Universidad de las Ciencias Informáticas

Facultad 4.

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

Componente para la mensajería instantánea y notificaciones sobre Node.js.

Autores:

Neysa Baldoquín Alonso

Armando Rodríguez Vallín

Tutores:

MSc. Rosalba Carralero Medina

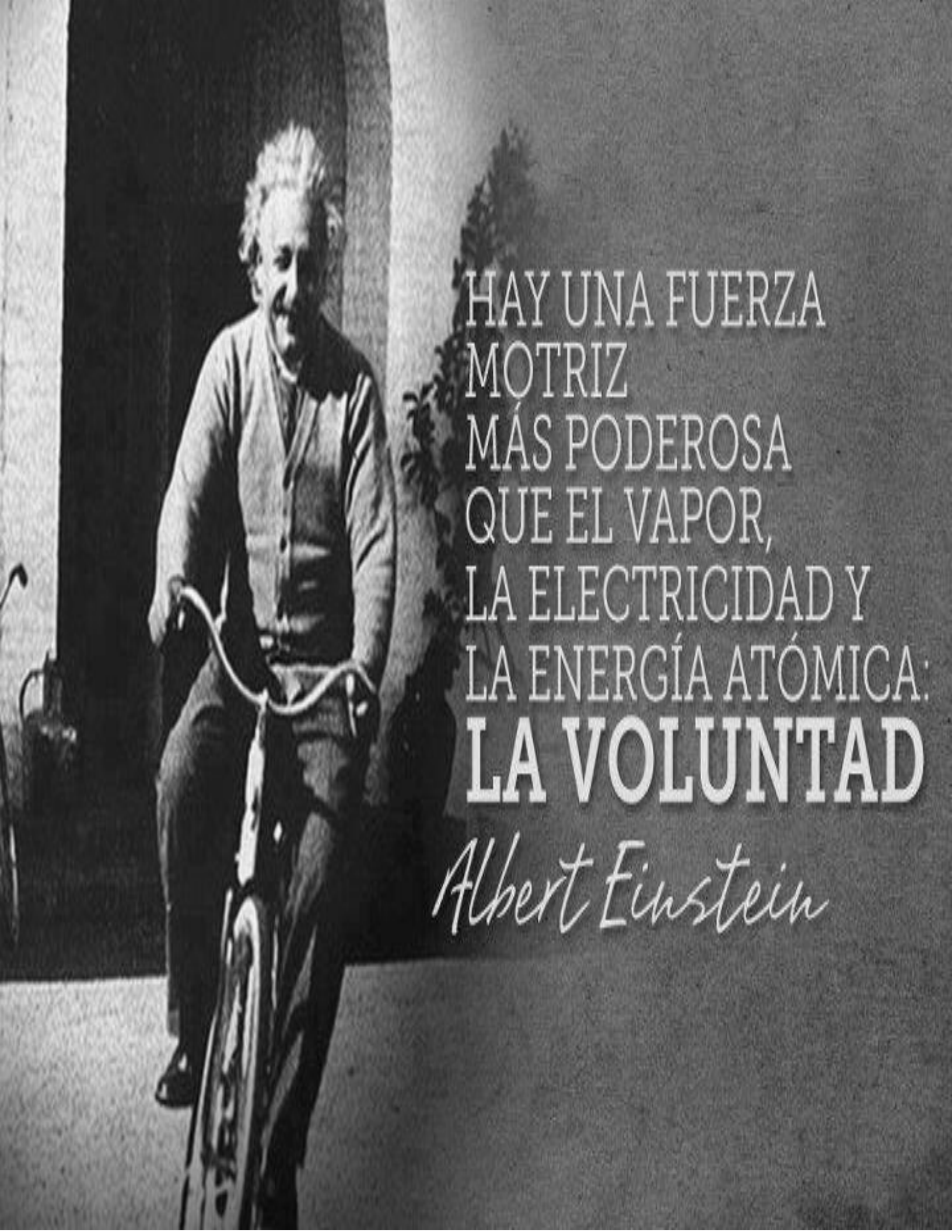
MSc. Iván Pérez Mallea

Co-Tutores:

Ing. Gustavo Crespo Sánchez

La Habana, Cuba.

Junio de 2015


HAY UNA FUERZA
MOTRIZ
MÁS PODEROSA
QUE EL VAPOR,
LA ELECTRICIDAD Y
LA ENERGÍA ATÓMICA:
LA VOLUNTAD
Albert Einstein

Declaración de Autoría:

Declaramos ser los únicos autores de este trabajo y concedemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los____ días del mes de _____del año 2015.

_____ _____

Neysa Baldoquín Alonso Armando Rodríguez Vallín

_____ _____

MSc. Rosalba Carralero Medina MSc. Iván Pérez Mallea

Agradecimientos:

Neysa:

Cuando ingresé a esta universidad trazándome como meta graduarme como ingeniera, veía como un sueño dar gracias a todas aquellas personas que me han ayudado tanto y que han estado a mi lado en todo momento. Mi primer agradecimiento va dirigido a mi mami, que Dios la tenga en la gloria, ejemplo de mujer incansable y luchadora, inteligente, respetada, tan humana, quien dio su vida para verme convertida en la mujer que soy hoy. Gracias mami por estar siempre a mi lado aunque no te pueda ver y haberme preparado para las cosas malas que he tenido que enfrentar en la vida, por ser mi fuerza de seguir adelante para realizar mis sueños. Mis agradecimientos también para mis dos hermanas, Zoybe gracias por ser mi ejemplo de no darme por vencida nunca, gracias por quererme tanto, por ser mi hombro de lágrimas levantando siempre mi autoestima prometiéndome que todo iba a pasar, gracias por serlo todo para mí. Aloima, gracias por quererme tanto, por quitarte lo tuyo para dármelo a mí, gracias por tus consejos y por tu amor. Quiero agradecer a mis tías Esther por ser como una madre, siempre pendiente y cuidándome, gracias por siempre estar para mí, y Luz María, gracias por no olvidarme por quererme tanto. Quiero dar gracias además a mis dos primas Yanet y Lili quienes me han acogido como una hermana. A mis cuñados Julio, por acogerme en su familia y preocuparse tanto, a Oldaniel, quien me ha querido como una hermana, gracias a los dos. A mi suegra Miladis, gracias por ser tan buena y comprensiva conmigo, gracias por ser como una madre, gracias a la abuelita Alicia que aunque no sea mi abuela carnal es el mejor ejemplo que tengo de abuela. Gracias a Magalis y Alfonso por ser su pelu, queriéndome como su propia sobrina. Gracias a Misle, Milania y a Natacha por preocuparse tanto por mí y hacerme parte de su familia. Quiero dar gracias a mis mejores amigos Niurka que más que una amiga ha sido mi hermana, en las malas y en las buenas, Ernesto Alfono otro hermano que me dio esta universidad, Daynier, el mejor amigo que se pueda desear, gracias por tu sincera amistad, por alegrarme el día cuando parecía imposible. Tengo que dar gracias a todos los profesores que he tenido en mi periodo de estudiante pues de todos saqué muy buenas enseñanzas, pero en especial a mi profe Rafael, que decir de usted, no si decirle profe o padre, gracias por sus consejos, por secar mis lágrimas, por levantar mi ánimo, por ser su niñita como cariñosamente me dice, gracias le doy con todo mi corazón. Agradezco a mis tutores, Rosalba gracias por ser además de tutora amiga, Gustavo que más que un co-tutor fue una tutor, Mayea gracias por tu ejemplo. Finalmente quiero agradecer a mi compañero de la vida Ernesto Vladimir Pereda Díaz, gracias por tu amor, por tu paciencia, por haberme hecho descubrir el amor, gracias por ser mi vida, y quiero decirte que eres mi ejemplo a seguir, jamás te voy a decepcionar.

Armando:

Agradezco en primer lugar a los que realmente hicieron posible la realización de este Trabajo de Diploma, mis tutores, Ernesto Vladimir Pereda y Rosalba Carralero Medina quienes supieron pellizcarme cuando fue necesario y me desvié del camino, a mi co-tutor Gustavo Crespo quien me ayudó cuando más lo necesite y para no ser extenso, agradezco a todos mis compañeros los de los grupos 4104 y 4505 porque todos me impregnaron sabiduría aunque no se hayan dado cuenta.

Dedicatoria:

Neysa:

A mi madre por haber sido mi ejemplo, mi fuerza para seguir mis sueños, la mujer y madre más perfecta que ha tenido este mudo, gracias por todo lo que me enseñaste y por lo que me pudiste enseñar, gracias mami.

Armando:

Quiero dedicarle este momento a mi familia que se lo merece mucho más que yo, porque solo Dios sabe cuántas cosas hicieron aunque yo aquí no lo demostrara.

Resumen:

Las herramientas para la mensajería instantánea y las notificaciones desde su creación han servido como base para las comunicaciones establecidas entre los seres humanos. Por su facilidad de utilización y sus ventajas se han convertido en un método muy difundido a nivel mundial, contribuyendo de esta manera a disminuir distancias entre los usuarios.

Como parte del trabajo de diploma que se presenta a continuación fue necesario realizar una búsqueda minuciosa de las tendencias actuales y de tecnologías de punta que son utilizadas a nivel mundial, para de esta manera lograr una adecuada velocidad de respuesta en el envío y recibo de mensajes y notificaciones, así como disminuir la diversidad tecnológica entre todos los productos desarrollados en el centro FORTES con el fin comunicacional. Fue necesario además, una investigación profunda del tipo de metodología a utilizar, debido a las características particulares que presenta el componente para su elaboración. Se llevaron a cabo pruebas de calidad, generadas a lo largo todo el ciclo de desarrollo de la aplicación, con el objetivo de garantizar el perfecto funcionamiento del sistema. Gracias a la gama de actividades desarrolladas, se obtuvo como resultado la creación de un sistema de mensajería instantánea y notificaciones con la utilización de salas grupales de chat y logrando la integración del mismo con las diferentes aplicaciones desarrolladas en el centro.

Palabras claves: componente, mensajería instantánea, notificaciones, salas grupales de chat.

Índice:

Introducción:

El mundo de las Tecnologías de la Información y las Comunicaciones, (TIC), avanza con gran rapidez y se reconoce el papel desempeñado por estas en el ámbito educativo lo que ha impulsado en gran medida el proceso de enseñanza-aprendizaje (PEA). Las TIC garantizan poner en práctica estrategias para la educación y la comunicación, para de esta manera establecer novedosas formas de enseñar y aprender haciendo uso de técnicas avanzadas respondiendo a su vez a un mundo tecnológico tan exigente y competitivo.

Uno de los resultados del desarrollo tecnológico son los Sistemas de Gestión de Aprendizaje o LMS[1] por sus siglas en inglés (*Learning Managent System*), los cuales facilitan la interacción entre los docentes y los estudiantes, dan soporte al seguimiento y evaluación, y aportan herramientas para la gestión de contenidos académicos. Su conceptualización está orientada a que estos sean fácilmente accesibles, amigables, intuitivos y flexibles, permitiendo ser usados en cualquier momento y lugar. En otras palabras, facilitan la "simulación" del modelo tradicional en el mundo virtual, por lo que también se les conoce como Entorno Virtual de Aprendizaje o VLE[2] por sus siglas en inglés (*Virtual Learning Environment*) (CAÑELLAS, 2014).

Una de las principales características de las propuestas educativas en modalidad no presencial es la interactividad, con lo cual todas aquellas instancias que ayuden a involucrar activamente al alumno en el proceso resultan beneficiosas en términos de su aprendizaje (RICCI, SANZ y DE GIUSTINI, 2005), dicha interactividad es posible gracias a los diversos mecanismos comunicacionales, como es el caso de los foros, la mensajería instantánea y las notificaciones. Se caracterizan a los foros como aplicaciones diseñadas para la comunicación y el intercambio de ideas, para aclarar dudas sobre una problemática determinada y brindan además otras posibilidades comunicativas (GOZÁLES, 2005). Los chat se definen como herramientas para la comunicación escrita entre dos personas a través de Internet, también permiten la interacción entre varios usuarios a través de los llamados chats públicos, estas son una de las aplicaciones más extendidas y de uso sencillo para la interacción entre las personas en el contexto digital (GOZÁLES, 2005). Finalmente las notificaciones se definen como la acción y efecto de dar noticia de algo

[1] LMS: Sistema para la gestión del aprendizaje.

[2] VLE: Entorno virtual de aprendizaje

con un cierto propósito. Un sistema de notificaciones se encarga de notificar de forma automática a los usuarios cuando un determinado evento ocurre. El mundo de los sistemas de notificaciones, permite que los usuarios puedan ser notificados en tiempo real y en cualquier lugar (CRUZ RUEDA, 2013).

El aprendizaje colaborativo mantiene una estrecha relación con los disímiles mecanismos de comunicación planteados con anterioridad, ya que el mismo ha permitido desplegar una gran variedad de mejoras en cuanto al proceso de enseñanza-aprendizaje, permitiendo de esta manera una mayor comunicación entre los profesores y los estudiantes, donde aspectos como la evaluación también han recibido su influencia brindándole a este proceso un alto grado de calidad. El empleo de las herramientas colaborativas permitirá digitalizar, difundir y acceder de forma rápida al conocimiento, creando un terreno fértil para el intercambio constante de todas las partes de donde llegue la información para compartirla y convertirla en saber (CASTELLANOS, MERIÑO y ESPINOSA, 2012).

Múltiples y diversos son los mecanismos que han sido implementados con este fin, como es el uso de la mensajería instantánea y las notificaciones. En la Universidad de las Ciencias Informáticas (UCI), específicamente el centro FORTES se tiene como objetivo fundamental el desarrollo de tecnologías para la formación. El mismo cuenta con una línea de producción de software denominada Ambiente Integrado de Aprendizaje, encargada de desarrollar los componentes comunes a las aplicaciones desarrolladas en el centro como la plataforma educativa(ZERA), el repositorio de objetos de aprendizaje(RHODA) y la herramienta de autores de creación de objetos de aprendizaje(CRODA). Esta línea permite utilizar principios tales como el logro de la colaboración entre los diferentes proyectos productivos además de agilizar la creación de software y la reutilización de cada uno de los procedimientos y/o metodologías que se crean o se usan en los mismos. La mayoría de los productos que surgieron antes de la creación de la línea tienen puntos en común; este es el caso de las comunicaciones y las notificaciones. La plataforma educativa ZERA utiliza mecanismos para garantizar la comunicación entre diferentes roles como es el caso de estudiante-estudiante, entre los cuales se realizan acciones como: envío de mensajes tanto a un buzón como instantáneos y notificaciones de apuntes compartidos, también se comunican los roles profesor-estudiante donde, además de enviarse mensajes como los anteriores, el profesor le asigna tareas al estudiante y una vez realizadas las mismas, el profesor las revisa y emite una evaluación de dichas tareas, notificándose estos eventos a los usuarios relacionados, y por último se encuentra la comunicación de profesor-profesor donde ejercen como acción principal las notificaciones de apuntes compartidos y el envío de mensajes. CRODA y RHODA están incluidas en el ámbito de herramientas

educativas de apoyo a los docentes, donde la comunicación establecida es de profesor-profesor, con acciones como: notificaciones de las acciones realizadas sobre los objetos de aprendizaje y el envío de mensajes. Las herramientas mencionadas con anterioridad tienen puntos comunes, como es el caso de que son desplegadas en ambientes donde los usuarios no necesariamente se encuentran geográficamente cerca, por lo que juegan un papel importante las herramientas de comunicación que acortan las distancias además de brindar mecanismos para notificar eventos o acciones desarrolladas por otros usuarios.

Actualmente existe una herramienta para la mensajería instantánea y las notificaciones, desarrollado con las tecnologías siguientes:

Symfony 2.3.7: Framework PHP, utilizado en el servidor para llevar a cabo la lógica de negocio, comunicación con la base de datos del sistema y con los servicios que provee el servidor de mensajería. Ejabberd 2.1.12: Servidor de mensajería que utiliza el estándar XMPP.

Postgresql 9.2: Sistema gestor de base de datos, utilizado para guardar los datos de la aplicación y los datos del servidor de mensajería.

CRODA, RHODA y ZERA, cuentan con soluciones individuales que no se pueden reutilizar en otros sistemas, además, cuentan con técnicas que sobrecargan el servidor y no son eficientes en cuanto a la interacción instantánea entre los usuarios y el sistema. Como parte de la herramienta existe una Interfaz de Programación de Aplicaciones o API[3] por sus siglas en inglés (*Application Programming Interface*), que permite a los sistemas que la integren, realizar operaciones sobre los usuarios y las notificaciones. Este sistema presenta condiciones muy particulares de uso a la hora de utilizar la sala de chat, orientada a un entorno similar a un grupo de estudiantes con un moderador al frente. Esta herramienta no incluye la mensajería interna con buzón (no chat), común en los productos desarrollados en el centro. El servidor Ejabberd tiene varias formas de persistir los datos, tanto de forma nativa en ficheros binarios, o con sistemas gestores de bases de datos SQL como Mysql o Posrgresql. El esquema de base de datos utilizado en la versión 2.1.12 tiene algunas deficiencias en cuanto al almacenamiento de los mensajes como es el caso de que se guardan los mensajes dobles, es decir, el mismo mensaje se guarda para el usuario que lo envía, y para el usuario que lo recibe, ocupando el doble de tamaño en la base de datos,

[3] API: Interfaz de programación de aplicaciones

los mensajes se asocian a las sesiones que ha creado el usuario, lo cual dificulta el acceso a los mensajes de un usuario, y como última desventaja se tiene que se utilizan bases de datos relacionales, teniendo como inconveniente, que no se utilizan las relaciones, trayendo consigo que no se eliminen en cascada los elementos relacionados al que se desea eliminar. Todas estas deficiencias no pueden ser cambiadas debido al funcionamiento propio del servidor Ejabberd, que depende de la estructura de la base de datos. El formato utilizado por Ejabberd para el intercambio de información con los clientes es XML, siendo el mecanismo propuesto por el estándar XMPP[4] (*Extensible Messaging and Presence Protocol*) que implementa este servidor. Este formato es un lenguaje etiquetado, lo cual presupone un uso de ancho de banda adicional a los datos trasmitidos relacionado con la apertura y cierre de las etiquetas propias del lenguaje. Por otro lado, el estándar XMPP provee una amplia gama de características que, en los sistemas desarrollados en el centro, que son orientados a la educación en entornos controlados, solo se usa una pequeña parte. Esto influye negativamente en la cantidad de información transmitida entre los clientes y el servidor, enviando información que no se usa.

El componente para la mensajería necesita un conjunto de herramientas para ser puesto en funcionamiento, y sería complejo integrarlo con algunas aplicaciones desarrolladas en el centro con tecnologías diferentes y versiones más antiguas de estas, como el caso de las desarrolladas con el *framework*[5] Symfony 1.X y Node.js (servidor web orientado a eventos) en cualquiera de sus versiones. Esta diversidad tecnológica hace que sea más difícil el despliegue de una aplicación que utilice el sistema de mensajería instantánea y notificaciones.

Por todo lo anterior expuesto, se plantea como **problema de investigación:**

¿Cómo estandarizar los mecanismos y tecnologías de mensajería y notificaciones de los productos desarrollados por el Centro FORTES garantizando una adecuada velocidad de respuesta y facilitando las labores de mantenimiento y construcción de dichos productos?

El **objeto de estudio** se enmarca en los componentes para la mensajería instantánea y las notificaciones, teniendo como **campo de acción** las comunicaciones basadas en la mensajería instantánea y las notificaciones en el ámbito educativo para la web.

[4] XMPP: Extensible de mensajería y Presencia de Protocolo.

[5] Framework: Marco de trabajo.

Para resolver el problema de la investigación planteado con anterioridad se expone como **objetivo general** del presente trabajo:

Desarrollar un componente estándar que unifique las tecnologías de mensajería y notificaciones garantizando una adecuada velocidad de respuesta y pueda ser reutilizado por las diferentes aplicaciones elaboradas en el centro FORTES.

Del objetivo general se derivan los siguientes **objetivos específicos**:

- ✓ Elaborar un análisis del estado del arte acerca de las tendencias tecnológicas actuales y estrategias utilizadas para la mensajería y notificaciones.
- ✓ Realizar el análisis y diseño del componente a partir de los resultados obtenidos en la fase de requerimientos.
- ✓ Implementar el componente diseñado de acuerdo a la estructura de diseño definida.
- ✓ Ejecutar pruebas de calidad a lo largo de todo el ciclo de desarrollo.

Se define como **hipótesis de la investigación:** La implementación de un componente para la mensajería instantánea y las notificaciones de los productos desarrollados por el centro FORTES, contribuirá a facilitar labores de mantenimiento y construcción de dichos productos garantizando una adecuada velocidad de respuesta.

Las **tareas identificadas** para dar cumplimiento al objetivo propuesto fueron las siguientes:

- ✓ Realizar un estudio de sistemas webs existentes para las comunicaciones instantáneas.
- ✓ Investigar sobre los diferentes *frameworks* de desarrollo, *plugins* y/o componentes con vista al desarrollo de elementos interactivos.
- ✓ Estudio de protocolos utilizados a nivel mundial para la mensajería instantánea.
- ✓ Análisis e investigación de metodologías de desarrollo de software aplicables a la solución.

Para el desarrollo de la investigación se hace necesario un estudio de los **métodos científicos**, para de esta forma demostrar la veracidad científica de las respuestas planteadas, es por ello que se definen los siguientes:

Métodos teóricos:

- ✓ **Análisis histórico – lógico:** Con la utilización de este método será posible conocer la evolución de la mensajería instantánea y las notificaciones, su desempeño y los resultados obtenidos en el ámbito educativo. Con la identificación de su necesidad y de sus propósitos se podrá indagar en sus tendencias actuales, de tal manera que contribuya esta investigación a la solución informática que se pretende lograr en este trabajo.
- ✓ **Analítico – sintético:** A través de este método se hace necesario llevar a cabo una búsqueda de documentos e información necesaria que fundamente el contenido teórico de la investigación. Dicha búsqueda se sustenta basada en la amplia gama bibliográfica destinada a la mensajería, que brinda la posibilidad de la realización de una búsqueda más profunda y mejor argumentada.
- ✓ **Inductivo – deductivo:** El uso de este método teórico tiene como principal objetivo la observación y la extracción de conclusiones que posibilita el análisis y la clasificación de los hechos, lo que permite identificar determinadas características de los sistemas comunicacionales así como el almacenamiento de evidencias que puedan ser reutilizables, siendo utilizado esto en la implementación y desarrollo de nuevas funcionalidades.

Métodos empíricos:

- ✓ **Modelación:** El método posibilita la modelación de los diferentes procesos por los que debe transitar el sistema que se desea conformar, a través de las problemáticas planteadas con anterioridad.
- ✓ **Observación:** Mediante la utilización de este método es posible la observación detallada de sistemas similares al que se pretende desarrollar, tomando como objetivo fundamental la captación y el análisis de elementos que puedan ayudar para la realización del sistema de mensajería que se desea desarrollar.

Resultados esperados:

Con el desarrollo del presente trabajo de diploma, se pretende lograr:

Un componente de software reutilizable que permita dotar a las aplicaciones desarrolladas en el centro FORTES de mecanismos de mensajería y notificaciones, así como un sistema de gestión para realizar operaciones de configuración en el componente. Este componente permitirá estandarizar las diferentes

soluciones ofrecidas en la actualidad en futuras versiones de estos productos así como la adición de características nuevas como salas grupales de Chat.

