UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad de Ciencias y Tecnologías Computacionales

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

Título: Implementación de un componente de mensajería instantánea para el XAVIA-PACS-RIS.

Autor:

Marielis Jiménez García

Tutores:

Ing. Dariel López Vera

Ing. Yenisel Valido Pérez

Co-Tutor:

Ing. Pablo Alfredo Martínez Quintana

La Habana

Noviembre 2023

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título **"*Implementación de un componente de mensajería instantánea para el XAVIA-PACS-RIS*"** concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma la presente a los 15 días del mes de noviembre del año 2023.

< Marielis Jiménez García >

_____

Firma del Autor

<>

_____

Firma del Tutor

<>

_____

Firma del Tutor

DATOS DE CONTACTO

AGRADECIMIENTOS

Agradezco:

- A mis tutores por su dedicación, entrega y brillante ayuda que han guiado este trabajo de diploma desde sus inicios hasta su terminación.
- A mis compañeros y amigos, en especial aquellos que convivieron conmigo todos estos años de universidad que sin su apoyo y ayuda no hubiese sido posible la terminación de esta carrera.
- En fin, doy gracias a todas aquellas personas que de una forma u otra contribuyeron a la realización de este trabajo de diploma.

DEDICATORIA

- A mis padres, mi hermano y mi pareja que con su apoyo, desinterés, dedicación, paciencia y amor hicieron posible la realización de mi carrera.

RESUMEN

El avance en la tecnología ha permitido una revolución en el campo de la medicina, especialmente con el desarrollo del sistema de almacenamiento y recuperación de imágenes médicas, XAVIA PACS-RIS. Sin embargo, el sistema actual tiene ciertas limitaciones en cuanto a la comunicación y la coordinación entre los profesionales médicos. La implementación de un componente de mensajería instantánea en el XAVIA PACS-RIS utilizando el estándar HIPAA (Ley de responsabilidad y portabilidad del seguro de salud) busca dar respuesta a problemas como la comunicación ineficiente, la falta de coordinación del cuidado del paciente, las amenazas a la seguridad y privacidad de la información médica y la accesibilidad limitada a la atención médica. El empleo de las herramientas y tecnologías seleccionadas para la implementación de la solución propició la correspondencia entre los resultados obtenidos y los esperados. Las aplicaciones de diversas pruebas de software demostraron la efectividad de la solución implementada y su correspondencia con estándares internacionales de calidad, seguridad y usabilidad.

Palabras claves: implementación, sistema, metodología, mensajería instantánea, XAVIA PACS-RIS.

ABSTRACT

Advances in technology have enabled a revolution in the field of medicine, especially with the development of the medical image storage and retrieval system, XAVIA PACS-RIS. However, the current system has certain limitations in terms of communication and coordination between medical professionals. The implementation of an instant messaging component in the XAVIA PACS-RIS using the HIPAA (Health Insurance Portability and Accountability Act) standard seeks to respond to problems such as inefficient communication, lack of coordination of patient care, threats to the security and privacy of medical information and limited accessibility to medical care. The use of the selected tools and technologies for the implementation of the solution led to the correspondence between the results obtained and those expected. The applications of various software tests demonstrated the effectiveness of the implemented solution and its correspondence with international standards of quality, security and usability.

Keywords: *implementation, system, methodology, instant messaging, XAVIA PACS-RIS.*

ÍNDICE

DECLARACIÓN DE AUTORÍA.....I

DATOS DE CONTACTO.....II

AGRADECIMIENTOS.....III

DEDICATORIA.....IV

RESUMEN.....V

ÍNDICE DE TABLAS.....IX

ÍNDICE DE FIGURAS.....X

OPINIÓN DE LOS TUTORES.....XI

AVAL DEL CLIENTE.....XII

INTRODUCCIÓN.....1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.....5

1.1 Conceptos asociados.....5

- **Mensajería instantánea:**.....5
- **XAVIA PACS-RIS:**.....5
- **Usuario:**.....5
- **Notificaciones:**.....5
- **Chat:**.....5
- **Mensaje:**.....5

1.2 Estándar HIPAA:.....6

1.3 Análisis de soluciones existentes.....7

1.3.1 TigerConnect.....7

1.3.2 Halo Health.....7

1.3.3 Spok.....7

1.3.4 toDus.....7
1.3.5 Telegram.....8
1.3.6 Valoración de los sistemas homólogos.....8
1.4 Selección de la metodología.....9
1.4.1 Metodología ágil.....9
1.4.2 Selección de la metodología.....9
1.5 Lenguajes.....11
1.5.1 Lenguaje de modelado.....11
1.5.2 Lenguaje de etiqueta de Hipertexto HTML.....11
1.5.3 Lenguaje de programación Python.....11
1.5.4 Marco de trabajo.....12
1.6 Herramientas utilizadas para el desarrolla del sistema.....12
1.6.1 Herramienta de modelado.....12
1.6.2 Herramientas de Base de datos MySQL.....13
1.6.3 Herramientas de seguridad.....13
1.7 Entorno de Desarrollo Integrado (IDE).....13
1.8 Conclusiones del capítulo.....14
CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO.....16
2.1 Propuesta de Solución.....16
2.1.1 Modelo conceptual.....16
2.1.2 Determinación de requisitos.....18
2.1.3 Técnica de captura de requisitos.....18
2.1.4 Requisitos no funcionales.....24
2.2 Historias de Usuarios (HU).....25
2.2.1 Plan de iteraciones.....26
2.3 Patrón arquitectónico:.....29
2.4 Prototipo de interfaz.....31
2.5 Patrones de diseño.....32
2.5.1 Patrones GRASP.....33

2.5.2 Patrones GOF.....34
2.5.3 Modelo de datos.....35
2.6 Conclusiones del capítulo.....36
CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.....37
3.1 Diagrama de despliegue.....37
3.2 Pruebas.....38
3.2.1 Estrategia de pruebas.....38
3.2.2 Pruebas unitarias.....38
3.2.3 Pruebas de Caja Negra.....41
3.2.4 Pruebas de Seguridad.....43
3.2.5 Pruebas de aceptación.....44
Las pruebas de aceptación son pruebas finales de software realizadas para determinar si un sistema cumple con los requisitos y expectativas del cliente. Se verifican funcionalidades, usabilidad, rendimiento, seguridad y compatibilidad, y pueden involucrar a los usuarios finales. Estas pruebas son cruciales para garantizar la calidad del sistema antes de su implementación y permiten hacer ajustes necesarios. (<< **Pruebas de aceptación: El Qué Y Porqué + Los Tipos que hay que Conocer>>s. f.**).....44
3.3 Resultados de las pruebas.....45
3.4 Impacto de la solución propuesta en el XAVIA PACS-RIS.....45
3.5 Conclusiones del capítulo.....46
CONCLUSIONES GENERALES.....47
RECOMENDACIONES.....48
REFERENCIAS BIBLIOGRAFICAS.....49
ANEXOS.....55

ÍNDICE DE TABLAS

Tabla 1 Comparación sistemas homólogos Fuente: Elaboración Propia
Tabla 2 Requisitos Funcionales Fuente: Elaboración Propia
Tabla 3 Historia de usuario Fuente: Elaboración Propia
Tabla 4 Plan de iteraciones Fuente: Elaboración Propia Tabla 5 Juego de Datos Fuente: Elaboración Propia
Tabla 6 Clases de equivalencia Fuente: Elaboración Propia
Tabla 7 Prueba de Aceptación Historia de Usuario Fuente: Elaboración Propia
Tabla 8 Validación de variables Fuente: Elaboración Propia

ÍNDICE DE FIGURAS

- Figura 1 Modelo conceptual Fuente: Elaboración Propia
- Figura2 Modelo Vista Plantilla
- Figura 3 Capa de Template Fuente: Elaboración Propia
- Figura4 Capa de Modelo Fuente: Elaboración Propia
- Figura5 Capa de Views Fuente: Elaboración Propia
- Figura 6 Prototipo de interfaz de Portada Fuente: Elaboración Propia
- Figura7 Prototipo de interfaz de Autenticación Fuente: Elaboración Propia
- Figura 8 Ejemplo de Patrones GRASP (Creador, Bajo Acoplamiento, Alta Cohesión y Controlador) Fuente: Elaboración Propia
- Figura 9 Ejemplo de Patrones GoF (Decorador)Fuente: Elaboración Propia
- Figura10 Modelo de Base de Datos Fuente: Elaboración propia
- Figura 11 Diagrama de Despliegue Fuente: Elaboración Propia
- Figura 12 Camino Básico Fuente: Elaboración propia
- Figura 15 Gráfico de no conformidades Fuente: Elaboración Propia

OPINIÓN DE LOS TUTORES

_____

Profesor Titular.

_____

Profesor Instructor

AVAL DEL CLIENTE

INTRODUCCIÓN

Con la llegada de las TIC, se ha generado un cambio significativo en la forma en que accedemos, pro cesamos y compartimos información. Estas tecnologías se han convertido en un componente esencia l en nuestra vida cotidiana, ya que nos permiten comunicarnos de manera instantánea y global, acced er a vastos volúmenes de conocimiento y facilitar la realización de tareas cotidianas de manera más e ficiente.

En el contexto específico de la sociedad cubana, las TIC han desempeñado un papel crucial en el est ablecimiento de las líneas de desarrollo. Han proporcionado herramientas y recursos para mejorar la educación, la salud, la administración pública, el comercio y muchos otros sectores. Estas tecnologías han contribuido a la modernización de los procesos y han fomentado la innovación en diversas áreas, impulsando así el crecimiento y el progreso del país.

En particular, la integración de las TIC en el ámbito de la salud ha brindado numerosos beneficios. La implementación de sistemas y plataformas digitales ha mejorado la eficiencia en la gestión de la infor mación médica, facilitando la comunicación entre profesionales de la salud y agilizando los procesos de diagnóstico y tratamiento. Esto ha llevado a una atención médica de mayor calidad, reduciendo los tiempos de espera y optimizando los recursos disponibles.

La mensajería instantánea se basa en el uso de programas conocidos como clientes de IM (*Instant M essaging)* que se instalan en una computadora o dispositivo móvil. Esta facilita de una manera directa la interacción personal entre servidor y usuario, llegando a audiencias específicas con mensajes insta ntáneos personalizados ya que permite integrar los datos de los clientes en su plataforma. Las apps d e mensajería instantánea son compatibles con contenido visual como voz interactiva, video e imágen es. Facilita conectar con los consumidores de formas más significativas y obtener datos valiosos para mejorar los indicadores clave de rendimiento del servicio al cliente, como la resolución al primer conta cto, el tiempo de gestión promedio y la satisfacción del cliente. Algunas compañías usan este servicio como parte de sus herramientas de productividad y comunicación; ya que permite asegurar que los e quipos tengan las herramientas adecuadas para tener éxito en sus funciones y un flujo de trabajo más ágil, evitando las repeticiones y facilitando la información que necesitan para dirigirse a los usuarios d e manera eficiente.(QUIROZ, A., 2022.)

El sistema XAVIA PACS RIS es una solución integral y avanzada para la gestión de imágenes médi-cas y la información relacionada con los pacientes en entornos de atención médica. Combina dos componentes esenciales: el PACS (Picture Archiving and Communication System) y el RIS (Radiology

Information System), para ofrecer una plataforma completa y eficiente para el manejo de datos clínicos y radiológicos en instituciones médicas.(IZAGUIRRE, I.L.V.,2020)

Se trata de un sistema crítico que debe estar disponible en todo momento y que debe garantizar la seguridad y privacidad de la información de los pacientes; a pesar de las ventajas del sistema XAVIA-PACS-RIS, se ha identificado una brecha en la comunicación y la colaboración entre los profesionales de la salud.

A partir de la situación anterior, se identifica como **problema de investigación**: ¿Cómo mejorar la comunicación y coordinación entre los diferentes actores del proceso de atención médica y los usuarios del sistema XAVIA-PACS-RIS?

Objeto de estudio: comunicaciones basadas en la mensajería instantánea en el ámbito de la salud.

El campo de acción: componentes para la mensajería instantánea y las notificaciones.

Se plantea como **objetivo general**: Desarrollar un componente estándar que unifique las tecnologías de mensajería y notificaciones garantizando una mejora en la colaboración, comunicación, la toma de decisiones y la coordinación entre los profesionales y usuarios del sistema XAVIA-PACS-RIS.

Objetivos específicos:

1. Caracterizar los procesos de negocio y soluciones tecnológicas empleadas en mensajería y notificaciones.

2.Diseñar los artefactos necesarios para la implementación de las funcionalidades de mensajería y notificaciones.

3. Implementar el componente diseñado siguiendo la estructura de diseño previamente definida, asegurando una integración efectiva y coherente con el sistema existente.

4. Realizar pruebas exhaustivas de calidad a lo largo de todo el ciclo de desarrollo, con el objetivo de validar el correcto funcionamiento del componente y garantizar su rendimiento óptimo en diferentes escenarios y situaciones.

Tareas de la investigación:

Para dar cumplimiento al objetivo planteado se proponen como tareas de la investigación:

1. Realización de un exhaustivo análisis de los sistemas web existentes que se utilizan para las comunicaciones instantáneas, con el objetivo de comprender su funcionamiento, características y limitaciones.

