del cuadrante muestra que requerimientos abarcan cada módulo, siendo el primer modulo el conjunto de requerimientos por encima de la linea diagonal y el segundo módulo el restante inferior.

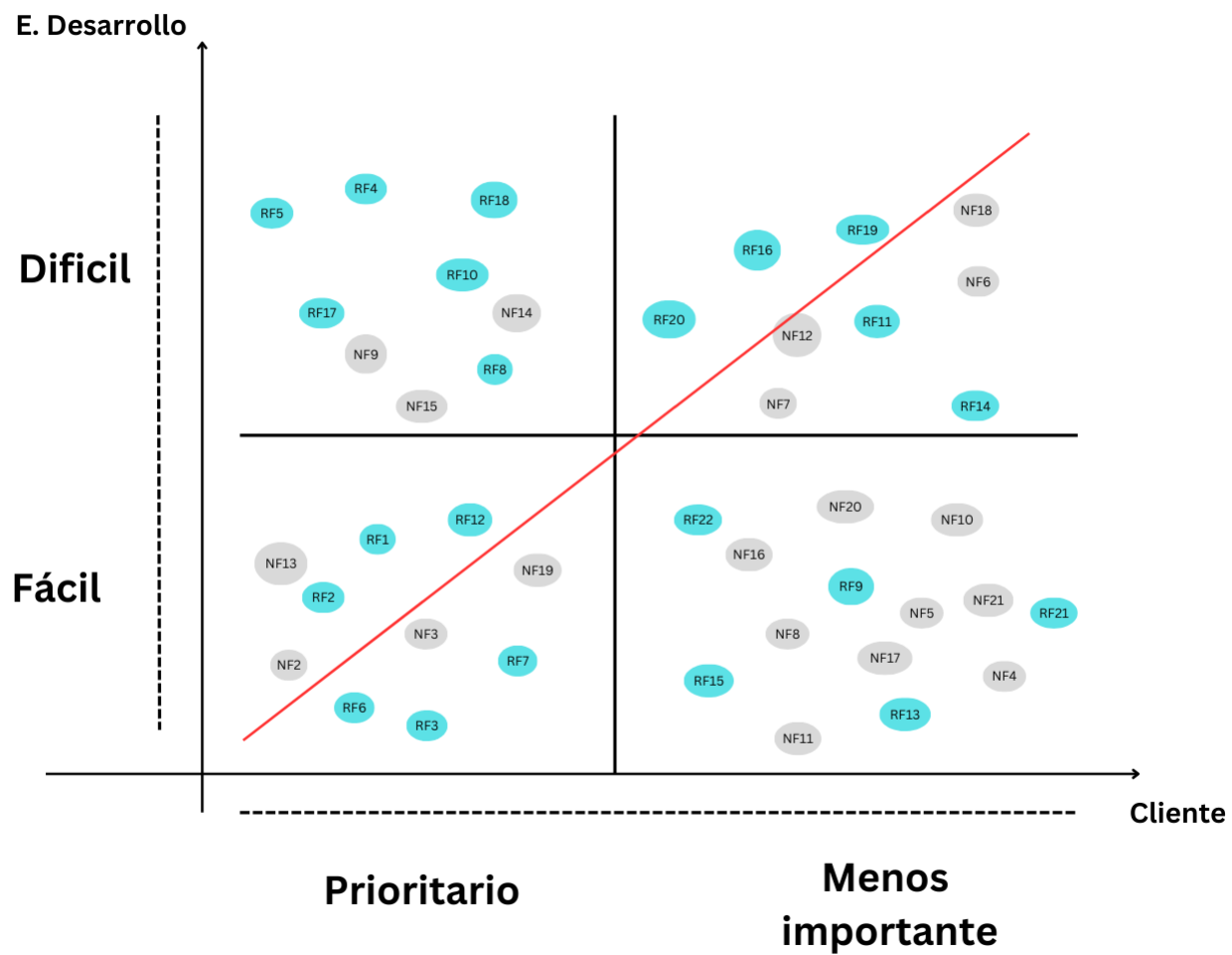


Figura 7: Diagrama de cuadrante de requerimientos

9.1. Módulo 1

Objetivo	Actividad	Nombre de actividad	Fecha de inicio	Duración en días	Fecha fin
RF4	Actividad 1	Establecer IP host y puerto del servidor	01-02-24	1	02-02-24
	Actividad 2	Crear un socket de servidor en modo IPv4	01-02-24	1	02-02-24
	Actividad 3	Bindar la IP y el puerto al socket	01-02-24	1	02-02-24
	Actividad 4	Crear las llaves y el certificado para la protección SSL	01-02-24	1	02-02-24
	Actividad 5	Bindar las llaves de seguridad al socket	01-02-24	1	02-02-24
	Actividad 6	Configurar el socket a modo escucha para un máximo de 1000 clientes	01-02-24	1	02-02-24
RF5	Actividad 7	Establecer IP host y puerto de conexión	02-02-24	1	03-02-24
	Actividad 8	Crear un socket de cliente con su hilo correspondiente	02-02-24	1	03-02-24
	Actividad 9	Conectar el socket cliente con el socket servidor que esta en escucha	02-02-24	1	03-02-24
	Actividad 10	Crear una función que permita enviar y recibir mensajes	02-02-24	1	03-02-24
RF18	Actividad 11	Crear un hilo para cada sala creada	05-02-24	1	06-02-24
	Actividad 12	Generar una llave para que un usuario pueda ingresar a la sala	05-02-24	1	06-02-24
RF10	Actividad 13	Diseñar e implementar el esquema de la base de datos	05-02-24	1	06-02-24
	Actividad 14	Manejar de limites de almacenamiento	05-02-24	1	06-02-24
	Actividad 15	Mostrar el historial de mensajes por la interfaz al abrir una sala	05-02-24	1	06-02-24
RF17	Actividad 16	Implementar un sistema para notificar a los administradores de los reportes recibidos	06-02-24	1	07-02-24
	Actividad 17	Diseñar un formulario simple que permita a los usuarios especificar el motivo del reporte	06-02-24	1	07-02-24
	Actividad 18	Agregar un boton en la interface que permita a los usuarios reportar un mensaje o comportamiento	06-02-24	1	07-02-24
NF14	Actividad 19	Crear un boton con la funcionalidad de enviar archivos multimedia	07-02-24	1	08-02-24
	Actividad 20	Carga progresiva (identificar el contenido critico para la visualizacion inicial)	07-02-24	1	08-02-24
	Actividad 21	Compresion de recursos estaticos (imágenes y codigo)	07-02-24	1	08-02-24
RF8	Actividad 22	Identificar las funcionalidades asociadas a roles	07-02-24	1	08-02-24
	Actividad 23	Asignar rol a un usuario	07-02-24	1	08-02-24
RF12	Actividad 24	Creación de estados (En linea, Ausente, No molestar)	08-02-24	1	09-02-24
	Actividad 25	Persistencia del estado	08-02-24	1	09-02-24
RF1	Actividad 26	Guardar usuarios y contraseñas en la base de datos	08-02-24	1	09-02-24
	Actividad 27	Implementar la doble autentificacion de usuarios	08-02-24	1	09-02-24
NF13	Actividad 28	Detectar las características del dispositivo	12-02-24	1	13-02-24
	Actividad 29	Adaptar la interfaz del sistema	12-02-24	1	13-02-24
	Actividad 30	Pruebas en multiples en dispositivos	12-02-24	1	13-02-24
RF2	Actividad 31	Validar credenciales	13-02-24	1	14-02-24
	Actividad 32	Verificar la doble autentificación	13-02-24	1	14-02-24
	Actividad 33	Permitir 3 intentos, caso contrario espera 1 minuto	13-02-24	1	14-02-24
NF2	Actividad 34	Realizar test de carga	14-02-24	1	15-02-24
RF20	Actividad 35	Creación de encuestas	14-02-24	1	15-02-24
	Actividad 36	Almacenar de encuestas	14-02-24	1	15-02-24
	Actividad 37	Gestión de respuestas	14-02-24	1	15-02-24
	Actividad 38	Asignar plazos y restricciones de tiempo	14-02-24	1	15-02-24
	Actividad 39	Acción que permita editar y eliminar encuesta	14-02-24	1	15-02-24
RF16	Actividad 40	Recuperar mensaje anterior	15-02-24	1	16-02-24
	Actividad 41	Editar mensaje	15-02-24	1	16-02-24
	Actividad 42	Enviar mensaje editado	15-02-24	1	16-02-24
	Actividad 43	Notificar a los usuarios restantes al editar un mensaje	15-02-24	1	16-02-24
RF19	Actividad 44	Definir de contenido inapropiado	19-02-24	1	20-02-24
	Actividad 45	Importar API de la I.A. Clarifai detección de dicho contenido	19-02-24	1	20-02-24
	Actividad 46	Implementar filtros	19-02-24	1	20-02-24