El presente trabajo de diploma cuenta con **tres capítulos**, los cuales están fundamentados a partir de la investigación llevada a cabo para la realización del mismo.

Capítulo I: Fundamentación teórica.

En el capítulo uno se abordan temas fundamentales para la realización de una fundamentación teórica profunda en la cual se hacen alusión a temas importantes como es el caso del estudio del estado del arte, donde es indispensable conocer el uso de la mensajería y las notificaciones a nivel mundial. A partir de este capítulo se debe tener una base introductoria de la necesidad de realización de la presente investigación. También se hace alusión del estado del arte que sustenta esta investigación así como el marco teórico referencial del trabajo. Se seleccionará una metodología que responda a las necesidades y características de la problemática planteada, también se realizará un análisis de las disímiles herramientas y tecnologías que se utilizan a nivel mundial para a partir de dicho análisis elaborar un criterio único permitiendo de esta manera que el sistema que se implemente esté acorde con las tendencias actuales.

Capítulo II: Presentación de la propuesta de solución.

En el capítulo número dos, se plasman tareas tales como la confección de un modelo conceptual que contiene los principales conceptos identificados para la realización del componente. Se realiza una descripción general del sistema, donde se tratarán además las características particulares del mismo. Posteriormente se muestra la lista de reserva, detallando en la misma los requisitos definidos por el cliente. Luego de la realización de estas tareas se procede a la confección de un plan de estimación de esfuerzo y de tiempo, dando paso a las historias de usuario que están organizadas por iteraciones, de las cuales se diseñará un plan de entregas basado en la duración de las mismas.

Capítulo III: Implementación y prueba.

En el capítulo tres se procede a la confección de las tareas de implementación, garantizándose el propósito para el cual se realizó la presente investigación como es la integración con otros sistemas, cerciorándose a su vez la seguridad de la aplicación. Para comprobar el correcto funcionamiento de cada requisito definido se trazan estrategias de prueba, que arrojan resultados significativos y de gran importancia.

Capítulo I: Fundamentación teórica de la Investigación.

Actualmente se utilizan como parte del avance científico-tecnológico diversos dispositivos que se integran al proceso educativo, para a través de los mismos establecer una comunicación más rápida y segura entre los usuario que utilicen este servicio. Las plataformas educativas tienen entre sus múltiples características, brindar soluciones que propicien una comunicación entre docentes y estudiantes, conformando herramientas que hacen posible que exista un aprendizaje colaborativo, un ejemplo de ello es la mensajería instantánea y las notificaciones que cobra gran auge en estos tiempos.

Para un mayor entendimiento de las principales definiciones que abarca este abanico tecnológico es necesario realizar un estudio que permita indagar en las principales características y funcionalidades de dichos conceptos. A continuación se listan los conceptos más importantes y a tener en cuenta en la presente investigación.

1.1. Definición de los principales conceptos asociados al problema

1.1.1. Comunicación síncrona y asíncrona

El acto de comunicar radica en la interacción entre dos o más personas que realizan el intercambio de mensajes, utilizando un canal que sirve de soporte en la transmisión de la información. Todas las formas de comunicación requieren un emisor, un mensaje y un receptor (DRAE, 2014).

Para el campo de la informática, se referenciaron las siguientes definiciones de los conceptos de comunicación sincrónica y asincrónica.

“La comunicación sincrónica es el intercambio instantáneo de información por Internet. Es un concepto que se enmarca dentro de la comunicación mediada por computadora, que es aquel tipo de comunicación que se da entre personas y que está mediatizada por ordenadores (CONDE, 2014).”

“La comunicación asincrónica es aquella que se establece entre dos o más personas de manera diferida en el tiempo, es decir, cuando no existe coincidencia temporal. Un ejemplo antiquísimo de comunicación asincrónica es la carta de papel. Actualmente los mails o correos electrónicos y los foros son un buen ejemplo del uso de este tipo de comunicación, desarrollada mediante ordenadores o computadores (CONDE, 2014).”

Desde el punto de vista educacional se pueden enfocar la comunicación sincrónica y asincrónica de la siguiente manera:

"...un sistema sincrónico es aquel que ofrece comunicación instantánea entre los estudiantes y los tutores. Por ejemplo, las charlas o las videoconferencias (MELIÁN, 2014)."

"Los sistemas asincrónicos ofrecen como ventaja el registro de las discusiones y aportes de los participantes, posibilitando el estudio con detenimiento del contenido antes de ofrecer su aporte o respuesta (MELIÁN, 2014)."

Partiendo de lo anteriormente expuesto, para el presente trabajo de diploma se utilizarán ambas comunicaciones. El servidor de mensajería instantánea a utilizar debe permitir establecer una comunicación entre los usuarios conectados simultáneamente, así como mostrar los mensajes no vistos cuando el usuario se conecte, guardando todos estos en el historial de conversaciones.

1.1.2. Notificaciones de eventos

Según el Diccionario de la Real Academia de la Lengua Española (DRAE, 2014) se define a las notificaciones como: *"la acción y efecto de notificar*", mientras que los eventos provienen del latín "*eventus"* y en varios países latinoamericanos se define como un suceso de importancia que se encuentra programado.

En el ámbito informático una aplicación de *Notifications Services* (aplicaciones de notificaciones) genera y envía mensajes a usuarios y otras aplicaciones que se hayan suscrito a la aplicación (MICROSOFT, 2005), además de que aparecen relacionadas a alertas que emiten algunos programas con el objetivo de advertir al usuario de un suceso. Con el avance de la ciencia y la tecnología es posible observar un amplio crecimiento de las notificaciones electrónicas no solo en ordenadores, también son muy utilizadas en diversos dispositivos ya sea en teléfonos móviles y en *tablets*[6]. Dichos dispositivos se basan en las notificaciones para mantener a sus usuarios al tanto de diferentes acontecimientos que consideran de interés, de este modo muchas compañías aprovechan estas facilidades para hacer llegar a sus seguidores noticias acerca de eventos, lanzamientos de productos, entre otros, lo que constituye una amplia gama de oportunidades para hacer publicidad.

[6] Tablets: Computadora portátil de mayor tamaño que un teléfono.

En la educación las notificaciones son muy utilizadas, ya que permite enviar mensajes y documentos a los usuarios que participen en un curso determinado. Los mensajes que son enviados entre los usuarios funcionan similar a los correos electrónicos, donde los mensajes que son enviados perduran en el tiempo (ORTÍ, 2012).

Para el desarrollo de la investigación se toma entre los conceptos antes planteados al que profundiza en el aspecto informático con alcance educativo planteado por Consuelo Ortí (ORTÍ, 2012) en un artículo publicado para la Universidad de Valencia, ya que el mismo recoge con gran profundidad las particularidades de las notificaciones para entornos educacionales.

1.1.3. Mensajería instantánea

La palabra mensaje proviene del provenzal "*massatge"* lo que significa "*recado que envía alguien a otra persona*", mientas que instantánea se define como "*un suceso que se produce inmediatamente"* (DRAE, 2014).

Esta tecnología permite el intercambio de mensajes directos en tiempo real entre uno o más interlocutores a través de Internet, el mismo hereda un gran número de funcionalidades del chat, pero a diferencia de este, se suele tratar al mismo como un programa independiente que ofrece otras posibilidades añadidas a la mera transmisión de mensajes de texto, como es el caso del intercambio de archivos, video-llamadas, llamadas de voz, entre otros (ROIG, 2012).

Otros autotores como Jean-Noël Anderruthy definen a este tipo de comunicación como: "…programa que permite dialogar en directo con un interlocutor distante mediante teclado, micrófono o teléfono".

Actualmente el concepto de mensaje está fuertemente ligado a los programas de mensajería instantánea y los servicios de correo electrónico. Existen una gran cantidad de aplicaciones que utilizan esta nueva técnica, donde las personas tienen la posibilidad de hablar desde cualquier lugar del mundo y en todo momento, permitiendo de esta manera un acortamiento de distancias y la poca inversión de tiempo a la hora de comunicarse.

En la mensajería instantánea la comunicación suele ser síncrona, es decir, los usuarios que van a mantener una conversación deben estar conectados al servicio en el mismo instante de tiempo y mientras dure la comunicación. Por otro lado existen otros clientes de mensajería que permiten comunicarse de forma asíncrona dejando los mensajes en una especie de "contestador" (RODIL, 2010).

La mensajería instantánea en el aspecto educativo permite enviar mensajes cortos a los participantes de un curso al que se esté suscrito. La misma puede ser utilizada tanto por docentes como por alumnos. Los mensajes son una vía de comunicación rápida, eficaz y habitual utilizados para realizar una notificación concreta o dar respuesta a una cuestión puntual (ORTÍ, 2012).

De las definiciones analizadas se toma el planteamiento realizado por Consuelo Ortí (ORTÍ, 2012), ya que es la que más se ajusta a las necesidades reales de la presente investigación.

1.1.4. Salas de chat

La palabra chat provine del inglés chat, la cual se traduce como charla, tiene como significado: "*intercambio de mensajes electrónicos a través de Internet que permite establecer una conversación entre dos o varias personas*" (DRAE, 2014).

Las salas de chat o chat-*rooms* como también son conocidas en el idioma inglés, son espacios donde la gente se reúne para charlar con otras personas que hay en la misma sala y mantener una conversación mediante el uso de mensajes electrónicos en tiempo real (FREIRÍA, 2008).

En la educación se le atribuye gran importancia a las salas de chat, ya que a través de las mismas es posible realizar actividades conjuntas entre estudiantes, discutir y analizar en forma colectiva entre el profesor y los estudiantes, efectuar preguntas colectivas, asesorar a los estudiantes y evaluar las participaciones de cada estudiante (GARCÍA, 2004).

El chat está asociado al ocio y la comunicación informal, caracterizándose por respuestas rápidas de frases cortas con estructuras gramaticales sencillas, discursos fragmentados, produciéndose en muchos casos discursos o conversaciones paralelas. Es por ello, que la clave para un desarrollo eficaz del mismo está en el moderador. La moderación de un chat es necesaria para dar los turnos de palabra y controlar que las conversaciones no se alejen del propósito del chat (ORTÍ, 2012).

Según los criterios analizados por varios autores, se escoge como concepto de sala de chat para esta investigación el planteado por Guillermo García en un artículo publicado con el título "Los chats y su uso en la educación" (GARCÍA, 2004), debido a que el mismo profundiza en aspectos significativos para el presente trabajo de diploma.

1.2. Análisis de LMS que incluyen mensajería instantánea en el mundo

Un LMS (*Learning Management System*) es un sistema de gestión de aprendizaje online, que permite administrar, distribuir, monitorear, evaluar y apoyar las diferentes actividades previamente diseñadas y programadas dentro de un proceso de formación completamente virtual (eLearning) (CAÑELLAS, 2014). Algunos de los LMS más conocidos a nivel mundial son los que se describen a continuación.

Moodle

Moodle es un Entorno Virtual de Aprendizaje (EVA), el cual permite al profesor la gestión de cursos virtuales para sus alumnos, o la utilización de un espacio en línea que brinde apoyo a la presencialidad. Esta plataforma utiliza la mensajería para dos temas: alertas automáticas para anunciar nuevas publicaciones en los foros, como puede ser que los estudiantes son informados por su tutor de la inscripción de un curso, y notificaciones de envíos de tareas, que los profesores reciben una notificación confirmando que un alumno envía una tarea asignada. Las características propias de la mensajería instantánea, como es el envío y recibo de mensajes también tiene su lugar en Moodle, ya que tanto los estudiantes como los profesores pueden establecer una conversación. (MOODLE, 2014).

Dokeos

Dokeos es un entorno de *e-learning*[7]y una aplicación de administración de contenidos de cursos y también una herramienta de colaboración. El entorno presenta como principales ventajas que es un software libre, dispone de numerosos recursos entre los que se destaca la posibilidad de realizar video-conferencias y tiene la capacidad de montar sus propias redes sociales para facilitar la comunicación entre los estudiantes. La plataforma utiliza el chat para propiciar comunicación en tiempo real entre los participantes, y la utilizacion de foros, los cuales permiten la comunicación a través de mensajes que pueden ser revisados y/o respondidos en cualquier momento, estas herramientas son síncronas y asíncronas. Dokeos presenta una sección para los anuncios, los que permiten que el profesor envíe un mensaje por email a los estudiantes y/o publicar una información importante directamente en su aula virtual. El buzón de tarea es otra de sus características distintivas, que básicamente facilita el intercambio de archivos entre alumnos y el docente del curso: El creador del curso puede enviar archivos a uno o a

[7] E-learning: Aprendizaje electrónico utilizado para la educación a distancia.

muchos estudiantes; los estudiantes pueden enviar archivos al creador del curso y también enviar archivos entre sí. Además, los archivos pueden incluir comentarios. (DOKEOS, 2014).

Claroline

Claroline es un software de código abierto para implementar fácilmente una plataforma dedicada al aprendizaje y la colaboración en línea. Dentro de sus características se encuentra que puede hacer anuncios, los cuales son una especie de tablón donde son publicados mensajes a los alumnos, dichos anuncios son organizados a través de los grupos de estudiantes definidos por el profesor. Los foros son parte también de esta plataforma, los cuales permiten crear discusiones públicas o privadas. Los debates es otra característica distintiva de Claroline, los cuales sirven como herramienta online para aclarar dudas o crear rubricas entre los estudiantes. La comunicación en esta plataforma se establece de forma síncrona y asíncrona. El almacenamiento y la gestión de chats es otro mecanismo que utiliza Claroline para establecer una comunicación entre docentes y estudiantes (CLAROLINE, 2014).

1.3. Análisis de LMS que incluyen mensajería instantánea en Cuba

En el ámbito educativo se han desarrollado un gran número de soluciones informáticas que pretender estrechar distancias con el objetivo de colaborar en el proceso de enseñanza-aprendizaje. Nuestro país no está exento de esta nueva tendencia, ya que ha desarrollado aplicaciones para colaborar con el proceso educativo, tanto dentro de nuestra isla como fuera de ella.

Sistema de mensajería utilizado en la plataforma educativa Zera

Zera cuenta con un sistema de mensajería instantánea y notificaciones para el marco de trabajo Xalix, el cual está desarrollo sobre el protocolo XMPP, utilizando Ejabberd como servidor de MI (mensajería instantánea). La aplicación incorpora entre sus funcionalidades el uso de debates colaborativos de forma simultánea en salones de charla con características enfocadas a la educación, haciendo posible la comunicación síncrona y asíncrona (AMADO y ABREU, 2014).

1.4. Análisis de sistemas para la mensajería instantánea y las notificaciones

Los clientes de mensajería instantánea permiten enviar y recibir mensajes de otros usuarios usando los mismos software clientes, sin embargo, últimamente han aparecido algunos clientes de mensajerías que ofrecen la posibilidad de conectarse a varias redes al mismo tiempo (aunque necesitan registrar usuario

distinto en cada una de ellas). Dichos clientes presentan características comunes, como es el caso del uso de avisos de presencia, indicando cuando el cliente de una persona en la lista de contactos se conecta o en qué estado se encuentra y si está disponible para tener una conversación (SAAVEDRA, 2009). Dentro de los sistemas más difundidos a nivel internacional se encuentran: Windows Live Messenger, Skype, Pidgin, Yahoo! Messenger y Facebook.

Windows Live Messenger es conocido como "El MSN", permite a los usuarios mantener conversaciones instantáneas a través de Internet, formar grupos de contactos, lo que se traduce en conversaciones simultáneas, y la aplicación de soporte de voz. Windows Live Messenger usa el protocolo abierto Jabber (SAAVEDRA, 2009).

Skype por su parte, es un sistema de telefonía y mensajería instantánea que cuenta con varios seguidores en el mundo debido a su utilidad para llamadas telefónicas entre dos equipos y las videoconferencias, sin duda dos de los aspectos más significativos que sitúan a esta solución como puntera en el mundo de las comunicaciones para equipos informáticos (CABELLO, 2013).

El cliente de mensajería instantánea multiplataforma Pidgin es capaz de conectarse a múltiples redes y cuentas de manera simultánea. Este sistema de mensajería es capaz de mostrar las conversaciones en pestañas, además puede realizar un historial o registro de las conversaciones en caso de requerirse una auditoría. Por otra parte esta herramienta permite el reemplazo de los nombres de los contactos de la lista y muestra un aviso o reproduce un sonido cuando un contacto se conecta/desconecta o cambia de estado, posibilita la transferencia de archivos y brinda soporte para *WebCam*[8]. Otro aspecto que lo hace distinguible es la gran cantidad de protocolos que soporta, algunos de ellos son AOL, *Instant Messenger*, *Bonjour*, *Gadu-Gadu*, *Groupwise Novell Messenger*, ICQ, XMPP/Jabber (*Gtalk*, Facebook entre otros), Yahoo! y Zephyr (SAINT-ANDRE, SMITH, y TRONCON, 2009).

Yahoo! Messenger es una aplicación de mensajería instantánea y su protocolo es desarrollado por el propio Yahoo. Dentro de sus principales características están el envío de mensajes de texto, notificaciones de eventos, transferencia de archivos y correos de voz, salas de chat, posibilidad de *plug-ins*, integración con el servicio de correo de Yahoo! y de otros servicios de la empresa, también es

[8] WebCam: Cámaras web o cámara de red.

distinguida esta aplicación por la interoperabilidad con otros mensajeros y los juegos online. Presenta además la comunicación síncrona y asíncrona entre sus ventajas (HERNÁNDEZ, 2014).

Por otro lado Facebook es una red social que permite conectar a las personas en internet. Presenta miles de seguidores por todo el mundo, haciéndose reconocer como modelo a seguir en cuanto a funcionalidad y diseño, por otra parte el chat de Facebook, además de permitir el intercambio instantáneo de mensajes entre los usuarios conectados, se encuentra estructurado de forma tal que no afecte la información mostrada en la página visitada, aspecto fundamental en los chats incluidos en sistemas web. Las notificaciones del Facebook pueden mostrarse como avisos que llegan a la cuenta de correo del usuario implicado, cada vez que surge un evento en la red social, algunos de estos eventos son la realización de un comentario acerca de una fotografía donde se menciona a un usuario determinado el cual será notificado mediante un correo (BROWN, 2013).

1.5. Protocolos más utilizados para la mensajería

Actualmente existe una gran diversidad de protocolos para la mensajería instantánea, de los cuales los más difundidos son: XMPP, Skype, Sip, ICR, Oscar, y la nueva tecnología WebSockets. A continuación se detallan las características más relevantes de los mismos.

XMPP (*Extensible Messaging and Presence Protocol*[9]) se describe como un protocolo libre y abierto de comunicación en tiempo real. Presenta una arquitectura descentralizada, por lo que puede usar un servidor propio, permitiendo a las organizaciones tener control de su experiencia de comunicaciones. Las aplicaciones originales de XMPP (de mensajería y presencia) se han extendido y ahora pueden encontrarse en administración de redes, sindicalización de contenidos, herramientas de colaboración, compartimiento de archivos, juegos, monitoreo de sistemas remotos, servicios web, computación en la nube, etc. Contiene como programas clientes a aplicaciones tales como Spark, Pidgin, MCabber y Talkonaut. XMPP tiene como desventajas la sobrecarga de datos de presencia, ya que cerca de un 70% se utiliza para el tráfico entre servidores con datos de presencia, y cerca de un 60% de estos son transmisiones redundantes. También se identifica como desventaja la escalabilidad, pues sufre el mismo problema de redundancia en los servicios de chatroom y de suscripción (XMPP, 2015).

[9] Extensible Messaging and Presence Protocol: Extensible de mensajería y Protocolo de Presencia

El código y protocolo de Skype permanecen cerrados y propietarios, esta aplicación incluye una característica denominada YY SkypeOut, que permite a los usuarios llamar a teléfonos convencionales, cobrándoseles diversas y bajas tarifas según el país de destino, pudiendo llamar a casi cualquier teléfono del mundo (JABBER, 2010).

SIP (*Session Initiation Protocol*[10]) por su parte, permite el control y señalización, mayoritariamente usado en los sistemas de telefonía IP. Entre las funciones que SIP posee se tienen las siguientes: localización de usuarios (proporciona soporte para la movilidad), disponibilidad del usuario, establecimiento y mantenimiento de una sesión, además de la autenticación y encriptación que son soportados por SSL/TLS (JABBER, 2010).

IRC (*Internet Relay Chat*) se describe como un protocolo de comunicación en tiempo real basado en texto, que permite la conferencia entre 2 o más personas y que está clasificado dentro de la mensajería instantánea. Las conversaciones se desarrollan en los llamados canales de IRC, que pueden ser locales al servidor al que se conectan los clientes o no. Los usuarios del IRC utilizan una aplicación cliente para conectarse con un servidor en el que funciona una aplicación INCD (IRC *Daemon* o servidor IRC), que gestiona los canales y las conversaciones (JABBER, 2010).

OSCAR (*Open System for Communication in Realtime*[11]) es el protocolo oficial del programa de mensajería instantánea de AOL y AIM. Es un desarrollo propietario y no ofrece ninguna documentación ni da acceso a sus fuentes. Es por ello que muchos diseñadores de programa con soporte para múltiples plataformas de mensajería han tenido que recurrir a la ingeniería inversa para conocer su forma de actuar y adaptar sus programas para hacerlos compatibles con AIM. Este protocolo utiliza paquetes binarios de longitud variable, de forma que permite una amplia variedad de servicios (chat, directorio, gestión, localización, etc.). Los clientes no se conectan directamente entre sí, lo hacen a través de servidores, que se responsabilizan de la entrega de los mensajes a sus destinatarios (JABBER, 2010).

WebSockets es una tecnología que hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor. Una vez que se establece una conexión WebSocket la conexión permanece abierta hasta que el cliente o el servidor, decide cerrar esta conexión. Esto significa que

[10] SIP: Protocolo de Iniciación de Sesión.

[11] OSCAR: Sistema abierto para la comunicación en tiempo real.

cuando el servidor quiere enviar datos al servidor, el mensaje se traslada inmediatamente. Con esta conexión abierta, el cliente o el servidor puede enviar un mensaje en un momento dado a la otra. Esto hace que la programación web en su totalidad sea realizada por eventos (MOZILA, 2011). Socket.io, por su parte, es una librería que permite manejar eventos en tiempo real mediante una conexión TCP y todo ello en JavaScript. Socket.IO proporciona comunicación en tiempo real entre su servidor Node.js y los clientes con WebSockets. Es realmente potente y se puede hacer todo tipo de aplicaciones en tiempo real empleando esta librería (MICROSOFT, 2015).

Para la confección del componente, después de realizar una investigación y análisis de protocolos, se empleará la tecnología WebScket con Socket.io, ya que los mismos se integran fácilmente para el manejo de comunicaciones en tiempo real.

