UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 4

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

Chat Inteligente de Desarrollo Colaborativo

Autores: Rosalia Martínez García
Leandro Campos Roja

Tutor: Ing. Arisbel Hechavarría Rojas
Co-tutor: Ing. Leannys Rodríguez Moreno

La Habana, 2014.
"Año 56 de la Revolución"

Declaración de autoría

Por este medio declaramos ser autores del presente trabajo de diploma y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio cada vez que se estime.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año 2014.

_____	_____
Firma de la tesista	Firma del tesista
Rosalia Martínez García	Leandro Campos Rojas

_____	_____
Firma de la tutora	Firma del tutor
Ing. Leannys Rodríguez Moreno	Ing. Arisbel Hechavarría Rojas

Agradecimientos de Leandro

A toda mi familia, en especial a mi hermana.

Quiero agradecer a mi madre y sobre todo a mi hermana Yani, al Cuñao, a Salvador, a toda mi familia, a mis amigos, en especial a todos aquellos que aún recuerdo y que realmente conozco, a mis queridos enemigos, a Linus Torvalds por crear Linux, a las comunidades de las UCI que tanto me han enseñado, en especial a humanOS, a toda la comunidad de software libre en el mundo, a Ubuntu y a San Google a quien le debo gran parte de mis conocimientos. Y por último pero no menos importante a mis tutores y a mi compañera de tesis, a todos gracias.

Rosalia agradecimientos y dedicatoria

A mi mamá Elizabeth, a mi abuela Leonor, a mi abuelo Jorge, a mi tía Eleonor y a mi papá Luis. Que siempre me apoyaron y guiaron por el mejor camino.

Le agradezco a mi novio y a su familia que me han acogido en su casa como una más, apoyándome en todo lo que me ha hecho falta.

A mi compañero de tesis que tantos dolores de cabeza me ha dado, pero que sin él no hubiese sido posible realizar este trabajo, gracias por ayudarme en todo.

A todas mis amistades de siempre y también a las que surgieron en el trayecto de estos 5 años, es difícil decir nombres pero cada una de estas, sabe cuando digo amistad, que ellas entran en este grupo.

Y pues claro a mis tutores Leannys y Arisbel, gracias por haber sido pacientes con nosotros y dedicarnos parte de su tiempo.

Dedicatoria

Dedico este trabajo a mi FAMILIA y en especial a mis Hermanos, que sigan los pasos que le indican sus sueños porque ¨ el que persevera triunfa¨.

Resumen

La Universidad de la Ciencias Informáticas posee varios centros de desarrollo de *software* que apoyan al gobierno cubano, con vista a informatizar el país. Estos centros concentran un gran número de desarrolladores de distintos niveles profesionales que interactúan a diario, intercambian puntos de vistas, soluciones y desarrollan productos de *software* en colaboración. Muchas de las tareas que se les orientan requieren de un proceso de investigación, documentación y auto-capacitación para darles solución. La Universidad posee una gran infraestructura tecnológica y una sub-red con un gran monto de información disponible, pero localizarla rápidamente es engorroso, por lo que surge la necesidad de facilitar el acceso a la información**.** Por otro lado existen varias comunidades que sirven de apoyo a los desarrolladores y que son una fuente de información y colaboración excelente. La interacción en estos sistemas se efectúa a través de foros, que permiten compartir opiniones o mensajes entre los usuarios pero no en tiempo real, mejorar el desempeño de estos permite que la comunicación sea instantánea y la colaboración fluida. Surge la necesidad de desarrollar un sistema que integre un motor de búsqueda y un sistema de mensajería instantánea: el Chat Inteligente de Desarrollo Colaborativo permitirá disminuir el tiempo de búsqueda de la información necesaria para dar solución a las tareas de desarrollo, creando un ambiente de colaboración, con distintas propuestas de solución y sugerencias. Una vez desplegado el sistema los desarrolladores podrán acceder más fácil a la información y contar con un canal de comunicación e intercambio en tiempo real.