2. Investigación de los diferentes frameworks de desarrollo, plugins y/o componentes disponibles que sean relevantes para el desarrollo de elementos interactivos en el contexto de la comunicación instantánea. Se explorarán las opciones más adecuadas y se evaluarán sus ventajas y desventajas.Además de la realización de un estudio detallado de los protocolos utilizados a nivel mundial para la mensajería instantánea. Se investigarán los estándares y tecnologías más comunes, analizando sus características, seguridad y escalabilidad.

3. Realización de un análisis e investigación exhaustiva de las metodologías de desarrollo de software aplicables a la solución propuesta. Se evaluarán las diferentes metodologías, como Agile, Scrum o DevOps, y se determinará cuál es la más adecuada para el desarrollo de la solución de comunicación instantánea. Se considerarán factores como la eficiencia, la flexibilidad y la colaboración en el equipo de desarrollo.

Métodos teóricos:

- Histórico-Lógico: Permitió realizar un estudio de los principales conceptos asociados a los componentes de mensajería instantánea.
- Hipotético-deductivo: se utilizó para guiar la investigación desde el planteamiento del problema hasta la verificación de la solución a partir de las validaciones, orientando la secuencia lógica de las tareas que se realizaron.
- Análisis y Síntesis: Se utiliza para identificar y analizar las diversas funcionalidades de las plataformas a nivel internacional que pueden ser aplicadas en la solución.
- Modelación: Permitió la creación del modelo conceptual para entender el contexto en el que se enmarca la investigación.

Métodos empíricos:

- ❖ Entrevistas: Las entrevistas pueden ser utilizadas para recopilar información más detallada sobre las necesidades y expectativas de los usuarios, así como para obtener retroalimentación sobre la solución de mensajería instantánea después de su implementación.

El presente documento está compuesto por tres capítulos:

Capítulo 1. Fundamentación teórica: en este capítulo se da a conocer, fundamentalmente, la teoría del trabajo y el estudio de componentes de mensajería instantánea; además, se profundiza en las metodologías, lenguajes, modelo de desarrollo de software y herramientas adecuadas que se han de utilizar para darle solución al trabajo.

Capítulo 2 En este capítulo se aborda la propuesta de solución a la problemática planteada. Se describen los requisitos funcionales y no funcionales, para dar cumplimiento a los objetivos planteados. Se realizan las historias de usuario, plan de iteraciones y demás artefactos exigidos por la Metodología AUP UCI V4 y sus fases.

Capítulo 3. Validación de la solución propuesta: En este capítulo se aborda las diferentes pruebas realizadas al componente y que son exigidas por la metodología AUP UCI V4 para el buen funcionamiento y calidad de la propuesta desarrollada. Se detalla con exactitud los diferentes resultados alojados por cada prueba y la respuesta a dichas no conformidades.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se da a conocer, fundamentalmente, la teoría del trabajo y el estudio de componentes de mensajería instantánea; además, se profundiza en las metodologías, lenguajes, modelo de desarrollo de software y herramientas adecuadas que se han de utilizar para darle solución al trabajo.

1.1 Conceptos asociados

Los conceptos asociados son de gran significación para conocer y comprender mejor el dominio, por lo cual se tuvo en cuenta los conceptos de: XAVIA PACS-RIS, mensajería instantánea, usuario, notificaciones, chat y mensaje.

- **Mensajería instantánea:**

Una forma de comunicación en tiempo real que permite el intercambio de mensajes de texto, voz y video entre los usuarios.

- **XAVIA PACS-RIS:**

Un sistema de gestión de imágenes médicas que permite la visualización, almacenamiento y distribución de imágenes médicas.

- **Usuario:**

Persona que interactúa con el chat y las salas de chat.

- **Notificaciones:**

Las notificaciones son generadas por el sistema, ejecutadas cuando se conecta o se desconecta un

- **Chat:**

Aplicación que se utiliza para establecer una comunicación entre dos o más usuarios.

- **Mensaje:**

Argumento enviado entre los usuarios para establecer una conversación produciéndose la activación directa del chat.

1.2 Estándar HIPAA:

HIPAA, que significa Health Insurance Portability and Accountability Act (Ley de Portabilidad y Responsabilidad del Seguro Médico), es una legislación de Estados Unidos promulgada en 1996 con el

propósito de abordar diversas preocupaciones relacionadas con la privacidad y seguridad de la información médica y de salud de los pacientes. (SAVAGE, M. y SAVAGE, L.C., 2020.)

El objetivo principal de HIPAA es proteger la privacidad y confidencialidad de la información de salud, al tiempo que establece estándares para garantizar su seguridad. La ley busca asegurar que la información médica de los pacientes sea manejada de manera adecuada y que solo sea accesible para las personas autorizadas.

HIPAA se aplica a diversas entidades en el sistema de atención médica, incluyendo proveedores de atención médica, planes de salud, profesionales de la salud, aseguradoras y proveedores de servicios de salud. También se aplica a cualquier parte que maneje, almacene o transmita información de salud electrónica.

La ley establece normas y reglamentos específicos para proteger la información de salud, como la Norma de Privacidad y la Norma de Seguridad.

En cuanto a su aplicación en los componentes de mensajería, es importante tener en cuenta que la comunicación electrónica que incluye información de salud está sujeta a los requisitos de privacidad y seguridad de HIPAA. Esto implica que los mensajes de texto, correos electrónicos u otros medios de comunicación utilizados para transmitir información médica deben cumplir con los siguientes principios:

1. Acceso controlado: Los sistemas de mensajería deben asegurarse de que solo las personas autorizadas tengan acceso a la información de salud y establecer mecanismos de autenticación para verificar la identidad de los usuarios.

2. Cifrado: La información de salud transmitida a través de los mensajes debe estar protegida mediante cifrado para evitar la interceptación no autorizada.

3. Conservación de registros: Se deben mantener registros de las comunicaciones para cumplir con los requisitos de retención y auditoría.

4. Consentimiento informado: Los pacientes deben otorgar su consentimiento informado para la comunicación electrónica de información médica, y se deben establecer mecanismos para que los pacientes retiren su consentimiento en cualquier momento.

1.3 Análisis de soluciones existentes

Existen diferentes componentes de mensajería instantánea que están estrechamente relacionados en el sector de la atención médica, estos presentan ciertas características que pueden ser de gran ayuda a la hora de mejorar el sistema. Es por ello que se decide hacer un análisis de varios sistemas desarrollados a nivel nacional e internacional con el objetivo de determinar las principales características como: cubano o extranjero, tecnología actualizada, gratuita o de pago, adaptación a las necesidades del cliente, acceso a la información, a continuación, se muestran algunos de ellos:

retrasar el tratamiento y la atención al paciente.

1.3.1 TigerConnect

Es una plataforma de mensajería instantánea que proporciona comunicación segura y en tiempo real entre los profesionales médicos, pacientes y equipos de atención médica. La plataforma cumple con los requisitos de seguridad y privacidad de HIPAA y ofrece características avanzadas como la integración de sistemas de registro médico electrónico (EMR). (TigerConnect,2023)

1.3.2 Halo Health

Es una plataforma de comunicación en el sector de la atención medica que permite a los equipos de atención colaborar y comunicarse de manera eficiente. Ofrece mensajería segura, alertas de emergencia, notificaciones escalables y funciones de gestión de tareas (Halo Health,2023)

1.3.3 Spok

Es una plataforma de comunicación que permite la mensajería instantánea, llamadas de voz y video en tiempo real, y la integración con sistemas EMR. La plataforma cumple con los requisitos de seguridad y privacidad de HIPAA y facilita la comunicación entre los profesionales médicos y los pacientes. (Spok,2023)

1.3.4 toDus

Es una aplicación de mensajería enfocada en la velocidad y seguridad de la información, es simple y gratis. Con toDus, puedes enviar mensajes, notas de voz, fotos, archivos de hasta 200Mb; también cr ear grupos de hasta 257 miembros e incluso agendar eventos. Puedes escribir a tus contactos del tel éfono que ya tengan toDus o invitarlos a unirse.(toDus,2023)

1.3.5 Telegram

Es una plataforma de mensajería instantánea que ofrece chat individual y grupal, llamadas de voz y video, envío de archivos. También proporciona canales públicos y opciones de autodestrucción de mensajes.(Telegram Messenger,2023)

1.3.6 Valoración de los sistemas homólogos

En el análisis de los sistemas homólogos a la propuesta de solución se determinó que a pesar de que eran software libre y permitían acceder a la información no cumplían con los requerimientos para darle solución al problema planteado.

Tabla 1. Comparación sistemas homólogos. (Fuente: elaboración propia)

Componentes de mensajería instantánea	Enfoque	Gratuito o de pago	Tecnología actualizada	Cumple con el estándar HIPAA
TigerConnect	Sector de la atención medica	Gratuito	Sí	Sí
Halo Health	Colaboración y gestión de tareas	Gratuito	Sí	Sí
Spok	Gestión de alarmas	De pago	Sí	Sí
toDus	Privacidad y seguridad	Gratuito	Sí	Si
Telegram	Canales públicos y disponibilidad	Gratuito	Sí	Sí

Se necesitaba un sistema que pudiese complementarse al XAVIA PACS-RIS, como una exigencia del cliente, que tuviera tecnología actualizada, que se pudiera desplegar de forma gratuita, que fuera fácil de mantener y que se adapte a las necesidades reales del cliente; aunque en su estudio aportaron algunas características que permitieron ayudar en selección de la metodología ya que fueron desarro-

lladas utilizando metodologías ágiles, en la selección de la arquitectura debido a que la mayoría se desarrollaron bajo la arquitectura modelo-vista-controlador y en el diseño de las interfaces de usuario dado que estas usan interfaces amigables.

1.4 Selección de la metodología

La metodología de desarrollo de software es el conjunto de técnicas y métodos que se utilizan para diseñar una solución de software informático. Es importante señalar que existen varias, de manera que es una decisión de cada equipo. (Carlemany, 2020)

Las metodologías también sirven para controlar el desarrollo del trabajo. Esto sirve para minimizar los márgenes de errores y anticiparse a esa situación. Otra ventaja de utilizar una metodología es que te hace ahorrar tiempo y gestionar mejor los recursos disponibles. Piattini define a la metodología de desarrollo de software como "*un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software*".(Piattini Velthuis, 2018)

La metodología indica cómo hay que obtener los diferentes productos parciales y finales. Las metodologías se clasifican en 2 grupos: ágiles y tradicionales. Para este trabajo se seleccionó la ágil debido a las características y ventajas que presenta.

1.4.1 Metodología ágil

La metodología ágil es un enfoque de gestión de proyectos que se centra en la entrega continua de soluciones iterativas y funcionales a través de un proceso colaborativo y flexible. En la metodología ágil, los equipos trabajan en colaboración para lograr objetivos específicos y se adaptan a los cambios en los requisitos a medida que surgen; estas se basan en cuatro valores principales y 12 principios, establecidos en el Manifiesto Ágil. (Metodologías Ágiles: Definición, Manifiesto, 2023)

1.4.2 Selección de la metodología

El Proceso Unificado Ágil o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado Racional (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. La UCI le ha realizado modificaciones con el fin de adaptarlo al ciclo de vida definido para la actividad productiva de dicha institución.

1.4.2.1 Ciclo de vida de AUP UCI V4:

El ciclo de vida de AUP UCI V4consta de cuatro fases:

- Inicio: En esta fase, se identifican los objetivos del proyecto, se definen los requisitos y se establecen los planes y la estructura del equipo. Los roles principales en esta fase son:

- Patrocinador del proyecto: es la persona que proporciona la financiación y el apoyo para el proyecto.

- Líder del proyecto: es la persona encargada de coordinar las actividades del equipo y de garantizar el cumplimiento de los objetivos del proyecto.

- Analista de negocio: es la persona encargada de identificar y definir los requisitos del proyecto.

- Arquitecto de soluciones: es la persona encargada de diseñar la arquitectura de la solución y de identificar las tecnologías necesarias para implementarla.

- Elaboración: En esta fase, se definen y se refinan los requisitos y se desarrolla un plan más detallado para la implementación de la solución. Los roles principales en esta fase son:

- Gerente de proyecto: es la persona encargada de coordinar y supervisar el equipo de desarrollo.

- Analista de sistemas: es la persona encargada de analizar los requisitos y diseñar la solución.

- Diseñador de interfaz de usuario: es la persona encargada de diseñar la interfaz de usuario de la solución.

- Diseñador de base de datos: es la persona encargada de diseñar la estructura de la base de datos.

- Construcción: Se implementa la solución y se realizan las pruebas necesarias para garantizar su calidad. Los roles principales en esta fase son:

- Programador: es la persona encargada de implementar la solución.

- Analista de pruebas: es la persona encargada de realizar las pruebas necesarias para garantizar la calidad de la solución.

- Gerente de configuración: es la persona encargada de gestionar la configuración del software y garantizar que se mantenga un registro de los cambios realizados.