Figura 8: Módulo 1

9.2. Módulo 2

Objetivo	Actividad	Nombre de actividad	Fecha de inicio	Duración	Fecha de fin
NF3	Actividad 1	Crear interfaz con jerarquía visual y buena organización	20-02-24	2	22-02-24
	Actividad 2	Poner iconos y etiquetas claras	20-02-24	2	22-02-24
	Actividad 3	Tener consistencia en el diseño	20-02-24	2	22-02-24
RF7	Actividad 4	Manejar eventos de conexión y desconexión de usuarios	22-02-24	1	23-02-24
	Actividad 5	Recuperar los nombres de los usuarios conectados	22-02-24	1	23-02-24
	Actividad 6	Mostrar los nombres solo al usuario que lo solicitó	22-02-24	1	23-02-24
RF6	Actividad 7	Crear formulario para la edición de perfiles	23-02-24	1	24-02-24
	Actividad 8	Crear formulario para el cambio de contraseñas	23-02-24	1	24-02-24
	Actividad 9	Autenticar y validar datos del usuario que quiere editar su perfil o contraseña	23-02-24	1	24-02-24
RF3	Actividad 10	Actualizar nueva contraseña	23-02-24	1	24-02-24
	Actividad 11	Verificar si el usuario es administrador	26-02-24	1	27-02-24
	Actividad 12	Crear funcionalidad de restringir ciertas acciones a usuarios (silenciar, denegar acceso a salas)	26-02-24	1	27-02-24
NF19	Actividad 13	Notificar a los usuarios afectados	26-02-24	1	27-02-24
	Actividad 14	Listar de usuarios en estado de restricción	26-02-24	1	27-02-24
	Actividad 15	Recuperar el correo electrónico del usuario	27-02-24	1	28-02-24
NF6	Actividad 16	Generar un token con tiempo de expiración de un minuto	27-02-24	1	28-02-24
	Actividad 17	Enviar token al correo electrónico	27-02-24	1	28-02-24
	Actividad 18	Comparar el token que ingresa el usuario y autenticar	27-02-24	1	28-02-24
NF6	Actividad 19	Introducción del proyecto	28-02-24	16	15-03-24
	Actividad 20	Requisitos del sistema	28-02-24	16	15-03-24
	Actividad 21	Configuración del entorno de desarrollo	28-02-24	16	15-03-24
RF11	Actividad 22	Arquitectura de software	28-02-24	16	15-03-24
	Actividad 23	Modelo de datos	28-02-24	16	15-03-24
	Actividad 24	Guía de instalación (usuarios)	28-02-24	16	15-03-24
NF6	Actividad 25	Inicio rápido	28-02-24	16	15-03-24
	Actividad 26	Funcionalidades del sistema	28-02-24	16	15-03-24
	Actividad 27	Guías paso a paso	28-02-24	16	15-03-24
RF11	Actividad 28	Políticas de privacidad	28-02-24	16	15-03-24
	Actividad 29	Diseñar emoticones	28-02-24	1	29-02-24
	Actividad 30	Acción de enviar emoticon	28-02-24	1	29-02-24
NF6	Actividad 31	Limitar cantidad de reacciones	28-02-24	1	29-02-24
	Actividad 32	Visualizar reacciones por debajo del mensaje	28-02-24	1	29-02-24
	Actividad 33	Notificar al usuario que envió el mensaje sobre las reacciones	28-02-24	1	29-02-24

Figura 9: Módulo 2.1

NF12	Actividad 34 Definir fondos predefinidos	29-02-24	1	01-03-24
	Realizar funcionalidad de que cada usuario pueda cargar sus			
	Actividad 35 propias imágenes de fondo	29-02-24	1	01-03-24
	Actividad 36 Hacer que los fondos modificados sean persistentes	29-02-24	1	01-03-24
	Actividad 37 Limitar las imágenes de fondo a un tamaño menor a 1920x1080	29-02-24	1	01-03-24
NF7	Actividad 38 Diseñar las tablas necesarias	01-03-24	1	02-03-24
	Actividad 39 Crear las llaves primarias y foraneas	01-03-24	1	02-03-24
	Actividad 40 Normalizar la base de datos	01-03-24	1	02-03-24
	Actividad 41 Agregar indices	01-03-24	1	02-03-24
	Actividad 42 Crear procesos almacenados y triggers	01-03-24	1	02-03-24
	Actividad 43 Optimizar consultas	01-03-24	1	02-03-24
RF14	Actividad 44 Limitar el de tamaño de archivos	04-03-24	1	05-03-24
	Agregar la funcionalidad de enviar y recibir archivos			
	Actividad 45 multimedia	04-03-24	1	05-03-24
	Actividad 46 Validar el tipo de archivo que se desea enviar	04-03-24	1	05-03-24
	Actividad 47 No permitir el envío de scripts maliciosos	04-03-24	1	05-03-24
NF18	Actividad 48 Implementar un reproductor de audio	05-03-24	1	06-03-24
	Actividad 49 Mediante un hilo hacer que funcione en segundo plano	05-03-24	1	06-03-24
	Crear un file oculto y protegido que funcione como un cache			
NF16	Actividad 50 para almacenar contraseña en caso de poner guardar contraseña	06-03-24	1	07-03-24
	Actividad 51 Almacenar la contraseña en el cache creado de forma segura	06-03-24	1	07-03-24
	La primera vez que inicia sesión al guardar se realizará doble			
	Actividad 52 autenticación	06-03-24	1	07-03-24
RF9	Actividad 53 Recuperar mensaje enviado	07-03-24	1	08-03-24
	Actividad 54 Verificar el tamaño del mensaje	07-03-24	1	08-03-24
	Actividad 55 Si es mayor a mil caracteres no se enviará el mensaje	07-03-24	1	08-03-24
NF5	Actividad 56 Verificar que los dispositivos tengan instalados python	08-03-24	1	09-03-24
	Actividad 57 Verificar que las dependencias esten instaladas	08-03-24	1	09-03-24
RF15	Actividad 58 Recuperar el historial de mensajes	08-03-24	1	09-03-24
	Actividad 59 Utilizando un find, rfind, lfind buscar por palabras clave	08-03-24	1	09-03-24
	Actividad 60 En caso de que exista uno o varios match, mostrar resultados	08-03-24	1	09-03-24
NF4	Actividad 61 Levantar el servidor en el dispositivo seleccionado por el cliente	11-03-24	1	12-03-24
	Actividad 62 Programar tareas cron para limpieza de archivos temporales	11-03-24	1	12-03-24
RF13	Actividad 63 Recuperar el atributo que define el nombre de la sala	11-03-24	1	12-03-24
	Actividad 64 Cambiar el atributo por el nombre que un usuario ingrese	11-03-24	1	12-03-24
	Actividad 65 Actualizar el registro del nombre de la sala en la base de datos	11-03-24	1	12-03-24
	Actividad 66 Notificar a los usuarios sobre el cambio	11-03-24	1	12-03-24
	Al momento de guardar la contraseña en la base de datos			
NF11	Actividad 67 configurar la encriptación con SHA2	12-03-24	1	13-03-24
NF20	Actividad 68 Importar Google Translate API para traducir mensajes	12-03-24	1	13-03-24
	Llamar la función que permita traducir mensajes cada que se la llame			
	Actividad 69 mediante la interfaz	12-03-24	1	13-03-24
	Cuando un usuario ingresa al chat, agregar el nombre del			
NF21	Actividad 70 usuario a una lista	13-03-24	1	14-03-24
	Actividad 71 Mostrar en la interfaz del que escribio el mensaje	13-03-24	1	14-03-24
RF21	Actividad 72 Filtrar la lista de usuarios por un nombre que se ingrese	13-03-24	1	14-03-24
	Actividad 73 Retornar usuarios con nombres similares	13-03-24	1	14-03-24
RF22	Actividad 74 Crear un apartado para mensajes fijados	14-03-24	1	15-03-24
	Actividad 75 Crear una copia del mensaje a los mensajes fijados	14-03-24	1	15-03-24
	Actividad 76 Acción de eliminar mensajes fijados	14-03-24	1	15-03-24