Palabras clave: *chat*, colaborativo, motor de búsqueda, tecnología.

Índice

Chat Inteligente de Desarrollo Colaborativo

Introducción

Desde sus inicios el ser humano se percató que su supervivencia dependía de la capacidad de adaptación a las condiciones que el entorno le imponía a un ritmo vertiginoso. La necesidad de comunicarse en cualquier momento y lugar, sin importar la distancia, el tiempo y las condiciones, ha llevado al hombre a desarrollar un conjunto de herramientas en su beneficio que a lo largo de los años evolucionan y se han perfeccionado de una manera muy significativa con ayuda de las tecnologías.

La informática es una ciencia nueva, que a diferencia de otras como las matemáticas o la química, tiene apenas un siglo de vida. A pesar de su corta existencia, la evolución ha sido siempre muy rápida (1), uno de los aportes en esta esfera que ha revolucionado en la sociedad son las Tecnologías de la Información y las Comunicaciones (TIC). Se han realizado descubrimientos y avances en este entorno, siempre con el objetivo de mejorar su utilización. La interacción del hombre con las TIC ha traído como consecuencias el surgimiento de ideas de cómo hacer atractivo y dinámico este intercambio. Relacionar las facilidades que brinda el desarrollo tecnológico en el devenir constante de las actividades que realiza el individuo como parte de la sociedad, tanto en su vida personal, como en su desempeño laboral, se ha convertido en el propósito que sustenta este proceso de transformación.

A escala mundial una de las relevancias en la Internet, es la proliferación de los *chat*. Esta herramienta ha provocado un impacto positivo en la sociedad, dando lugar a una nueva forma de establecer vínculos interpersonales. La ciber-charla o conversación virtual como se le puede llamar es una de las prácticas sociales y discursivas en la que las personas conectadas se expresan de manera espontánea. El *chat*[1] surgió y se erigió como fenómeno social emergente y se transformó en una de las herramientas más populares. Es una práctica social que crece día a día y que ha logrado convertirse en una de las tecnologías más usadas.

Los *chat* son sistemas de mensajería instantánea[2] empleados por los usuarios como espacios privados o públicos, en el cual pueden platicar sin prejuicios y expresar su punto de vista respecto a un tema determinado. Se tiene completo control sobre la conversación y se puede

1 **Chat** (en español: charla), se refiere a una comunicación escrita, entre dos o más personas, realizada de manera instantánea a través de la red (91).

2 **La mensajería instantánea** es una forma de comunicación en tiempo real entre dos o más personas basada en texto. El texto es enviado a través de dispositivos conectados a una red como Internet.

elegir la información personal a mostrar. Estos sistemas en muchas ocasiones son empleados para preguntar dudas, intercambiar opiniones y solicitar información o colaboración sobre un tema determinado, creando un perfecto espacio colaborativo en tiempo real. En la actualidad los *chat* han adquirido gran importancia y por ello se ha hecho necesario que se introduzcan cambios para extender sus funcionalidades. Inteligente es definido por la Real Academia en Español como "la capacidad de resolver problemas, conocimiento, comprensión, acto de entender" (2); por lo que un *chat* inteligente es la extensión de las características de un *chat* usando técnicas de inteligencia artificial o técnicas de programación y algoritmos complejos que propicien propiedades que simulen la capacidad de comprender y satisfacer necesidades de sus usuarios tales como: saber si el otro participante de una conversación privada está prestando atención a la conversación, con el fin de identificar si existe interés en la misma, notificar cuando se conecta o desconecta otro usuario y compartir estados de ánimo.