- Transición: Se realiza la implementación completa de la solución y se lleva a cabo la transferencia del conocimiento y la capacitación al equipo de soporte. Los roles principales en esta fase son:

- Especialista en implementación: es la persona encargada de implementar la solución en el entorno del cliente.

- Especialista en capacitación: es la persona encargada de capacitar al equipo de soporte en la utilización de la solución.

- Especialista en soporte: es la persona encargada de proporcionar soporte técnico y solucionar problemas técnicos.

1.5 Lenguajes

1.5.1 Lenguaje de modelado

El lenguaje de modelado a utilizar será Lenguaje Unificado de Modelado (UML) en su versión 2.5. UML es un lenguaje de modelado visual ampliamente utilizado en la industria del software para visualizar, especificar, construir y documentar sistemas de software. En particular, el uso de diagramas de secuencia UML podría ser útil para modelar el flujo de mensajes entre los diferentes componentes del sistema de mensajería instantánea y los sistemas de información de salud (PACS, RIS, EHR), asegurando que se cumplan los requisitos de privacidad y seguridad establecidos por el estándar HIPAA. (UML, 2018)

1.5.2 Lenguaje de etiqueta de Hipertexto HTML

HTML5 es un lenguaje de marcado utilizado para crear páginas web y aplicaciones web. Se utiliza para describir la estructura y el contenido de una página web, incluyendo el texto, imágenes, enlaces y otros elementos interactivos. (Lenguaje HTML, s. f.)

En el contexto de la implementación de un componente de mensajería instantánea para el XAVIA-PACS-RIS, HTML podría utilizarse para crear la interfaz de usuario del sistema de mensajería instantánea, incluyendo elementos como cajas de entrada de texto, botones de envío, ventanas de chat y otros elementos de la interfaz de usuario.

1.5.3 Lenguaje de programación Python

Python en su versión 3.10, es un lenguaje interpretado, interactivo y orientado a objetos. Incorpora módulos, excepciones, tipado dinámico, tipos de datos de muy alto nivel y clases. Python combina un poder destacado con una sintaxis muy clara. Tiene interfaces a muchas llamadas de sistema y bibliotecas, así como a varios sistemas de ventana, y es extensible en C o C++. También es usable como un lenguaje de extensión para aplicaciones que necesitan una interfaz programable. Por último, Python es portable: corre en muchas variantes de Unix, en Mac y en Windows 2000 y posteriores.(Python, 2022)

1.5.4 Marco de trabajo

Django en su versión 4.2, es un framework web de alto nivel para Python que se utiliza para construir aplicaciones web complejas y escalables. Es gratuito, de código abierto y sencillo de instalar; solo se necesita poseer una documentación muy completa y extensa, además emplea un sistema de modelos que indica a los Gestores lo que deben hacer. (Django s. f.)

Posee una arquitectura de modelo-vista-controlador (MVC) y ofrece características de seguridad avanzadas, lo que lo hace adecuado para la implementación de sistemas de mensajería instantánea seguros.

1.6 Herramientas utilizadas para el desarrolla del sistema

1.6.1 Herramienta de modelado

Las herramientas de modelado de sistemas informáticos, son herramientas que se emplean para la creación de modelos de sistemas que ya existen o que se desarrollarán, permitiendo crear un modelo del sistema a bajo costo y riesgo mínimo.

Las buenas herramientas de modelado cumplen con las siguientes características:

- Permiten una visión descendente del sistema.
- Poseen componentes gráficos con algo de apoyo textual.
- El modelo resultante debe ser transparente (fácil de comprender).
- Poseen mínima redundancia (el aumento de redundancia, disminuye la transparencia del modelo y aumenta las tareas de mantenimiento). (Visual, 2018)

➢ Visual Paradigm

La herramienta de modelado seleccionada fue Visual Paradigm para UML en su versión 8.0, debido a que es una herramienta para el desarrollo de aplicaciones utilizando modelado UML. Esta herramienta es ideal para ingenieros de software, analistas de sistemas y arquitectos de sistemas que están interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. (UML, 2018)

Visual Paradigm ha sido concebido para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas y constituye una herramienta disponible en varias ediciones. (Visual, 2018)

Se caracteriza por:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Licencia: gratuita y comercial.
- Diagramas de procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Generación de código - modelo a código, diagrama a código.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.

La utilización de esta herramienta permite ampliar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software, así como, aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software.(Saeedi y Visvizi, 2021)

1.6.2 Herramientas de Base de datos MySQL

MySQL en su versión 5.7 es una base de datos relacional de código abierto que es conocida por su escalabilidad y rendimiento. MySQL cuenta con características de seguridad avanzadas, como la encriptación de datos y el control de acceso basado en roles, lo que lo hace adecuado para cumplir con los requisitos de HIPAA.(Qué es MySQL,2019)

1.6.3 Herramientas de seguridad

Let's Encrypt 2.6: Es una autoridad de certificación de código abierto que proporciona certificados SSL/TLS gratuitos y automatizados para la encriptación de datos.(Let's Encrypt,2023)

1.7 Entorno de Desarrollo Integrado (IDE)

Un IDE es el entorno digital empleado para desarrollar cualquier tipo de software cuyo objetivo es agilizar todo el proceso de diseño de software, ofreciendo un servicio integral al programador. IDE es el acrónimo del término inglés (*Integrated Development Environmen)*, lo que es lo mismo, Entorno de Desarrollo Integrado. (La Rioja, 2022)

Es el escenario digital utilizado en programación para desarrollar aplicaciones, juegos, entre otros; es imprescindible tanto en el ámbito del Desarrollo de Aplicaciones Web (DAW) como en el Desarrollo de Aplicaciones Multiplataforma (DAM). Hace que la tarea del programador sea más sencilla gracias a las herramientas que tiene incorporadas como compiladores, depuradores o bibliotecas, y esto se traduce en un aumento de la productividad.(La Rioja, 2022)

Cualquier IDE debe tener una serie de características básicas que garanticen que la experiencia del usuario será satisfactoria. Todo IDE debe contar con:

- Editor de código: se trata de un editor de texto creado exclusivamente para trabajar con el código fuente de programas informáticos.
- Compilador: es un programa encargado de traducir las instrucciones en código fuente, escritas en lenguaje de programación, a código objeto, el único lenguaje que el ordenador entiende.
- Depurador o debugge: es un programa que permite probar y buscar errores en otros programas.
- Linker: es la herramienta con la que combinar diferentes archivos de código fuente para convertirlos en un único fichero ejecutable.
- Refactorización de código: proceso en el que se recurre a funciones como el reformateo o la encapsulación para mejorar el código fuente.(La Rioja, 2022)

Para la implementación de la plataforma se tuvo en cuenta el Visual Estudio Code en su versión 1.71. Visual Studio es un editor de código gratuito y de código abierto que admite una amplia variedad de lenguajes de programación, incluyendo Python y Django. Visual Studio Code también admite la integración con herramientas de modelado como Visual Paradigm y herramientas de seguridad como Let's Encrypt.

1.8 Conclusiones del capítulo

Los resultados expuestos en este capítulo permitieron:

1. El análisis de los conceptos fundamentales asociados a la implementación de un componente de mensajería instantánea para el XAVIA PACS-RIS, permitieron la comprensión de contenidos relacionados con este tema.
2. El estudio comparativo de soluciones similares para obtener una mayor comprensión sobre el componente que se quiere desarrollar.
3. La identificación de las características propuestas para la solución y el análisis de las tecnologías que distinguen a soluciones similares para la implementación de un componente de mensajería instantánea para el XAVIA PACS-RIS, contribuyó a la selección de las herramientas y tecnologías más adecuadas para desarrollar la propuesta de solución, destacando como metodología de desarrollo AUP UCI V4, como marco de trabajo el framework Django con la base de datos MongoDB, Visual estudio Code como IDE de desarrollo y como herramienta de modelado *Visual Paradigm* usando como lenguaje UM

CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

En este capítulo se aborda la propuesta de solución a la problemática planteada. Se describen los requisitos funcionales y no funcionales, para dar cumplimiento a los objetivos planteados. Se realizan las historias de usuario, plan de iteraciones y demás artefactos exigidos por la Metodología AUP UCI V4 y sus fases.

2.1 Propuesta de Solución

Utilizando la información recopilada en el capítulo anterior, se propone el desarrollo de un componente de mensajería instantánea que brinde a los usuarios la posibilidad de tener acceso al mismo de forma rápida y sencilla, en cualquier momento, y a través de cualquier dispositivo. El desarrollo del componente permitirá facilitar la comunicación entre profesionales de la salud y los usuarios, que utilizan el sistema XAVIA PACS-RIS. Podrá ser utilizado como sistema independiente o como herramienta dentro de cualquier sistema web, que debe interactuar necesariamente con el flujo de trabajo de dicho sistema al cual se integre. Utilizado como sistema independiente se realizará a través de un servidor web, donde el usuario interactuará directamente con el sistema garantizando todas las funcionalidades que la aplicación puede llevar a cabo. Esta interacción se realizará bajo el flujo de trabajo petición-respuesta, donde la petición puede ser visualizada por el usuario, mientras la respuesta es garantizada por el sistema.

2.1.1 Modelo conceptual

Un modelo conceptual es una descripción general de cómo un sistema se organiza y opera. Su tarea principal es especificar y describir los conceptos base. Este es un paso muy importante en cualquier metodología de análisis y diseño orientada a objetos, pues de aquí parte el diagrama de clases que modela la aplicación. Los errores que se cometan en la modelación de este diagrama pueden dar al traste con una aplicación incompleta o que no cumpla las expectativas de los clientes. («Modelos con-

ceptuales para definir las plataformas digitales | IDA Chile», 2016)

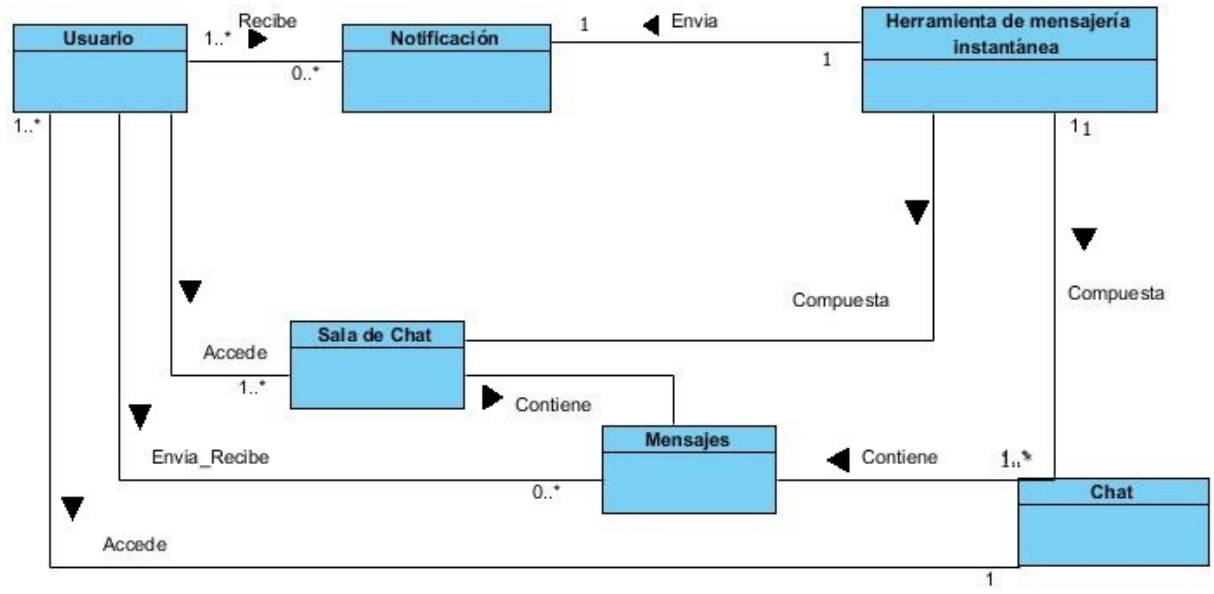

Figura 1 Modelo conceptual (Fuente: Elaboración Propia)

Usuario: Un usuario se refiere a una persona que utiliza una plataforma, aplicación o servicio para participar en interacciones y comunicarse con otros usuarios.

Notificación: Una notificación es un mensaje o aviso que se muestra a un usuario para informarle sobre algún evento o actividad relevante en una herramienta de mensajería instantánea o chat. Estas notificaciones pueden incluir la recepción de nuevos mensajes, menciones directas, actualizaciones de estado de otros usuarios, recordatorios de eventos o cualquier otra información importante.

Herramienta de mensajería instantánea: Aplicación o plataforma que permite a los usuarios enviar y recibir mensajes de texto en tiempo real. Estas herramientas facilitan la comunicación directa entre personas, ya sea en uno a uno o en grupos, a través de una interfaz de chat.

Sala de chat: Es un entorno virtual en una plataforma de chat o mensajería instantánea donde múltiples usuarios pueden participar en una conversación simultánea.

Mensajes: Son textos o contenidos que los usuarios envían o reciben durante una conversación. Los mensajes se envían de un usuario a otro o se comparten en una sala de chat, y permiten la interacción y la transmisión de información entre los participantes

Chat: Es una herramienta de comunicación en tiempo real que facilita la interacción y colaboración entre miembros de un equipo de trabajo, departamentos o incluso organizaciones enteras. El chat es una forma eficiente de comunicación que permite compartir información, discutir proyectos, tomar decisiones y coordinar actividades de manera ágil y efectiva.