Figura 10: Módulo 2.2

10. Diagrama de Gantt

10.1. Diagrama de Gantt modulo 1

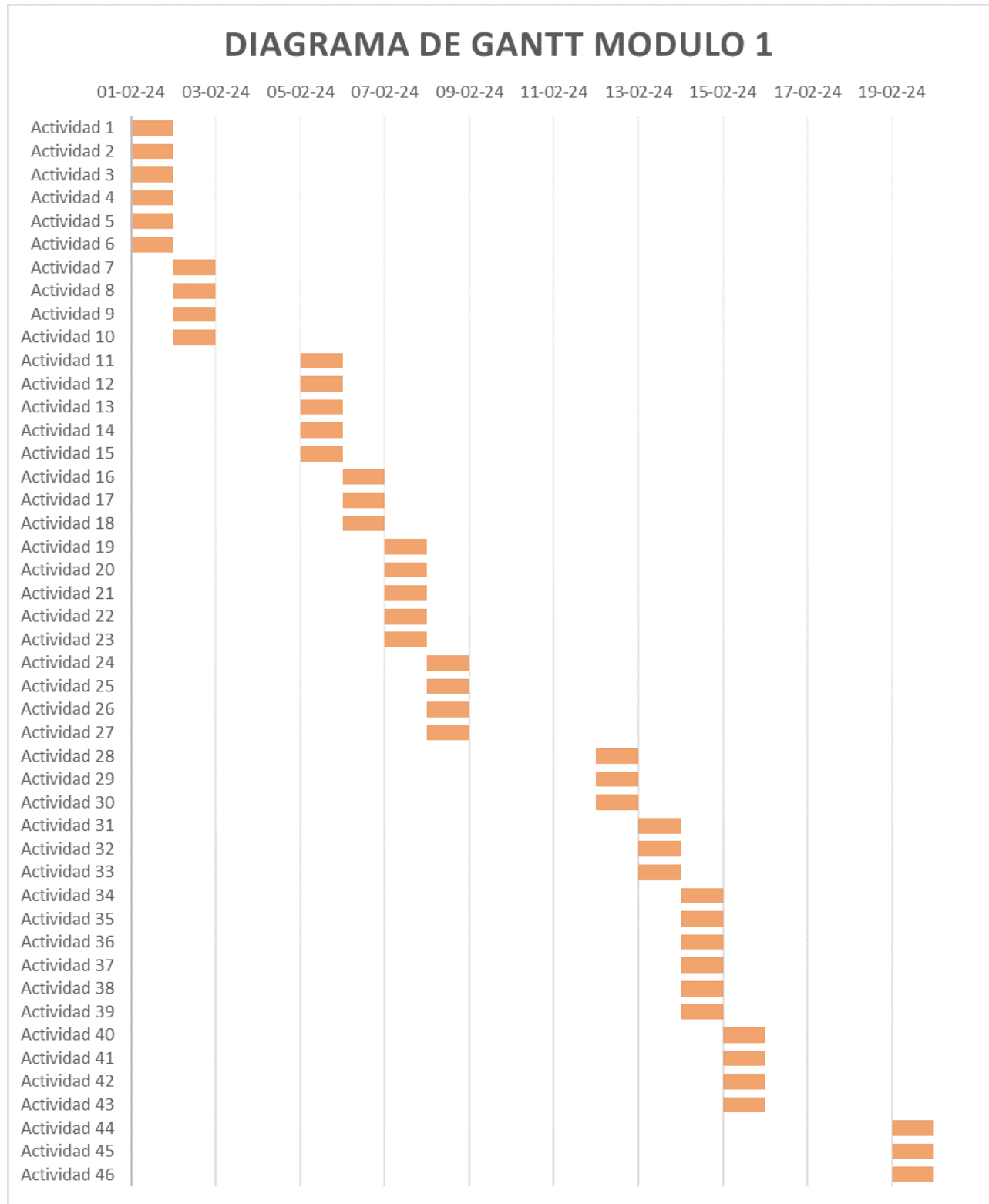


Figura 11: Gantt módulo 1

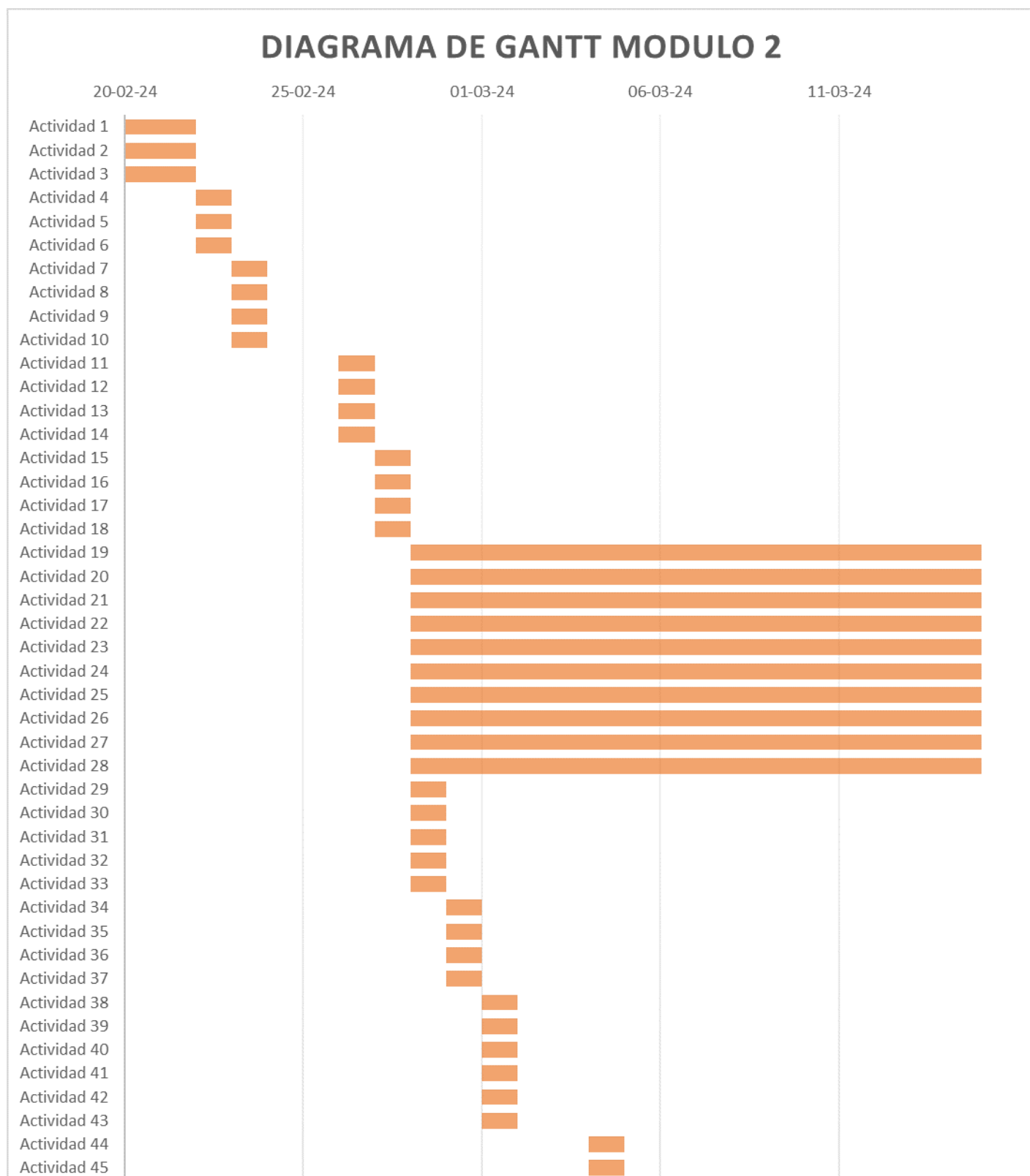
10.2. Diagrama de Gantt modulo 2

Figura 12: Gantt módulo 2

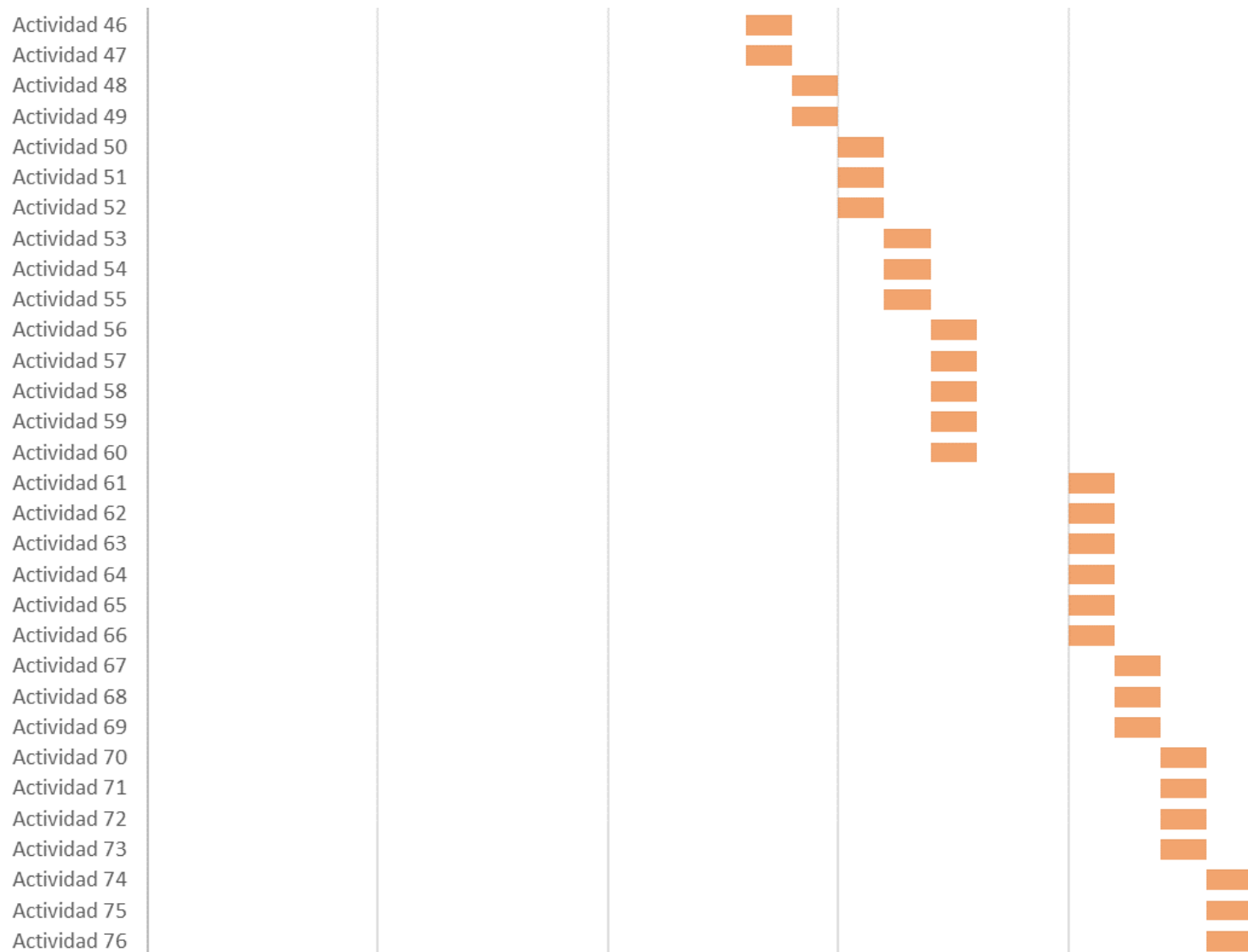
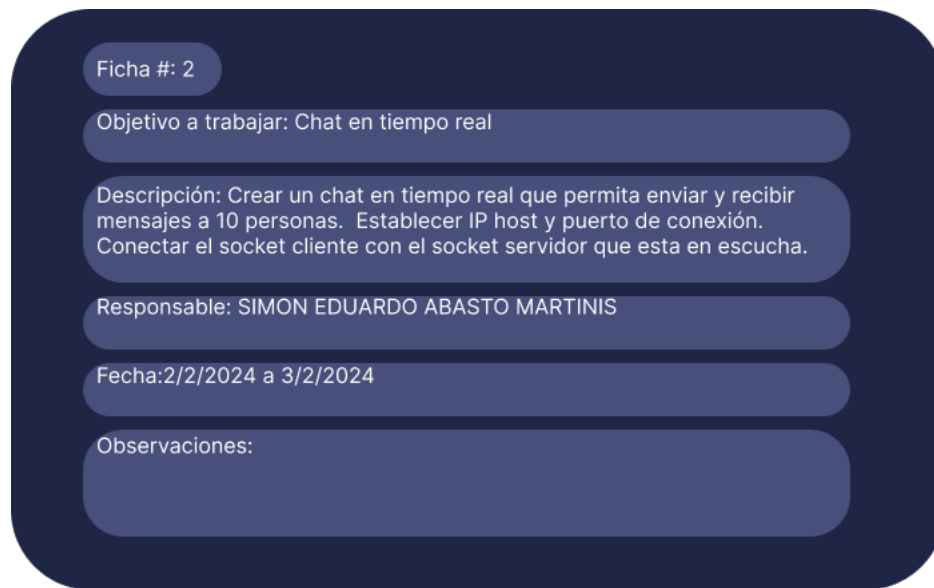


Figura 13: Gantt módulo 2

10.3. Fichas de requerimientos

Un formulario de ficha de requerimientos con un fondo oscuro y elementos de texto en tonos más claros. El formulario contiene los siguientes campos: 'Ficha #: 2', 'Objetivo a trabajar: Chat en tiempo real', 'Descripción: Crear un chat en tiempo real que permita enviar y recibir mensajes a 10 personas. Establecer IP host y puerto de conexión. Conectar el socket cliente con el socket servidor que esta en escucha.', 'Responsable: SIMON EDUARDO ABASTO MARTINIS', 'Fecha: 2/2/2024 a 3/2/2024' y 'Observaciones:'.

Ficha #: 2

Objetivo a trabajar: Chat en tiempo real

Descripción: Crear un chat en tiempo real que permita enviar y recibir mensajes a 10 personas. Establecer IP host y puerto de conexión. Conectar el socket cliente con el socket servidor que esta en escucha.

Responsable: SIMON EDUARDO ABASTO MARTINIS

Fecha: 2/2/2024 a 3/2/2024

Observaciones:

Figura 14: Gantt módulo 2

Se tiene todas las fichas de requerimientos en el repositorio creado para el proyecto.



Figura 15: QR del repositorio

11. Factibilidad

11.1. Factibilidad económica

Los siguientes datos son obtenidos gracias a la investigación de la situación actual de los recursos humanos para el desarrollo de software, lo que nos permite sacar conclusiones del valor hora-hombre de los programadores considerando el tiempo de auto aprendizaje, entre otros.