La Universidad de las Ciencias Informáticas (UCI) no está exenta de la utilización del *chat* como un medio más, para una comunicación rápida e interactiva. Los desarrolladores que se encuentran en la UCI en muchas ocasiones no cuentan con todos los conocimientos necesarios para darle solución a las tareas que se le asignan, por lo que se ven en la necesidad de documentarse sobre el tema correspondiente. Pero este proceso de auto-capacitación o aprendizaje autodidacta adquiere gran complejidad, debido a la amplia documentación e información existente en numerosos sitios *web*, blogs, comunidades y enciclopedias que deben visitar en busca de la información requerida, este proceso se torna más complejo cuando no se tiene acceso a Internet. Los meta-buscadores[3] son una grandiosa fuente de acceso al conocimiento, pero normalmente ofrecen una larga lista de coincidencias con el texto que se está buscando y no siempre el resultado es la información buscada, sino un texto que contiene parte de la cadena de texto a buscar y por lo tanto es incluido como un posible resultado.

Lo previamente planteado hace **ineficiente el proceso de búsqueda**, obligando al desarrollador a descargar una vasta documentación con un elevado volumen de información que se ven forzados a leer y revisar en busca de los conocimientos necesarios, proceso que puede llegar a ser muy engorroso, traduciéndose en una **pérdida de tiempo** que podría aprovecharse en el proceso de desarrollo. Todo para darle solución a un problema común para varios usuarios, cuya solución ha sido ofrecida de forma magistral por un usuario en un foro, blog o en una de las comunidades de la Universidad y luego de un arduo esfuerzo y un largo

3 El **meta-buscador** es un sistema que localiza información en los buscadores más usados y carece de base de datos propia.

tiempo de búsqueda, finalmente es encontrada y en muchas ocasiones es mejor porque en ella contribuyeron varios usuarios con su opinión y sus experiencias, e incluso se exponen varias alternativas.

Esta búsqueda exhaustiva incurre negativamente en la cuenta de Internet de los desarrolladores, que en muchas ocasiones descargan información que ya está presente en la Universidad en diferentes sitios *web*, que contienen una gran infraestructura de documentos, texto y artículos validados que pueden ser útiles, **pero se encuentran dispersos**. Estos sitios de la Universidad permiten comentar un tema determinado, lo que propicia emitir criterios e intercambiar opiniones e ideas y crear un **debate colaborativo**, **pero no en tiempo real**. Muchos de estos sistemas necesitan de un moderador para revisar y autorizar los comentarios que serán publicados, lo que ralentiza las publicaciones de los mismos y el intercambio de información.

A partir de lo expuesto se plantea como **problema a resolver**:

¿Cómo facilitar el acceso a la información y la comunicación en tiempo real entre los desarrolladores de la UCI?

Para resolver el presente problema de investigación se planteó como **objeto de estudio:** la mensajería instantánea. Enmarcado en el **campo de acción** los sistemas de mensajería instantánea para el desarrollo colaborativo.

Para solucionar el problema de investigación existente se define como **objetivo general**: Desarrollar un Chat Inteligente de Desarrollo Colaborativo (CIDC) para facilitar el acceso a la información y la comunicación en tiempo real entre los desarrolladores de la UCI.

Se plantea como **idea a defender:** con el desarrollo de un Chat Inteligente de Desarrollo Colaborativo se facilitará el acceso a la información y la comunicación en tiempo real entre los desarrolladores de la UCI.

Para alcanzar el objetivo general se definieron los siguientes **objetivos específicos:**

1. Elaborar el marco teórico relacionado con el objeto de estudio.
2. Diseñar la propuesta de solución.
3. Desarrollar la implementación y prueba de la propuesta de solución.

En vista a lograr el cumplimiento de los objetivos específicos señalados se definen las siguientes **tareas de la investigación**:

1. Estudio sobre los sistemas de mensajería instantánea, comunidades de desarrollo colaborativo, blogs destinados al desarrollo de *software* y meta-buscadores.
2. Definición de la plataforma del producto final.
3. Selección de la metodología de desarrollo de *software* a emplear en la implementación de la propuesta de solución.
4. Identificación de las tecnologías, herramientas y técnicas más empleadas en el desarrollo de aplicaciones de mensajería instantánea y en los meta-buscadores.
5. Identificación de las funcionalidades del sistema.
6. Elaboración de los artefactos correspondientes al diseño de la propuesta de solución.
7. Implementación del Chat Inteligente de Desarrollo Colaborativo.
8. Realización de pruebas definidas por la metodología de desarrollo a utilizar.