2.1.2 Determinación de requisitos

El análisis de requisitos permite al desarrollador o desarrolladores especificar la función y el rendimiento del software. Los requisitos fueron identificados gracias a entrevistas realizadas al cliente, donde fueron determinados una serie de necesidades que marcan puntos claves para el desarrollo de la aplicación. Los requisitos funcionales (RF) describen la interacción entre el sistema y su ambiente, independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema. (ESTÉVEZ, J.R., 2020.)

2.1.3 Técnica de captura de requisitos

En el ámbito del desarrollo de software, los analistas emplean una variedad de enfoques para obtener los requisitos del cliente. A lo largo de la historia, se han utilizado diversas técnicas, como entrevistas y talleres de grupo, para recopilar los requisitos del sistema. Los analistas combinan estos métodos de manera flexible para identificar con precisión los requisitos específicos de las partes interesadas, con el objetivo de crear un sistema que satisfaga las necesidades del cliente. A continuación, se presentan algunas de estas técnicas:

- Exploración en profundidad: Mediante entrevistas detalladas, se busca obtener una comprensión exhaustiva de las necesidades y expectativas de los diferentes sectores clave de la organización. Se selecciona cuidadosamente una muestra representativa de personas relacionadas con el sistema, centrándose en aquellas áreas que se verán más afectadas o que utilizarán el sistema con mayor frecuencia.
- Documentación estructurada: En lugar de entrevistas, se pueden utilizar formularios o plantillas que permitan a las partes interesadas proporcionar información detallada sobre los requisitos. Estos documentos pueden ser extensos, especialmente en sistemas complejos, y contener múltiples secciones que cubran diversos aspectos del sistema.

- Definición de metas mensurables: Los requisitos presentados por los usuarios se consideran objetivos generales a largo plazo. Sin embargo, es fundamental analizarlos desde la perspectiva del sistema para identificar los objetivos críticos que darán forma a los comportamientos y funcionalidades específicas del sistema. Además, se establecen indicadores de progreso y métricas para evaluar en cualquier momento el avance del proyecto y garantizar que se estén cumpliendo los objetivos establecidos.

2.1.4 Requisitos funcionales

Los requisitos funcionales describen la interacción entre el sistema y su ambiente independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema. Los requisitos definidos por el cliente se listan a continuación:

Los requisitos funcionales del sistema son los siguientes:

Tabla 2 Requisitos Funcionales (Fuente: Elaboración Propia)

No.	Requisito Funcional	Prioridad	Descripción
RF_1	Crear Usuario	Alta	Permite a los usuarios registrarse en el sistema de mensajería, proporcionando la información necesaria para crear una cuenta única y personalizada.
RF_2	Autenticar Usuario	Alta	Garantiza que solo los usuarios autorizados puedan acceder al sistema de mensajería, verificando su identidad a través de credenciales como nombre de usuario y contraseña.
RF_3	Modificar Usuario	Alta	Permite a los usuarios realizar cambios en su perfil, c

			omo actualizar la informaci ón personal, cambiar la fot o de perfil o editar la confi guración de privacidad.
RF_4	Eliminar Usuario	Alta	Da la opción a los usuario s de eliminar de forma per manente su cuenta y todo s los datos asociados a ell a del sistema de mensajer ía.
RF_5	Listar/Mostrar Usuario	Alta	Proporciona una lista de u suarios registrados en el si stema o permite mostrar in formación específica de u n usuario en particular.
RF_6	Buscar usuario	Alta	Facilita la búsqueda de us uarios dentro del sistema de mensajería, ya sea por nombre, alias u otros criter ios de búsqueda.
RF_7	Cambiar contraseña	Alta	Permite a los usuarios mo dificar su contraseña actu al por una nueva, brindand o mayor seguridad y contr ol sobre su cuenta.

RF_8	Listar/Mostrar notificaciones	Alta	Muestra una lista de notificaciones recibidas por el usuario, como nuevos mensajes, solicitudes de amistad o actualizaciones relevantes.
RF_9	Configurar notificaciones	Alta	Permite al usuario personalizar las preferencias de notificación, como elegir el tipo de eventos por los que desea recibir notificaciones y la forma en que se mostrarán.
RF_10	Crear sala de chat	Media	Permite a los usuarios crear nuevas salas de chat, proporcionando un espacio para interactuar con otros usuarios sobre temas específicos.
RF_11	Editar sala de chat	Media	Permite a los usuarios realizar cambios en las configuraciones o características de una sala de chat existente, como su nombre, descripción o configuración de privacidad.

RF_12	Eliminar sala de chat	Media	Permite a los usuarios eliminar una sala de chat que ya no sea necesaria o relevante.
RF_13	Mostrar/Listar chat	Media	Muestra una lista de conversaciones o chats disponibles para el usuario o muestra el contenido completo de un chat específico.
RF_14	Iniciar conversación privada con un usuario	Media	Permite al usuario iniciar una conversación privada uno a uno con un usuario específico.
RF_15	Enviar mensaje a un usuario	Alta	Permite al usuario enviar mensajes a usuario dentro del sistema de mensajería.
RF_16	Recibir mensaje de un amigo	Alta	Permite al usuario recibir mensajes enviados por usuarios dentro del sistema de mensajería.
RF_17	Cerrar conversación	Media	Permite al usuario finalizar

			una conversación o chat específico, cerrando la sesión y dejando de recibir notificaciones relacionadas con esa conversación.
RF_18	Mostrar estado del usuario	Baja	Muestra el estado actual del usuario, como disponible, ocupado, ausente o desconectado, para que otros usuarios puedan saber si está disponible para interactuar.
RF_19	Mostrar mensajes de historial	Baja	Permite al usuario ver el historial completo de mensajes en una conversación o chat, incluyendo los mensajes enviados y recibidos anteriormente.
RF_20	Salir del sistema	Baja	Permite al usuario cerrar sesión y salir completamente del sistema de mensajería.

2.1.4 Requisitos no funcionales

Los requisitos no funcionales describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema. Los requerimientos no

funcionales incluyen restricciones como el tiempo de respuesta (desempeño), la precisión, recursos consumidos, seguridad, entre otros.

Usabilidad:

- RNF_1 El sistema a implementar contará con una interfaz que podrá ser accedida por cualquier tipo de usuario que cree una cuenta en el sistema.
- RNF_2 El sistema debe mantener informado al usuario sobre los eventos que se generen a través de la interacción que presente con el mismo.
- RNF_3 La herramienta de chat estará disponible desde todas las páginas de la aplicación a la cual se integre.

Soporte:

- RNF_4 El componente de mensajería debe integrarse a cualquier aplicación desarrollada para la web o simplemente puede ser accedida cuando se ejecute como una aplicación independiente
- RNF_5 Debe ser ejecutado en cualquier tipo de navegador.

Seguridad:

- RNF_6 El sistema debe cumplir con el estándar HIPAA para los componentes de mensajería instantánea.
- RNF_7 Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos.

Hardware:

- RNF_8 La PC servidor debe cumplir con las siguientes características:

  PC Dual Core a 3.0 GHZ de micro y superior.

  4 GB de memoria RAM como mínimo.

  Disco duro de 40 GB como mínimo
- RNF_9 La PC cliente debe cumplir con las siguientes características:

  PC Pentium IV a 3.0 GHZ de micro como mínimo.

  1 GB de memoria RAM como mínimo.

Disco duro de 10 GB como mínimo.

- RNF_10 Los hardware clientes y servidor deben poseer una tarjeta de interconexión de red que permita acceder y responder a las peticiones respectivamente.

2.2 Historias de Usuarios (HU)

Las Historia de Usuarios es la técnica que propone el escenario 4 de la metodología AUP-UCI para encapsular los requisitos funcionales del sistema. Con el empleo de esta técnica se describió qué se espera como salida de la implementación, y cómo se ve beneficiado el usuario final. Se expresa en lenguaje natural y sencillo, considerando que los usuarios finales tienden a desconocer el lenguaje técnico. A continuación, se muestra la historia de usuario correspondientes al requisito funcional

Enviar mensaje:

Tabla 3 Historia de usuario (Fuente: Elaboración Propia)

Historia de Usuario : Enviar mensaje	
Número: 15	**Nombre del requisito: Enviar mensaje**
Programador:Marielis Jiménez García	**Iteración Asignada: 1**
Prioridad:Alta	**Tiempo Estimado:8h**
Riesgo en Desarrollo: Alto	**Tiempo Real:10h**
Descripción: Permitir al usuario el envío de mensajes a otros usuarios del sistema permitiendo comunicar-se entre sí.	
Observaciones: • Ingresar los datos de Usuario y Contraseña. • Implementar una interfaz intuitiva para escribir y enviar mensajes. • Asegurarse de que los mensajes se entreguen correctamente y se muestren en la conv ersación correspondiente.	

En el proceso de desarrollo del sistema de gestión se definió para mantener la organización, el nivel de prioridad para el negocio de cada uno de los requisitos obtenidos en la entrevista con el cliente. Los niveles se dividen en:

• Alta: Funcionalidades que son sumamente primordiales en la aplicación, que por su función no deberían faltar en el editor, es decir, la aplicación depende de estas para su correcto funcionamiento.

• Media: Funcionalidades que no son muy primordiales, pero otras dependen de estas para poder ejecutarse.

• Baja: Funcionalidades que su existencia en la aplicación no es significativa, es decir, la aplicación puede funcionar correctamente sin estas.

2.2.1 Plan de iteraciones

Un plan de iteraciones está constituido por un conjunto secuencial de actividades y tareas, cada una tiene recursos asignados y pueden depender a su vez de otras tareas. El plan de iteración se realiza una vez por cada iteración.

El contenido de la iteración puede variar, dependiendo de la posición dentro del ciclo de vida y de la naturaleza del proyecto. El plan de iteración incluye porciones relevantes de todas las disciplinas para cada iteración en particular, la prioridad de implementación se evalúa en base al cliente y el equipo de desarrollo, y el impacto del riesgo en base al juicio de experto. Los planes de iteración por lo general muestran un planeamiento detallado de quién va a realizar una tarea/actividad de acuerdo en conformidad a qué criterios de evaluación.

Tabla 4 Plan de Iteraciones (Fuente: Elaboración Propia)

No. Iteración	Historia de Usuario	Prioridad	Esfuerzo Estimado	
	Crear Usuario	Alta	8	25
	Eliminar Usuario	Alta	3	
	Modificar Usuario	Alta	2	

1	Buscar Usuario	Alta	2	
	Listar/Mostrar Usuario	Alta	2	
	Autenticar Usuario	Alta	8	

2	Cambiar contraseña	Alta	6	14
	Listar/Mostrar notificaciones	Alta	4	
	Configurar notificaciones	Alta	4	
	Crear sala de chat	Media	5	26
	Editar sala de chat	Media	6	
	Eliminar sala de chat	Media	3	
3	Salir del sistema	Baja	3	9
4	Mostrar/Listar mensajes de historial	Media	2	
	Iniciar conversación privada con un usuario	Baja	3	
	Mostrar estado del usuario	Media	3	
	Enviar mensaje a un usuario	Baja	3	
	Recibir mensaje de un usuario	Media	6	
	Cerrar conversación	Media	3	

2.3 Patrón arquitectónico:

Los patrones arquitectónicos son soluciones reutilizables y probadas para problemas comunes en el diseño de software. Estos patrones proporcionan una estructura y un enfoque para organizar los componentes y las interacciones dentro de un sistema, permitiendo a los diseñadores aprovechar soluciones exitosas que han sido desarrolladas y refinadas a lo largo del tiempo. (PALMERO, E., 2022.)

La arquitectura Modelo-Vista-Controlador (MVC) es un patrón de diseño ampliamente utilizado en el desarrollo de aplicaciones de software. Proporciona una estructura modular y organizada que separa la lógica de negocio, la presentación de la interfaz de usuario y la gestión de datos en tres componentes principales: el modelo, la vista y el controlador. (AGUILAR, J.M., 2019)

Django es conocido por seguir el patrón de diseño Modelo-Vista-Controlador (MVC) de manera rigurosa, lo que le ha llevado a ser considerado un framework MVC en sí mismo. Este enfoque arquitectónico proporciona una estructura sólida y bien definida para el desarrollo de aplicaciones web.

En Django, las responsabilidades del patrón MVC se separan y gestionan de la siguiente manera:

- Modelo (M): Django se encarga de la capa de acceso a la base de datos, que es fundamental en cualquier aplicación web. El modelo en Django se define mediante la utilización de clases que representan las entidades y relaciones de la base de datos. Django proporciona un ORM (mapeo objeto-relacional) potente y flexible que permite interactuar con la base de datos de forma sencilla y segura. Esta capa se encarga de gestionar la creación, lectura, actualización y eliminación de los datos, así como de realizar consultas complejas.