1.6. Características presentes en los sistemas analizados

Tabla1: Análisis de características de los sistemas

Características	Moodle	Dokeos	Claroline	Zera	Windows Live Messenger	Skype	Pidgin	Yahoo! Messenger	Facebook
Comunicación síncrona		x	x	x	x	x	x	x	x
Comunicación asíncrona	x	x	x	x	x	x	x	x	x
Notificaciones	x	x	x	x	x	x	x	x	x
Mensajería instantánea	x	x	x	x	x	x	x	x	x
Salas de chat	x		x	x	x	x	x	x	x
Foros	x	x	x						

Video-conferencias		x			x	x			
Soporte de voz					x	x		x	
Transferencia de archivos		x			x		x	x	x
Conexión con otros servicios					x		x	x	x

Después de haber realizado un análisis de cada uno de los LMS y aplicaciones disponibles en la web para establecer comunicaciones, se procede a seleccionar las características que formarán parte de la aplicación para la mensajería instantánea a desarrollar. Para la selección se tuvo en cuenta que las funcionalidades deben cumplir con las características y requisitos propuestas por el cliente. Las características son:

- ✓ Comunicación síncrona
- ✓ Comunicación asíncrona
- ✓ Notificaciones
- ✓ Mensajería instantánea
- ✓ Salas de chat

1.7. Ambiente de desarrollo

Con la selección y utilización correcta de una metodología de desarrollo que se adecue a las necesidades que requiere la aplicación que se desea implementar, se garantiza la reutilización de procesos, mayor comunicación y organización del equipo de desarrollo, además de garantizar una mayor calidad en el producto final. Para lograr la calidad mencionada con anterioridad es necesario además la selección de herramientas y tecnologías que apoyen el proceso de construcción del software, dichas tecnologías deben ajustarse a un mundo tecnológicamente cambiante como el que se vive en la actualidad. En el centro FORTES específicamente en la Línea de Ambiente Integrado de Aprendizaje se tiene la premisa de

desarrollo de componentes que se integren en los productos desarrollados en el centro, para ello se toma como una necesidad la disminución de la diversidad tecnológica de las soluciones informáticas que se implementen. Por lo antes expuesto se impone una correcta selección de tres elementos: metodología, herramienta y tecnología, debido a que la herramienta a desarrollarse pertenece a dicha línea, se acoge a las especificidades de la misma, no significando esto que se utilicen otras tecnologías que optimicen y agilicen el desarrollo de la aplicación.

1.7.1. Metodología de desarrollo de software

La Ingeniería de Software es una disciplina o área de la Informática o Ciencias de la Computación, que ofrece métodos y técnicas para desarrollar y mantener software de calidad (PRESSMAN, 2005), dicho planteamiento refiere la importancia que se le acredita a la ingeniería de software como proceso de modelado y diseñado para entender las etapas de desarrollo y la organización que debe seguir el software para su correcta elaboración. Para la confección de un software con alto grado de calidad es necesario regirse por una metodología, la cual está compuesta por una serie de pasos, técnicas y procedimientos que garantizan con su correcta utilización la evolución de la solución informática que se desee implementar. El proceso de selección y clasificación de una metodología para la construcción de un producto no es tarea sencilla producto a la gran diversidad de propuestas que existen actualmente, de su correcta selección depende el éxito o el fracaso de un proyecto. Para la selección adecuada de una metodología que respalde esta investigación se realizó un análisis entre las metodologías ágiles y las tradicionales.

Las metodologías tradicionales son definidas como robustas o pesadas y se hace muy difícil aplicarlas en un entorno cambiante. Este tipo de metodología es muy rígida debido a la extensa documentación que la acompaña, por lo que no resulta ser la vía más factible para muchos proyectos con entornos cambiantes y están guiadas por una potente planificación acompañada durante todo el ciclo de desarrollo por los que atraviesa el software. Se destaca además que el modelado es imprescindible dando lugar a la elaboración de gran cantidad de artefactos, existe una amplia gama de roles y se espera que no ocurran cambios de gran impacto durante el proyecto. En la actualidad, en escenarios donde los requisitos cambian habitualmente se hace necesario una metodología que sea más ligera y versátil (PRESSMAN, 2005).

Las metodologías ágiles se caracterizan por ser más flexibles, además de que es altamente adaptable a los cambios que puedan efectuarse en el trayecto del desarrollo del producto de software. Cuentan

además con la virtud de ser orientadas a proyectos pequeños, dando lugar a la existencia de pocos roles que sean más pequeños y flexibles, las mismas están orientadas a la generación de código con ciclos muy cortos de desarrollo pero sin obviar los procesos esenciales que hacen posible la calidad requerida para obtener el producto final y tienen la ventaja de involucrar directamente el cliente con el equipo de desarrollo trabajando en conjunto y estableciendo una estrecha comunicación. Se destaca que se generan pocos artefactos a través de ella y el modelado es prescindible, además de que con este patrón metodológico se esperan cambios durante el proyecto (GUPO ISSI, 2003).

Un resumen más sencillo de la comparación que se establece entre las dos metodologías antes mencionadas, se muestra en el **Anexo 1: Comparación de metodologías**, tomado del libro "Ingeniería de software un enfoque práctico" (PRESSMAN, 2005)

Por las ventajas planteadas con anterioridad y el acercamiento de sus características a los requisitos que presenta con la presente investigación, se seleccionó para la composición de este trabajo una metodología ágil.

Cada metodología tiene características propias, es por ello que se hace necesario realizar una investigación dentro de las mismas para seleccionar cuál es la más factible y la que mejor se ajusta al sistema que se implementará.

Extreme Programming (XP) se define como metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Surge como respuesta y posible solución a problemas derivados del cambio en los requerimientos (ESCRIBANO, 2002). XP está diseñada para entornos dinámicos específicamente pensada para equipos pequeños y pone énfasis en la comunicación informal. Basado en lo anterior su creador *Kent Beck expresó "Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar".*

SCRUM es una metodología que está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos: el desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días, donde el resultado de cada sprint es un incremento ejecutable que se muestra al cliente, la segunda característica importante son las

reuniones a lo largo proyecto. Las reuniones son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (LETELIER y PENADÉS, 2006).

CRYSTAL METHODOLOGIES se basa de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros) (HIGHSMITH J., 2000).

Dynamic Systems Development Method (DSDM) define el marco para desarrollar un proceso de producción de software. Nace con el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos, propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases (JEFFRIES, 2001).

Adaptive Software Development (ASD) presenta entre sus principales características que es iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo (POPPENDIECK M., 2003).

Feature -Driven Development (FDD) es una metodología que define un proceso iterativo que consta de cinco pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software (SCHWABER K., 2001).

En la metodología Lean Development (LD), los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios (STAPLETON J., 1997).

En la tabla que se muestra en el **Anexo 2: Ranking de agilidad**, tomado del sitio web crystalmethodologies.org (HIGHSMITH J., 2000). Se comparan las distintas metodologías en cuanto a sus características, donde se arrojan resultados que demuestran que Scrum y XP son las más ágiles.

1.7.2. Selección de la metodología

De las metodologías antes analizadas se puede concluir que en todas se refleja que son menos orientadas al documento, exigiendo una cantidad más pequeña ya que las mismas son más orientadas al código. XP destaca entre las metodologías ágiles mencionadas debido a su flexibilidad, su simplicidad y la comunicación, una de las características más sobresaliente de la misma es su capacidad para manejar cambios que se puedan dar con el avance del proyecto además de su agilidad en la conformación de los procesos y el equipo de desarrollo debe ser pequeño. El producto que se desea conformar debe tener entre sus principales aspectos el contacto directo con el cliente y la estrecha comunicación que se debe establecer con el mismo, el equipo de desarrollo es pequeño y sobresalta además los cambios que puedan ocurrir en trayecto de la conformación de la solución. Por lo antes planteado se selecciona la metodología XP para el desarrollo de esta investigación.

XP presenta un ciclo de vida iterativo, está centrado en la arquitectura y posee cuatro variables fundamentales: Coste, Tiempo, Calidad y Alcance. Surge como respuesta y posible solución a los problemas derivados al cambio en los requerimientos, aumentando de esta manera la productividad (ESCRIBANO, 2002) y básicamente está compuesta por cuatro fases fundamentales **Anexo 3: Fases de la metodología XP**, tomado del artículo “Introducción a Extreme Programming” (ESCRIBANO, 2002).

1.7.3. Lenguaje de modelado

Como lenguaje de modelado para la realización de la solución informática se utilizará al Lenguaje de Modelado Unificado (UML). UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre un sistema de software. Este lenguaje

pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar (RUMBAUGH, JACOBSON y BOOCH, 1998).

Según Erwin Schrödinger: *"Si no se puede explicar lo que se hace, el trabajo carece de valor"*, de esta base parte la necesidad de la modelación para explicar los diferentes procesos por los que atraviesa un sistema. El UML necesita ser lo suficientemente expresivo para manejar todos los conceptos que se originan, tales como la concurrencia y distribución, así como los mecanismos de ingeniería de software (RUMBAUGH, JACOBSON y BOOCH, 1998).

A partir de los beneficios que brinda la herramienta, es posible una visualización de los contenidos, teniendo en cuenta los diferentes métodos por los que debe transitar la aplicación para su edificación. La principal ventaja de su utilización es la sencillez de la representación simplificada de procesos de software.

1.7.4. Herramientas para el modelado

Para la visualización grafica de diseños que respondan con las características y requisitos que debe cumplir un software para su elaboración, es necesario realizar una correcta selección de una herramienta para el modelado. El objetivo de un modelado es capturar de la mejor forma posible las partes esenciales que componen al sistema que se desea conformar, para ello se necesita una herramienta que responda a las necesidades del proyecto. Las herramientas soportan los procesos de desarrollo de software modernos. Hoy, es impensable desarrollar software sin utilizar un proceso soportado por herramientas (JACOBSON y JACOBSON, 1995). El empleo de las herramientas proporciona una serie de ventajas como la automatización de procesos repetitivos, el mantenimiento de elementos estructurados y la gestión de grandes cantidades de información. Es esencial que las mismas gestionen la información representada por la serie de modelos y artefactos, fomentando y soportando las actividades creativas que son el núcleo fundamental del desarrollo (RUMBAUGH, JACOBSON y BOOCH, 1999).

Las herramientas CASE (Ingeniería Asistida por Computadoras) se destinan a automatizar los aspectos claves de todo el proceso de desarrollo de un sistema. Existen cuatro razones para adoptar las herramientas CASE: incrementar la productividad del analista, mejorar la comunicación entre analistas y usuarios, integrar las actividades del ciclo de vida y analizar y valorar el impacto de los cambios en el mantenimiento (KENDALL y KENDALL, 2005).

Visual Paradigm For UML es una Herramienta Case caracterizada por ser una de las herramientas que soporta las últimas versiones del UML y el modelado de procesos de negocios. La herramienta provee un generador de mapeo de objetos relacionales para los lenguajes de programación Jaba, PHP, y .NET. *Visual Paradigm* está disponible en varias ediciones, cada una destinada a las necesidades del usuario. Una de las características más importantes de este producto es que facilita a las organizaciones la diagramación visual y el diseño de sus proyectos de sistema, que les brinda la posibilidad de integrar y desplegar sus aplicaciones empresariales de misión crítica y de sus bases de datos subyacentes. Esta herramienta ayuda a los equipos de desarrollo de software para sobresalir todo el modelo de acumulación de trabajo y desplegar el proceso de desarrollo de software, lo que permite maximizar y acelerar tanto las contribuciones individuales como las de equipo de desarrollo (CRUZ, 2015).

1.7.5. Tecnologías del lado del cliente

Las tecnologías del lado del cliente son aquellas que se ejecutan en el navegador del usuario, cargando páginas dinámicas que se procesan en el cliente. El código necesario para crear los efectos y funcionalidades se incluye dentro del archivo HTML, y CSS se aplica para para la apariencia de la página.

HTML es la abreviatura de *Hyper Text Markup Language*, y es el lenguaje que todos los programas navegadores usan para presentar información en la *World Wide Web* (WWW). Este es un lenguaje muy sencillo que se basa en el uso de etiquetas, encerrado texto dentro de un par de paréntesis angulares (<>).Para crear un documento HTML se necesita un procesador de textos para escribir y editar el código HTML. Este podrá ser cualquiera que no formatee el texto, ya que el leguaje HTML está basado en el código ASCII (acrónimo inglés *de American Standard Code for Information Interchange* lo que en español significa Código Estándar Estadounidense para el Intercambio de Información) (HOGAN, 2011).

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función

de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, entre otras configuraciones (HOGAN, 2011).

1.7.6. Frameworks del lado del cliente

Los *frameworks* del lado del cliente utilizados para el desarrollo de la investigación son JQuery, Angular y Boostrap, de los cuales a continuación se muestra una breve introducción.

JQuery es un *framework* Javascript, o lo que viene siendo un producto que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Los *frameworks* son unas librerías de código que contienen procesos o rutinas ya listos para usar. Los programadores los utilizan para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar. JQuery implementa una serie de librerías que nos permiten programar sin preocuparnos del navegador con el que nos está visitando el usuario, ya que funcionan de exacta forma en todas las plataformas más habituales (ÁLVAREZ, 2009).

AngularJS es un proyecto de código abierto, realizado en JavaScript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo como es el MVC o Modelo Vista Controlador según sus siglas. En pocas palabras, es lo que se conoce como un *framework* para el desarrollo, en este caso sobre el lenguaje JavaScript con programación del lado del cliente. Este JavaScript pretende que los programadores mejoren su código HTML, ya que se puede producir un HTML altamente semántico, es decir, que cuando sea leído se entienda de manera clara qué es lo que hace o para qué sirve cada cosa. Lógicamente, AngularJS viene cargado con todas las herramientas que los creadores ofrecen para que los desarrolladores sean capaces de crear un HTML enriquecido (AZAUSTRE, 2013).

Bootstrap, es un framework originalmente creado por Twitter, que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Esta técnica de diseño y desarrollo se conoce como "*responsive design*" o diseño adaptativo. Los diseños creados con Bootstrap son simples, limpios e intuitivos, proporcionando agilidad a la hora de

cargar y al adaptarse a otros dispositivos. Trae consigo varios elementos con estilos predefinidos fáciles de configurar como: botones, menús desplegables, formularios, incluyendo todos sus elementos e integración con JQuery para ofrecer ventanas y herramientas para eventos y efectos que le brinden dinamismo al proyecto (MONTERO, 2013).

1.7.7. Lenguajes de desarrollo de software del lado del cliente

Los *frameworks* contienen un conjunto de librerías de código y módulos ya listos que resumen las tareas de creación de elementos recurrentes en el desarrollo de aplicaciones, a la vez que definen una arquitectura para el desarrollo de software. JavaScript es el lenguaje escogido para el desarrollo del lado del cliente, el mismo es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (TRUJILLO y RODRÍGUEZ, 2011).

1.7.8. Tecnologías del lado del servidor

Los lenguajes del lado servidor son aquellos que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Para ello se utilizan tecnologías que garantizan esta interacción, como es el caso de Node.js.

Node.js es un entorno de programación en la capa del servidor basado en el lenguaje de programación ECMA-Script asíncrono. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web. Es similar en su propósito a Twisted o Tornado de Python, Perl Object Environment de Perl, React de PHP, libevent o libev de C, EventMachine de Ruby, vibe.d de D y de Java existe Apache MINA, Netty, Akka, Vert.x, Grizzly o Xsocket. Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor. Implementa algunas especificaciones de CommonJS, así como la incorporación de varios módulos básicos compilados en el propio binario, como por ejemplo el módulo de red, que proporciona una capa para programación de red asíncrona y otros módulos fundamentales, como por ejemplo Path, FileSystem, Buffer, Timers y el de propósito más general Stream. Es posible utilizar módulos desarrollados por terceros, ya sea como

archivos node precompilados, o como archivos en JavaScript plano. Los módulos JavaScript se implementan siguiendo la especificación CommonJS para módulos, utilizando una variable de exportación para dar a estos scripts acceso a funciones y variables implementadas por los módulos (KIESSLING y JUNGE, 2015).

1.7.9. Frameworks del lado del servidor

Los *frameworks* del lado del servidor analizados son los más utilizados por Node.js, los cuales se describen a continuación:

Express, es un entorno de programación en la capa del servidor basado en el lenguaje de programación Javascript. Express.js, según sus creadores, es un *framework* de desarrollo de aplicaciones web minimalista y flexible para Node.js. Está inspirado en Sinatra, además es robusto, rápido, flexible y muy simple. Entre otras características, ofrece *router* de URL (Get, Post, Put) y facilidades para motores de plantillas (Jade, EJS, JinJS) (NÚÑEZ, 2012).

Hapi es un marco de configuración céntrico para crear aplicaciones web y API facilitando la construcción de estas en un instante. Hapi requiere como mínimo tener la versión 0.10.x de Node.js para poder crear un servidor. Se ha creado alrededor de la idea de que es mejor que la configuración de código y la lógica de negocio que debe ser aislado de la capa de transporte (HERNÁNDEZ, 2013).

Koa.js es un *framework* web de próxima generación expresivo y escrito para Node.js. Aprovecha generadores, que son una característica de borde sangrado de JavaScript, y todavía no se han hecho en las versiones estables de Node.js. Koa es un marco extremadamente ligero, utilizando generadores para hacer aplicaciones web y APIs (NPM, 2015).

1.7.10. Selección del framework del lado del servidor

Express.js es el *framework* de aplicaciones web más popular para Node.js utilizado. Es la base de la dependencia en la mayoría de las aplicaciones web para Node.js. Los siguientes dos marcos de trabajo más populares son Koa y Hapi respectivamente, en el **Anexo 4: Frameworks para Node.js**, se realiza un estudio comparativo de los mismos. El primer paso para cuando se trabaja en una aplicación web Node.js, es la creación de un servidor básico, aspecto en los que Express y Koa son muy parecidos, no siendo así con el *framework* Hapi, En cuanto al enrutamiento de ficheros, Hapi toma la ventaja, debido a que logra un trabajo limpio y legible en este sentido. Por otra parte Express es el marco más maduro de los tres

frameworks analizados, ofrece una forma sencilla de crear un servidor y promueve la reutilización de código (GLOCK, 2013).

Aunque la utilización de estos tres marcos posibilita el desarrollo de una buena aplicación en Node.js, Express es sin duda el marco más popular y el más reconocido de los tres, con la mayor comunidad de desarrollo para Node.js, siendo el más estable para el desarrollo de aplicaciones. Es por ello que se selecciona a Express como *framework* del lado del servidor para el desarrollo de la propuesta de solución en el presente trabajo.

1.7.11. Sistema Gestor de Base de Datos

Una base de datos es el conjunto de datos informativos organizados en un mismo contexto para su uso y vinculación. Un Sistema Gestor de Base de Datos es un sistema de software que permite la definición de bases de datos; a continuación s e exponen los dos sistemas más utilizados en la actualidad.

Los sistemas gestores de Base de Datos SQL como su nombre lo indica están basados en el lenguaje de ese mismo nombre SQL, el cual es un estándar para el trabajo con bases de datos relacionales. SQL incluye operaciones tanto de definición como de manipulación de datos. También soporta diversos valores predeterminados, abreviaturas y escrituras alternas, en particular podemos realizar operaciones relacionales de restringir, proyectar y juntar los datos. La mayoría de los productos SQL permiten la ejecución de instrucciones SQL de manera directa (es decir, en forma interactiva desde una terminal en línea) y también como parte de un programa de aplicación (es decir, las instrucciones SQL pueden estar incrustadas, lo que significa que pueden estar entremezcladas con las instrucciones del lenguaje de programación de dicho programa). En el caso de que las instrucciones estén incrustadas, el programa de aplicación puede estar escrito comúnmente en una variedad de lenguajes (como COBOL, Java. PL/1, etc.) (DATE, 2009).

En el caso de las NoSQL (no solo SQL) son utilizadas para referirse a una base de datos (BD) de código abierto que no ofrece una interfaz SQL pero sí utiliza el modelo relacional. NoSQL no impone una estructura de datos en forma de tabla y relaciones entre las mismas, en este caso es más flexible ya que permite almacenar información de forma clave-valor (Cassandra, Riak, etc.), documentos (CouchDB, MongoDB, BaseX, etc.), mapeo de columnas o grafos (Neo4j, InfoGrid, HyperGraphDB, etc.) (BAHIT, BURGA, DÍAZ y MONTERO, 2012).

La diferencia entre SQL y NoSQL es que resuelven escenarios completamente diferentes y excluyentes (para lo que es ideal SQL no lo es NoSQL y al revés). Los sistemas SQL permiten combinar de forma eficiente diferentes tablas para extraer información relacionada, mientras que los NoSQL no lo permite o de forma muy limitada. Por otra parte las bases de datos NoSQL permiten fácilmente distribuir grandes cantidades de información, mientras que distribuir bases de datos relacionales (SQL) requiere una cuidadosa planificación. Las bases de datos NoSQL escalan horizontalmente, añadiendo más servidores para hacer frente a cargas más grandes. Por otro lado, las bases de datos SQL, suelen escalar de forma vertical, añadiendo más y más tráfico a un solo servidor. Los sistemas NoSQL pueden compartir automáticamente los datos a través de servidores, sin necesidad de realizar algunas maniobras complejas de codificación equilibrando la carga entre varios servidores, proporcionando un sistema más robusto (MARIJA, 2014).

Después de haber realizado un análisis de estos dos sistemas, se selecciona el Sistema Gestor de Base de Datos NoSql para la administración de la Base de Datos de la propuesta de solución a desarrollar.

MongoDB

MongoDB es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre. Esto significa que cada entrada o registro puede tener un esquema de datos diferentes, con atributos o columnas que no tienen por qué repetirse de un registro a otro. Las características que más sobresalen de MongoDB son su velocidad y su rico pero sencillo sistema de consulta de los contenidos de la base de datos. Se podría decir que alcanza un balance perfecto entre rendimiento y funcionalidad. En MongoDB, cada registro o conjunto de datos se denomina documento, los cuales se pueden agrupar en colecciones, que son el equivalente a las tablas en una base de datos relacional (sólo que las colecciones pueden almacenar documentos con muy diferentes formatos, en lugar de estar sometidos a un esquema fijo). Se pueden crear índices para algunos atributos de los documentos, de modo que MongoDB mantendrá una estructura interna eficiente para el acceso a la información por los contenidos de estos atributos. Los distintos documentos se almacenan en formato BSON, o Binary JSON, que es una versión modificada de JSON que permite búsquedas rápidas de datos (SEGUIN, 2014).

Teniendo en cuenta estas características sobre todo la propiedad de guardar los datos en formato BSON, y no en XML, particularidad muy útil para la propuesta de solución, se decide seleccionar este gestor para la administración de la base de datos.

1.7.12. Sistema de control de versiones

Un sistema de control de versiones es una herramienta capaz de registrar todos los cambios hechos en el contenido de un proyecto, guardando versiones del producto en todas sus fases del desarrollo.

Git es un software de control de versiones pensando para facilitar la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones, cuando estas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o *front end* como StGIT, sin embargo, se ha convertido en un sistema de control de versiones con funcionalidad plena (RODRÍGUEZ, 2013).

1.7.13. Editor de textos

Un editor de texto se define como la principal herramienta para la creación del código que conformará a una determinada aplicación.

Sublime Text es un editor de texto y editor de código fuente está escrito en C++ y Python para los *plugins*. Desarrollado originalmente como una extensión de Vim, con el tiempo fue creando una identidad propia. Tiene soporte nativo para 43 lenguajes de programación y texto plano, el Syntax Highlight configurable o remarcado de sintaxis es completamente configurable a través de archivos de configuración del usuario y otros como la búsqueda dinámica que permite realizar búsquedas de expresiones regulares o por archivos, proyectos, directorios, una conjunción de ellos o todo a la vez (SUGRUE, 2014).