Para llevar a cabo la investigación, se utilizan los métodos de investigación de nivel teórico y empírico.

Métodos teóricos:

Analítico-Sintético

En la investigación se establece un estudio bibliográfico y una base sólida de fundamentos teóricos que permitan relacionar el análisis documental y el estado del tema de investigación, así como las técnicas, metodologías y herramientas necesarias para desarrolla la propuesta de solución**.**

Modelación

La modelación es el proceso mediante el cual se crea una representación o modelo para investigar la realidad. Se evidencia la utilización de este método en la elaboración de gráficos interface para un mejor entendimiento del problema de investigación y solución.

Dentro de los **métodos empíricos** se emplea**:**

Observación

Este método es el instrumento que permite estudiar más de cerca el objeto de investigación, las acciones, causas y consecuencias. Se puede observar como funciona la relación entre los desarrolladores de la Universidad y el nivel de acceso a la información y comunicación en

tiempo real entre los mismos. Además, de realizarse un estudio de la situación de la mensajería instantánea y la estrecha relación que puede tener con los desarrolladores.

La presente investigación consta de introducción, conclusiones, recomendaciones, referencias bibliografía, además de tres capítulos que tratan los temas que se muestran a continuación.

Capítulo 1: Fundamentación teórica.

En este capítulo se aborda todo lo relacionado con la fundamentación teórica que sustenta la investigación acerca del estudio del marco teórico, las tecnologías y herramientas a utilizar, además se exponen conceptos de gran importancia para la comprensión del trabajo.

Capítulo 2: Diseño de la propuesta de solución.

Este capítulo tiene como objetivo principal el modelo de diseño de la propuesta de solución y de este modo crear las bases para la implementación. Se detallan las historias de usuarios, donde se recogen todas las funcionalidades a implementar. Además, se muestran otros artefactos que se generan en las distintas fases de la metodología de desarrollo de *software* seleccionada.

Capítulo 3: Implementación y prueba.

Se describen los elementos necesarios para la implementación del CIDC. Además, de las pruebas al sistema para comprobar que el mismo cumple con los requerimientos planteados. Se muestran los resultados alcanzados en las pruebas.

Capítulo 1: Fundamentación teórica

Introducción

En el presente capítulo se realiza un estudio del marco teórico, referente a los sistemas de mensajería instantánea (en adelante se empleará las siglas MI) y de los conceptos fundamentales de la investigación. También, se dan a conocer las herramientas seleccionadas, características fundamentales de las tecnologías a utilizar para dar solución al problema planteado y la metodología escogida para el desarrollo de la aplicación.

1.1 Mensajería instantánea

La MI no es una idea novedosa, en su concepción y expansión intervinieron varias instituciones que hicieron de esta una realidad y un medio de comunicación no presencial, útil en cualquier contexto. Su historia nos hace retroceder muchos años atrás:

"*Una primera forma de mensajería instantánea fue la implementada en el sistema PLATO usado al principio de la década de 1970. Más tarde, el sistema talk implementado en UNIX/LINUX, comenzó a ser ampliamente usado por ingenieros y académicos en las décadas de 1980 y 1990 para comunicarse a través de Internet. ICQ ("I seek you" en español "te busco") cliente de mensajería instantánea creado por la empresa* israelí *Mirabilis, a finales de 1990,* que fue el primero desarrollado para ordenadores *con sistema operativo distinto de UNIX/LINUX...*" (3).

Posteriormente con el auge de las computadoras (PC) la MI se convirtió en una herramienta de comunicación, hasta llegar a Internet, donde se difundió su uso y se convirtió en un canal de comunicación muy empleado y valorado por los usuarios. Ha sido potenciado con nuevas características para hacer dicho enlace más ameno, implementándose novedosas y variadas tecnologías que hacen uso y explotan al máximo las virtudes de la MI, como la telefonía IP (VoIP) y videoconferencia, que permiten integrar la capacidad de transmitir audio y video (3). En la actualidad existen una serie de protocolos que han evolucionado para aumentar la confiabilidad de la comunicación a distancia y que se asemeje cada vez más a la comunicación oral presencial.