- Vista (V): En Django, la vista se encarga de seleccionar los datos que se mostrarán al usuario y definir cómo se presentarán. Esta parte del patrón MVC se logra utilizando vistas y plantillas. La vista en Django se define como una función o una clase que procesa la solicitud del usuario y devuelve una respuesta. La vista utiliza el modelo para obtener los datos necesarios y los pasa a una plantilla, que se encarga de renderizar la respuesta final que se enviará al usuario. Las plantillas en Django permiten separar la lógica de presentación del código de la vista, lo que facilita la creación de interfaces de usuario dinámicas y flexibles.

- Controlador (C): En Django, la responsabilidad del controlador recae en el propio framework. Django utiliza la URLconf (configuración de URL) para mapear las URL entrantes a las funciones o clases de vista correspondientes. Cuando se recibe una solicitud, Django examina la URL solicitada y la compa-

ra con las rutas definidas en la URLconf. A continuación, invoca la función de vista adecuada para manejar la solicitud. Esto permite una estructura organizada y coherente para el enrutamiento de solicitudes y la delegación a las vistas correspondientes.

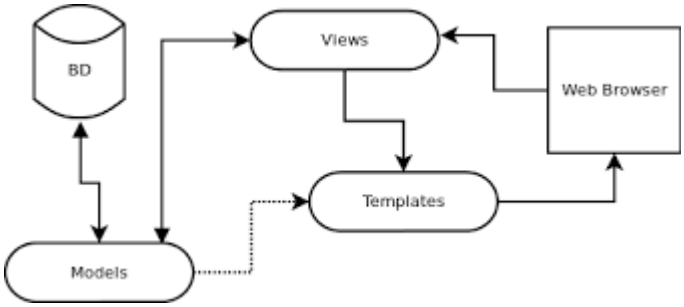

Figura 2 Modelo Vista Plantilla(Fuente: Django s. f.)

Plantilla

```
  mensajes = conversacion.mensajes.all()
        return render(request, 'mensajes.html', {
            'conversacion': conversacion,
            'mensajes': mensajes,
            'form': form
        })
```

Fig 3 Capa de Template (Fuente: Elaboración Propia)

Modelo

```
def post(self, request, conversacion_id):
        # Obtener la conversación y su formulario de mensaje
        conversacion = Conversacion.objects.get(id=conversacion_id)
        form = MensajeForm(request.POST, request.FILES)

        if form.is_valid():
            # Crear y guardar el nuevo mensaje
            mensaje = form.save(commit=False)
            mensaje.conversacion = conversacion
            mensaje.remitente = request.user
            mensaje.save()
            return redirect('mensajes', conversacion_id=conversacion_id)
```

Fig 4 Capa de Modelo(Fuente: Elaboración Propia)

Vistas

```
class ConversacionesView(View):
    def get(self, request):
        # Obtener todas las conversaciones del usuario
         conversaciones_enviadas = Conversacion.objects.filter(remitente=request.u-
ser)
        conversaciones_recibidas = Conversacion.objects.filter(destinatario=reques-
t.user)
        return render(request, 'conversaciones.html', {
            'conversaciones_enviadas': conversaciones_enviadas,
            'conversaciones_recibidas': conversaciones_recibidas
        })

class MensajesView(View):
    def get(self, request, conversacion_id):
        # Obtener la conversación y sus mensajes
        conversacion = Conversacion.objects.get(id=conversacion_id)
        mensajes = conversacion.mensajes.all()
        form = MensajeForm()
        return render(request, 'mensajes.html', {
            'conversacion': conversacion,
            'mensajes': mensajes,
            'form': form
        })
```

Fig 5 Capa de Views(Fuente: Elaboración Propia)

2.4 Prototipo de interfaz

Un prototipo de interfaz es una representación visual o interactiva preliminar de cómo será la interfaz de un sistema o aplicación. Es una versión tangible y de baja fidelidad que permite a los diseñadores, desarrolladores y usuarios experimentar y evaluar la apariencia, la estructura y la funcionalidad de la interfaz antes de su implementación final.

El prototipo de interfaz ayuda a los equipos de desarrollo a visualizar y comunicar ideas, identificar posibles problemas y realizar ajustes antes de invertir tiempo y recursos en el desarrollo completo del sistema. También permite a los usuarios finales comprender mejor la apariencia y el flujo de la interfaz, y proporcionar comentarios valiosos para su mejora.

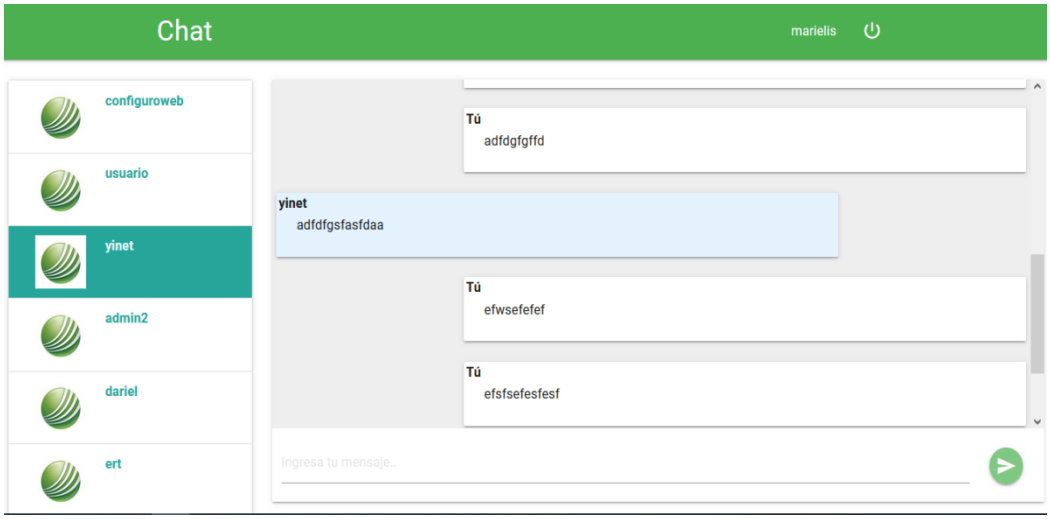

Fig 6 Prototipo de interfaz de Portada(Fuente: Elaboración Propia)

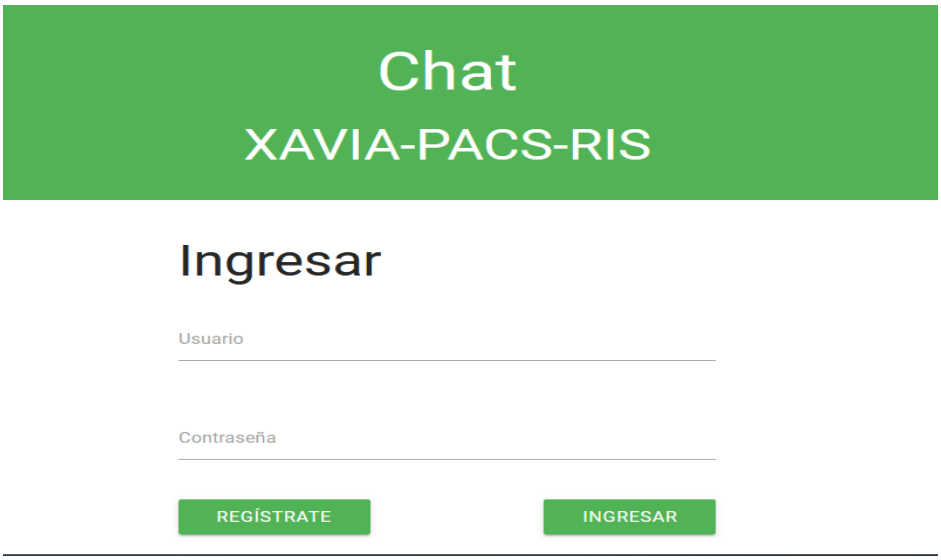

Fig 7 Prototipo de interfaz de Autenticación (Fuente: Elaboración Propia)

2.5 Patrones de diseño

La calidad del producto final de un software está directamente influenciada por un buen diseño, y gran parte de ello se basa en la correcta aplicación de los patrones de diseño y arquitectónicos existentes en la actualidad. Los patrones de diseño son soluciones recurrentes a problemas comunes que sur-

gen durante el proceso de diseño de software. Son como plantillas predefinidas que se pueden personalizar para abordar problemas de diseño recurrentes en el código (Torresburriel Estudio,2022)

Al utilizar el framework Django, se garantiza la aplicación de varios de estos patrones, lo que permite a los desarrolladores seguir buenas prácticas de programación y ahorrar tiempo y recursos. A continuación, se presentan los principales patrones utilizados en el diseño e implementación de la solución.

2.5.1 Patrones GRASP

Los patrones GRASP (por sus siglas en inglés, *General Responsibility Assignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos.(Gof y Grasp,2019a)

Los patrones GRASP utilizados en la solución propuesta fueron:

- Creador: este patrón es quien guía el proceso de asignación de responsabilidades relacionadas con la creación de objetos. Tiene el objetivo de asignarle a la clase B la responsabilidad de crear una instancia de la clase A.
- Bajo acoplamiento: Plantea que se debe poder reutilizar las funcionalidades de las distintas clases, con un nivel de dependencia mínima. Este patrón se evidencia en todas las aplicaciones web que funcionen sobre Django, pues cada pieza de las aplicaciones tiene un propósito clave, que puede modificarse sin afectar otras piezas. Por ejemplo, se puede cambiar la URL de cierta parte de la aplicación sin tener que afectar la implementación subyacente o se puede modificar el HTML de una página sin tener que tocar el código Python que la renderiza.
- Alta cohesión: Asigna responsabilidades de manera tal que la cohesión siga siendo alta, o sea que las funcionalidades de las clases estén altamente relacionadas de forma tal que exista un a colaboración entre ellas para compartir el esfuerzo y no caiga todo el peso sobre una única c lase. Usar este patrón simplifica el mantenimiento y favorece el bajo acoplamiento.
- Controlador: Permite manejar todos los eventos del sistema, al servir de intermediario entre la s interfaces y el algoritmo que las implementa.

```
def chat_view(request):
    if not request.user.is_authenticated:
```

```
        return redirect('index')
    if request.method == "GET":
        return render(request, 'chat/chat.html',
                      {'users': User.objects.exclude(username=request.user.user-
name)})
```

Fig 8 Ejemplo de Patrones GRASP (Creador, Bajo Acoplamiento, Alta Coh esión y Controlador) (Fuente: Elaboración Propia)

2.5.2 Patrones GOF

Patrones publicados por Gamma, Helm, Johnson y Vlossodes en 1995: patrones de la banda de los cuatro (del inglés, *Gang of Four*). Esta serie de patrones permiten ampliar el lenguaje, aprender nuev os estilos de diseño y además introducir más notación UML. Existen 23 patrones GoF de los cuales 1 5 se utilizan con frecuencia. Los patrones de diseño del grupo GoF se clasifican en tres grandes cate gorías basadas en su propósito: creacionales, estructurales y de comportamiento. A continuación, se describe el patrón GoF utilizado en la solución propuesta(Gof y Grasp,2019b):

- Decorador: Patrón estructural que extiende la funcionalidad de un objeto dinámicamente de m anera tal que es transparente a sus clientes, utiliza una instancia de una subclase de la clase original que delega las operaciones al objeto original. Este patrón se evidencia en la clase Logi n require (decorador que trae Django por defecto, para acceder a una clase).

```
@csrf_exempt

def message_list(request, sender=None, receiver=None):

    """

    List all required messages, or create a new message.

    """

    if request.method == 'GET':

         messages = Message.objects.filter(sender_id=sender, receiver_id=receiver,
is_read=False)

          serializer = MessageSerializer(messages, many=True, context={'request':
request})

        for message in messages:
```

```
            message.is_read = True

            message.save()

        return JsonResponse(serializer.data, safe=False)

    elif request.method == 'POST':

        data = JSONParser().parse(request)

        serializer = MessageSerializer(data=data)

        if serializer.is_valid():

            serializer.save()

            return JsonResponse(serializer.data, status=201)

        return JsonResponse(serializer.errors, status=400)
```

Fig 9 Ejemplo de Patrones GoF (Decorador) (Fuente: Elaboración Propia)

2.5.3 Modelo de datos

EL modelo de datos es un lenguaje orientado a hablar de una base de datos. Típicamente un modelo de datos permite describir:

- Las estructuras de datos de la base: El tipo de los datos que hay en la base y la forma en que se relacionan.
- Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar la realidad deseada.
- Operaciones de manipulación de los datos: típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base.