1.8. Conclusiones parciales del capítulo

Partiendo de la investigación antes realizada se pudo arribar a las conclusiones siguientes, las cuales arrojan resultados a tener en cuenta durante todo el ciclo de desarrollo del software:

1. Se realizó un análisis del estado del arte sobre los sistemas de mensajería instantánea y notificaciones para determinar las características que debía contener la aplicación para su desarrollo, así como las herramientas necesarias para la confección del mismo.
2. La identificación de características comunes presentes en los sistemas analizados que no deben faltar en el desarrollo de la solución como son: el envío de mensajes privados, notificaciones configurables, salas de chat, comunicación síncrona y asíncrona, y la integración con otros sistemas.
3. Se propone el desarrollo de una aplicación reutilizable y con una adecuada velocidad de respuesta, aplicando la metodología de desarrollo de software XP y lenguajes como JavaScript, el f*ramework* Node.js, Angular, Express, JQuery, Websockets, Boostrap y el sistema gestor de base de datos MogoDB; con el objetivo de lograr disminuir la diversidad tecnológica existente en las aplicaciones de mensajería desarrolladas en el centro.

Capítulo II: Presentación de la propuesta de solución.

Después de un estudio del estado del arte y de los elementos a tener en cuenta durante el periodo que abarca esta investigación, es preciso determinar los principales procesos que conformarán el sistema que se desea implementar. Para lograr estos objetivos se definirán los requisitos que formarán parte de la aplicación, ejecutando a su vez las diferentes etapas que define la metodología XP.

2.1. Modelo conceptual

El modelo conceptual se define como un artefacto necesario para visualizar los principales conceptos del dominio del sistema. El mismo sirve como base para la comprender los conceptos o definiciones presentes en el sistema a desarrollar combinándolos con los requisitos funcionales y no funcionales definidos. Un modelo conceptual debe tener como uno de sus principales objetivos saber explicar cuáles son los conceptos y cómo se relacionan en la descripción del problema (RUMBAUGH, JACOBSON y BOOCH, 1999).

2.1.1. Principales conceptos

- ✓ **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 usuario.
- ✓ **Herramienta de Mensajería Instantánea y Notificaciones:** Implanta una comunicación entre los usuarios que necesariamente estén registrados en el sistema.
- ✓ **Chat:** Aplicación que se utiliza para establecer una comunicación entre dos o más usuarios.
- ✓ **Sala de Chat:** Herramienta que se utiliza para establecer debates con fines colaborativos entre dos o más usuarios.
- ✓ **Mensaje:** Argumento enviado entre los usuarios para establecer una conversación produciéndose la activación directa del chat.

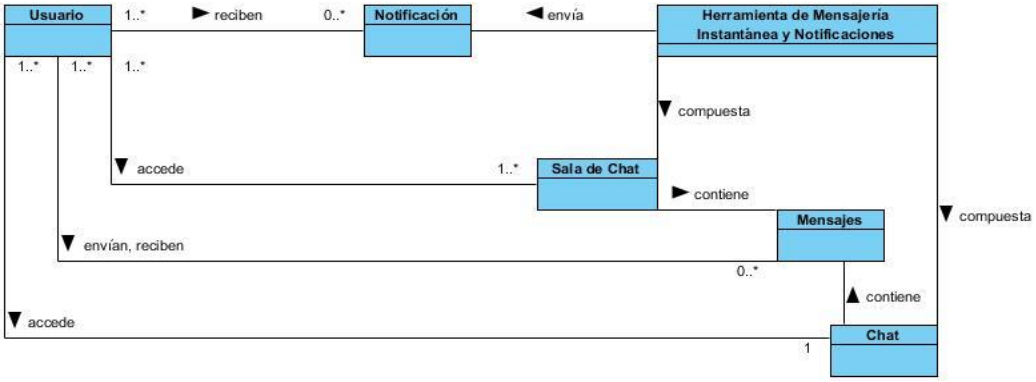


Imagen 2: Modelo conceptual

2.2. Descripción general del sistema a desarrollar

La herramienta de mensajería instantánea y notificaciones a implementar, está desarrollada sobre la tecnología asíncrona socket.io y como servidor de mensajería utiliza el entorno de programación Node.js.

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. Cuando se ejecute la aplicación como sistema independiente se realizará utilizando 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.

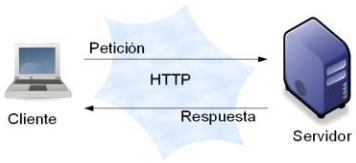


Imagen 3: Funcionamiento de la aplicación como sistema independiente

En el caso de cuando se integre el componente a otro sistema, el mismo debe capturar los datos de la creación de las sesiones. Con la implementación del componente el usuario será capaz de establecer una comunicación directa con otros usuarios, así como recibir avisos de notificaciones. El componente debe ser capaz de tomar los recursos necesarios para mantener el flujo de información haciendo que este proceso sea bidireccional y transparente al usuario.

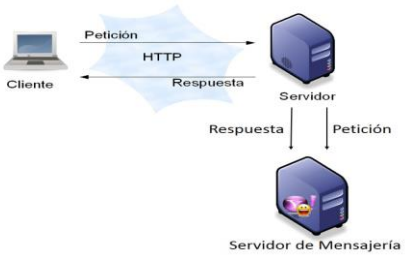


Imagen 4: Funcionamiento de la aplicación integrada con otro sistema

2.2.1 Características particulares del sistema

A partir de la descripción que se muestra a continuación, se presentarán las principales características que conformaran el sistema, permitiendo a través de las mismas brindar una visualización de las acciones que podrá realizar el usuario.

Lista de amigos: El sistema visualizará un panel a la derecha de la pantalla donde mostrará los usuarios conectados y su estado de disponibilidad. En la parte superior de este panel se encontrarán un conjunto de opciones con las cuales el usuario podrá interactuar de manera directa.

Opciones: Las opciones permitirán realizar tareas tales como: la configuración de las notificaciones, eliminar un amigo, entrar a las salas de chat, entre otras.

Mensajes: Los mensajes estarán colocados a continuación de la lista de amigos y serán visualizados cuando el receptor o el transmisor deseen establecer comunicación. Las conversaciones podrán ser minimizadas por el propio usuario y serán mostradas una a continuación de la otra.

Configurar Notificaciones: Esta opción brinda al usuario la posibilidad de configurar algunas características de las notificaciones. Entre estas características están el activar o desactivar el sonido de alerta ante una nueva notificación y la posición que ocupará en pantalla dicha notificación.

Notificación: Cuando se produzca una notificación, será colocada en la posición que el usuario haya definido, y será activada cuando se conecte o se desconecte un usuario.

Sala de chat: La opción de entrar a una sala de chat podrá darle la oportunidad al usuario de visualizar y seleccionar la sala a la cual desea pertenecer dentro del listado de salas de chat existentes, y a su vez podrá crear y eliminar una o más salas si así lo desea.

2.3. Lista de reserva

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 (QUIROGA, 2014).

Contará con una API (Interfaz de Programación Aplicada) que permitirá:

RF 1.1. ***Crear usuarios:***

El sistema debe permitir crear un nuevo usuario.

RF 1.2. ***Obtener listado de usuarios:***

El sistema debe permitir obtener la lista de usuarios.

RF 1.3. ***Obtener datos de usuario:***

El sistema debe permitir obtener los datos de un usuario.

RF 1.4. ***Editar usuarios:***

Garantiza que el sistema edite los datos de un usuario.

RF 1.5. ***Cambiar contraseña directamente:***

Permite cambiar la contraseña de un usuario en el sistema.

RF 1.6. ***Cambiar usuario*****:**

Cambia el nombre de usuario en el sistema.

RF 1.7. ***Crear sala de chat:***

El usuario puede crear una sala de chat en el sistema.

RF 1.8. ***Editar sala de chat:***

El usuario puede editar una sala de chat en el sistema.

RF 1.9. ***Eliminar sala de chat:***

Brinda la posibilidad de eliminar una sala de chat en el sistema.

RF 1.10. *Obtener listado de sala de chat:*

El sistema debe permitir mostrar las salas de chat existentes.

RF 1.11. *Enviar una notificación a un usuario o a un grupo de usuarios:*

El sistema debe permitir enviar notificaciones a los usuarios.

RF 1.12. *Autenticar usuario:*

El sistema debe autenticar correctamente a un usuario.

Usuario:

RF 1.13. Gestionar usuario en el sistema:

- ✓ **RF 1.13.1. *Insertar usuario:***

  Inserta a un usuario en el sistema una vez que el mismo se haya autenticado.

- ✓ **RF 1.13.2. *Modificar datos de usuario:***

  El usuario es capaz de modificar sus datos personales.

RF 1.14. Autenticar usuario en el sistema:

El usuario debe autenticarse para poder acceder a todas las funcionalidades del sistema.

Chat

RF 1.15. Consultar mensajería instantánea:

- ✓ **RF 1.15.1*. Iniciar conversación privada con un amigo:***

  El usuario podrá establecer una comunicación con uno o más usuarios.

- ✓ **RF 1.15.2. *Enviar mensaje a un amigo:***

  El usuario será capaz de enviar un texto a otro usuario y el mismo ser recibido por la persona a la cual se le envió el mensaje.

- ✓ **RF 1.15.3. *Recibir mensaje de un amigo:***

  El usuario receptor podrá visualizar los mensajes que le han sido enviados.

- ✓ **RF 1.15.4. *Cerrar conversación:***

  Se podrá cerrar la ventana de una conversación según la preferencia del usuario.

- ✓ **RF 1.15.5. *Mostrar estado del usuario:***

  Se mostrará la disponibilidad del usuario a través de un ícono que tomará diferentes colores para indicar su estado.

- ✓ **RF 1.15.6. *Cambiar estado del usuario:***

  Se podrá cambiar el estado del usuario según la preferencia del mismo.

RF 1.16. Mostrar mensajes de historial:

Mostrará los diferentes mensajes que ha escrito y recibido el usuario en la ventana de conversación.

RF 1.17. Gestionar amigo para la mensajería instantánea:

- ✓ **RF 1.17.1. *Mostrar amigo:***

Se muestran los amigos del usuario en la lista de contactos.

- ✓ **RF 1.17.2. *Eliminar amigo:***

Se puede eliminar al usuario de la lista de contactos.

Notificaciones

RF 1.18. Mostrar notificaciones:

Se mostrarán las nuevas notificaciones que sean enviadas al usuario.

RF 1.19. Configurar notificaciones:

El usuario puede establecer las configuraciones que estime convenientes para las notificaciones.

- ✓ **RF 1.19.1. *Ajuste de sonido:***

  Activa o desactiva el sonido según la preferencia del usuario.

- ✓ **RF 1.19.2. *Ajuste de posición:***

  Opción que es utilizada con el fin de definir la posición que adoptará la notificación una vez que se ejecute.

Sala de Chat

RF 1.20. Gestionar sala de chat*:*

- ✓ **RF 1.20.1. *Mostrar sala de chat:***

  Muestra el listado de salas de chat disponibles.

- ✓ **RF 1.20.2. *Crear sala de chat:***

  Posibilita creación de una nueva sala de chat.

- ✓ **RF 1.20.3. *Eliminar sala de chat:***

  Elimina la(s) sala(s) de chat que el usuario estime conveniente.

RF 1.21. Consultar sala de chat:

- ✓ **RF 1.21.1. *Iniciar conversación grupal:***

  Se establece una conversación entre todos los usuarios conectados en la sala de chat de la cual se hace uso.

- ✓ **RF 1.21.2. *Enviar mensajes a la sala de chat:***

  El usuario puede enviar mensajes a los demás usuarios que pertenezcan a la sala de chat de la cual esté haciendo uso.

- ✓ **RF 1.21.3. *Recibir mensajes en la sala de chat:***

  Se reciben los mensajes de los usuarios que pertenezcan a una misma sala de chat.

- ✓ **RF 1.21.4. *Cerrar sala de chat:***

  Brinda la posibilidad de cerrar la sala de chat cuando el usuario lo estime conveniente.

- ✓ **RF 1.21.5. *Mostrar estado del usuario en la sala de chat:***

  Evidencia el estado de disponibilidad de todos los usuarios pertenecientes a una misma sala de chat.

- ✓ ***RF 1.21.6. Cambiar estado del usuario en la sala de chat:***

  El usuario puede cambiar el estado de disponibilidad.

RF 1.22. Salir del sistema:

El usuario puede salir del sistema cundo lo estime conveniente.

2.3.1. Requisitos no funcionales

Los requisitos no funcionales (RNF) describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema. Este tipo de requisito incluye restricciones como el tiempo de respuesta (desempeño), la precisión, recursos consumidos, seguridad, entre otros (QUIROGA, 2014).

Dentro de los requisitos no funcionales identificados para el curso de esta investigación están los siguientes:

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, siempre y cuando sea compatible con las siguientes características:

- ✓ Internet Explorer 8.0 y superior.
- ✓ Mozilla Firefox 7.0 y superior.
- ✓ Chrome 7.0 y superior.
- ✓ Opera 12.1 y superior.
- ✓ Safari 5.1 y superior.

Hardware

RNF 6: 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 7: 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.

Precisiones sobre diseño e implementación

RNF 8: El marco de trabajo en el cual se realizará la aplicación será sobre Node.js v0.10.35

RNF 9: Se empleará como editor de texto Sublime Text en su v3.0.

RF 10: Se utilizará MongoDB como Sistema Gestor de Base de Datos.

RF 11: Se utilizará programación orientada a eventos.

Apariencia o interfaz externa

RNF 12: La apariencia que adoptará la herramienta a implementar será sencilla y de agradable vista al usuario, brindándole al mismo las opciones necesarias para que interactúe con ella de forma rápida y fácil. Este sistema o componente será implementado siguiendo las especificaciones de diseño de XAUCE, ya que el mismo es el que marca los productos de FORTES.

2.5. Descripción de los roles que interactúan con el sistema

Tabla 5: Descripción de roles

Rol	Descripción
Usuario	Este rol tiene acceso a todas las partes del sistema siempre y cuando este registrado y autenticado. El mismo puede acceder a la herramienta de chat pudiendo realizar todas las actividades que brinda esta herramienta, dígase: enviar mensajes, recibir mensajes y cerrar conversación, también puede eliminar a un amigo y mostrar historial. Las notificaciones serán activadas cuando el usuario ejecute eventos tales como la conexión y desconexión del sistema. Podrá ser partícipe de las salas de chat enviando y recibiendo mensajes de la sala a la cual pertenezca. Este rol desempeñará todas las actividades que propone la aplicación para su puesta en funcionamiento.

2.6. Exploración

Es la fase en la que se define el alcance general del proyecto. En ella, el cliente define lo que necesita mediante la redacción de sencillas historias de usuarios y los programadores estiman los tiempos de desarrollo en base a esta información. Las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración. La fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado (JOSKOWICZ, 2008).

Para la aplicación de esta fase en la construcción de la aplicación, fue necesario que el cliente definiera en conjunto con el equipo de desarrollo las historias de usuario, para posteriormente los programadores estimaran el tiempo de duración que compondrá cada historia.

2.7. Planificación

Esta etapa es una fase corta, donde el cliente y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y las entregas asociadas a estas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación, donde el resultado de esta fase es la obtención de un Plan de Entregas (JOSKOWICZ, 2008). Las estimaciones de esfuerzo asociado a la implementación de las historias de usuario la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias (LETELIER y PENADÉS, 2009).

Al planificar por tiempo, método que es utilizado para la planificación efectuada, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar (LETELIER y PENADÉS, 2009). Dentro de la fase de planificación es necesario realizar una estimación de los esfuerzos, para ello la colaboración del cliente en conjunto con el equipo de desarrollo se vuelve vital, dando paso a que el mismo sea parte de cada paso que se genere dentro del proceso de desarrollo del software. Se tuvo en cuenta para realizar una correcta planificación, la priorización de aquellas historias de usuario que el cliente selecciona como de gran importancia para el negocio. La siguiente tabla muestra las estimaciones de esfuerzo para las diferentes historias de usuario.

Estimación de esfuerzo por Historia de Usuario (HU).

Tabla 6: Estimación del esfuerzo por HU

Historias de Usuario(HU)	Estimación del tiempo(días)
HU 1: Gestionar usuario	10
HU 2 : Autenticar usuario en el sistema	3

HU 3: Consultar mensajería instantánea	15
HU 4: Mostrar historial para la mensajería instantánea	5
HU 5: Gestionar amigo para la mensajería instantánea	10
HU 6: Mostrar notificaciones	5
HU 7: Configurar notificaciones	10
HU 8: Gestionar sala de chat	15
HU 9: Consultar sala de chat	10
HU 10: Salir del sistema	3

2.7.1. Historia de usuarios (HU)

Dentro de la fase de planificación de la metodología XP, se realizan las historias de usuario, estas tienen el mismo propósito que los casos de uso. Las mismas son escritas por los propios clientes, tal y como ellos visualicen las necesidades del sistema. Las historias de usuario son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario. También conducirán el proceso de creación de los test de aceptación que son empleados para verificar que las historias de usuarios han sido implementadas correctamente (ESCRIBANO, 2002). Las historias de usuario pueden ser consultadas en el **Anexo: 5 Historias de usuario (HU).**

2.7.2. Iteraciones

Luego de un análisis realizado para conformar las historias de usuario para definir la estimación de esfuerzos para cada una, es necesario entonces proseguir con la siguiente etapa del ciclo de desarrollo de la metodología XP. En la etapa de iteraciones se puntualiza la prioridad de cada funcionalidad que definen las historias de usuario. Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle

como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios (JOSKOWICZ, 2008). La realización de las tareas para la implementación de las historias de usuario, fueron realizadas según el grado de importancia que se le acredita a cada una. Se tuvo en cuenta las implementaciones de funcionalidades que dependían de otras para la confección de las mismas, las cuales fueron resueltas en una misma iteración con el objetivo de que no exista atraso en cuanto al plan de entregas propuesto.

2.7.3. Plan de duración de las iteraciones

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores (LETELIER y PENADÉZ, 2006). Para la confección de un plan de iteraciones se deben de tener en cuenta los siguientes factores: leer las historias de usuario y a través de las mismas detallar tareas, a continuación se deben asignar los quehaceres, teniendo en cuenta a su vez la estimación y la selección de dichas tareas para cada programador (ESCRIBANO, 2002).Siguiendo este precepto se confecciona el plan de duración de cada una de las iteraciones propuestas, donde se define el tiempo de duración de cada iteración por cada HU.

Plan de duración de iteraciones

Tabla 21: Plan de duración de iteraciones

Iteración	HU a implementar	Prioridad	Días	Total de días de cada iteración	Semanas
	HU 1: Gestionar usuario	Alta	10		
1	**HU 2:** Autenticar usuario en el sistema	Alta	3	23	3
	HU 5: Gestionar amigo para la mensajería instantánea	Media	10		

	HU 3: Consultar mensajería instantánea	Alta	15	35	5
2	**HU 4:** Mostar historial para la mensajería instantánea	Media	5		
	HU 6: Mostrar notificaciones	Alta	5		
	HU 7: Configurar notificaciones	Media	10		
	HU 8: Gestionar sala de chat	Alta	15	28	3
3	**HU 9:** Consultar sala de chat	Media	10		
	HU 10: Salir del sistema	Media	3		
Total de días				86	11

2.7.4. Plan de entregas

Después de tener ya definidas las historias de usuario es necesario crear un plan de publicaciones, en inglés "*Release plan*", donde se indiquen las historias de usuario que se crearán para cada versión del programa y las fechas en las que se publicarán estas versiones. Un *Release plan* es una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias que serán implementadas en cada versión del programa. Después de haber realizado un plan de publicaciones tienen que estar claros estos cuatro factores: los objetivos que se deben cumplir (que son principalmente las historias que se deben desarrollar en cada versión), el tiempo que tardarán en desarrollarse y publicarse las versiones del programa, el número de personas que trabajarán en el desarrollo y cómo se evaluará la calidad del trabajo realizado (CASTILLO, 2014).

A partir de las iteraciones antes propuestas se pretende lograr por cada una los siguientes objetivos:

1ra Iteración: Preparar las funcionalidades básicas para el acceso al sistema.

2da Iteración: Desarrollar las funcionalidades de la aplicación acordadas con el cliente.

3ra Iteración: El producto debe estar prácticamente elaborado, dando paso a que se desarrollen funcionalidades que son de rápida realización para de esta manera dar fin al proceso de desarrollo del software.

Plan de entregas por iteraciones

Tabla 22: Plan de entregas por iteraciones

Entregable	Final 1ra Iteración 4ta_semana 26 de febrero de 2015	Final 2da Iteración 2da_semana 6 de abril de 2015	Final 3ra Iteración 1ra_semana 6 de mayo de 2015
HU 1: Gestionar usuario **HU 2:** Autenticar usuario en el sistema **HU 5:** Gestionar amigo para la mensajería instantánea	**Finalizada**	**-**	**-**
HU 3: Consultar mensajería instantánea **HU 4:** Mostrar historial para la mensajería instantánea **HU 6:** Mostrar notificaciones **HU 7:** Configurar notificaciones	**-**	**Finalizada**	**-**
HU 8: Gestionar sala de chat **HU 9:** Consultar sala de chat **HU 10:** Salir del sistema	**-**	**-**	**Finalizada**

2.8. Tarjetas Clases – Responsabilidades – Colaboración (CRC)

El uso de las tarjetas C.R.C (*Class, Responsabilities and Collaboration*) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica (CASTILLO, 2014).

Para la confección de las tarjetas CRC es imprescindible guiarse por el diagrama de clases que describe las funcionalidades y las clases de la aplicación, en el **Anexo 6: Diagrama de clases**, se visualiza lo planteado anteriormente. A continuación se expone un ejemplo de cómo quedarían conformadas las tarjetas CRC, el resto de las mismas está disponible en el **Anexo 7: Tarjetas CRC**.

Tabla 23: Tarjetas CRC (No: 1)

Tarjetas CRC	
Número: 1	**Nombre de la clase:** controlador-vista
Responsabilidades: Controlador del lado del cliente (Angular), que contiene toda la lógica de cómo mostrar los datos, de cómo pedirlos en el servidor, de los eventos y se encarga de cambiar los datos en las plantillas.	
Colaboraciones: client-pages	

2.9. Arquitectura de la información

La arquitectura de información, tiene como propósito la búsqueda de mejores diseños para la presentación de la información y su comprensión, así como la usabilidad, que estudia el conjunto de características del diseño y funcionamiento de una interfaz de usuario, para obtener una correcta operación y comprensión de los contenidos, es una disciplina cuya actividad está dirigida a lograr la máxima satisfacción del usuario durante el proceso de interacción con los productos de información. Una información estructurada y coherente, sin dudas, facilita, tanto su consulta como el proceso de asimilación e introducción en la práctica (BUSTAMANTE, 2004).

La arquitectura de la información brinda muchos beneficios al ubicar rápidamente la información, encontrar con el menor esfuerzo, establecer relaciones o enlaces, además de reducir costos de mantenimiento y

procesos de reingeniería. Para una empresa es importante que sus clientes encuentren la información, y que esta información conduzca al usuario a tomar una decisión (GONZALES, 2003).

A continuación se muestra un mapa de navegación que refleja la organización de los contenidos dentro de la aplicación a desarrollar. Dicho esquema está elaborado con el objetivo de que el usuario tenga una orientación sobre cómo encontrar las funcionalidades que distinguen al software.