1.1.1 Protocolos de mensajería instantánea

Los protocolos son instrucciones, normativas o reglas que permiten guiar una acción o que establecen ciertas bases para el desarrollo de un procedimiento. Comunicación, por su parte, es una noción con múltiples usos, que se emplea para nombrar a la difusión y recepción de mensajes. Se trata del conjunto de pautas que posibilitan que distintos elementos que forman

parte de un sistema establezcan comunicaciones entre sí, intercambiando información. Los protocolos de comunicación instituyen los parámetros que determinan, cuál es la semántica y sintaxis que deben emplearse en el proceso comunicativo en cuestión. Las reglas fijadas por el protocolo también permiten recuperar los eventuales datos que se pierden en el intercambio de información (4). Existen varios protocolos que permiten la comunicación bidireccional entre un cliente y un servidor, entre los que se destacan:

1.1.1.1 *Internet Relay Chat*

El protocolo de IRC[4], es basado en textos, con el cliente más simple, puede ser cualquier programa que emplee *socket* [5]y sea capaz de conectar con el servidor. IRC es un sistema de teleconferencia, que (a través del uso del modelo cliente-servidor) está adaptado para poder ejecutarse en muchas máquinas de una manera distribuida. Una configuración típica consiste en un solo proceso (el servidor) que forma un punto central o intermediario para que los clientes (u otros servidores) puedan conectarse entre sí, realizar la entrega de mensajes y otras funciones. Un cliente es cualquier dispositivo que se conecta a un servidor, siempre que no sea otro servidor. A cada cliente se distingue por un único identificador (*nickname*) que tiene una longitud máxima de nueve caracteres. Además del identificador, los servidores deben tener la siguiente información acerca de todos los clientes: el verdadero nombre del anfitrión, el usuario del cliente en ese anfitrión y el servidor al que está conectado el cliente.

1.1.1.2 Protocolo Extensible de Mensajería y Comunicación de Presencia

El protocolo XMPP[6], antes conocido por jabber, es uno de los protocolos de MI más usados, sobre todo por la comunidad de *software* libre, por ser gratuito, una tecnología abierta, pública y de fácil comprensión. Es un conjunto de tecnologías abiertas para la MI, pluripartidista de *chat*, llamadas de voz y video, colaboración, *middleware* ligero, sindicación de contenidos y el enrutamiento generalizado de datos XML. Fue desarrollado originalmente en la comunidad de código abierto jabber, para proporcionar una alternativa abierta y descentralizada de los servicios de MI cerrados en ese momento (5).

4 **IRC**: *Internet Relay Chat* es un sistema multi-usuario, multi-canal de chateo.

5 **Socket**: es el punto final de una comunicación bidireccional entre dos programas que intercambian información a través de Internet.

6 **XMPP**: Protocolo Extensible de Mensajería y Comunicación de Presencia, por sus siglas en inglés *Extensible Messaging and Presence Protocol*.

El Chat Multi-Usuarios (MUC por sus siglas en inglés *Multi-User Chat*) es una extensión de XMPP, para el intercambio de información multipartidista similar a *Internet Relay Chat* (IRC), por el cual varios usuarios pueden intercambiar mensajes XMPP en el contexto de una sala o canal de comunicación. Además, de las características estándar de las salas de *chat*, como los temas de las habitaciones e invitaciones, el protocolo define un sólido modelo de control de la sala, incluyendo la habilidad de prohibir a los usuarios, nombrar moderadores de las habitaciones y administradores y exigir contraseña para unirse a la sala. Debido a que las habitaciones MUC están basadas en XMPP, pueden ser utilizadas no solo para intercambiar mensajes de texto plano sino para una amplia variedad de configuraciones XML (6).