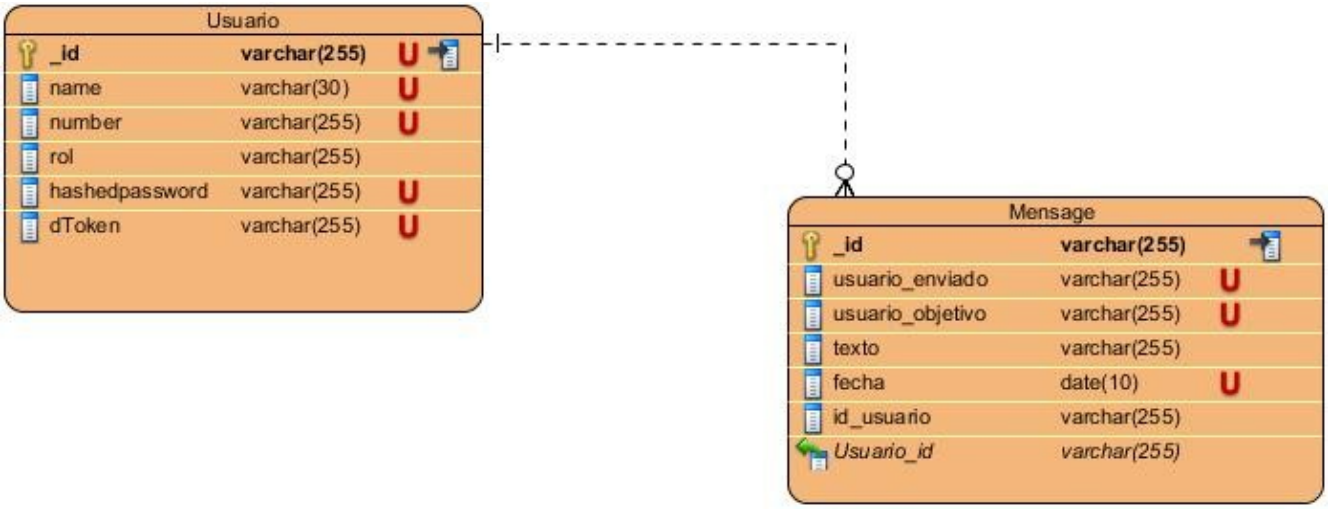

Fig 10 Modelo de Base de Datos (Fuente: Elaboración propia)

2.6 Conclusiones del capítulo

- Se identificaron un total de 30 requisitos, donde se definen 20 como funcionales y 10 se caracterizan por ser no funcionales, los cuales serán parte del funcionamiento de la aplicación.
- Se estableció como estilo arquitectónico la arquitectura cliente-servidor, la cual responde a las necesidades y flujo de trabajo de la aplicación a desarrollar.
- Se utilizaron patrones de diseños como: Modelo Vista Controlador, GRASP y Gof; los cuales contribuirán a facilitar labores de mantenimiento en futuras versiones del software.
- Se confeccionó un modelo de datos que recoge el funcionamiento de la base de datos del sistema.

CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En este capítulo se aborda las diferentes pruebas realizadas al componente y que son exigidas por la metodología AUP UCI V4 para el buen funcionamiento y calidad de la propuesta desarrollada. Se detalla con exactitud los diferentes resultados alojados por cada prueba y la respuesta a dichas no conformidades.

3.1 Diagrama de despliegue

Los diagramas de despliegue son una herramienta visual que se utiliza para representar la infraestructura física de un sistema, incluyendo procesadores, nodos y dispositivos de hardware. También muestran las conexiones de comunicación entre estos elementos y la ubicación de los archivos de software en el hardware correspondiente.

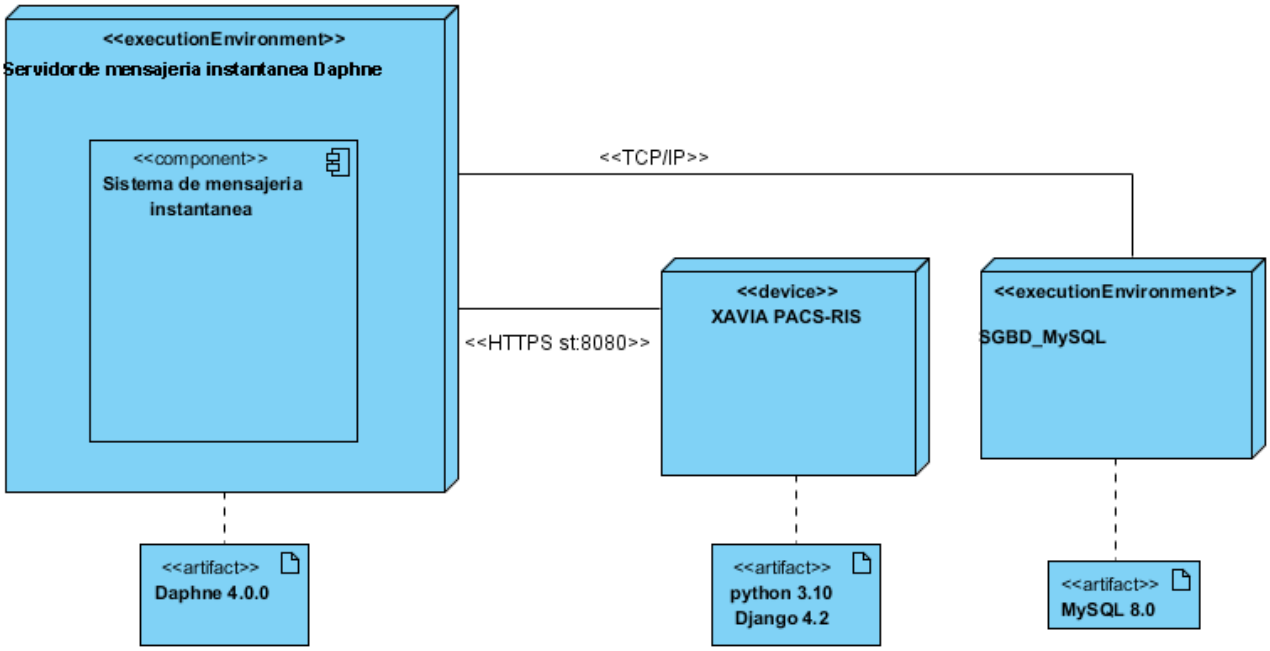

Fig 11 Diagrama de Despliegue (Fuente: Elaboración Propia)

3.2 Pruebas

Las pruebas de software comprenden una serie de procesos cuyo objetivo es verificar el funcionamiento adecuado de un sistema o aplicación en distintos momentos. Estas pruebas cubren todas las etapas de desarrollo, desde su creación hasta su implementación. Resulta especialmente interesante poder ejecutar las pruebas de forma automática para detectar posibles problemas que puedan surgir cuando un cambio en una parte del sistema afecta inadvertidamente a otras áreas. En general, las pruebas implican operaciones del sistema que evalúan los resultados bajo condiciones controladas. Esto confiere una importancia vital a la realización de pruebas de software.(Pruebas del Sistema,2020)

3.2.1 Estrategia de pruebas

La Metodología AUP UCI V4 enfatiza la realización de pruebas de software al finalizar cada iteración para garantizar la calidad y satisfacción del cliente. Se destacan las pruebas unitarias y de aceptación como las más importantes y ágiles en el proceso. Las pruebas unitarias evalúan el código desarrollado, mientras que las pruebas de aceptación verifican si el software cumple con los criterios definidos. Estas pruebas regulares permiten detectar y corregir errores tempranamente, asegurando una entrega funcional y de calidad.

3.2.2 Pruebas unitarias

Las pruebas unitarias son pruebas automatizadas que se realizan a nivel de código para evaluar individualmente cada unidad o componente. Su objetivo es verificar que cada unidad funcione correctamente según su diseño y requisitos. Para llevar a cabo estas pruebas, se crean casos de prueba específicos que cubren diferentes escenarios y condiciones posibles. Estos casos de prueba se ejecutan de forma automatizada utilizando frameworks de prueba, se comprueba si la unidad bajo prueba produce los resultados esperados y se comporta de acuerdo con las especificaciones establecidas. Se verifican las entradas y salidas esperadas, se evalúa el flujo de control y se manejan casos de error. Si se encuentran discrepancias entre los resultados reales y los esperados, se considera que la prueba ha fallado y se deben identificar y corregir los posibles errores en el código. Estas pruebas tempranas ayudan a detectar y corregir errores de manera eficiente, mejoran el modularidad y brindan confianza en la calidad del software.

Estas pruebas se basan en el minucioso examen de los detalles procedimentales, donde se comprueban los caminos lógicos del software proponiendo casos de pruebas que examinen que todas las condiciones y/o bucles estén correctos para determinar si el estado real coincide con el esperado no afirmado. (Pressman, 2021)

La prueba de caja blanca implica probar el código del software para lo siguiente:

- Agujeros de seguridad internos
- Rutas rotas o mal estructuradas en los procesos de codificación
- Las entradas específicas fluyen a través del código
- Rendimiento esperado
- Funcionalidad de bucle condicional
- Pruebe cada enunciado, objeto y función individualmente

Para realizar la prueba se decidió utilizar el Camino Básico lo que demuestra un conjunto de pasos base del programa, lo que se quiere lograr es que cada sentencia de código se ejecute mínimo una vez.

Para calcularla, se tiene tres opciones:

- Restar las aristas menos los nodos y sumar 2:
    - V(G) = Aristas – Nodos + 2
- Sumar 1 al número de nodos predicados
    - V(G) = Nodos predicados + 1
- Contar el número de regiones (espacios «encerrados entre nodos y aristas», también se tiene en cuenta el espacio «exterior» a todos los nodos y aristas).
    - V(G) = Regiones

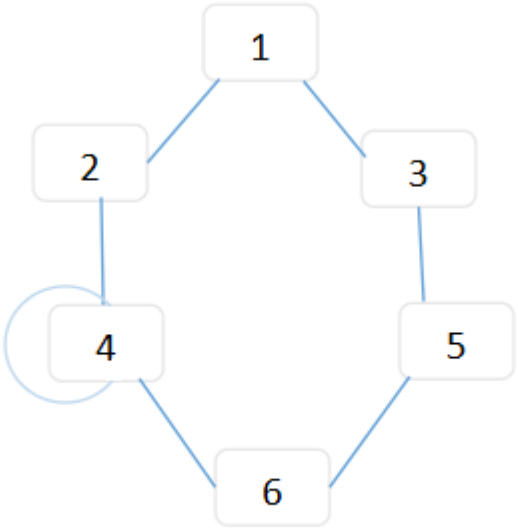

Fig 12 Camino Básico (Fuente: Elaboración propia)

Caminos:

1,2,4,6

1,2,4,4,6

1,3,5,6

Tabla 5 Juego de Datos (Fuente: Elaboración Propia)

Camino	**Mensaje**	**Estado**
1,2,4,6	**Mostrar mensaje**	**Satisfecho**
1,2,4,4,6	**Mostrar mensaje**	**Satisfecho**
1,3,5,6	**Enviar mensaje**	**Satisfecho**

Luego de realizar la prueba de caja blanca a la función listar mensaje se determinó que la complejidad ciclomática era 3 por lo que el método es sencillo y de poco riesgo para el sistema.

3.2.3 Pruebas de Caja Negra

Las pruebas de caja negra son un tipo de técnica de prueba utilizada en el desarrollo de software. En estas pruebas, se evalúa el sistema sin conocer o considerar su estructura interna o el funcionamiento interno de los componentes. En lugar de eso, las pruebas de caja negra se centran en examinar el comportamiento y las salidas del sistema en función de entradas específicas. El objetivo de las pruebas de caja negra es validar la funcionalidad del sistema y verificar si cumple con los requisitos y expectativas establecidos, sin necesidad de conocer los detalles internos de su implementación. Estas pruebas se basan en las especificaciones y los casos de uso, y se enfocan en probar diferentes escenarios y condiciones que el sistema puede enfrentar en el mundo real. (KeepCoding,2022)

Durante las pruebas de caja negra, se proporcionan entradas de prueba al sistema y se registran las salidas resultantes. Se comparan las salidas obtenidas con las esperadas según los requisitos y se verifican posibles discrepancias o errores. Además de las entradas y salidas, también se pueden evaluar otros aspectos, como el rendimiento, la seguridad y la usabilidad del sistema.

Las pruebas de caja negra ofrecen varias ventajas en el proceso de desarrollo de software. Al no requerir conocimiento interno del sistema, pueden ser realizadas por equipos diferentes o personas no involucradas en la implementación. Esto permite una perspectiva imparcial y objetiva sobre la funcionalidad del sistema. Además, estas pruebas pueden identificar problemas que no han sido considerados o detectados durante las pruebas de caja blanca (pruebas basadas en la estructura interna).

Permite encontrar errores tipo:

- Errores de funcionalidad
- Errores de interfaz
- Errores de rendimiento
- Errores de seguridad
- Errores de usabilidad
- Errores de escalabilidad

Existen varias técnicas que se pueden utilizar para llevar a cabo las pruebas de caja negra, la selecci onada entre ellas es:

La partición de equivalencia es una técnica utilizada en las pruebas de software que se basa en dividi r el conjunto de posibles entradas en grupos o particiones, donde se espera un comportamiento simila r del sistema. El objetivo de esta técnica es reducir el número de casos de prueba necesarios para cu

brir todas las posibles combinaciones de entradas, centrándose en probar representantes de cada par tición.

La idea principal detrás de la partición de equivalencia es que, si una entrada dentro de una partición es válida y produce un resultado esperado, se asume que todas las entradas dentro de esa partición t ambién serán válidas y producirán el mismo resultado esperado. Del mismo modo, si una entrada den tro de una partición es inválida y produce un resultado inesperado, se asume que todas las entradas dentro de esa partición también serán inválidas y producirán un resultado inesperado.