Recurso Humanos	Horas	Días
Modulo 1	60	15
Modulo 2	110	22
Total	170	37

Cuadro 1: Horas de desarrollo por Módulos

Considerando el actual mercado de desarrollo de software por ingenieros de software el precio por hora de desarrollo ronda por los **25.00 Bs.- (25.00 BOLIVIANOS POR HORA)** dandonos un costo total del proyecto por 37 días de desarrollo de :

23,185 Bs.- (VEINTITRES MIL CIENTO OCHENTA Y CINCO 00/100 BOLIVIANOS)

Costos de recursos tecnológicos:

En este proyecto , no existe la necesidad de inversión económica en servicios de terceros, ya que para el desarrollo del sistema no es necesario algún software de pago o hardware específico mas de los actualmente disponible.

11.2. Factibilidad legal

Probar que nuestro sistema cumple con la factibilidad legal como objetivo principal es demostrar que el sistema no transgrede ningún artículo de la LEY N° 164 Artículo 75 al 77 que establece que todos los organismos ejecutivo , legislativo , judicial y electoral en todos sus niveles, promoverán y priorizarán la utilización de software libre.

CAPÍTULO SEGUNDO GOBIERNO ELECTRÓNICO Y SOFTWARE LIBRE

Artículo 75. (GOBIERNO ELECTRÓNICO).

- I. El nivel central del Estado promueve la incorporación del Gobierno Electrónico a los procedimientos gubernamentales, a la prestación de sus servicios y a la difusión de información, mediante una estrategia enfocada al servicio de la población.
- II. El Órgano Ejecutivo del nivel central del Estado, elaborará los lineamientos para la incorporación del Gobierno Electrónico.

Artículo 76. (ALCANCE). El Estado fijará los mecanismos y condiciones que las entidades públicas aplicarán para garantizar el máximo aprovechamiento de las tecnologías de la información y comunicación, que permitan lograr la prestación de servicios eficientes. Artículo 77. (SOFTWARE LIBRE).

- I. Los Órganos Ejecutivo, Legislativo, Judicial y Electoral en todos sus niveles, promoverán y priorizarán la utilización del software libre y estándares abiertos, en el marco de la soberanía y seguridad nacional.
- II. El Órgano Ejecutivo del nivel central del Estado, elaborará el plan de implementación de software libre y estándares abiertos en coordinación con los demás órganos del Estado y entidades de la administración pública.

De acuerdo a lo anterior, las personas involucradas en el desarrollo y asesoría del proyecto han aceptado condiciones como mantener la confidencialidad y el manejo ético de la información, protegiendo en todo momento la integridad de cada una de las personas y los datos involucrados, por lo que el proyecto está totalmente dentro del marco legal.

11.3. Factibilidad técnica

Analizamos si están disponibles el hardware y software necesarios para la realización del proyecto y si cuentan con la capacidad de llevar a cabo todas las alternativas de desarrollo y diseño que se este considerando.

El sistema estará basado en un lenguaje de programación Python 3.9.2 utilizando sus librerías que nos permitan desarrollar un sistema de chat para 1000 usuarios cumpliendo con las normas de seguridad y privacidad. El sistema será compatible con sistemas Windows, Linux, MacOS para que los usuarios no tengan limitaciones de uso.

Componente	Especificaciones mínimas
Procesador del equipo	CPU intel I3
Memoria RAM	4 GB
Disco Duro/Solido	160GB
Tarjeta de video	1GB
Mouse	Estándar
Teclado	Estándar
Monitor	1024 x 768 píxeles
% Tarjeta de red	Ethernet 10/100 Mbps

Cuadro 2: Hardware

Componente	Componente Requisitos del sistema
Sistema operativo	Windows, Linux , MacOS

Cuadro 3: Software

Python 3.9.2
Librerías Python
PosgreSQL

Cuadro 4: Lenguajes y Herramientas a utilizar

12. Robustez

12.1. Seguridad

El sistema contara con Robustez de seguridad al usar la autenticación en dos pasos mediante contraseña y código de verificación enviado al correo electrónico para prevenir daños de la integridad de los datos de los usuarios.

12.2. Entrada

El sistema contara con Robustez de entrada para evitar posibles daños y ataques contra la integridad del sistema y los datos de los usuarios, se usara Filtrado para eliminar o neutralizar caracteres especiales y códigos maliciosos que podrían utilizarse en ataques de inyección, como SQL injection

12.3. Adaptividad

El sistema contara con Robustez de adaptabilidad, se refiere a la capacidad de un sistema de software para funcionar de manera eficiente y sin interrupciones en diferentes entornos y condiciones.

12.4. Manejo de errores

El sistema contara con Robustez de manejo de errores, se refiere a la capacidad de un sistema para gestionar errores y continuar operando de manera segura incluso cuando se producen situaciones inesperadas o problemas. Será capaz de detectar, informar y, cuando sea posible, recuperarse de errores sin afectar gravemente la integridad o la disponibilidad del sistema.

13. Capacitación

Para asegurar una adopción exitosa del sistema por parte de los administradores, se llevará a cabo un programa de capacitación. Esta capacitación se centrará en proporcionar a los administradores las habilidades necesarias para utilizar y mantener eficientemente las funciones ofrecidas por el sistema y a los usuarios para el correcto uso del sistema. La duración de la capacitación sera de 1 semana.

13.1. Iniciar Sesión y Registro de Usuarios

Se capacitará a los administradores y usuarios sobre el proceso de registro en el sistema, la gestión de credenciales y el inicio de sesión, también la forma en la que se relaciona con la base de datos.

13.2. Funciones Básicas de Chat

Se proporcionará una guía detallada sobre el funcionamiento enviar y recibir mensajes, crear y unirse a salas de chat, así como la gestión de contactos.

13.2.1. Configuración de Perfiles y Preferencias

Los administradores y los usuarios aprenderán a gestionar los perfiles, cambiar contraseñas, establecer estados personalizados.

13.3. Funciones Avanzadas

Se proporcionará capacitación detallada sobre funciones avanzadas como el envío de archivos multimedia, la creación de encuestas, y el uso de emoticones y reacciones.

13.4. Entorno de Desarrollo

Se proporcionará información detallada sobre el entorno de desarrollo, incluyendo la versión de Python utilizada, las dependencias y la configuración recomendada.

13.5. Comunicación cliente servidor

los administradores seran capacitados en la comunicación cliente-servidor y la gestión de bases de datos PostgreSQL.

13.6. Seguridad y Encriptación

Se enfatizará la importancia de la seguridad, incluida la encriptación SSL para la comunicación cliente-servidor y la encriptación de contraseñas de usuarios.

13.7. Documentación y Mantenimiento

Se proporcionará orientación detallada sobre la documentación para administradores y usuarios, así como las mejores prácticas de mantenimiento del sistema.

14. Soporte

Para garantizar una experiencia positiva para los usuarios, se proporcionará un servicio de soporte que abarcará desde la resolución de problemas técnicos hasta la orientación sobre el uso adecuado del sistema.

14.1. Centro de Ayuda en Línea

Tutoriales paso a paso y preguntas frecuentes para que los usuarios encuentren respuestas a sus consultas de manera autónoma.

14.2. Asistencia Técnica por Correo Electrónico

Los usuarios podrán enviar consultas y problemas técnicos a través de una dirección de correo electrónico designada. Se garantizará una respuesta veloz

14.3. Chat en Vivo

Se proporcionará un servicio de chat en vivo para consultas inmediatas. Este servicio estará disponible durante horas específicas.