Mapa de navegación

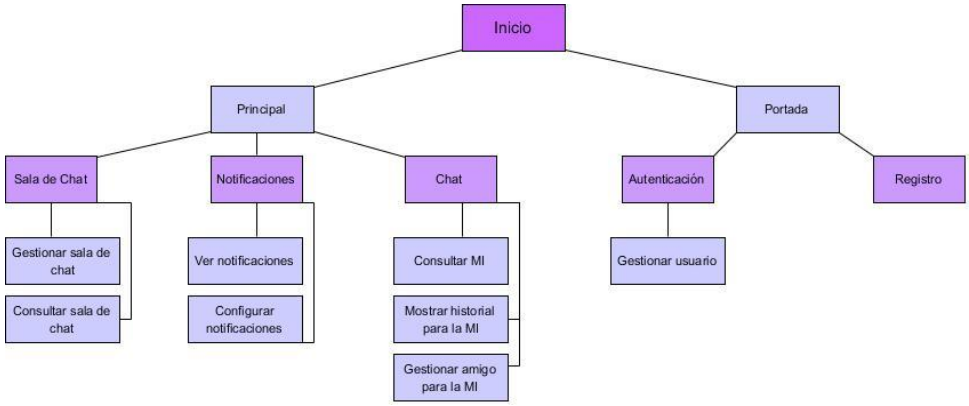


Imagen: 6 Arquitectura de la información del sistema

La imagen presenta una portada que estará visualizada con la ventana de registro, una vez efectuado el mismo se procederá a la autenticación. La funcionalidad de gestionar usuario tiene entre sus principales características la opción de modificar los datos que posee el usuario.

Luego de haberse ejecutado el registro y la autenticación se procede a la pantalla principal de la aplicación, la cual está compuesta por salas de chat, notificaciones y el propio chat. Si se accede a la sala de chat, se mostrarán las salas de chat existentes además de mostrar todas las características que brinda esta funcionalidad, como es el caso del envío y recibo de mensajes. En el caso de la gestión de las salas de chat, se utilizará para configurar todo lo referente a las mismas, como es el caso de añadir una nueva sala de chat. Las notificaciones por su parte pueden ser visualizadas y configuradas. El chat contiene funcionalidades como consultar mensajería instantánea, donde se puede enviar y recibir mensajes y posee también la funcionalidad de mostrar historial, que visualiza los mensajes almacenados.

Los colores y el formato utilizados en la confección de la aplicación serán los que el centro (XAUCE) ha definido para su línea de trabajo, predominando de esta manera el color naranja para cada vista.

2.10. Descripción de arquitectura del sistema

Según Christopher Alexander los patrones representan (…*un problema que ocurre una y otra vez en nuestro entorno, y describe la esencia de la solución a ese problema, de tal modo que pueda utilizarse esta solución un millón de veces más, sin siquiera hacerlo de la misma manera dos veces.*). Seguidamente se muestra una descripción de cómo se aplican los patrones y estilos arquitectónicos en la aplicación implementada.

2.10.1. Patrón arquitectónico

El diseño arquitectónico es un proceso creativo en el que se intenta establecer una organización del sistema que satisfaga los requerimientos funcionales y no funcionales del propio sistema. La arquitectura de un software puede basarse en un modelo o estilo arquitectónico particular. Un estilo arquitectónico es un patrón de organización de un sistema, tal como una organización cliente-servidor, una arquitectura por capas y un estilo de repositorio de datos (SOMMERVILLE, 2005).

El modelo arquitectónico cliente-servidor es un modelo de sistema en el que dicho sistema se organiza como un conjunto de servicios y servidores asociados, más unos clientes que acceden y usan los servicios. La ventaja más importante de este estilo arquitectónico es que es una arquitectura distribuida. Se puede hacer un uso efectivo de los sistemas en red con muchos procesadores distribuidos. Es fácil añadir un nuevo servidor e integrarlo con el resto del sistema o actualizar los servidores de forma transparente sin afectar el resto del sistema (SOMMERVILLE, 2005).

Se escoge como modelo arquitectónico a la arquitectura cliente-servidor, ya que para la confección de la aplicación se seleccionó el *framework* Node.js, el cual utiliza este tipo de estilo para hacer que las páginas web sean dinámicas, lo cual se debe al constante intercambio de información que se realiza entre el cliente y el servidor.

2.10.2. Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, los mismos brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (TEDESCHI, 2004). Node.js y Express se utilizan para implementar la clase controladora, que a su vez se vincula con Angular mediante su función scope para

generar las vistas, y mongoose que es la librería para la BD que se conecta con MongoDB, entonces se implementar el patrón MVC (modelo, vista, controlador).

El patrón MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. En estos nuevos tiempos de desarrollo tecnológico surge la necesidad de crear software más robusto, con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos. El fundamento principal de este patrón es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo mismo, *Model, Views & Controllers* (ÁLVAREZ, 2014).

El Modelo es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos habitualmente se almacenan en una base de datos, por lo que en los modelos se tendrá todas las funciones para acceder a las tablas (ÁLVAREZ, 2014). Mediante el desarrollo de modelos de datos es posible definir la estructura de los datos a almacenar, los cuales son implementados utilizando Json, lenguaje por excelencia para el almacenamiento de MongoDB. En estas clases se definen a su vez las funcionalidades que se pueden realizar en la aplicación como es el caso de las operaciones de crear, eliminar, modificar y mostrar.

En las Vistas, como su nombre nos hace entender, se almacena el código de la aplicación realizada que va a producir la visualización de las interfaces de usuario, o sea, el código que permitirá mostrar los estados de la aplicación en HTML (ÁLVAREZ, 2014). En las vistas se tienen los códigos HTM, CSS y JQuery. Angular, por su parte tiene dentro de sus principales características el uso de la estructura MVC, lo que le posibilita separar muy bien la responsabilidad de cada tecnología en su ámbito: CSS, HTML y Javascript, y las comunica cuando lo considera necesario.

El Controlador Contiene el código necesario para responder a los eventos que se solicitan en la aplicación, como visualizar un elemento, envío y recibo de mensajes y el envío de notificaciones. En realidad es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de la aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo (ÁLVAREZ,

2014). En la aplicación las clases controladoras están definidas por las clases app.js y la clase socketio.js, la cual se encarga de los eventos referentes con socket, que a su vez es llamada por la clase app.js.

Otro de los patrones utilizados es el GRASP[12](*General Responsibility Assignment Software Patterns*), los cuales describen los principios fundamentales para asignar responsabilidades a los objetos (LARMAN, 1998).

El patrón Experto se basa en asignar la responsabilidad al experto en la información, es decir a la clase que contiene la información necesaria para cumplir la responsabilidad. De esta manera se logra que las clases tengan un mejor comportamiento y hacen que las mismas sean más cohesivas, permitiendo a su vez que sean mayores las posibilidades de soporte (LARMAN, 1998). Este tipo de patrón se puede observar en las clases auth.service.js, y socket.service.js las cuales se encargan de controlar en la aplicación las tareas de autenticación, envío y recepción de eventos de socketio por las clases clientes. En estas clases es donde se crean o dan acceso a objetos que permiten que se realicen las tareas de autenticación y de manipulación de los eventos generados tantos por el cliente como por el servidor. A continuación se muestra un ejemplo de lo planteado con anterioridad.

```
// When the user disconnects.. perform this

function onDisconnect(socket) {}

// When the user connects.. perform this

function onConnect(socket) {
    // When the client emits 'info', this listens and executes
    socket.on('info', function(data) {
        console.info('[%s] %s', socket.address, JSON.stringify(data, null, 2));
    });

    // When the client emits 'enviar_mensaje', this listens and executes
     socket.on('enviar_mensaje', function(data) {
        socket.emit('enviar_mensaje', {
            usuario: data.name,
            usuario_envia: data.envia_name,
            mensaje: data.msg
        });

        socket.broadcast.emit('enviar_mensaje', {
            usuario_recibe: data.name,
            usuario_envia: data.envia_name,
            mensaje: data.msg
        });
    });

    // Insert sockets below
    require('../api/message/message.socket').register(socket);
    require('../api/thing/thing.socket').register(socket);
    require('../api/user/user.socket').register(socket);
}
```

Imagen 7: Clase socket.io.js

[12] GRASP: Responsabilidad general Patrones de Software Asignación.

El Creador se encarga de la asignación de responsabilidades basadas en la creación de objetos. Tiene como uno de sus principales propósitos la búsqueda de un creador que se debe conectar con el objeto producido en cualquier evento, una vez escogido el creador se da soporte al bajo acoplamiento (LARMAN, 1998). Las clases que permiten la creación o instanciación de clases, son las llamadas modelos, tales como user.model.js, en la cual se puede crear un objeto de tipo *User*, facilitando que otras clases puedan hacer llamadas o instancias de este objeto antes de ser creado. A continuación se muestra un ejemplo.

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var crypto = require('crypto');
var authTypes = ['github', 'twitter', 'facebook', 'google'];

var UserSchema = new Schema({
  name: String,
  email: { type: String, lowercase: true },
  role: {
    type: String,
    default: 'user'
  },
  hashedPassword: String,
  provider: String,
  salt: String,
  facebook: {},
  twitter: {},
  google: {},
  github: {},
  datatoken: String
```

Imagen 8: Clase user.model.js

El Bajo Acoplamiento mide el grado en que una clase está conectada a otra, tiene conocimiento de otra, o de alguna manera depende de otra. Es un patrón evaluativo, ya que un bajo acoplamiento permite que el diseño de clases sea más independiente, reduce el impacto de los cambios y aumenta la reutilización (LARMAN, 1998). En las clases main.controller.js, socket.service.js y auth.service.js se puede afirmar que hacen uso del bajo acoplamiento, debido a que en estas se puede observar un acoplamiento de control, que se produce cuando un módulo le envía a otro elementos de control que determinan la lógica de ejecución del mismo.

```
enviarMensaje: function(name, envia_name) {

    //aparecer cartel
    $('#' + name).css('display', 'inline');

    $('.close').click(function() {
        $(this).parent().parent().css('display', 'none');
    });

    $('textarea').keypress(
        function(e) {
            if (e.keyCode == 13) {
                var msg = $(this).val();
                $(this).val('');

                if (msg != '')
                    $('<div class="msg_b">' + msg + '</div>').insertBefore('.msg_push');
                $('.msg_body').scrollTop($('.msg_body')[0].scrollHeight);
```

Imagen 9: Clase socket.service.js

La Cohesión mide el grado en que están relacionadas las responsabilidades de una clase, es además un patrón evaluativo, ya que entre más alta sea la cohesión es más fácil de comprender, de utilizar, de mantener y de cambiar (LARMAN, 1998). En la clase socket.service.js se encuentra un conjunto de funcionalidades que solamente pueden ser ejecutadas en esta clase, la cual tiene como funcionalidad la comunicación directa con el servidor.

```
/* global io */
'use strict';

angular.module('tesisAppApp')
    .factory('socket', function(socketFactory) {

        // socket.io now auto-configures its connection when we ommit a
        connection url
        var ioSocket = io('', {
            // Send auth token on connection, you will need to DI the Auth
            service above
            // 'query': 'token=' + Auth.getToken()
            path: '/socket.io-client'
        });

        var socket = socketFactory({
            ioSocket: ioSocket
        });

        return {
            socket: socket,
```

Imagen 10: Clase socket.service.js

El patrón Controlador es aquel que representa al sistema completo y es el manejador artificial de los eventos. Consiste en la asignación de responsabilidades para controlar el flujo de los eventos que genera el sistema a clases específicas (LARMAN, 1998). Se evidencia su uso en la clase app.js, dado que en esta se manejan los eventos realizados por la aplicación, tanto en la parte del cliente como en la del servidor.

```
process.env.NODE_ENV = process.env.NODE_ENV || 'development';

var express = require('express');
var mongoose = require('mongoose');
var config = require('./config/environment');

// Connect to database
mongoose.connect(config.mongo.uri, config.mongo.options);

// Populate DB with sample data
if (config.seedDB) {
    require('./config/seed');
}

// Setup server
var app = express();
var server = require('http').createServer(app);
var socketio = require('socket.io')(server, {
    serveClient: (config.env === 'production') ? false : true,
    path: '/socket.io-client'
});
require('./config/socketio')(socketio);
require('./config/express')(app);
require('./routes')(app);
```

Imagen 11: Clase app.js

Los patrones de creación son parte de los patrones Gof[13] (*Gan of Four*), los cuales son utilizados para estructurar y encapsular una clase u objeto, se define además cómo un objeto que puede delegar responsabilidades a otros objetos (LARMAN, 2003).

El patrón Singlenton hace referencia a aquellos métodos que solo pueden ser llamados si es a través de una instancia de la clase que los contiene (LARMAN, 2003). Un ejemplo de esto se visualiza en el archivo main.controler.js, donde se crea una instancia de la clase auth, dentro de la misma se encuentran métodos como isLoggedIn, isAdmin, los cuales solo pueden ser llamados si es a través de una instancia de la clase auth.

```
// controlador ruta /main
app.controller('MainCtrl', function($scope, $http, $modal, $rootScope, socket, User, $location, Auth) {
    $scope.isCollapsed = true;
    $scope.isLoggedIn = Auth.isLoggedIn;
    $scope.isAdmin = Auth.isAdmin;
    $scope.getCurrentUser = Auth.getCurrentUser;
    $scope.users = User.query();
    var name=Auth.getCurrentUser;
    $scope.$on('$estado',socket.allChangeStatus());
    $scope.$on('$enviar_mensaje', socket.recibirMensaje(name));
```

Imagen 12: Clase auth

Los patrones de comportamiento son aquellos que se encargan de establecer la forma en que los objetos se comunican en la aplicación (LARMAN, 2003). El patrón Iterator se encarga de recorrer una lista sin conocer su estructura interna, lo cual es aplicado mediante el uso de la directiva ng-repeat, que permite recorrer un conjunto de elementos sin conocer el tipo de datos de los elementos que va a recorrer.

```
<div class="" ng-repeat="user in users" ng-show="isLoggedIn()">

    <a href="" ng-click="nuevaConversacion(user.name, getCurrentUser().name)">
        <a href="" ng-click="nuevaConversacion(user.name,getCurrentUser().name)" style="margin-left: 5px; margin-right:
        5px; " class="icon-user"></a>
        <a href="" ng-click="nuevaConversacion(user.name, getCurrentUser().name)" class="" style="padding-rigth:100px;">
        {{user.name}}</a>
        <a href="" id="{{user.name}}" style=""class="estado-disponible"></a>
    </a>
</div>
```

Imagen 13: Directiva ng-repeat

[13] GOF: Gan de los Cuatro.

2.11. Modelo de datos

Un modelo de datos es un conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Los modelos de datos comprenden aspectos relacionados con: estructuras y tipos de datos, operaciones y restricciones. Entre sus principales características se encuentra que son independientes del sistema gestor de base de datos, presentan un nivel mayor de abstracción, mayor capacidad semántica y son más enfocados al diseño de alto nivel (ZORRILLA, 2011).

A continuación se muestra un diagrama que describe cómo está compuesta la base de datos del sistema, la cual define dos aspectos fundamentales los usuarios (User) y los mensajes (Message). En cada tabla se almacenan datos que son de vital importancia, ya que los cuales son llamados desde funciones para obtener dichos elementos.

Imagen 14: Modelo de dato

2.12. Conclusiones parciales del capítulo

Después de haber realizado las etapas de exploración, planificación e iteración que incluye la metodología XP se obtienen las siguientes conclusiones:

1. Se identificaron un total de 55 requisitos, donde se definen 43 como funcionales y 12 se caracterizan por ser no funcionales, los cuales serán parte del funcionamiento de la aplicación.
2. Después de un análisis sobre el tiempo de duración con el cual se desarrolló la aplicación, se estableció un plan de entregas, dividido por iteraciones que se resumen en un total de 86 días para dar conclusión al software.
3. Se estableció como estilo arquitectónico la arquitectura cliente-servidor, la cual responde a las necesidades y flujo de trabajo de la aplicación a desarrollar.
4. 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.
5. Se confeccionó un modelo de datos que recoge el funcionamiento de la base de datos del sistema.

Capítulo III: Implementación y pruebas.

Luego de haber elaborado la lista de reserva que define las funcionalidades del sistema a desarrollar, así como la confección de un plan de entregas para garantizar el cumplimiento de las tareas propuestas, y el establecimiento de un estilo y un modelo arquitectónico, se da paso a la próxima fase de desarrollo de la aplicación que se pretende lograr. Durante el presente capítulo se realizarán las acciones de implementación de la aplicación. Una vez terminada la fase de implementación se procederá a especificar y utilizar los tipos y mecanismos de pruebas que se ejecutarán para garantizar una adecuada calidad del software.

3.1. Implementación

Una vez que se conoce qué funciones debe desempeñar el sistema y que se ha decidido cómo se van a organizar los distintos componentes, es el momento de pasar a la etapa de implementación. Antes de escribir una sola línea de código es fundamental haber comprendido bien el problema que se pretende resolver y haber aplicado principios básicos de diseño que permitan construir un sistema de calidad (GLASS, 2003).

En esta próxima fase las historias de usuario que quedaron definidas serán implementadas, cumpliendo además con el plan de iteraciones propuesto. Las tareas de implementación estarán basadas en los patrones y estilos arquitectónicos antes propuestos, para de esta forma obtener un software reutilizable. En la fase de implementación el cliente debe estar muy compenetrado con el equipo de desarrollo, ya que el mismo proporciona las historias de usuario y las mismas son expresamente cortas y de alto nivel, además de que no contienen los detalles necesarios para realizar el desarrollo del código (JOSKOWICZ, 2008).

XP propone que la implementación del código sea realizado a través de pares de programadores, ya que al ejecutar esta práctica se minimizan los errores y se logran mejores diseños. Esta buena práctica proporciona que los problemas sean solucionados en forma más rápida, que el proyecto termine con más personas que conozcan los detalles de cada parte del código y que el equipo aprenda a trabajar unido generando una buena dinámica y que la información fluya rápidamente. Otra recomendación importante de esta metodología es la utilización de estándares, para lograr que el código sea entendible y que se

faciliten las tareas de recodificación (JOSKOWICZ, 2008). Por las ventajas que estas buenas prácticas implican para el desarrollo de un sistema, se implementaron dichas habilidades al producto de software efectuado.

3.1.1. Integración con otros sistemas

Para lograr que el software desarrollado logre la integración con otros sistemas de la web, se tuvo en cuenta la utilización de sesiones y cookies, los cuales se describen como datos que quedan archivados en los navegadores cuando un usuario se registra en un sistema. A través de esos datos, la aplicación desarrollada debe ser capaz de obtenerlos y usarlos como datos de registro. Esta acción brinda la posibilidad de que el usuario mientras esté registrado en una aplicación web, no necesite registrarse nuevamente para utilizar el componente de mensajería instantánea y notificaciones.

3.1.2. Seguridad

Para garantizar la seguridad fue necesario la aplicación de estrategias de autenticación. La utilización de token facilita que se cree el mismo por cada usuario registrado, el token posee un tiempo de duración, mientras no se cierre el navegador el token permanece activo con los datos de registro del usuario, evitando que se vuelvan a insertar los datos de registro. En el caso de la base de datos se estableció una contraseña para garantizar accesos no deseados. Se dio tratamiento especial además a las palabras y términos no adecuados en el contenido de los mensajes, los cuales no serán enviados para de esta manera asegurar la integridad del software. Fue necesario también aplicar estrategias para evitar inyecciones SQL y XSS, poniéndose en práctica en aquellos textos que se envíen como script, los cuales son analizados, para posteriormente ser convertidos y enviados como texto plano.

Node.js posee características que facilitan la seguridad del sistema, como es el caso de:

- ✓ Manejo de rutas, garantizando de esta manera que solo se acceda a las rutas definidas en la aplicación.
- ✓ El archivo .htaccess, analiza un conjunto de elementos de seguridad como es el caso de la norma File Access que se encarga de bloquear el acceso a ficheros ocultos y directorios, además de bloquear el acceso a ficheros de código y de resguardo. En este mismo archivo se encuentra el elemento ERRORS, que se asegura de mostrar el error 404 *Not Found* (no encontrado), que es ejecutado cuando se accede a carpetas y directorios no encontrados.

Angular también posee directivas de seguridad, llevadas a cabo cuando:

- ✓ Se insertan datos no validos en el sistema, respondiendo a su vez con mensajes de error.
- ✓ Elementos como ng-minlength y ng-maxlength permiten validar el número mínimo y máximo de elementos que se inserten, $valid, por ejemplo, es una función que se utiliza mayormente en el envío de formularios, ya que la misma verifica si los datos que se insertaron en el formulario son correctos.

3.2. Tareas de ingeniería

A través de las tareas de ingeniería los desarrolladores tendrán una guía por la cual orientarse para la implementación del código. Dichas tareas son escritas en lenguaje técnico, lo que le posibilita al programador una visualización de la estructura a nivel de código del software. A continuación será expuesto un ejemplo de estas tareas, el resto de las mismas puede visualizarse en el **Anexo 8: Tareas de ingeniería**.

Tabla 37: Tareas de implementación (No: 8)

Tarea	
Número: 8	
Nombre de la tarea: HU 3: Consultar mensajería instantánea.	
Tipo de tarea: Desarrollo.	**Estimación:** 15
Fecha de inicio: 27/02/2015	**Fecha de fin:** 14/03/2015
Descripción: Crear una función en el main.controler.js que se encargue del envío y recibo de mensaje denominada nuevaConversacion.	

3.3. Diagrama de despliegue

Un modelo de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos de hardware sobre los cuales pueden ejecutarse los elementos software. A través de este diagrama se conocerá la arquitectura física del sistema antes de comenzar su desarrollo, por tanto podremos modelar los nodos y las conexiones del modelo de despliegue tan pronto como comience el flujo de trabajo de los requisitos (RUMBAUGH, JACOBSON y BOOCH, 1999).

Aunque en la metodología XP no se define entre uno de sus puntos el modelado del diagrama de despliegue, los autores de la presente investigación han decidido añadirlo para ofrecer un mejor entendimiento del funcionamiento del sistema.

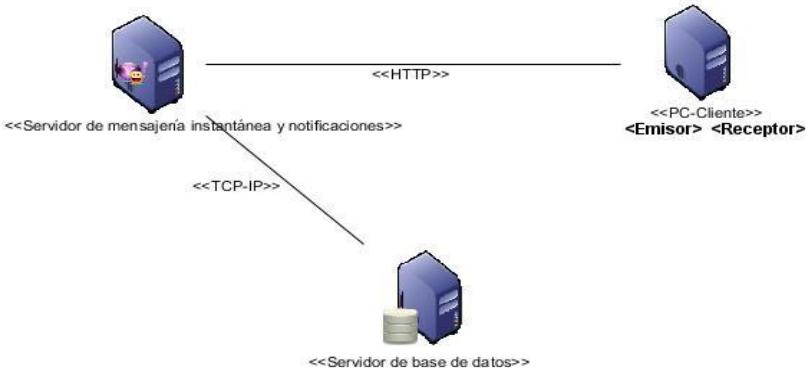


Imagen 15: Diagrama de despliegue

3.4. Estándares de código

El uso de estándares es muy promovido por XP, de manera que sea fácilmente entendible por el equipo de desarrollo, y que de esta manera se faciliten las labores de recodificación (JOSKOWICZ, 2008). Es por ello que para la confección de sistemas informáticos, el centro FORTES ha identificado una serie de estándares, los cuales serán utilizados en el desarrollo del software a efectuarse, además de otros estándares definidos por los desarrolladores de la aplicación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Al inicio de un desarrollo de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Para un programador comprender bien un sistema de software, influye directamente la legibilidad del código fuente. La mantenibilidad del

código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Aunque estas dos características (legibilidad y mantenibilidad) son el resultado de muchos factores, la técnica de codificación es una faceta del desarrollo de software en la que todos los programadores influyen especialmente. Establecer un estándar de codificación asegura que un equipo de programadores mantenga un código de calidad. Para contribuir a que un sistema de software sea fácil de comprender y mantener, se puede aplicar de forma continuada un estándar de codificación bien definido, utilizar técnicas de programación apropiadas, y posteriormente, efectuarse revisiones del código de rutinas (PEREDA, 2014).

Dentro de los estándares definidos por los autores de la presente investigación, se tiene que para el ordenamiento y la estructura del código fue utilizado el archivo .jshintrc, el cual contiene un conjunto de normas que reorganizan y velan por el uso de estándares de código. Un ejemplo de los mismos es la norma strict, la cual obliga a que todos los documentos de javascript contengan la línea 'use strict', entre otros muchos que facilitan la organización y la limpieza del código.

Los estándares establecidos por el centro FORTES pueden ser observados en cuanto a la estructura como es el caso de:

- ✓ Adición de un solo espacio después de cada delimitador coma.
- ✓ Adición de un solo espacio alrededor de los operadores (==, &&,…).
- ✓ Adición de una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último.
- ✓ Uso de llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.
- ✓ Añade una línea en blanco antes de las declaraciones *return*, a menos que el valor devuelto solo sea dentro de un grupo de declaraciones (tal como una declaración *if*).
- ✓ Para cada clase controladora se definió nombre.controller.js
- ✓ Para todas las funciones y variables se utilizó el estándar Camel, donde todos los nombres comienzan con minúscula y si es un nombre compuesto la primera inicial de la segunda palabra se escriben con mayúscula.

3.5. Validación de entradas y errores del sistema

Para obtener un software con un alto grado de calidad es necesario verificar aquellas entradas de datos efectuadas por el usuario. Se debe verificar cada acción para que el sistema responda acorde a dicha acción ejecutada. En la aplicación existen dos entradas de datos fundamentales, la primera que es gestionar usuario (HU 1) y la segunda autenticar usuario (HU 2).

La funcionalidad gestionar usuario está compuesta por tres acciones, mostrar usuario, modificar usuario e insertar usuario. Cuando un usuario se registra se le solicita su nombre, contraseña y la dirección de correo electrónico. Para efectuar un correcto registro, el sistema verifica el número de caracteres del nombre, donde si el mismo es menor que 2 y supera a los 15 caracteres el sistema envía un mensaje de error. En igual caso ocurre con la contraseña, la cual debe tener como mínimo un total de 8 caracteres y un máximo de 16, y por último la dirección de correo electrónico debe tener el formato correo@dominio.org. En el caso de modificar, el usuario debe de completar todos los campos que se mostrarán en la ventana, siendo cada uno de ellos de carácter obligatorio. La autenticación por su parte, tiene como entrada la dirección de correo y la contraseña con el cual el usuario fue registrado anteriormente. Cada campo de la autenticación es de carácter obligatorio, es decir, el usuario debe completar cada campo que aparece visualizado. También serán validados elementos como el correo y la contraseña, los cuales deben cumplir los mismos parámetros que los mencionados anteriormente para ejecutar el registro de usuario, de no ser de esta manera el sistema enviará mensajes con los errores presentados.

3.6. Estrategias de prueba

La validación del software se utiliza para mostrar que el sistema se ajusta a su especificación y que cumple con las expectativas del usuario. Implica procesos de comprobación, como las inspecciones y revisiones. Las pruebas deben pasar por un proceso que consta de tres etapas en la cual se prueban los componentes del sistema, la integración del sistema y el sistema con los datos del cliente (SOMMERVILLE, 2005). En los próximos puntos se determinará las estrategias a seguir para la validación del software, donde se definirán elementos referentes a las pruebas a realizar según la metodología escogida y el tipo de sistema que se implementó.

3.6.1. Tipos de prueba

Para la realización de las pruebas al software elaborado se ha trazado como estrategia la utilización de dos tipos de pruebas, pruebas de caja blanca y pruebas de caja negra. Las de caja blanca están centradas en verificar qué líneas específicas de código funcionan tal como están definidas. También se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, los bucles y condiciones, y examinado el estado del programa en varios puntos. Las pruebas de caja negra están centradas en los requisitos funcionales, las cuales son realizadas sobre la interfaz del software y están enfocadas en las entradas y salidas y no en el código fuente (PRESSMAN, 2005).

Las pruebas de caja blanca y de caja negra definen niveles de pruebas. Las pruebas unitarias, pertenecen a las pruebas de caja blanca y las de aceptación y de integración a las de caja negra. A continuación se hace referencia a los niveles de prueba utilizados.

3.6.2. Niveles de prueba

Las pruebas unitarias: Son definidas como una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas deben ser guardadas junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo (JOSKOWICZ, 2008). La utilización de los test automatizados es uno de los factores que más calidad aporta a un proyecto. Los sistemas de *tests* empleados por defecto en Angular, tanto unitarios como de extremo a extremo, se basan en el uso de Karma, un lanzador de *tests* que se integra con distintos *frameworks* (Jasmine, QUnit, Mocha, etc.) (AZAUSTRE, 2013), donde en la aplicación confeccionada fue utilizado Jasmine. El *framework* Jasmine es orientado a Desarrollo Dirigido por Comportamientos[14] (BDD) para realizar pruebas unitarias (*Unit Testing*) de código JavaScript (GODÍNEZ, 2015). Se pueden encontrar en Jasmine características que favorecen la aplicación de test para el proyecto desarrollado, tales como el soporte de pruebas asíncronas, sintaxis natural para poder probar comportamientos con BDD y personalización de diferentes componentes como matchers y Mocking (AZAUSTRE, 2013). A continuación se muestra un ejemplo de las pruebas unitarias efectuadas:

[14] BDD: Behavior Driven Development.