Se realizaron 6 casos de prueba a continuación se muestra un ejemplo de los mismos:

Tabla 6 Clases de equivalencia **(**Fuente: Elaboración Propia)

Identificador	Entrada (Clase)	Clases válidas	Clases inválidas
Usuario	Usuario	Usuario=A-Z	Usuario=0-9, carácteres especiales
Contraseña	Contraseña	Contraseña=A-Z,0-9, carácteres especiales	Contraseña=¨ ¨

Id del Escenario	Escenario	Usuario	Contraseña	Respuesta del sistema	Resultado de la prueba
EC1	Autenticarse	marielisjg V	Pelusa 9902. i	La contraseña no puede contener espacios en blanco	No satisfactorio
EC2	Autenticarse	Johnfp V	J200390.. V	El usuario se autentico correctamente	Satisfactorio
EC3	Autenticarse	Marielis99 i	Pelusa9902. V	El usuario no cumple los	No satisfactorio

				requisitos	
EC4	Autenticarse	John.x i	Jakskz i	El usuario y contraseña no son válidos	No satisfactorio

3.2.4 Pruebas de Seguridad

La prueba de seguridad de software es una evaluación sistemática y controlada de un sistema o aplicación para identificar y mitigar vulnerabilidades y riesgos de seguridad. Su objetivo es descubrir debilidades que podrían permitir acceso no autorizado, manipulación de datos o ataques maliciosos. Se utilizan técnicas y herramientas especializadas para evaluar áreas como autenticación, protección de datos, seguridad de red, gestión de sesiones y protección contra ataques. Las pruebas de seguridad buscan garantizar la protección de la información y la integridad del sistema. (admin,2020b)

Para llevar a cabo estas pruebas de seguridad, se utilizó la herramienta Netsparker. Netsparker es una herramienta que detecta y reporta vulnerabilidades en aplicaciones web, como inyecciones de SQL y cross-site scripting (XSS), independientemente de la plataforma o tecnología utilizada en la construcción de las aplicaciones.

La tecnología única y precisa de escaneo basado en pruebas (Proof-Based ScanningTM) de Netsparker no solo informa sobre las vulnerabilidades encontradas, sino que también proporciona una prueba de concepto para confirmar que no se trata de falsos positivos. Esto evita la necesidad de verificar manualmente las vulnerabilidades identificadas.(«Netsparker Web Application Security Scanner ⋆ QMA MSS», 2019)

3.2.5 Pruebas de aceptación

Las pruebas de aceptación son pruebas finales de software realizadas para determinar si un sistema cumple con los requisitos y expectativas del cliente. Se verifican funcionalidades, usabilidad, rendimiento, seguridad y compatibilidad, y pueden involucrar a los usuarios finales. Estas pruebas son cruciales para garantizar la calidad del sistema antes de su implementación y permiten hacer ajustes necesarios. (<< **Pruebas de aceptación: El Qué Y Porqué + Los Tipos que hay que Conocer>>s. f.**)

Tabla 7 Prueba de Aceptación Historia de Usuario **(**Fuente: Elaboración Propia)

Historia de Usuario : Enviar mensaje	
Número: 15	**Nombre del requisito:** Enviar mensaje
Programador:Marielis Jiménez García	**Iteración Asignada: 1**
Prioridad:Alta	**Tiempo Estimado:8h**
Riesgo en Desarrollo: Alto	**Tiempo Real:10h**
Descripción: Permitir al usuario el envío de mensajes a otros usuarios del sistema permitiendo comunicarse entre sí.	
Condiciones de ejecución: • El usuario debe introducir el texto del mensaje y luego presionar el botón enviar.	
Entrada/Pasos de ejecución: 1. Se introduce el mensaje. 2. Presionar el botón enviar.	
Resultados esperados: Se debe enviar el mensaje.	
Evaluación de la prueba: Satisfactoria.	

3.3 Resultados de las pruebas

A continuación, se muestra un gráfico final con los resultados arrojados en las cuatro iteraciones de pruebas donde se obtuvo un total de 15 no conformidades (NC) significativas: 5 con prioridad normal, 2 con prioridad baja, 4 con prioridad alta y 4 informativas. Los resultados obtenidos de las pruebas de aceptación por cada iteración pueden verse en la siguiente figura de este documento.

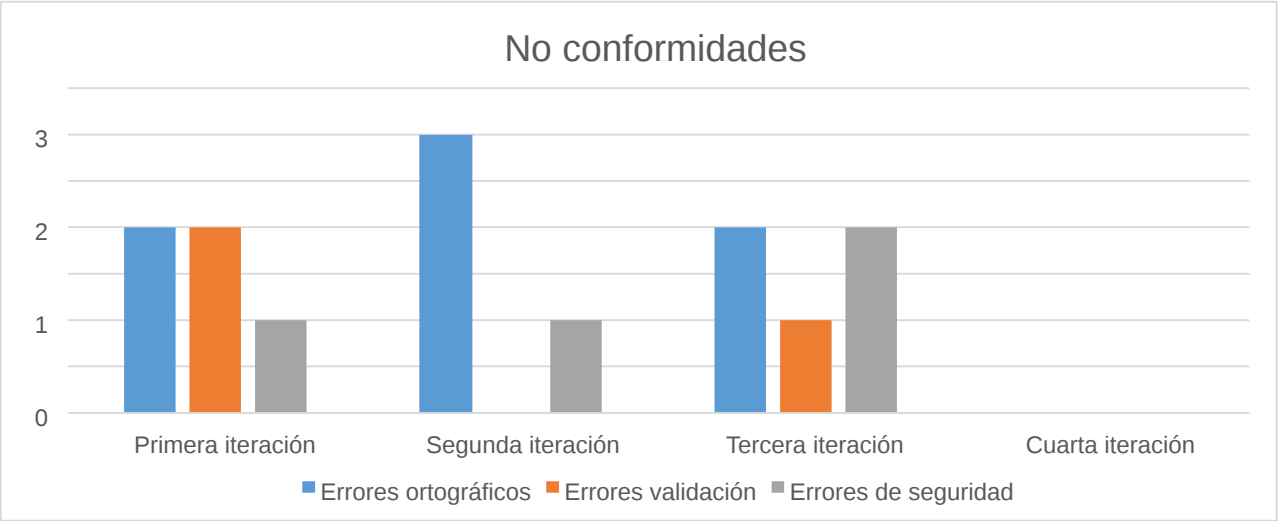

Fig 15 Gráfico de no conformidades (Fuente: Elaboración Propia)

3.4 Impacto de la solución propuesta en el XAVIA PACS-RIS

Al analizar el componente de mensajería instantánea, se puede suponer que este sistema brinda a los usuarios la posibilidad de tener acceso al mismo de forma rápida y sencilla, en cualquier momento, y a través de cualquier dispositivo. El desarrollo del componente permitirá facilitar la comunicación entre profesionales de la salud y los usuarios, que utilizan el sistema XAVIA PACS-RIS. Esta propuesta de solución se presume que tendrá un gran impacto en el sistema XAVIA PACS-RIS ya que permitirá una mejora en la comunicación entre los profesionales del mismo.

Luego del análisis, diseño y codificación de la solución se puede demostrar que:

Tabla 8 Validación de variables **(**Fuente: Elaboración Propia)

Características	Antes de la solución	Después de la solución
No existía una forma de comunicarse dentro del propio sistema.	El XAVIA PACS-RIS no contaba con un componente de mensajería instantánea propio del sistema.	Posee un componente de mensajería instantánea que permite la comunicación dentro del sistema.

Por tanto, se puede afirmar que con el desarrollo del sistema de mensajería instantánea para el XAVIA PACS-RI contribuirá a la comunicación de los profesionales del mismo.

3.5 Conclusiones del capítulo

- La implementación de pruebas durante el ciclo de vida del software, como las pruebas unit arias y de aceptación, mostro ser esencial para garantizar un buen funcionamiento y mejor ar la calidad de los resultados.
- Las pruebas de aceptación resultaron un paso crucial antes de la implementación del softw are ya que permitieron la verificación del correcto funcionamiento y aceptabilidad de los re quisitos planteados por el cliente
- Los resultados de las pruebas demostraron la efectividad de la solución para ser desplega da.

CONCLUSIONES GENERALES

Considerando los resultados descritos en este informe, la necesidad y el objetivo planteado por la investigación se arriba a las siguientes conclusiones:

1. El exhaustivo análisis de diversas fuentes, herramientas y soluciones tecnológicas para la distribución de contenido permitió identificar las características clave del sistema propuesto y su adaptabilidad al contexto nacional. Esto aseguró que el sistema cumpliera con los requisitos específicos del entorno y ofreciera una solución adecuada para mejorar la comunicación.

2. La implementación de las herramientas y tecnologías seleccionadas demostró una correspondencia exitosa entre los resultados obtenidos y los esperados. Esto garantizó un nivel de precisión en el análisis y diseño del sistema, lo que resulta fundamental para su eficacia y correcto funcionamiento.

3. Las exhaustivas pruebas de software realizadas, tanto en el código como en las interfaces implementadas, arrojaron resultados satisfactorios. Estas pruebas validaron la seguridad del código y su capacidad técnica en el entorno específico para el cual se desarrolló el sistema. Este proceso es esencial para asegurar la calidad y la confiabilidad del sistema.

RECOMENDACIONES

Los resultados obtenidos en este trabajo sientan las bases para la realización de proyectos futuros que permitan enriquecer y escalar la solución propuesta. Por lo que se recomienda que:

- Integrar el complemento al sistema XAVIA PACS-RIS.
- Desarrollar funcionalidades que permitan el envío de archivos, multimedia y mensajes de voz.
- Crear una aplicación para el uso de la misma en teléfonos móviles.
- Integración con otros sistemas relacionados al sistema XAVIA PACS-RIS.

REFERENCIAS BIBLIOGRAFICAS

1. Admin. (2020a, enero 1). ¿Qué es una prueba de caja BLANCA? Técnicas, muestras y tipos. *Ebooks Online*, https://ebooksonline.es/qué-es-una-prueba-de-caja-blanca-tecnicas-muestras-y-tipos/
2. Admin. (2020b, enero 1). ¿Qué es una prueba de seguridad? Tipos con ejemplo. *Ebooks Online*, https://ebooksonline.es/qué-es-una-prueba-de-seguridad-tipos-con-ejemplo/
3. Agile Unified Process - EcuRed. [en línea], [sin fecha]. [. Disponible en: https://www.ecured.cu/Agile_Unified_Process.
4. AGUILAR, J.M., 2019. ¿Qué es el patrón MVC en programación y por qué es útil? *campusMVP.es* [en línea]. Disponible en: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx.
5. BARRAGÁN QUIZHPE, C.F., 2017. Adaptación de las normas ISO 27001 e HIPPA para la reducción de riesgos en la seguridad en hospitales nivel I del IESS. En: Accepted: 2017-11-10T19:13:20Z [en línea]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/7544.
6. CADAVID, A.N., 2013. Revisión de metodologías ágiles para el desarrollo de software. Prospectiva [en línea], vol. 11, no. 2. ISSN 22161368, 16928261. DOI 10.15665/rp. v11i2.36. Disponible en: http://ojs.uac.edu.co/index.php/prospectiva/article/view/36.
7. Comenzando - Let’s Encrypt - Certificados SSL/TLS Gratuitos, [no date]. Available from: https://letsencrypt.org/es/getting-started/
8. Cómo escribir casos de prueba para software: ejemplos y tutorial, [no date].Available from: https://es.parasoft.com/blog/how-to-write-test-cases-for-software-examples-tutorial/
9. Configuración y uso de Python — documentación de Python - 3.12.0, [no date].Available from: https://docs.python.org/es/3/using/index.html
10. Daphne / ASGI won’t use the DJANGO_SETTINGS_MODULE I specify · Issue #230 · django/channels · GitHub, [no date]. Available from: https://github.com/django/channels/issues/230
11. Deploying Django Channels using Daphne · Abu Ashraf Masnun, [no date].Available from: http://masnun.rocks/2016/11/02/deploying-django-channels-using-daphne/

12. Documentación de Django | Documentación de Django | Django.Available from: https://docs.djangoproject.com/es/4.2/

13. Framework Web Django (Python) - Aprende desarrollo web | MDN, 2022. Available from: https://developer.mozilla.org/es/docs/Learn/Server-side/Django

14. Django es un framework web extremadamente popular y completamente funcional, escrito en Python. El módulo muestra por qué Django es uno de los frameworks de servidores web más populares, cómo configurar un entorno de desarrollo y cómo empezar a usarlo para crear tus propias aplicaciones web.

15. Django. *Django Project* [en línea], [sin fecha]. Disponible en: https://docs.djangoproject.com/es/4.2/contents/.

16. ESTÉVEZ, J.R., 2020. La ingeniería de requisitos en el desarrollo de aplicaciones informáticas. *Revista Cubana de Informática Médica* [en línea], vol. 12, no. 2.ISSN 1684-1859. Disponible en: https://revinformatica.sld.cu/index.php/rcim/article/view/375.