14.4. Reuniones Periódicas

Se programarán 2 reuniones periódicas cada media semana para consultas de dudas finales

15. Diagramas

15.1. Diagrama de clases

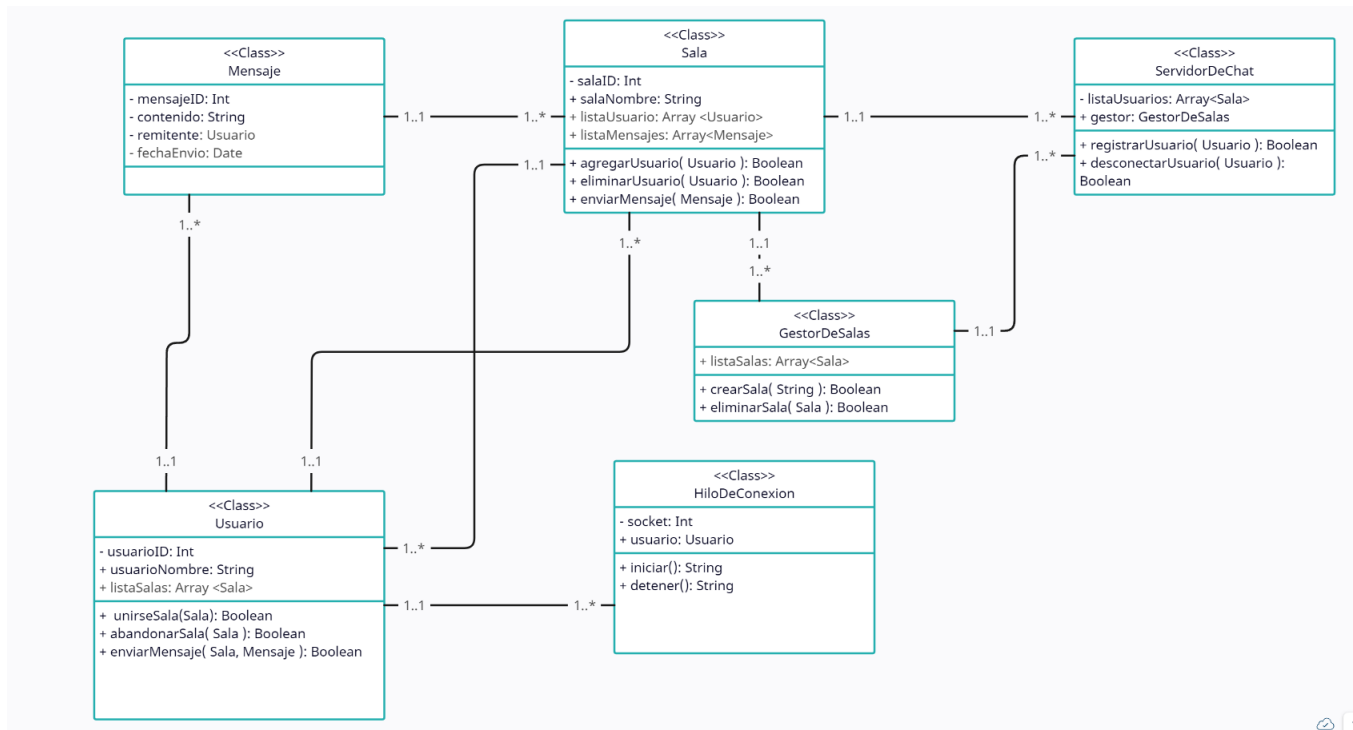


Figura 16: Diagrama de Clases

15.2. Diagrama de casos de uso

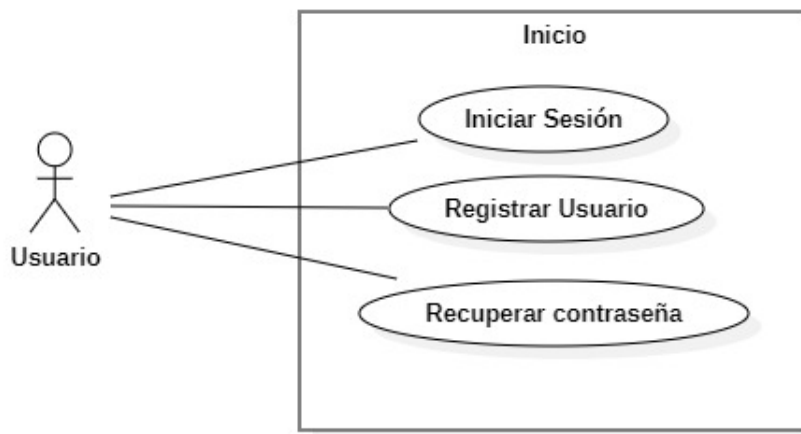


Figura 17: Diagrama de caso de uso

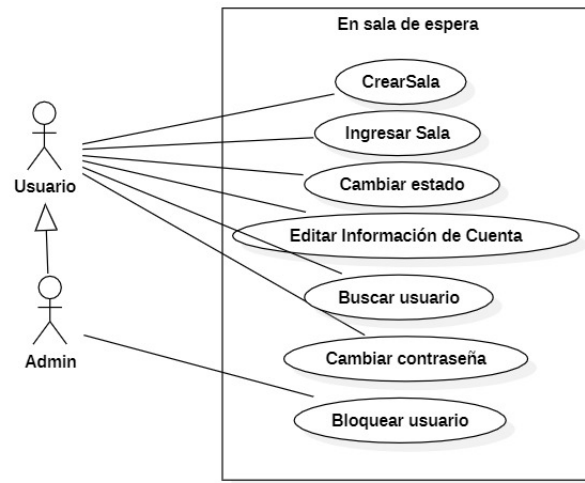


Figura 18: Diagrama de caso de uso 2

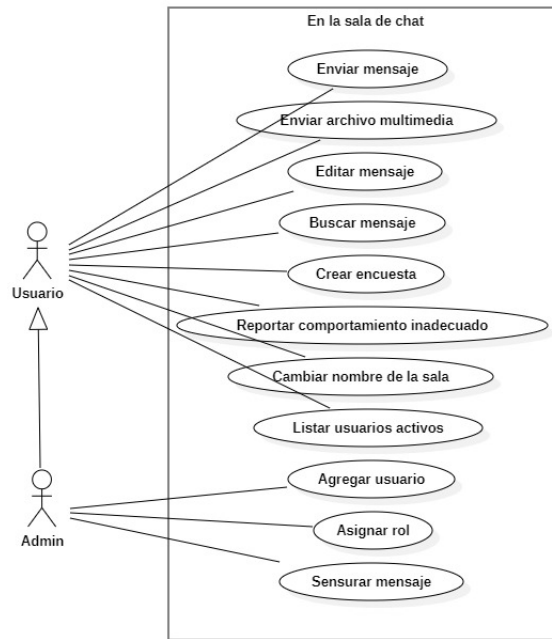


Figura 19: Diagrama de caso de uso 3

15.3. Diagrama de arquitectura por capas

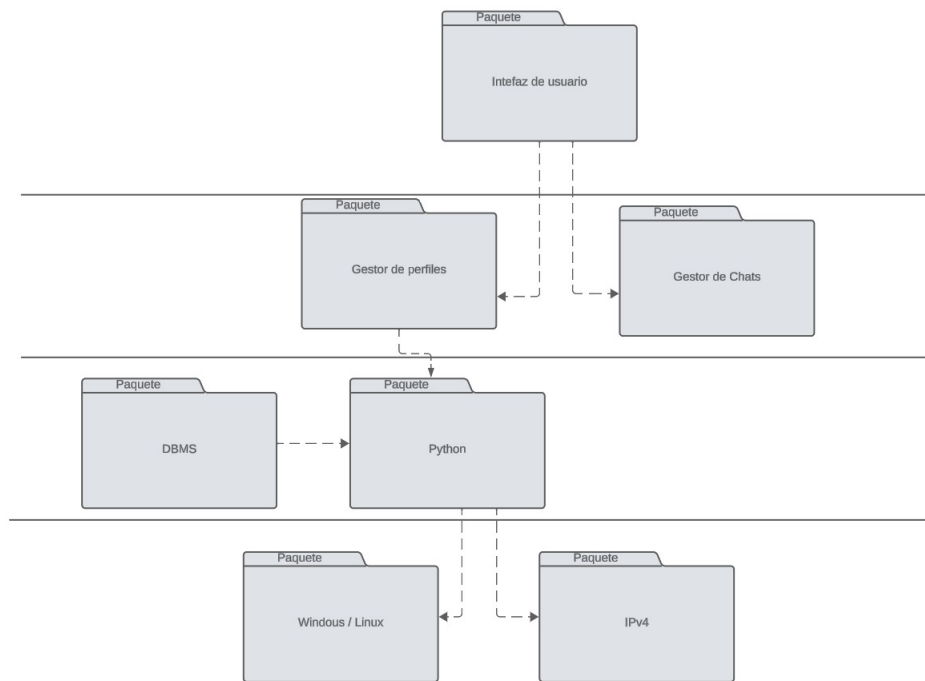


Figura 20: Diagrama de arquitectura por capas

15.4. Diagrama de despliegue

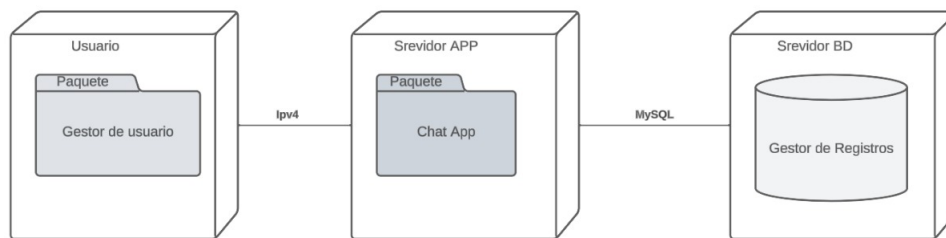


Figura 21: Diagrama de despliegue

15.5. Diagrama de estados

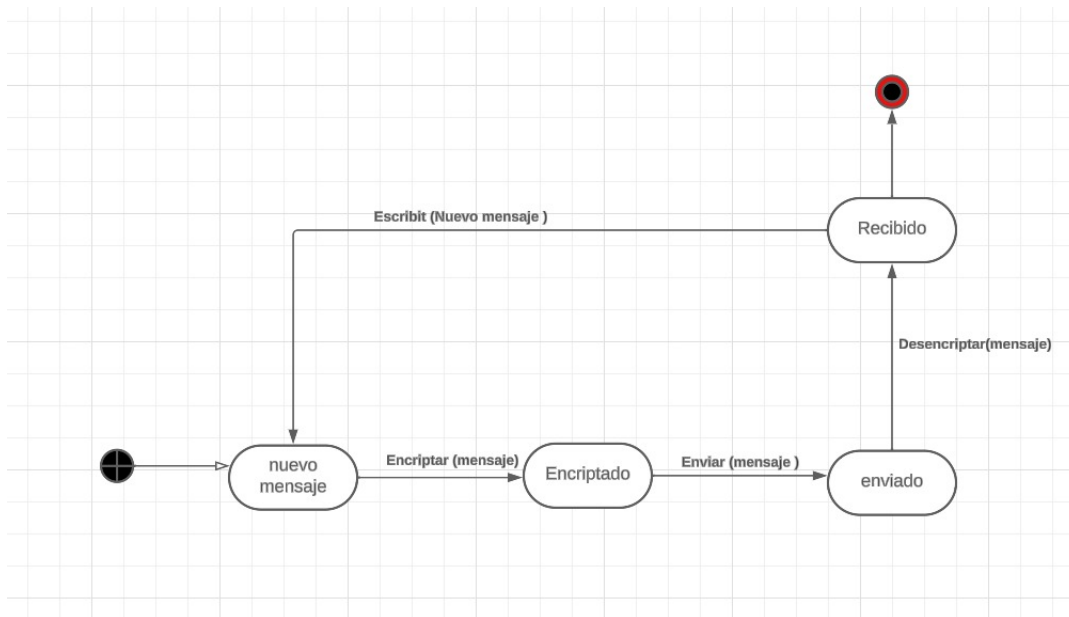


Figura 22: Diagrama de Estados

15.6. Diagrama de actividades

Diagrama de actividades Iniciar Sesión