```
describe('E2E Testing for Contact App', function(){
    var contactPageObjects = require('../ContactListPageObjects.js');

        beforeEach( function () {
        browser.get('/');//navigates the router to the route
        browser.waitForAngular();
    });

    it('Should have a contact Add Button', function () {
        expect(contactPageObjects.addContactButton).toBeDefined();
    });

    it('Should open a modal form', function () {
        contactPageObjects.addContactButton.click();
```

Imagen 16: Resultados de las pruebas unitarias

Pruebas de integración

El proceso del sistema implica construir este a partir de sus componentes y probar el sistema resultante para encontrar problemas que puedan surgir debido a la integración de los componentes. Cuando se planifica la integración en un proceso como XP, el cliente se implica en el proceso de desarrollo y decide qué funcionalidad debería incluirse en cada incremento del sistema (SOMMERVILLE, 2005).

Para la aplicación de las pruebas de integración fue necesario seguir una serie de pasos, los cuales son descritos de la siguiente manera:

- ✓ Crear un HTML que se encargue de conectar el servidor de la aplicación desde otra PC.
- ✓ Ubicar en la región del sistema a utilizar el HTML generado anteriormente.

Pruebas de aceptación

Es la etapa final en el proceso de pruebas antes de que se acepte que el sistema se ponga en funcionamiento. Se prueba con los datos proporcionados por el cliente más que con datos de prueba simulados, debido a la diferencia existente entre los datos reales y los de prueba. Las pruebas de aceptación pueden revelar problemas en los requerimientos, donde los recursos del sistema no cumplen las necesidades del usuario o donde el desempeño del sistema es inaceptable (SOMMERVILLE, 2005).

3.6.3. Diseño de casos de prueba

Para la realización de los diseños de caso de prueba se tomaron en cuenta cada historia de usuario y su número de iteración. En el **Anexo 9: Diseños de caso de prueba**, se encuentran todos los diseños identificados para la realización de las pruebas**.**

3.6.4. Resultados obtenidos

Resultados de las pruebas unitarias: Las pruebas unitarias realizadas con Karma y el *framework* Jasmine tuvieron como resultado un alto nivel en cuanto a la calidad del código que se implementó. Gracias a las mismas se disminuyó considerablemente el tiempo de despliegue de la aplicación evitando errores de programación.

Resultados de las pruebas de integración: Las pruebas unitarias arrojaron resultados satisfactorios para el cumplimiento de los objetivos planteados al comienzo de la presente investigación. El componente desarrollado logró integrarse perfectamente con otro sistema, donde los usuarios registrados en el mismo, pasaron a ser parte de la aplicación desarrollada.

Resultados de las pruebas de aceptación: Las pruebas fueron realizadas en tres iteraciones con el objetivo de detectar los posibles errores e inconsistencias que pudiera presentar la aplicación en su puesta en funcionamiento. Las iteraciones fueron aplicadas a las pruebas de integración y a las pruebas de aceptación, para de esta manera corregir los errores encontrados. Del resultado de dichas pruebas fueron arrojados los siguientes elementos:

1ra Iteración: Se obtuvo un total de 16 inconformidades, de ellas 14 se identificaron como significativas, 1 se identificó como no significativa y 1 recomendación.

2da Iteración: Se identificaron 10 inconformidades, de ellas 7 se identificaron como significativas, 1 se identificó como no significativa y 2 recomendaciones.

3ra Iteración: Se identificaron 3 no conformidades, de ellas 2 se identificaron como no significativas y 1 como recomendación.

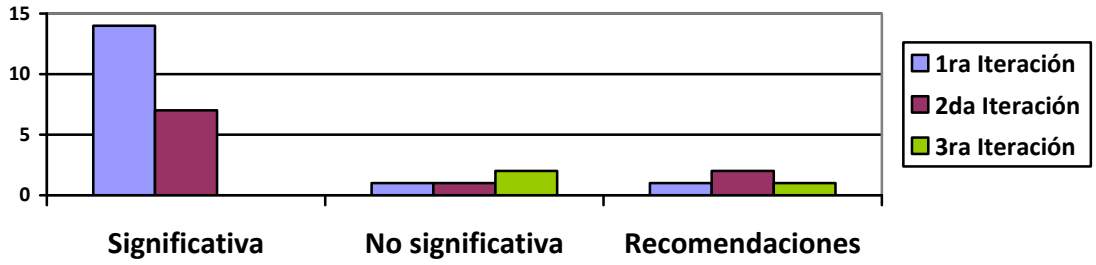


Imagen 17: Resultados de las pruebas de aceptación

3.7. Conclusiones parciales del capítulo:

Al concluir este capítulo se da cierre a la realización de la aplicación, obteniendo los siguientes resultados:

1. Se realizó el proceso de implementación de la aplicación a través de la creación de tareas de ingeniería, utilizando a su vez estándares de código.
2. Se aplicaron estrategias de pruebas y se definieron diseños de casos de prueba para garantizar la calidad óptima del software.
3. Los resultados obtenidos en la implementación del sistema, permitieron identificar mejoras en el proceso de detección de vulnerabilidades, para lo cual se establecieron normas para garantizar la seguridad del sistema desarrollado.

Conclusiones generales:

Al término de la investigación se arriba a las siguientes conclusiones, evidenciando el cumplimiento de los objetivos propuestos:

1. Se realizó un análisis del estado del arte sobre los sistemas de mensajería instantánea y notificaciones para determinar las características que debía contener la aplicación para su desarrollo, así como las herramientas necesarias para la confección del mismo.
2. Se construyó una aplicación reutilizable y con una adecuada velocidad de respuesta aplicando la metodología XP y lenguajes como JavaScript, el f*ramework* Node.js, Angular, Express, JQuery, Websockets y Boostrap, logrando de esta manera disminuir la diversidad tecnológica existente en las aplicaciones de mensajería desarrolladas en el centro.
3. Se obtuvo un componente de mensajería instantánea y notificaciones capaz de integrarse con las aplicaciones web desarrolladas para el centro FORTES, que garantiza la comunicación entre los usuarios a través del chat y salas de chats.
4. Se establecieron normas de seguridad para garantizar que la aplicación confeccionada no contenga vulnerabilidades.
5. La aplicación de las pruebas arrojaron resultados satisfactorios, permitiendo de esta manera la validación de los requisitos establecidos y garantizando la calidad del sistema.

Recomendaciones:

Seguidamente se muestran las recomendaciones identificadas para futuras versiones del software creado:

1. Insertar funcionalidades como foros de debates y trasferencias de archivo utilizando la mensajería instantánea para el envío de los mismos.
2. Insertar en el chat funcionalidades como bloquear usuario, ordenar historial por fechas y uso de emoticonos.
3. Mostrar imágenes de los usuarios como avatar en la lista de amigos.
4. Mostar las notificaciones utilizando el correo electrónico por el cual se registra y se autentica el usuario.

Referencias bibliográficas:

1. **Alfonso Melián. 2014**. Definición de “comunicación sincrónica” y “comunicación asincrónica”. [En línea] 5 de febrero de 2014. https://sites.google.com/site/actividad3alfonsobelloniniz/5-definir-comunicacion-sincronica-y-comunicacion-asincronica.
2. **Álvarez, Miguel Angel. 2014**. desarrolloweb.com. [En línea] 2 de enero de 2014. http://www.desarrolloweb.com/articulos/que-es-mvc.html.
3. —. **2009. Introducción a JQuery**. [En línea] 2009. http://www.desarrolloweb.com/articulos/introduccion-JQuery.html.
4. **Arquitectura de información y usabilidad: nociones básicas para los profesionales de la información.** Bustamante, Antonio Montes de Oca Sánchez de. 2004. 1, La Habana: Universidad de La Habana, 2004.
5. **Arquitectura de la Información: diseño e implementación**. Cam, Celso Gonzales. 2003. Lima: Departamento de Ciencias de la Información Pontificia Universidad Católica del Perú, 2003.
6. **AulaClic. 2014.** Aula Clic. Unidad 7. Foros y Grupos de discusión (I). [En línea] septiembre de 2014. http://www.aulaclic.es/internet/t_7_1.htm.
7. **Beck M. Fowler, J. Brant. 1999**. controlchaos.com. [En línea] Addison-Wesley, 1999. www.controlchaos.com.
8. **Beyond methods and CASE: The software engineering process with its integral support.** Ivar Jacobson, Sten Jacobson. 1995. s.l.: Object Magazine, 1995.
9. **Brown, Pamela. 2013**. NSA mines Facebook for connections, including Americans' profiles - CNN.com. [En línea] 30 de 9 de 2013. [Citado el: 05 de 06 de 2014.] http://www.cnn.com/2013/09/30/us/nsa-social-networks/index.html.
10. **Cabello, Carlos. 2013**. AnexoM - Blog oficial de Jazztel. [En línea] 2013.
11. —. 2013. AnexoM - Blog oficial de Jazztel. [En línea] 2013.
12. **Cañellas, Alicia Mayor. 2014**. LMS y LCMS: Funcionalidades y beneficios. centrocp.com. [En línea] 13 de julio de 2014. www.centrocp.com/lms-y-lcms-funcionalidades-y-beneficios/.

13. **Claroline. 2014. Claroline**. [En línea] 2014. http://www.claroline.net/?lang=es.

14. **Conde, Maritza. 2014**. Comunicación sincrónica y asincrónica. [En línea] 5 de febrero de 2014. http://maritzacondeitzmoyotl.wikispaces.com/COMUNICACI%C3%93N+SINCRONICA+Y+ASINC.

15. **Control de versiones con Subversion**. Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. 2011. 2011.

16. **Corporation, Oracle. 2014**. Java Magazine. 2014.

17. **Cruz, Daniel Sierra. 2015**. slideshare. Visual Paradigm For Uml. [En línea] 2015. http://es.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

18. **Daileen Jardínez Amado, Jeiser Medrano Abreu. 2014**. Herramienta de mensajería instantánea y notificaciones para el marco de trabajo Xalix. 2014.

19. **Daniel Sierra Cruz. 2015**. slideshare. Visual Paradigm For Uml. [En línea] 2015. http://es.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

20. **Data Model Patterns: Conventions of thought**. Hay, David C. 1996. S.l. : Dorset House Publishing,, 1996. ISBN 0-932633-29-3.

21. **Desarrollo ágil con AngulaJS**. Azaustre, Carlos. 2013. 2013.

22. **Dickson, Mike. 2007**. An examination into Trillian basic 3.x contact identification. [En línea] 13 de 01 de 2007. [Citado el: 12 de 02 de 2015.]

23. **Dokeos. 2014**. Dokeos. [En línea] 2014. http://www.dokeos.com/es.

24. **DRAE. 2014**. Real Academia Española. Diccionario de la Real Academia Española. [En línea] 2014. http://lema.rae.es/drae/?val=notificaciones.

25. **Dubretic, Marija. 2014**. NoSQL vs SQL – ¿Tiempo para un Cambio? | Spanish blog. [En línea] 2014. https://blog.udemy.com/es/nosql-vs-sql-tiempo-para-un-cambio/.

26. **E Kendall, Julie Kendall. 2005**. books.google. Análisis de diseño de sistema. [En línea] 2005. www.books.google.cu/.

27. **El reto impuesto por las herramientas colaborativas en la gestión del conocimiento en la Universidad de las Ciencias Informáticas**. Yidian Yosbel Castellanos, Yusdel Meriño, Dariela Elvira Espinosa. 2012. La Habana: Revista Cubana de las Ciencias Informáticas, 2012.
28. **El uso de foros como espacio educativo**. Martínez, María Guadalupe Lara. 2010. México: s.n., 2010.
29. **Escribano, Gerardo Fernández. 2002**. Introducción a Extreme Programming. 2002.
30. **Estándar de codificación (XALIX).** Díaz, Ernesto Vladimir Pereda. 2014. La Habana: UCI, 2014. V2.0.
31. **Estudio sobre el uso de los foros virtuales para favorecer las actividades colaborativas**. Salvat, Begoña Gros. 2014. España: Universidad de Barcelona, 2014.
32. **Eugenia Bahit, Indira Burga, María José Montes Díaz, Milagros Infante Montero, Sergio Infante Montero. 2012**. Hackers & Developers Magazine #2. 2012.
33. **Freiría, Germásn Alberto Tizón. 2008**. Las TIC en educación. S.l.: Lulupress.inc, 2008.
34. **García, Guillermo Roquet. 2004**. Los chat y su uso en la educación. [En línea] 12 de agosto de 2004. http://es.slideshare.net/EtiaHR/los-chat-y-su-uso-en-educacin.
35. **Glass, Robert L. 2003**. Facts and Fallacies of Software Engineering. S.l.: Addison-Wesley, 2003. ISBN 0321117425.
36. **Glock, Jonathan. 2013**. Node.js Framework Comparison: Express vs. Koa vs. Hapi. [En línea] 2013. https://www.airpair.com/node.js/posts/nodejs-framework-comparison-express-koa-hapi.
37. **Godínez, Marco. 2015**. JavascriptMX. [En línea] Javascript.org, 2015. http://javascriptmx.com/blog/pruebas-con-jasmine/.
38. **Introdución a los sistemas de Bases de Datos**. Date, C J. 2009. 2009.
39. **Irene Rodil Jiménez, Camino Pardo de la Vega. 2010**. Operaciones auxiliares con tecnologías de la información y las comunicaciones. España: Paraninfo, 2010.
40. **J. Highsmith, K Orr. 2000**. crystalmethodologies.org. Adaptative Software Development: A Colaborative Approach to Managing Complex Systems. [En línea] 2000. www.crystalmethodologies.org.
41. **James Rumbaugh, Ivar Jacobson, Grandy Booch. 1999**. El Lenguaje Unificado de Modelado. Manual de Referencia. La Habana: Félix Varela, 1999.

Anexos:

Anexo 1: Comparación de metodologías

Tabla 2: Comparación de metodologías

Metodología Ágil	Metodología Tradicional
Pocos artefactos. El modelo es prescindible, modelos desechables.	Más Artefactos. El modelo es esencial, mantenimiento de modelos.
Pocos Roles, más genéricos y flexibles.	Más roles, más específicos.
No existe un contrato tradicional, debe ser bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Orientada a proyectos pequeños. Corta duración(o entregas frecuentes), equipos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos.
La arquitectura se va definiendo y mejorando a lo largo del proyecto.	Se promueve a que la arquitectura se defina temporalmente en el proyecto.
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo.	Énfasis en las definiciones del proyecto: roles, actividades y artefactos.
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en heurísticas provenientes de estándares seguidos por el entorno de desarrollo.
Se esperan cambios durante todo el proyecto.	Se espera que no ocurran cambios durante el proyecto.
Diferencia entre metodologías ágiles y tradicionales.	

Anexo 2: Ranking de agilidad

Tabla 3: Ranking de agilidad

	ASD	Crystal	DSDM	FDD	LD	Scrum	XP
Sistema como algo cambiante	5	4	3	3	4	5	5
Colaboración	5	5	4	4	4	5	5
Características de las metodologías(CM)							
Resultados	5	5	4	4	4	5	5
Simplicidad	4	4	3	5	3	5	5
Adaptabilidad	5	5	3	3	4	4	3
Excelencia técnica	3	3	4	4	4	3	4
Prácticas de colaboración	5	5	4	3	3	4	5
Media CM	4,4	4,4	3,6	3,8	3,6	4,2	4,4
Media Total	4,8	4,5	3,6	3,6	3,9	4,7	4,8
Ranking de agilidad(Los valores más altos representan una mayor agilidad)							

Anexo 3: Fases de la metodología XP

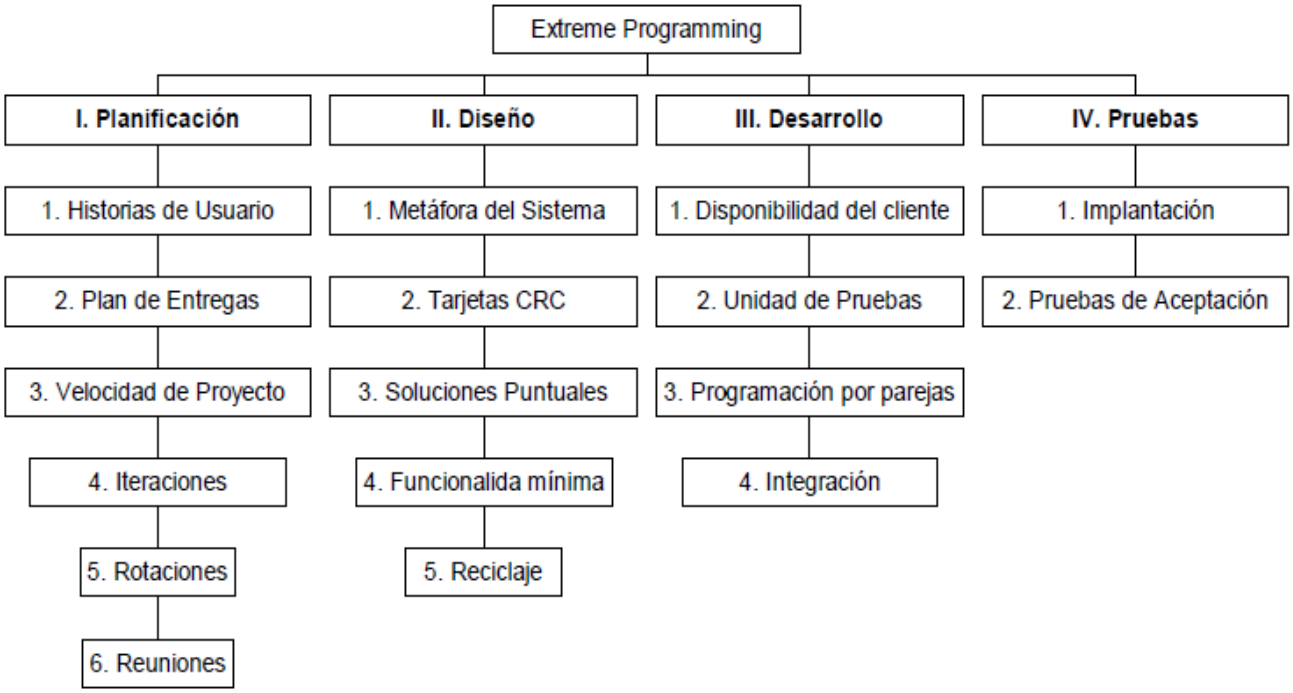


Imagen 1: Trabajando con XP

Anexo 4: Frameworks para Node.js

Tabla 4: Frameworks para Node.js

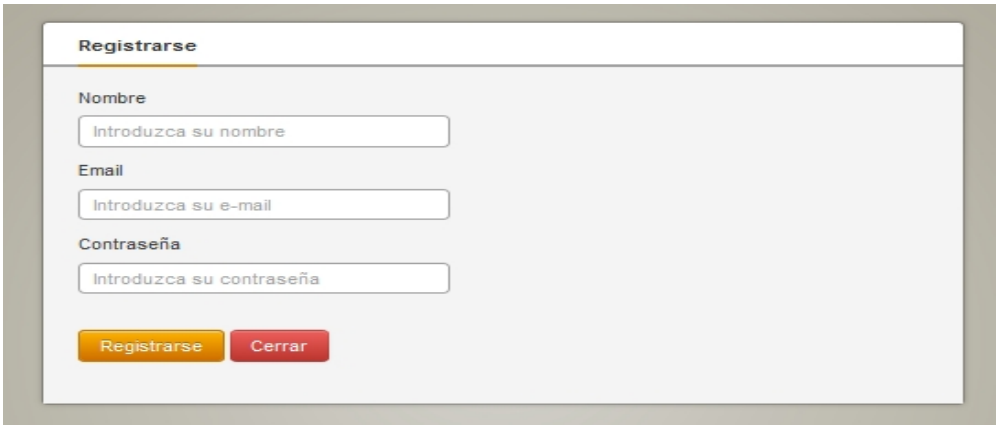
Framework para Node.js			
Métricas	**Express**	**Koa js**	**Hapi**
Github Proyectos	16 158	4846	3283
Colaboradores	163	49	95
Dependencia de Paquetes	3828	99	102
Preguntas	11419	72	82

Anexo 5: Historias de usuario

HU 1: Gestionar usuario en el sistema

- ✓ ***RF Insertar usuario***

Tabla 7: Historia de usuario (HU 1: No: 1)

Historia de usuario	
Número: 1	**Nombre del requisito:** Insertar usuario
Iteración Asignada: 1	
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Bajo	
Descripción: Se visualizará una ventana donde se podrá introducir el nombre, el correo electrónico y la contraseña del usuario.	
Observaciones: El usuario no tiene que poseer una cuenta para realizar esta acción.	
Prototipo de interfaz: 	

- ✓ ***RF Modificar datos de usuario***

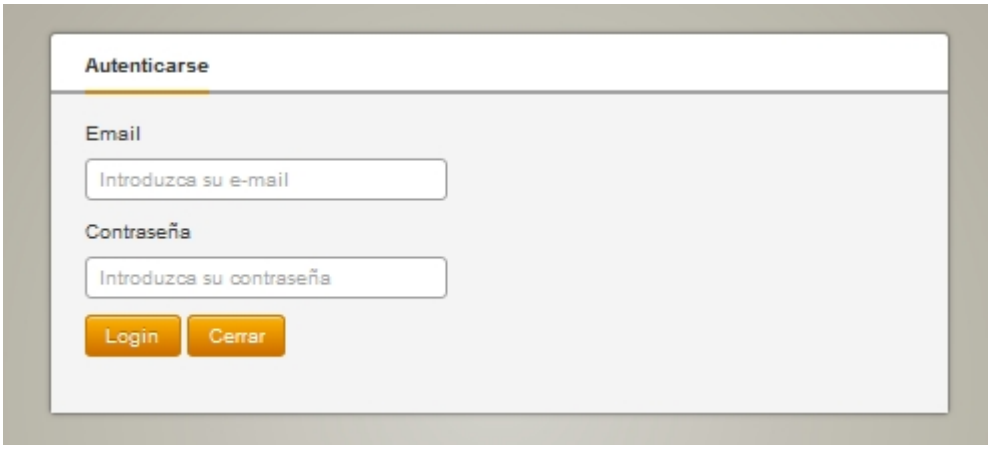
Tabla 8: Historia de usuario (HU 1: No: 2)