17. Gof y Grasp. *prezi.com* [en línea], [sin fecha]. Disponible en: https://prezi.com/mymp91_tnj3g/gof-y-grasp/.

18. GUERRERO, C.A., SUÁREZ, J.M. y GUTIÉRREZ, L.E., 2013. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica* [en línea], vol. 24, no. 3.ISSN 0718-0764. DOI 10.4067/S0718-07642013000300012. Disponible en: http://www.scielo.cl/scielo.php?script=sci_abstract&pid=S0718-07642013000300012&lng=es&nrm=iso&tlng=es.

19. Halo Health - Your Trusted Online Pharmacy and Clinic. *Halo Health* [en línea], [sin fecha]. Disponible en: https://haloishere.com/.

20. HENRYM, 2019. Deploy django-channels with Apache2 and Daphne. *Stack Overflow*. Online. 21 February 2019. Available from: https://stackoverflow.com/q/54809234

21. “HIPAA Regulations: A New Era of Medical-Record Privacy?” by George J. Annas, [no date].Available from: https://scholarship.law.bu.edu/faculty_scholarship/1283/

22. Informática Médica - EcuRed, [no date].Available from: https://www.ecured.cu/Inform%C3%A1tica_M%C3%A9dica

23. Informática médica. Tomo 1. Computación. Segunda edición – ECIMED, [no date]. Available from: http://www.ecimed.sld.cu/2019/11/12/3143/

24. INSTANTÁNEA, Protocolos para la mensajería instantánea Son usados en la mensajería, FUENTES, Algunos De Ellos Son Propietarios Y. Por Tal Motivo No Ofrecen Documentación Ni Dan Acceso a Sus, PARTE, Por Otra, TOUCH, hay otros que son libres y están muy bien documentados a disposición de todos los usuarios Sistemas Operativos compatibles MSN Messenger, FRING, NIMBUZZ, EBUDDY, and PALRINGO, [no date]. Protocolo para la mensajería instantánea – EcuRed. Available from: https://www.ecured.cu/Protocolo_para_la_mensajer%C3%ADa_instant%C3%A1nea

25. IZAGUIRRE, I.L.V., [sin fecha]. Sistema de Información Radiológica XAVIA RIS.

26. KeepCoding, R. (2022, junio 3). *¿Qué es una prueba de caja negra?* |KeepCoding Tech School.https://keepcoding.io/blog/que-son-las-pruebas-de-caja-negra/

27. La aplicación de la informática a la práctica médica ha permitido el desarrollo TICs - Concepto, ventajas, desventajas y ejemplos, [no date]. Available from: https://concepto.de/tics/

28. Ley de Transferencia y Responsabilidad de Seguro Médico (HIPAA) de 1996. [en línea], [sin fecha]. Disponible en: https://www.cancer.org/es/cancer/asuntos-financieros-y-cobertura/leyes-de-los-seguros-medicos/que-es-hipaa.html.

29. Ley HIPAA. *Comité para la Protección de los Seres Humanos en la Investigación (CPSHI)* [en línea], [sin fecha]. Disponible en: https://www.uprm.edu/irb/ley-hipaa/.

30. Lo que aprendí sobre WebSockets al crear una aplicación de chat en tiempo real usando Socket.IO., [no date]. Available from: https://ichi.pro/es/lo-que-aprendi-sobre-websockets-al-crear-una-aplicacion-de-chat-en-tiempo-real-usando-socket-io-4234324206852

31. Manual de HTML, [no date]. Available from: https://desarrolloweb.com/manuales/manual-html.html

32. MAEZ, D., 2023. Examining Interprofessional Communication in Healthcare. *Quality Improvement/Patient Safety Symposium* [en línea], Disponible en: https://digitalrepository.unm.edu/hsc_qips/77.

33. Mensajería instantánea: qué es, características, tipos, ejemplos, [no date]. Available from: https://www.lifeder.com/mensajeria-instantanea/

34. Metodologías Ágiles: Definición, Manifiesto, Principios, SCRUM, Kanban. [en línea], 2023. Disponible en: https://innovaromorir.com/metodologias-agiles-definicion-manifiesto-principios-scrum-kanban/.

35. MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN. [en línea], 2022. Disponible en: https://developer.mozilla.org/es/docs/Glossary/MVC.

36. NOGUERAS, A., 2022. Metodologías ágiles para la gestión de proyectos: Guía básica. EALDE Business School [en línea]. Disponible en: https://www.ealde.es/principales-metodologias-agiles/.

37. NOSOWSKY, R. y GIORDANO, T.J., 2006. The Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy Rule: Implications for Clinical Research. Annual Review of Medicine [en línea], vol. 57, no. 1.DOI 10.1146/annurev.med.57.121304.131257. Disponible en: https://doi.org/10.1146/annurev.med.57.121304.131257.

38. PACS-RIS 3.0 | Universidad de las Ciencias Informáticas, [no date].Available from: https://www.uci.cu/investigacion-y-desarrollo/productos/xavia/pacs-ris-30

39. (PDF) Metodología Agile Unified Process (AUP | Rodrigo González Rodríguez - Academia.edu, [no date]. Available from: https://www.academia.edu/33406233/Metodolog%C3%ADa_Agile_Unified_Process_AUP

40. PALMERO, E., 2022. ¡Conoce algunos de los patrones de arquitectura de software que utilizan los expertos! *Eykkon* [en línea]. Disponible en: https://www.eykkon.com/blog/patrones-de-arquitectura-de-software/.

41. Patrones en la Arquitectura de Software. [en línea], [sin fecha]. Disponible en: https://es.linkedin.com/pulse/patrones-en-la-arquitectura-de-software-elmo-renato-castro-ramirez.

42. Plan de Iteración | certified. [en línea], [sin fecha]. Disponible en: https://certified-es.blogspot.com/2009/11/plan-de-iteracion.html.

43. Pressman, Roger S*. Ingeniería de Software-Un enfoque practico 9ma.*MEXICO: s.n.,2021.

44. Protocolos de mensajería instantánea en profesordeinformatica.com, [no date].Available from: https://www.profesordeinformatica.com/servicios/mensajeria_instantanea/protocolos

45. Pruebas del Sistema. *manuel.cillero.es* [en línea], [sin fecha]. Disponible en: https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/pruebas/sistema/.

46. Qué es MySQL: Características y ventajas. OpenWebinars.net [en línea], 2019. Disponible en: https://openwebinars.net/blog/que-es-mysql/.

47. *¿Qué es la prueba de aceptación del usuario(UAT):Una guía completa?-Tipos De Pruebas.(s. f.).*Recuperado 2 de noviembre de 2022, de https://spa.myservername.com/what-is-user-accceptance-testing/

48. ¿Qué es la informática médica? - Spiegato, [no date]. Available from: https://spiegato.com/es/que-es-la-informatica-medica

49. ¿Qué es un prototipo? | Curso de Interacción Persona-Ordenador. [en línea], [sin fecha]. Disponible en: https://mpiua.invid.udl.cat/fases-mpiua/prototipado/que-es-un-prototipo/.

50. ¿Qué es una metodología ágil y qué ventajas conlleva? *appvizer.es* [en línea], [sin fecha]. Disponible en: https://www.appvizer.es/revista/organizacion-planificacion/gestion-proyectos/metodologia-agil.

51. Qué son los patrones de diseño y por qué utilizarlos | Torresburriel Estudio. [en línea], 2022. Disponible en: https://torresburriel.com/weblog/que-son-los-patrones-de-diseno-y-por-que-utilizarlos/.

52. QUIROZ, A., 2022. ¿Qué es y para qué sirve la mensajería instantánea? *B2Chat* [en línea]. Disponible en: https://www.b2chat.io/blog/mensajeria-instantanea/mensajeria-instantanea-tipos-para-que-sirve/.

53. REYES, R.A.M., [sin fecha]. Los Patrones como un Medio del Diseño Orientado a Objetos.

54. SAVAGE, M. y SAVAGE, L.C., 2020. Doctors Routinely Share Health Data Electronically Under HIPAA, and Sharing With Patients and Patients' Third-Party Health Apps is Consistent: Interoperability and Privacy Analysis. En: Company: Journal of Medical Internet ResearchDistributor: Journal of Medical Internet ResearchInstitution: Journal of Medical Internet ResearchLabel: Journal of Medical Internet Researchpublisher: JMIR Publications Inc., Toronto, Canada, *Journal of Medical Internet Research* [en línea], vol. 22, no. 9.DOI 10.2196/19818. Disponible en: https://www.jmir.org/2020/9/e19818.

55. SINGUREANU, C., [sin fecha]. Pruebas de caja blanca: tipos, proceso, herramientas y mucho más. *https://www.zaptest.com/es* [en línea]. Disponible en: https://www.zaptest.com/es/pruebas-de-caja-blanca-que-es-como-funciona-retos-metricas-herramientas-y-mas.

56. SUÁREZ-OBANDO, Fernando and CAMACHO SÁNCHEZ, Jhon, 2013. Estándares en informática médica: generalidades y aplicaciones. *Revista Colombiana de Psiquiatría*. 1 September 2013. Vol. 42, no. 3, p. 295–302. DOI 10.1016/S0034-7450(13)70023-4.

57. Telegram Messenger. [en línea], [sin fecha]. Disponible en: https://telegram.org/.

58. TigerConnect - Clinical Healthcare Software - Home. TigerConnect [en línea], [sin fecha]. Disponible en: https://tigerconnect.com/.

59. toDus. [en línea], [sin fecha]. Disponible en: https://www.todus.cu/landing.

60. Variación de AUP para la UCI (AUP-UCI) - Metodologías tradicionales, [no date]. Available from: https://1library.co/article/variaci%C3%B3n-aup-uci-aup-uci-metodolog%C3%ADas-tradicionales.zkwgn934

61. Ventajas y desventajas de la mensajería instantánea – Creative District, [no date]. Available from: https://creativedistrictblog.wordpress.com/2016/02/01/ventajas-y-desventajas-de-la-mensajeria-instantanea/

62. Videotutorial Principios GRASP - Fundamentos de la programación: Diseño orientado a objetos | LinkedIn Learning, antes Lynda.com. *LinkedIn* [en línea], [sin fecha]. Disponible en: https://es.linkedin.com/learning/fundamentos-de-la-programacion-diseno-orientado-a-objetos-9062878/principios-grasp.

63. Visual Paradigm para UML es una herramienta para desarrollo. *prezi.com* [en línea], [sin fecha]. Disponible en: https://prezi.com/3v2w1mpsptxx/visual-paradigm-para-uml-es-una-herramienta-para-desarrollo/.

64. WebSocket | Un canal de comunicación para la web en tiempo real - IONOS, [no date]. Available from: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-websocket/

ANEXOS

Historia de Usuario : Configurar notificaciones	
Número: 9	**Nombre del requisito: Configurar notificaciones**
Programador:Marielis Jiménez García	**Iteración Asignada: 2**
Prioridad:Alta	**Tiempo Estimado:6h**
Riesgo en Desarrollo: Bajo	**Tiempo Real:8h**
Descripción: Como usuario del sistema de mensajería, quiero tener la capacidad de configurar mis preferencias de notificación para personalizar cómo y cuándo recibir notificaciones sobre eventos relevantes.	
Criterios de Aceptación: ❖ Como usuario, quiero poder acceder a la configuración de notificaciones desde mi perfil o ajustes de cuenta. ❖ Como usuario, quiero poder seleccionar el tipo de eventos para los cuales deseo recibir notificaciones, como nuevos mensajes. ❖ Como usuario, quiero poder establecer la frecuencia de las notificaciones	
Observaciones: • Asegurarse de que las preferencias de notificación se guarden correctamente y se reflejen de manera consistente en todas las plataformas. • Implementar una interfaz intuitiva y fácil de usar para configurar las preferencias de notificación.	

Anexo#1: Historia de Usuario: Configurar notificaciones (Fuente: Elaboración Propia)

Historia de Usuario : Mostrar mensajes de historial	
Número: 19	**Nombre del requisito:** Mostrar mensajes de historial
Programador:Marielis Jiménez García	**Iteración Asignada: 3**
Prioridad:Bajo	**Tiempo Estimado:8h**
Riesgo en Desarrollo: Bajo	**Tiempo Real:10h**

Descripción:

El sistema debe ser capaz de mostrar los mensajes del historial de chat en la interfaz de usuario. Esto permitirá a los usuarios ver el historial de conversaciones anteriores y revisar cualquier información relevante.

Criterios de Aceptación:

- Cada mensaje del historial debe mostrar el remitente, la fecha y hora de envío, y el contenido del mensaje.
- Los usuarios podrán desplazarse hacia arriba y hacia abajo en el historial de mensajes para ver mensajes anteriores y posteriores.
- El historial de mensajes se actualizará automáticamente cuando se reciban nuevos mensajes en tiempo real.
- Se proporcionará una opción para borrar el historial de mensajes si se desea.

Observaciones:

- Es importante asegurarse de que la interfaz de usuario sea intuitiva y fácil de usar para los usuarios.
- El diseño y la presentación de los mensajes del historial deben ser coherentes con el resto de la interfaz del sistema.

Anexo#2 Historia de Usuario: Mostrar mensajes de historial (Fuente: Elaboración Propia)