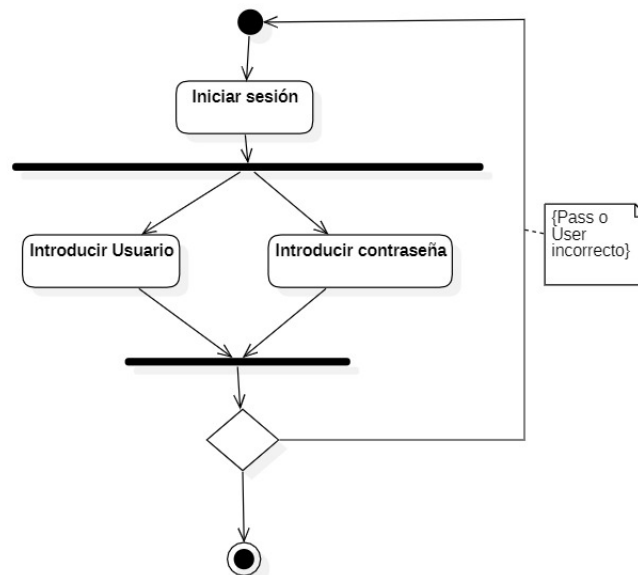


Figura 23: Diagrama de actividades iniciar sesión

Diagrama de actividades Eliminar registro

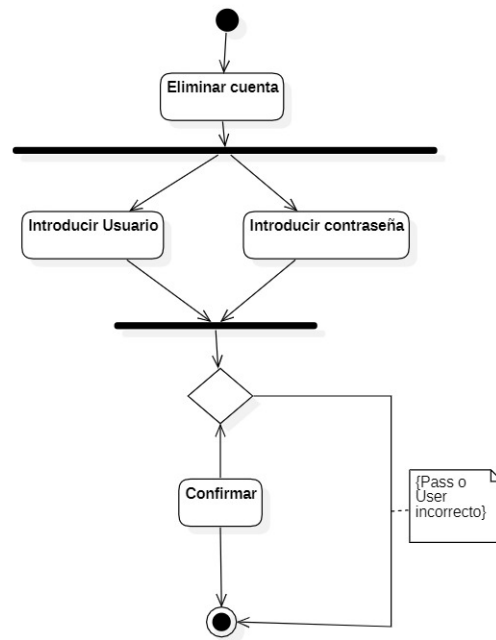


Figura 24: Diagrama de actividades eliminar registro

15.7. Diagrama de secuencia

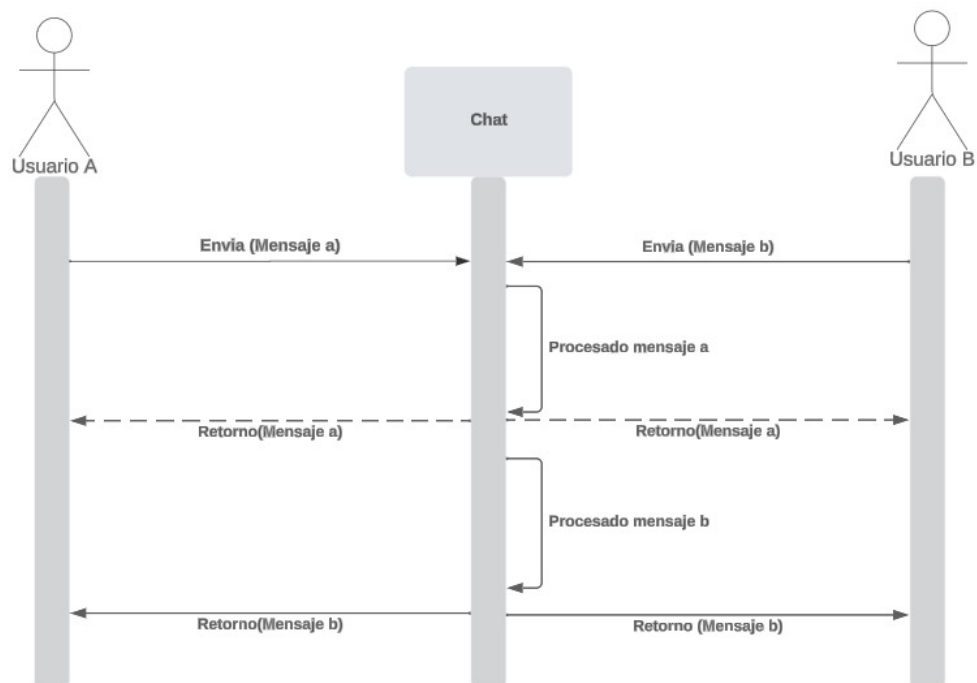


Figura 25: Diagrama de secuencia

15.8. Diagrama de colaboración

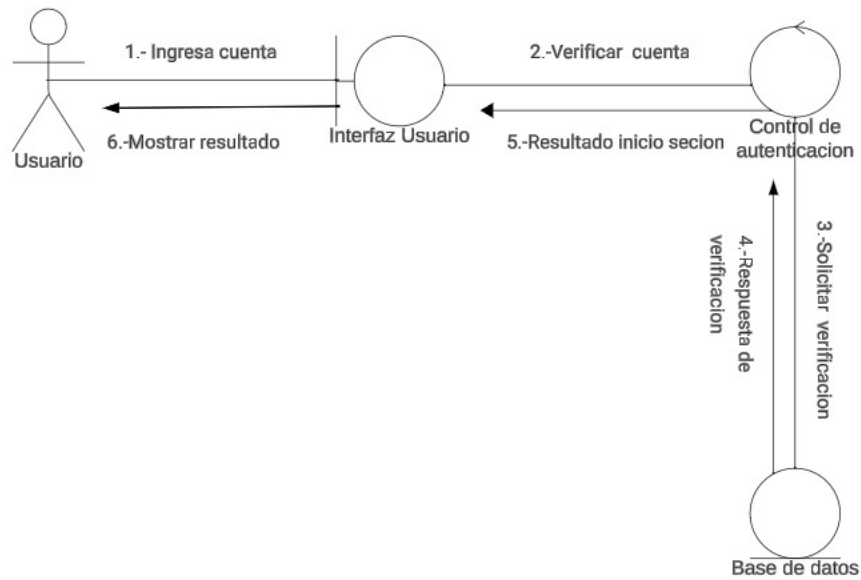


Figura 26: Diagrama de colaboración ingresar cuenta

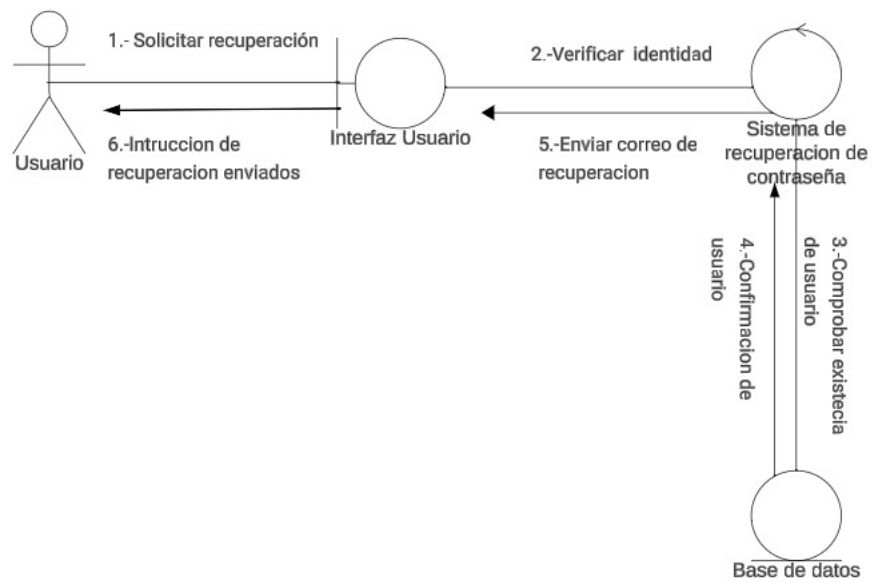


Figura 27: Diagrama de colaboración Recuperar cuenta

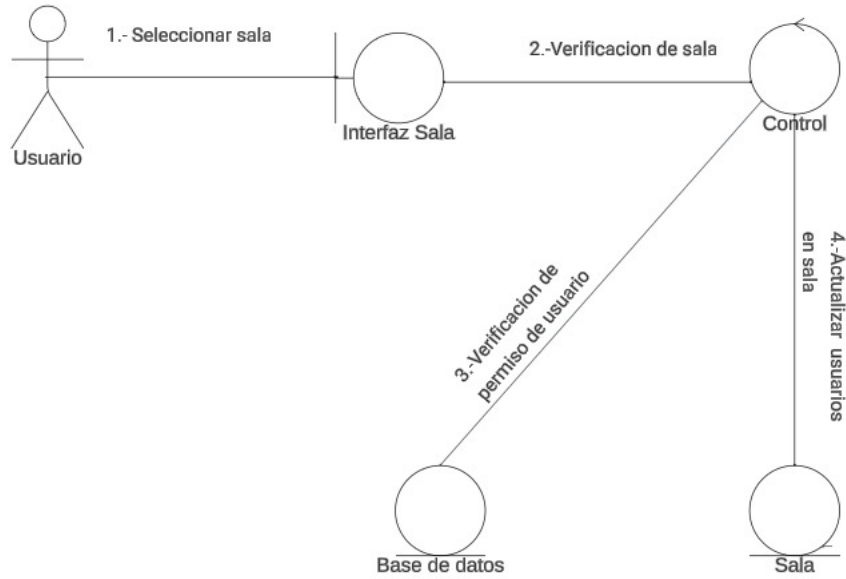


Figura 28: Diagrama de colaboración Seleccionar sala

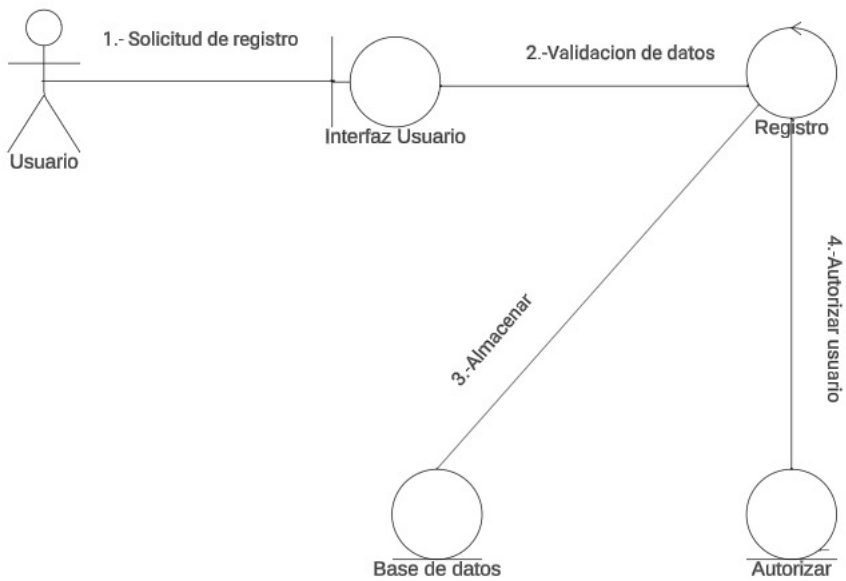


Figura 29: Diagrama de colaboración Solicitud de registro



Figura 30: Diagrama de colaboración envío de archivos multimedia

16. Versiones

- Versión 1.0 (Fecha de lanzamiento: [20/01/2024]) El sistema cuenta con el servidor y el cliente implementado, cuenta con una interfaz gráfica sencilla sin mucho detalle.

- Versión 2.0 (Fecha de lanzamiento: [1/02/2024]) El sistema cuenta con capacidades aumentadas como la interfaz gráfica mejorada, la implementación de múltiples salas de chat y el rol de administrador para controlar los chats.



Enlace: <https://github.com/Frosmin/Chad5.git>

Chad5 v2.0

Latest

El sistema cuenta con capacidades aumentadas como la interfaz gráfica mejorada, la implementación de múltiples salas de chat y el rol de administrador para controlar los chats.

▼ Assets 2

 Source code (zip)	44 minutes ago
 Source code (tar.gz)	44 minutes ago





Chad5 V1.0



El sistema cuenta con el servidor y el cliente implementado, cuenta con una interfaz gráfica sencilla sin mucho detalle.

▼ Assets 2

 Source code (zip)	45 minutes ago
 Source code (tar.gz)	45 minutes ago