1.1.1.3 Otros protocolos

MSN: *Microsoft Notification Protocol* en español Protocolo de Notificación de Microsoft, es utilizado por el cliente de MI *Messenger* de Microsoft. Es un protocolo privativo, sus fuentes no están disponibles, se puede hacer uso de este a través de los sistemas de Microsoft tales como el cliente de mensajería de esta compañía Skype (7).

OSCAR: el protocolo empleado por AIM (en muchos sistemas se emplea este como identificador del protocolo en lugar de OSCAR) el sistema de MI del gigante de Internet AOL (América Online). Es de desarrollo propietario y no ofrece ninguna documentación, ni da acceso a sus fuentes (8).

ICQ: el primer protocolo de mensajería actualmente en declive.

Estos protocolos son privados, por lo que se desechan como una posible alternativa para el desarrollo de la propuesta de solución, la cual será implementada usando tecnologías libres siguiendo la línea de trabajo de la UCI sobre la migración al *software* libre.

1.1.2 Tecnologías *web* de comunicación bidireccional.

La comunicación bidireccional consiste en: un emisor envía un mensaje por medio de un canal al receptor, quien lo recibe y envía la retroalimentación (9). Entre las tecnologías *web* que permiten la comunicación bidireccional entre un servidor y un cliente se encuentran.

1.1.2.1 Tecnología Ajax

Los *chat* que emplean la tecnología Ajax[7] son utilizados en sistemas de mensajería sencillos, estos sistemas hacen uso de consultas SQL a la base de datos periódicamente, donde solicitan toda la información a mostrar al usuario. Puede brindar todos los servicios normales de un sistema de mensajería que haga uso de un protocolo, pero su rendimiento es inferior, requiere de un servidor que soporte la constante solicitud del cliente. La cantidad de usuarios que pueden estar conectados simultáneamente depende del rendimiento del servidor, se puede incurrir en una sobrecarga al sobrepasar el límite de solicitudes que este logra atender al mismo tiempo y provocar un error en el sistema. Ajax es una tecnología para hacer consultas a la base de datos sin necesidad de recargar toda la página *web*, es usada con la filosofía de solicitud-respuesta, pero cuando la respuesta no existe al realizar la solicitud, hay que emplear una estrategia de demora, para que espere hasta que exista. Lo que se traduce en solicitudes periódicas hasta que el servidor disponga de la información requerida por el usuario. En un sistema de MI este método incurre negativamente en el rendimiento del sistema y resulta muy engorroso el control de eventos por lo que se desecha la tecnología Ajax como medio de intercambio de información en el *chat*. Se decide optar por una tecnología que brinde las mismas opciones y que esté orientado a la plataforma *web* pero que no afecte la capacidad funcional del servidor. Un medio de comunicación bidireccional que permita el intercambio de información en tiempo real, la respuesta está en un sistema conocido por WebSocket.

1.1.2.2 WebSocket

WebSocket es un protocolo que pretende superar la limitación estructural del protocolo HTTP [8] que lo vuelve ineficaz para las aplicaciones *web* alojadas en los exploradores, para que se mantengan conectados con el servidor a través de una conexión persistente. El protocolo WebSocket permite la comunicación bidireccional entre las aplicaciones *web* y los servidores *web* a través de un solo socket TCP[9]. La interacción de WebSocket comienza con un protocolo de enlace en el que las dos partes (explorador y servidor) confirman mutuamente su intención de comunicarse a través de una conexión persistente. A continuación se muestra como se envía un cúmulo de paquetes de mensajes a través de TCP en ambas direcciones (10).

7 **Ajax**: *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML),

8 **HTTP**: Protocolo de Transferencia de Hipertexto.

9 **TCP**: Protocolo de Transporte de Comunicación.