Historia de usuario	
Número: 2	**Nombre del requisito:** Modificar datos del usuario
Iteración Asignada: 1	
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Bajo	
Descripción: La opción se encuentra en el menú de opciones, donde al seleccionar modificar datos de usuario se visualizará una ventana donde se podrá introducir las nuevas características como son el nombre, el correo electrónico y la contraseña del usuario.	
Observaciones: El usuario debe estar autenticado para realizar esta acción (HU 2: No: 3).	
Prototipo de interfaz: 	

HU 2: Autenticar usuario en el sistema.

- ✓ ***RF Autenticar usuario en el sistema***

Tabla 9: Historia de usuario (HU 2: No: 3)

Historia de usuario	
Número: 3	**Nombre del requisito:** Autenticar usuario en el sistema
Iteración Asignada: 1	
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: Bajo	
Descripción: Se mostrará en la parte superior de la lista de amigos la opción para autenticarse, que se mostrará una ventana que le permitirá al usuario autenticarse introduciendo su correo electrónico y su contraseña.	
Observaciones: El usuario debe haberse registrado con anterioridad(HU 1:No: 1)	
Prototipo de interfaz: 	

HU 3: Consultar mensajería instantánea

- ✓ ***RF Iniciar conversación privada con un amigo***
- ✓ ***RF Enviar mensaje a un amigo***
- ✓ ***RF Recibir mensaje de un amigo***
- ✓ ***RF Cerrar conversación***
- ✓ ***RF Mostrar estado del usuario***
- ✓ ***RF Cambiar estado del usuario***

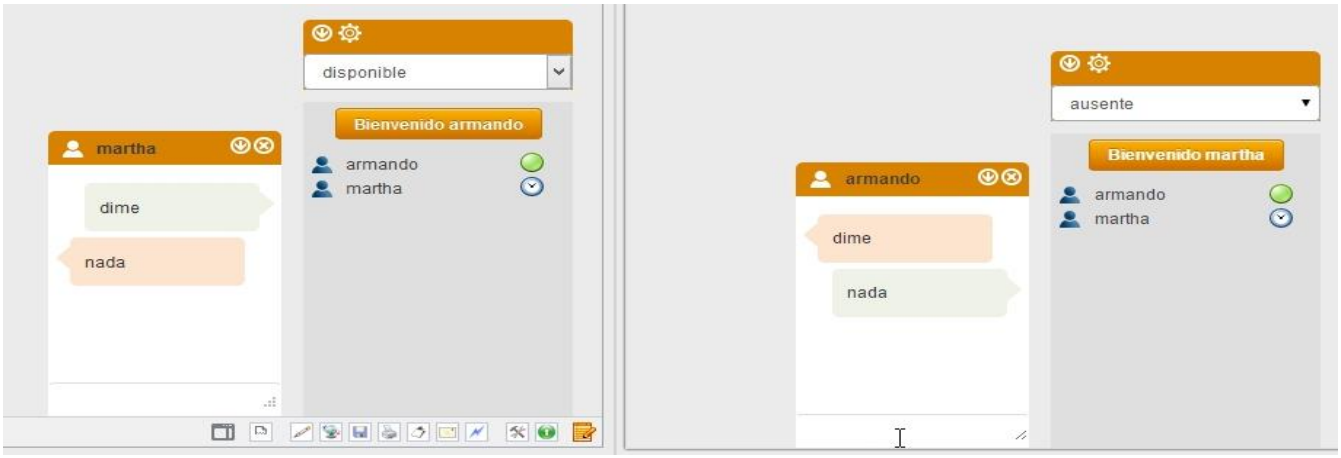
Tabla 10: Historia de usuario (HU 3: No: 4)

Historia de usuario	
Número: 4	**Nombre del requisito:** Iniciar conversación privada con un amigo. Enviar mensaje a un amigo. Recibir mensaje de un amigo. Cerrar conversación. Mostrar estado del usuario. Cambiar estado del usuario.
Iteración Asignada: 2	
Prioridad: Alta	**Tiempo Estimado:** 15 días
Riesgo en Desarrollo: Bajo	
Descripción: El usuario podrá establecer una comunicación con uno o más usuarios. Después de que el usuario emisor envíe un mensaje al usuario receptor se desplegará la ventana de conversación privada mostrando el mensaje enviado. La opción de cerrar conversación estará ubicada en la parte superior derecha de la ventana de	

conversación, y será capaz de cerrar una conversación si el usuario selecciona el ícono(X). En la parte superior de la lista de contactos se mostrará el estado del usuario el cual puede ser visualizado de disímiles colores, este estado puede ser modificado por el usuario cuando lo estime conveniente ejecutando clic sobre el botón de disponibilidad y cambiando al estado con el cual desea el usuario ser visualizado. En el caso de que se encuentre con color rojo, significará que el usuario se encuentra ocupado, cuando es de color verde tendrá como significado de que el usuario se encuentra disponible para entablar una conversación y cuando el ícono tome color azul será para indicar que el usuario está ausente.

Observaciones: El usuario debe estar previamente autenticado en el sistema (HU 2: No: 3).

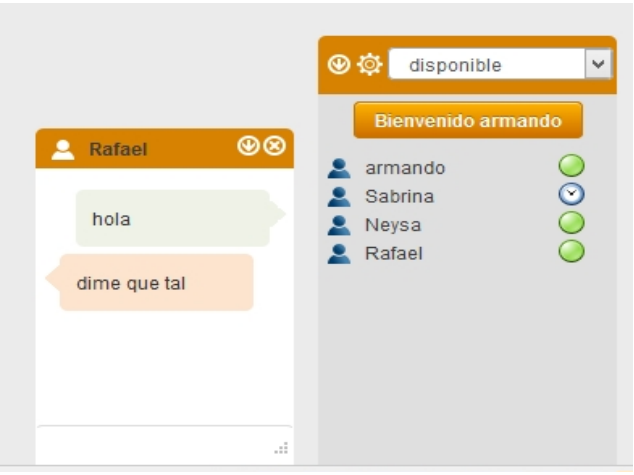
Prototipo de interfaz:



HU 4: Mostar historial para la mensajería instantánea

- ✓ ***RF Mostrar mensajes de historial***

Tabla 11: Historia de usuario (HU 4: No: 5)

Historia de usuario	
Número: 5	**Nombre del requisito:** Mostrar mensaje de historial
Iteración Asignada: 2	
Prioridad: Media	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Bajo	
Descripción: Para acceder a esta opción el usuario debe haber abierto una conversación, donde se mostrarán los mensajes almacenados de conversaciones anteriores.	
Observaciones: El usuario debe estar previamente autenticado en el sistema (HU 2: No: 3) y debe haber mantenido una conversación con otro usuario (HU 3: No: 4).	
Prototipo de interfaz: 	

HU 5: *Gestionar amigo para la mensajería instantánea*

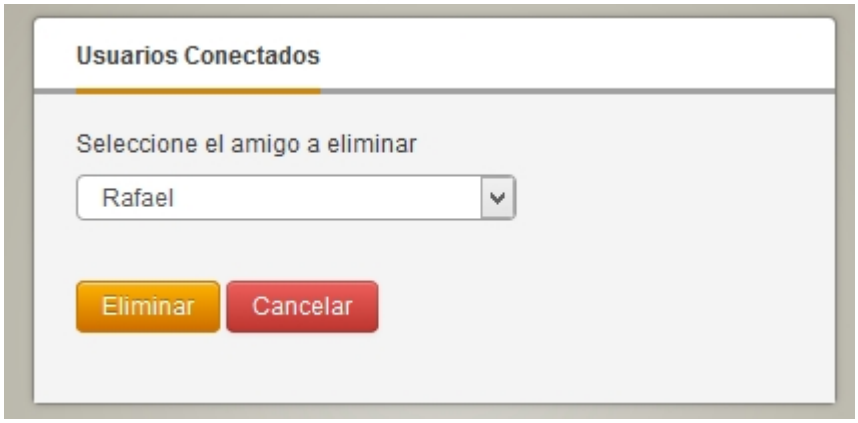
- ✓ ***RF Mostar amigo***

Tabla 12: Historia de usuario (HU 5: No: 7)

Historia de usuario	
Número: 7	**Nombre del requisito:** Mostrar amigo
Iteración Asignada: 1	
Prioridad: Media	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Bajo	
Descripción: Se mostrarán en la lista de contactos el nombre de cada amigo y su estado de disponibilidad.	
Observaciones: El usuario debe estar previamente autenticado(HU 2: No: 3).	
Prototipo de interfaz: 	

- ✓ ***RF Eliminar amigo***

Tabla 13: Historia de usuario (HU 5: No: 8)

Historia de usuario	
Número: 8	**Nombre del requisito:** Eliminar amigo
Iteración Asignada: 1	
Prioridad: Media	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Bajo	
Descripción: El usuario debe haber marcado el menú de opciones y luego seleccionar la opción eliminar amigo. Será mostrada una ventana donde el usuario seleccionará el amigo que desea eliminar de su lista de contactos.	
Observaciones: El usuario debe estar previamente autenticado (HU 2: No: 3) y debe tener al amigo que desea eliminar en la lista de contactos(HU 5: No: 7).	
Prototipo de interfaz: 	

HU 6: Mostrar notificaciones

- ✓ ***RF Mostar notificaciones***

Tabla 14: Historia de usuario (HU 6: No: 9)

Historia de usuario	
Número: 9	**Nombre del requisito:** Mostar notificaciones
Iteración Asignada: 2	
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Bajo	
Descripción: La notificación inicialmente aparece en la esquina superior izquierda de la pantalla y son ejecutadas ante eventos como la conexión y desconexión de un usuario.	
Observaciones: El usuario debe estar previamente autenticado (HU 2: No: 3).	
Prototipo de interfaz: 	

HU 7: Configurar notificaciones

- ✓ ***RF Ajuste de sonido***
- ✓ ***RF Ajuste de posición***

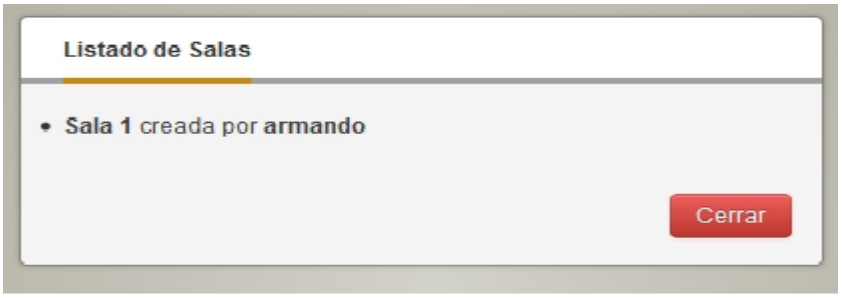
Tabla 15: Historia de usuario (HU 7: No: 10)

Historia de usuario	
Número: 10	**Nombre del requisito:** Configurar notificaciones
Iteración Asignada: 2	
Prioridad: Media	**Tiempo Estimado:**10 días
Riesgo en Desarrollo: Bajo	
Descripción: Para realizar esta acción se debe seleccionar en el panel que se encuentra en la parte superior de la lista de contactos la opción configurar notificación para posteriormente realizar las configuraciones que sean convenientes. Posteriormente se mostrará una ventana con las configuraciones a las cuales el usuario tiene acceso como son la activación y desactivación del sonido y el posicionamiento de las notificaciones. El posicionamiento se mostrará por defecto en la parte superior izquierda de la pantalla, y puede ser cambiado para la parte superior derecha o izquierda de la pantalla.	
Observaciones: El usuario debe estar previamente autenticado (HU 2: No: 3).	
Prototipo de interfaz: 	

HU 8: Gestionar sala de chat

- ✓ ***Mostrar sala de Chat***

Tabla 16: Historia de usuario (HU 8: No: 11)

Historia de usuario	
Número: 11	**Nombre del requisito:** Mostrar sala de chat.
Iteración Asignada: 3	
Prioridad: Alta	**Tiempo Estimado:** 4 días
Riesgo en Desarrollo: Bajo	
Descripción: La opción para entrar en una sala estará ubicada en el menú de opciones que se ubica en la parte superior de la lista de contactos, donde se seleccionará la sala que se desee mostrar.	
Observaciones: El usuario debe estar previamente autenticado (HU 2: No: 3) y deben haber salas creadas (HU: 8 No: 12).	
Prototipo de interfaz: 	

- ✓ ***Crear Sala de Chat***

Tabla 17: Historia de usuario (HU 8: No: 12)

Historia de usuario	
Número: 12	**Nombre del requisito:** Crear sala de chat.
Iteración Asignada: 3	
Prioridad: Media	**Tiempo Estimado:** 6 días
Riesgo en Desarrollo: Alta	
Descripción: Para crear una sala de chat el usuario debe seleccionar en el menú de opciones crear sala de chat, donde se mostrará una ventana para introducir el nombre de la sala.	
Observaciones: El usuario debe estar previamente autenticado (HU 2: No: 3).	
Prototipo de interfaz: 	

- ✓ ***Eliminar Sala de Chat***

Tabla 18: Historia de usuario (HU 8: No: 13)

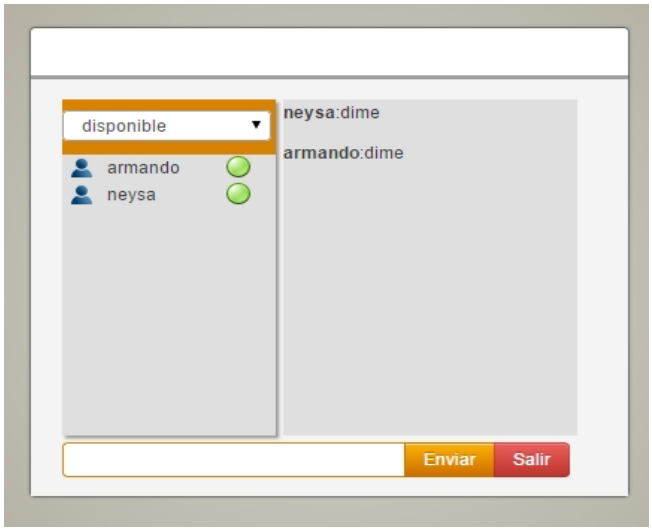
Historia de usuario	
Número: 13	**Nombre del requisito:** Eliminar sala de chat.
Iteración Asignada: 3	
Prioridad: Media	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Alta	
Descripción: Se seleccionará la(s) sala(s) de chat que se deseen eliminar y posteriormente se marcará el ícono "eliminar sala de chat" que se encuentra en la parte inferior izquierda de la ventana.	
Observaciones: El usuario debe estar previamente autenticado (HU 2: No: 3) y deben haber salas creadas (HU: 8 No: 12).	
Prototipo de interfaz: 	

HU 9: Consultar sala de chat

- ✓ **Mostrar sala de chat**
- ✓ ***Iniciar conversación grupal***
- ✓ ***Enviar mensaje a la sala de chat***
- ✓ ***Recibir mensaje en la sala de chat***
- ✓ ***Cerrar sala de chat***
- ✓ ***Mostrar estado del usuario en la sala de chat***
- ✓ ***Cambar estado del usuario en la sala de chat***

Tabla 19: Historia de usuario (HU 9: No: 14)

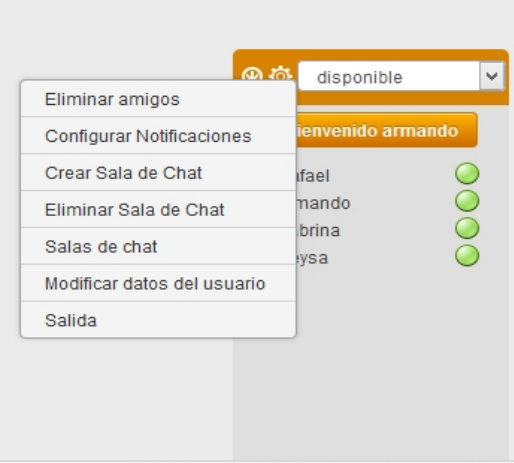
<table>
<tr><th colspan="3">Historia de usuario</th></tr>
<tr><td>Número: 14</td><td colspan="2">Nombre del requisito: Iniciar conversación grupal
Enviar mensajes a la sala de chat
Recibir mensajes en la sala de chat
Cerrar sala de chat
Mostrar estado del usuario</td></tr>
<tr><td colspan="3">Iteración Asignada: 3</td></tr>
<tr><td colspan="2">Prioridad: Media</td><td>Tiempo Estimado: 10 días</td></tr>
<tr><td colspan="3">Riesgo en Desarrollo: Bajo</td></tr>
<tr><td colspan="3">Descripción: Una vez seleccionada o creada la sala a la cual se desea pertenecer, se mostrará la lista de contactos a continuación de la ventana donde se recibirán y enviarán los mensajes entre los usuarios pertenecientes a una misma sala. En la parte superior izquierda de la lista de contactos estará ubicado el estado de disponibilidad del usuario, el cual puede ser modificado por el usuario. Para salir de la sala de se seleccionará un ícono que se visualizará al lado del nombre de la sala de chat que ha sido previamente seleccionada.</td></tr>
</table>

Observaciones: El usuario debe estar previamente autenticado (HU 2: No: 3) y haber seleccionado de la lista de salas de chat en la cual desea ser partícipe (HU 8: No: 11).
Prototipo de interfaz: 

HU 10: Salir del Sistema

- ✓ **Salir del sistema**

Tabla 20: Historia de usuario (HU 10: No: 15)

Historia de usuario	
Número: 15	**Nombre del requisito:** Salir del sistema
Iteración Asignada: 3	
Prioridad: Media	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: Alta	
Descripción: La acción permitirá al usuario salir del sistema cuando lo desee.	
Observaciones: El usuario debe estar previamente autenticado (HU 2: No: 3).	
Prototipo de interfaz: 	

Anexo 6: Diagrama de clases

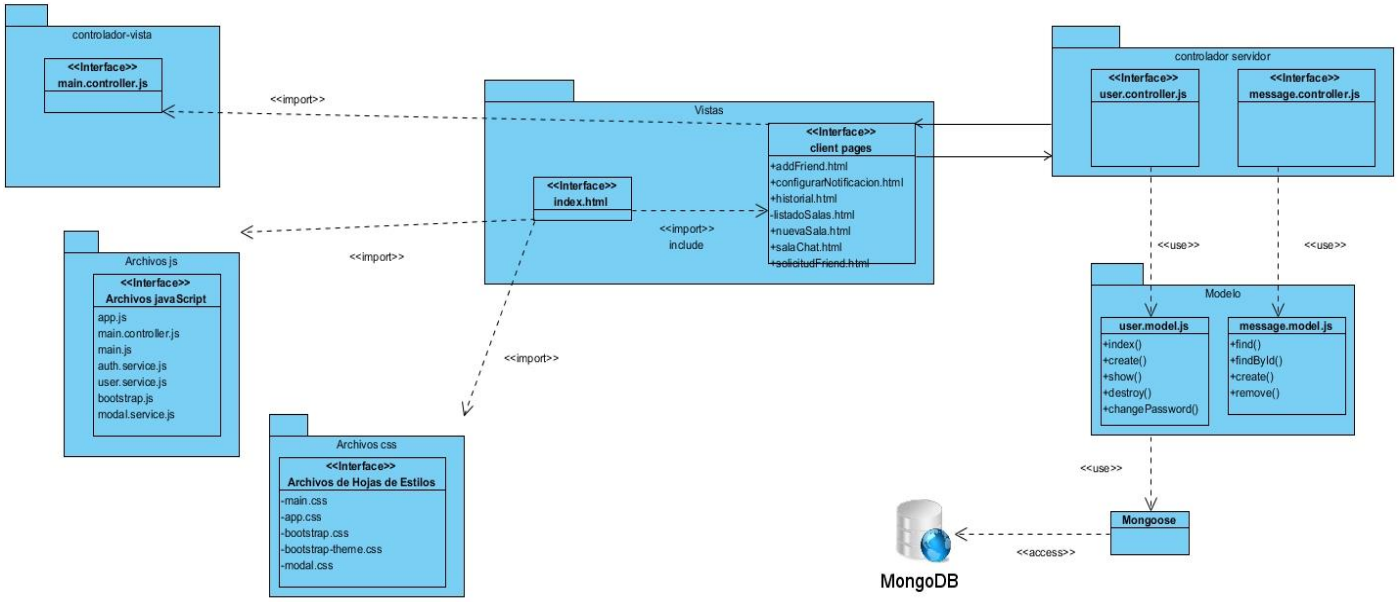


Imagen 5: Diagrama de clases

Anexo 7: Tarjetas CRC

Tabla 24: Tarjetas CRC (No: 2)

Tarjetas CRC	
Número: 2	**Nombre de la clase:** vistas
Responsabilidades: Contiene en su interior una página principal donde se ejecuta la aplicación, una plantilla que cambia en cada caso de uso, una librería o framework Angular, una librería o framework JQuery y una librería o framework bootstrap.	
Colaboraciones: archivos js, archivos css, controlador-servidor, controlador-vista	

Tabla 25: Tarjetas CRC (No: 3)

Tarjetas CRC	
Número: 3	**Nombre de la clase:** vistas
Responsabilidades: Contiene en su interior una página principal donde se ejecuta la aplicación, una plantilla que cambia en cada caso de uso, una librería o framework Angular, un librería o framework JQuery y una librería o framework bootstrap.	
Colaboraciones: archivos js, archivos css, controlador-servidor, controlador-vista	

Tabla 26: Tarjetas CRC (No: 4)

Tarjetas CRC	
Número: 4	**Nombre de la clase:** controlador-servidor
Responsabilidades: Controlador del lado del servidor [node], es el encargado de realizar las consultas al modelo, crear los mensajes nuevos y eliminarlos.	
Colaboraciones: modelo, vistas	

Tabla 27: Tarjetas CRC (No: 5)

Tarjetas CRC	
Número: 5	**Nombre de la clase:** modelo
Responsabilidades: Fichero donde se declara el modelo de cada entidad, en este caso de mensaje.	
Colaboraciones: modelo, vistas	

Tabla 28: Tarjetas CRC (No: 6)

Tarjetas CRC	
Número: 6	**Nombre de la clase:** Archivos js
Responsabilidades: Contiene en su interior un conjunto de librerías y archivos JavaScript que son incluidos en la aplicación para brindarle a esta funcionalidad.	
Colaboraciones: vistas	

Tabla 29: Tarjetas CRC (No: 7)

Tarjetas CRC	
Número: 7	**Nombre de la clase:** Archivos css
Responsabilidades: Contiene en su interior un conjunto de archivos u hojas de estilo que son incluidos en la aplicación para brindarle mejor maquetación y diseño a las vistas que se le muestran al cliente.	
Colaboraciones: vistas	

Anexo 8: Tareas de implementación

Iteración 1

Tabla 30: Tareas de implementación (No: 1)

Tarea	
Número: 1	
Nombre de la tarea: Selección de estándares de código.	
Tipo de tarea: Investigación	**Estimación:** 2
Fecha de inicio: 12/01/2015	**Fecha de fin:** 13/01/2015
Descripción: Investigar los estándares de código que se usarán para implementar para la aplicación.	

Tabla 31: Tareas de implementación (No: 2)

Tarea	
Número: 2	
Nombre de la tarea: Creación de la estructura de la aplicación.	
Tipo de tarea: Desarrollo	**Estimación:** 5
Fecha de inicio: 14/01/2015	**Fecha de fin:** 19/01/2015
Descripción: Para la creación de la estructura del sistema se tiene en cuenta la estructura del *framework* Yeoman que presenta dos carpetas: client y server, las cuales contienen la configuración del servidor así como los módulos que conforman la API y en client se almacena todo lo referente a las vistas.	

Tabla 32: Tareas de implementación (No: 3)

Tarea	
Número: 3	
Nombre de la tarea: Confección del modelo de datos.	
Tipo de tarea: Desarrollo.	Estimación: 3
Fecha de inicio: 20/01/2015	**Fecha de fin:** 23/01/2015
Descripción: Definir el modelo que soporte el flujo de información donde se crearan las clases message.model.js y usser.model.js, perteneciente en la API a los módulos Usser y Message.	

Tabla 33: Tareas de implementación (No: 4)

Tarea	
Número: 4	
Nombre de la tarea: Diseño y creación de las interfaces.	
Tipo de tarea: Desarrollo y diseño.	**Estimación:** 7
Fecha de inicio: 24/01/2015	**Fecha de fin:** 31/01/2015
Descripción: Crear las interfaces que se encuentran en el directorio client/app/account/login. Todo lo relacionado con los CSS y los Javascript se encuentran en la carpeta client/app/components/chat.	

Tabla 34: Tareas de implementación (No: 5)

Tarea	
Número: 5	
Nombre de la tarea: HU 1: Gestionar usuario.	
Tipo de tarea: Desarrollo.	**Estimación:** 10
Fecha de inicio: 1/02/2015	**Fecha de fin:** 11/02/2015
Descripción: Crear la clase main.controler.js para posteriormente implementar las funcionalidades modificar y eliminar, los cuales se encargan de modificar y eliminar a los usuarios respectivamente, estas son vistas independientes, una que permite modificar datos y la otra que permite eliminar.	

Tabla 35: Tareas de implementación (No: 6)

Tarea	
Número: 6	