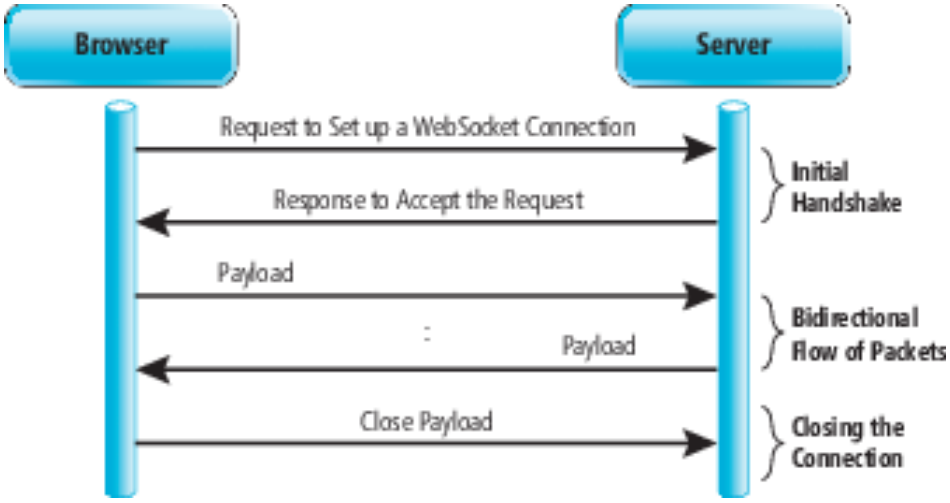

Figura 1: Esquema del protocolo WebSocket

WebSocket presenta algunas ineficiencias en cuanto a compatibilidad con los navegadores, es una tecnología nueva y como todos los usuarios no poseen navegadores modernos, no es posible asegurar que tengan soporte para el protocolo. Se decide optar por una alternativa más potente conocida como Socket.IO.

1.1.2.3 Socket.IO

Socket.IO utiliza principalmente el protocolo WebSocket, pero si es necesario emplea otros métodos, como son *sockets* de Adobe Flash, sondeo JSONP[10] y sondeo largo Ajax, pero mantiene la misma interfaz. A pesar de que se puede utilizar simplemente como un contenedor para WebSocket, brinda muchas más características, incluyendo la radiodifusión a múltiples *sockets*, el almacenamiento de datos asociados con cada cliente y comunicación asincrónica entrada y salida (11).

Socket.IO es una librería que nos permite manejar eventos en tiempo real mediante una conexión TCP. Es potente y se puede hacer todo tipo de aplicaciones en tiempo real. (12) Esta librería se puede incluir a node.js (en lo adelante indistintamente se utiliza nodejs, node.js o Node para referirse a esta tecnología).

1.1.2.4 Node.js

Node fue creado por Ryan Dahl y presentado por primera vez en el JSConf2009. Es un intérprete JavaScript del lado del servidor que cambia la noción de cómo debería trabajar un

10 **JSONP**: *JavaScript Object Notation with Padding* es un subconjunto de la notación literal de objetos de JavaScript con relleno.

servidor. Su meta es permitir a un programador construir aplicaciones altamente escalables y escribir código que maneje decenas de miles de conexiones simultáneas en una sola máquina física.

¿Qué problema resuelve Node?

Proporcionar una manera fácil para construir programas de red escalables. En lenguajes como Java y PHP, cada conexión genera un nuevo hilo que potencialmente viene acompañado de 2 MB de memoria. En un sistema que tiene 8 GB de RAM, esto da un número máximo de conexiones concurrentes cercanas a los 4.000 usuarios. A medida que crece su base de clientes, si desea que su aplicación soporte más usuarios, necesitará agregar más servidores, esto suma en cuanto a los costos de servidor del negocio, a los de tráfico y a los laborales. Además de estos costos, están los problemas técnicos potenciales, un usuario puede estar usando diferentes servidores para cada solicitud, así que cualquier recurso compartido debe almacenarse en todos los servidores. Por todas estas razones, la parte más sensible en toda la arquitectura de aplicación *web* (incluyendo el rendimiento del tráfico, la velocidad de procesador y la velocidad de memoria) es el número máximo de conexiones concurrentes que podrá manejar un servidor.