Nombre de la tarea: HU 2: Autenticar usuario en el sistema.	
Tipo de tarea: Desarrollo.	**Estimación:** 3
Fecha de inicio: 12/02/2015	**Fecha de fin:** 15/02/2015
Descripción: Dentro de la clase main.controler.js crear el controlador LoginCtrl y después crear la funcionalidad de login (form), el cual debe obtener los datos, los valida y procede a autenticar a los usuarios en la aplicación. El logingOauth se encarga de desconectar al usuario.	

Tabla 36: Tareas de implementación (No: 7)

Tarea	
Número: 7	
Nombre de la tarea: HU 5: Gestionar amigo para la mensajería instantánea.	
Tipo de tarea: Desarrollo.	**Estimación:** 10
Fecha de inicio: 16/02/2015	**Fecha de fin:** 26/02/2015
Descripción: En la clase main.controller.js, crear la función verFriends que se encargada de visualizar los amigos. Crear dentro de esta misma clase la función eliminarFriends para eliminar los amigos.	

Iteración 2

Tabla 38: Tareas de implementación (No: 9)

Tarea	
Número: 9	
Nombre de la tarea: HU 4: Mostar historial para la mensajería instantánea.	
Tipo de tarea: Desarrollo.	**Estimación:** 5
Fecha de inicio: 15/03/2015	**Fecha de fin:** 20/03/2015
Descripción: Crear la clase socket.service.js y la función historial para que muestre los mensajes almacenados.	

Tabla 39: Tareas de implementación (No: 10)

Tarea	
Número: 10	
Nombre de la tarea: HU 6: Mostrar notificaciones.	
Tipo de tarea: Desarrollo.	**Estimación:** 5
Fecha de inicio: 21/03/2015	**Fecha de fin:** 26/03/2015
Descripción: Crear dos tipos de notificaciones, una para los usuarios conectados y otra para los usuarios desconectados, definidos en la función bienvenida.	

Tabla 40: Tareas de implementación (No: 11)

Tarea	
Número: 11	
Nombre de la tarea: HU 7: Configurar notificaciones.	
Tipo de tarea: Desarrollo.	**Estimación:** 10
Fecha de inicio: 27/03/2015	**Fecha de fin:** 6/04/2015
Descripción: Dentro de la clase main.controler.js crear la función configurarNotificacion.	

Iteración 3

Tabla 41: Tareas de implementación (No: 12)

Tarea	
Número: 12	
Nombre de la tarea: HU 8: Gestionar sala de chat.	
Tipo de tarea: Desarrollo.	**Estimación:** 15
Fecha de inicio: 7/04/2015	**Fecha de fin:** 22/04/2015
Descripción: Crear la funcionalidad salaChat dentro de la clase main.controller.js, la cual se encarga de crear una sala de chat. Para eliminar se creará la función eliminarSalaChat.	

Tabla 42: Tareas de implementación (No: 13)

Tarea	
Número: 13	
Nombre de la tarea: HU 9: Consultar sala de chat.	
Tipo de tarea: Desarrollo.	**Estimación:** 10
Fecha de inicio: 23/04/2015	**Fecha de fin:** 3/05/2015
Descripción: Dentro la clase main.controler.js crear un controlador SingupCtrl que contenga la función nuevaConverzacionSala, asegurando el envío y recibo de mensajes dentro de la sala de chat.	

Tabla 43: Tareas de implementación (No: 13)

Tarea	
Número: 14	
Nombre de la tarea: HU 10: Salir del sistema.	
Tipo de tarea: Desarrollo.	**Estimación:** 10
Fecha de inicio: 4/04/2015	**Fecha de fin:** 7/05/2015
Descripción: Dentro la clase main.controler.js crear la función logouth.	

Anexo 9: Diseños de casos de prueba

Iteración 1

Tabla 44: Diseños de casos de prueba (HU 1_PR 1)

Caso de prueba de aceptación
Código: HU 1_PR 1
Nombre: Insertar usuario.
Descripción: La acción se ejecuta cuando un usuario desea ingresar en el sistema.
Condiciones para la ejecución: El usuario no puede estar registrado en el sistema.
Entrada/Pasos de ejecución: 1. Seleccionar en el menú de opciones la opción insertar usuario 2.Introducir el nombre, correo y contraseña

Resultado esperado: El usuario ya es parte del sistema
Evaluación de la prueba: Satisfactoria

Tabla 45: Diseños de casos de prueba (HU 1_PR 2)

Caso de prueba de aceptación
Código: HU 1_PR 2
Nombre: Modificar datos del usuario.
Descripción: Se ejecuta la acción cuando el usuario desea modificar sus datos.
Descripciones para la ejecución: El usuario debe estar autenticado en el sistema.
Entrada/Pasos de ejecución: 1. Seleccionar en el menú de opciones la opción modificar datos de usuario 2. Llenar los campos actuales para verificar que el usuario está conectado 3. Llenar los campos nuevos a modificar
Resultado esperado: Se modifican los cambios efectuados por el usuario.
Evaluación de la prueba: Satisfactoria

Tabla 46: Diseños de casos de prueba (HU 2_PR 3)

Caso de prueba de aceptación
Código: HU 2_PR 3
Nombre: Autenticar usuario en el sistema.
Descripción: Se ejecuta cuando el usuario desee acceder al sistema.
Descripciones para la ejecución: El usuario debe estar previamente registrado en el sistema.
Entrada/Pasos de ejecución: 1. Seleccionar la opción de autenticación 2. Se deben introducir datos como el correo y la contraseña con las que se registró el usuario
Resultado esperado: El usuario quedará autenticado y se visualizará en la lista de contactos.
Evaluación de la prueba: Satisfactoria

Iteración 2

Tabla 47: Diseños de casos de prueba (HU 3_PR 4)

Caso de prueba de aceptación
Código: HU 3_PR 4
Nombre: Iniciar conversación privada con un amigo.
Descripción: Se ejecuta una vez que el usuario de clic en la lista de contactos al amigo con el cual quiere hablar.
Descripciones para la ejecución: El usuario debe estar previamente autenticado en el sistema.

Entrada/Pasos de ejecución: 1. Clic en el usuario que con el cual desee establecer una conversación.
Resultado esperado: Se muestra una ventana con el nombre del usuario con el cual se quiera mantener una conversación.
Evaluación de la prueba: Satisfactoria

Tabla 48: Diseños de casos de prueba (HU 3_PR 5)

Caso de prueba de aceptación
Código: HU 3_PR 5
Nombre: Enviar mensaje a un amigo.
Descripción: Se ejecuta cuando el usuario desee enviar un mensaje a un amigo.
Descripciones para la ejecución: El usuario debe estar previamente autenticado en el sistema.
Entrada/Pasos de ejecución: 1. Se escribe un mensaje en la ventana de mensajes.
Resultado esperado: Se muestra el mensaje enviado.
Evaluación de la prueba: Satisfactoria

Tabla 49: Diseños de casos de prueba (HU 3_PR 6)

Caso de prueba de aceptación
Código: HU 3_PR 6
Nombre: Recibir mensaje de un amigo.
Descripción: El usuario receptor recibe el mensaje del usuario emisor.
Descripciones para la ejecución: El usuario debe estar autenticado en el sistema.
Entrada/Pasos de ejecución: 1. El usuario emisor envía un mensaje 2. El mensaje es mostrado en la ventana de mensajes del receptor.
Resultado esperado: Se muestra el mensaje enviado en la ventana del receptor.
Evaluación de la prueba: Satisfactoria

Tabla 50: Diseños de casos de prueba (HU 3_PR 7)

Caso de prueba de aceptación
Código: HU 3_PR 7
Nombre: Cerrar conversación.
Descripción: Se cierra la ventana de mensajes si el usuario lo desea.
Descripciones para la ejecución: El usuario debe estar previamente autenticado
Entrada/Pasos de ejecución: 1. Seleccionar el botón de cerrar que se encuentra en la parte superior derecha de la

ventana de mensajes.
Resultado esperado: Se cierra la ventana de mensajes.
Evaluación de la prueba: Satisfactoria

Tabla 51: Diseños de casos de prueba (HU 3_PR 8)

Caso de prueba de aceptación
Código: HU 3_PR 8
Nombre: Mostrar estado de un usuario.
Descripción: Se muestra el estado de disponibilidad del usuario.
Descripciones para la ejecución: El usuario debe estar previamente autenticado.
Entrada/Pasos de ejecución: 1. Se muestra un ícono con el estado de disponibilidad del usuario.
Resultado esperado: Se muestra el estado de disponibilidad del usuario.
Evaluación de la prueba: Satisfactoria

Tabla 52: Diseños de casos de prueba (HU 3_PR 9)

Caso de prueba de aceptación
Código: HU 3_PR 9
Nombre: Cambiar estado de un usuario.
Descripción: Se muestra el estado de disponibilidad del usuario.
Descripciones para la ejecución: El usuario debe estar previamente autenticado.
Entrada/Pasos de ejecución: 1. Se muestra un ícono con el estado de disponibilidad del usuario.
Resultado esperado: Se muestra el estado de disponibilidad del usuario.
Evaluación de la prueba: Satisfactoria

Tabla 53: Diseños de casos de prueba (HU 4_PR 10)

Caso de prueba de aceptación
Código: HU 4_PR 10
Nombre: Mostrar mensajes de historial
Descripción: Son mostradas las conversaciones anteriores en la ventana de mensajes.
Descripciones para la ejecución: El usuario debe estar autenticado.
Entrada/Pasos de ejecución: 1. Abrir una conversación 2.Mostar las conversaciones anteriores del usuario

Resultado esperado: Se muestran las conversaciones anteriores del usuario.
Evaluación de la prueba: Satisfactoria

Tabla 54: Diseños de casos de prueba (HU 5_PR 11)

Caso de prueba de aceptación
Código: HU 5_PR 11
Nombre: Mostrar amigos.
Descripción: Se muestra la lista de contactos una vez que el usuario es autenticado.
Descripciones para la ejecución: El usuario debe estar previamente autenticado en el sistema.
Entrada/Pasos de ejecución: 1. Se muestran los amigos una vez que el usuario se autentica en el sistema.
Resultado esperado: Los amigos son visualizados en la lista de contactos.
Evaluación de la prueba: Satisfactoria

Tabla 55: Diseños de casos de prueba (HU 5_PR 12)

Caso de prueba de aceptación
Código: HU 5_PR 12
Nombre: Eliminar amigo.
Descripción: Se eliminará el amigo que el usuario estime conveniente.
Descripciones para la ejecución: El usuario debe estar autenticado.
Entrada/Pasos de ejecución: 1. Se seleccionará la opción de eliminar amigo que se encuentra en el botón de opciones. 2. Se seleccionará el usuario al cual se desee eliminar.
Resultado esperado: Se elimina a un amigo.
Evaluación de la prueba: Satisfactoria

Tabla 56: Diseños de casos de prueba (HU 6_PR 13)

Caso de prueba de aceptación
Código: HU 6_PR 13
Nombre: Mostrar notificaciones.
Descripción: Se mostrarán las notificaciones una vez que ocurran los eventos de conexión y desconexión de un usuario.
Descripciones para la ejecución: El usuario debe estar previamente autenticado.

Entrada/Pasos de ejecución: 1. Se mostrarán las notificaciones una vez que el usuario se conecte o se desconecte.
Resultado esperado: Serán mostradas las notificaciones.
Evaluación de la prueba: Satisfactoria

Tabla 57: Diseños de casos de prueba (HU 7_PR 14)

Caso de prueba de aceptación
Código: HU 7_PR 14
Nombre: Ajuste de sonido.
Descripción: Se escucha el sonido de las notificaciones.
Descripciones para la ejecución: El usuario debe estar autenticado.
Entrada/Pasos de ejecución: 1. Se selecciona el botón de opciones y posteriormente configurar notificaciones 2.Se marca o desmarcar el sonido
Resultado esperado: Se escucha o no el sonido de la notificación según la elección del usuario.
Evaluación de la prueba: Satisfactoria

Tabla 58: Diseños de casos de prueba (HU 7_PR 15)

Caso de prueba de aceptación
Código: HU 7_PR 15
Nombre: Ajuste de posición.
Descripción: Se ajusta la posición en la pantalla de las notificaciones.
Descripciones para la ejecución: El usuario debe estar autenticado.
Entrada/Pasos de ejecución: 1. Se selecciona el menú de opciones y posteriormente configurar notificación 2.Seleccionar la posición para mostrar las notificaciones
Resultado esperado: Se muestran las notificaciones donde el usuario definió su posición.
Evaluación de la prueba: Satisfactoria

Iteración 3

Tabla 59: Diseños de casos de prueba (HU 8_PR 16)

Caso de prueba de aceptación
Código: HU 8_PR 16
Nombre: Crear sala de chat.
Descripción: Se creará la sala de chat que el usuario estime conveniente.
Descripciones para la ejecución: El usuario debe estar autenticado en el sistema.

Entrada/Pasos de ejecución: 1.Seleccionar el menú de opciones y la opción crear sala de chat 2.Introducir nombre de la sala
Resultado esperado: Se creará la sala de chat.
Evaluación de la prueba: Satisfactoria

Tabla 60: Diseños de casos de prueba (HU 8_PR 17)

Caso de prueba de aceptación
Código: HU 8_PR 17
Nombre: Mostar sala de chat.
Descripción: Se mostrarán las salas de chat existentes.
Descripciones para la ejecución: El usuario debe estar autenticado en el sistema.
Entrada/Pasos de ejecución: 1. Seleccionar el menú de opciones y marcar sala de chat 2.Muestra el listado de salas
Resultado esperado: Muestra el listado de salas de chat existentes.
Evaluación de la prueba: Satisfactoria

Tabla 61: Diseños de casos de prueba (HU 8_PR 18)

Caso de prueba de aceptación
Código: HU 8_PR 18
Nombre: Eliminar sala de chat.
Descripción: Se mostrarán las salas de chat existentes.
Descripciones para la ejecución: El usuario debe estar autenticado en el sistema.
Entrada/Pasos de ejecución: 1. Seleccionar el menú de opciones y marcar sala de chat 2.Muestra el listado de salas
Resultado esperado: Muestra el listado de salas de chat existentes.
Evaluación de la prueba: Satisfactoria

Tabla 62: Diseños de casos de prueba (HU 9_PR 19)

Caso de prueba de aceptación
Código: HU 9_PR 19
Nombre: Iniciar conversación grupal.
Descripción: Se mostrará la ventana de conversación.
Descripciones para la ejecución: El usuario debe estar previamente autenticado.
Entrada/Pasos de ejecución: 1. Seleccionar la sala de chat con la cual se desee establecer una conversación grupal.

Resultado esperado: Se mostrará la ventana de conversación grupal
Evaluación de la prueba: Satisfactoria

Tabla 63: Diseños de casos de prueba (HU 9_PR 20)

Caso de prueba de aceptación
Código: HU 9_PR 20
Nombre: Enviar mensaje a la sala de chat.
Descripción: Se enviará mensajes para la sala de chat.
Descripciones para la ejecución: El usuario debe estar autenticado.
Entrada/Pasos de ejecución: 1. Seleccionar la sala de chat 2.Escribir mensaje en la ventana
Resultado esperado: Se envía el mensaje a la sala de chat.
Evaluación de la prueba: Satisfactoria

Tabla 64: Diseños de casos de prueba (HU 9_PR 21)

Caso de prueba de aceptación
Código: HU 9_PR 21
Nombre: Recibir mensaje en la sala de chat.
Descripción: Se recibe el mensaje enviado en la sala de chat.
Descripciones para la ejecución: El usuario debe estar autenticado.
Entrada/Pasos de ejecución: 1. Se recibe el mensaje enviado mostrándose en la ventana de conversación.
Resultado esperado: Se muestra el mensaje.
Evaluación de la prueba: Satisfactoria

Tabla 65: Diseños de casos de prueba (HU 9_PR 22)

Caso de prueba de aceptación
Código: HU 9_PR 21
Nombre: Mostar estado del usuario en la sala de chat.
Descripción: Se muestra el estado de disponibilidad de los usuarios.
Descripciones para la ejecución: El usuario debe estar previamente autenticado.
Entrada/Pasos de ejecución: 1. Se muestra el estado del usuario
Resultado esperado: Es mostrado el estado del usuario.

Evaluación de la prueba: Satisfactoria

Tabla 66: Diseños de casos de prueba (HU 10_PR 23)

Caso de prueba de aceptación
Código: HU 10_PR 23
Nombre: Salir del sistema
Descripción: Se cierra por completo la aplicación.
Descripciones para la ejecución: El usuario debe estar previamente autenticado.
Entrada/Pasos de ejecución: 1. Se selecciona en el menú de opciones la opción salir
Resultado esperado: Se cierra el sistema.
Evaluación de la prueba: Satisfactoria