Node resuelve este problema cambiando la forma en que se realiza una conexión con el servidor. En lugar de generar un nuevo hilo para cada conexión (y de asignarle la memoria acompañante), cada una dispara una ejecución de evento dentro del proceso del motor de Node. Sus desarrolladores afirman que nunca se quedará en punto muerto, porque no se permiten bloqueos directamente para llamadas entrada/salida. Un servidor que lo ejecute puede soportar decenas de miles de conexiones concurrentes. Node.js en realidad usa el motor V8 JavaScript (intérprete ultra-rápido escrito en C++) desarrollado por Google y le da otro propósito para usarlo en el servidor. Utiliza lo que se conoce como modelo de programación orientado por eventos (13). Las aplicaciones que emplean estos protocolos para intercambiar mensajes, son los *chat*.

1.1.3 ¿Qué es un chat?

Ciber-charla o más conocida con el nombre de *chat***,** existe una variedad de definiciones que incluyen enfoques de lo que es un *chat*.

En el diccionario panhispánico el término *chat* se define como *"conversación entre personas conectadas a Internet, mediante el intercambio de mensajes electrónicos"* (14).

Capítulo 1: Fundamentación teórica

"Un chat es una conversación realizada por medios informáticos. La palabra chat *es un anglicismo, usado para describir este tipo de conversación. Chatear es entonces el hecho de participar en este tipo de conversación"* (15).

"Chat es una palabra inglesa que significa charlar, platicar o conversar. Dado que la informática es una disciplina universal y que el inglés es el idioma imperante en el mundo, con el fin de relacionar personas de diferentes regiones entre sí, el "arte" de conversar por medio de computadoras terminó llamándose chatear" (16).

"Chatear supone escribir en un teclado mediante el lenguaje escrito y recibir una respuesta por parte de otra persona que, a la distancia, estará utilizando el mismo método con programas idénticos o similares" (17).

En el presente trabajo, una vez estudiadas estas definiciones, se llega a la conclusión de que un *chat* es la comunicación escrita entre dos o más personas de manera instantánea a través de la red.

A continuación se refleja el estudio realizado de los sistemas similares, como el *chat* de: Facebook, Zera y EVA.

1.1.4 Estudio de Sistemas que emplean la mensajería instantánea unida a otros componentes.

Existen muchos sistemas que emplean la MI con fines diversos, pero no todos agrupan varias aplicaciones en un solo espacio virtual. Lograr que todos estos componentes interactúen entre sí puede llegar a ser un reto para los desarrolladores, forzándolos a crear tecnologías que mejoran el funcionamiento y la aceptación de estos sistemas.

1.1.4.1 Chat de Facebook

El *chat* de Facebook es un fuerte exponente del potencial de la MI, es una de las herramientas utilizadas por los usuarios que acceden para fomentar sus relaciones a esta red social. En junio de 2013, el número de usuarios de Facebook fue de 1,15 millones con 819 mil millones de usuarios activos en todo el mundo. Facebook en sí es un conglomerado de aplicaciones que se fusionan para formar un espacio virtual interactivo, dinámico y con una estética muy detallista. En vista a facilitar su uso esta compañía desarrolla y difunde una serie de aplicaciones y servicios que se integran tanto con clientes de MI de escritorio como con los navegadores *web* y otras aplicaciones (18).

El *chat* de Facebook emplea en su versión actual el protocolo XMPP, ofrece al usuario una serie de información útil, tales como usuarios en línea o conectados, manejo de lista de contactos, historial de conversaciones, notificaciones vía correo electrónico, notificaciones en tiempo real, así como gestos o acciones de los usuarios, ejemplo: si un usuario está escribiendo, si ha dejado de escribir y la fecha y hora del mensaje (19).

1.1.4.2 *Chat de* Zera

En la UCI existen varios centros de desarrollo de *software*, entre los que se encuentra el Centro de Tecnologías para la Formación (FORTES) y dentro del mismo un proyecto destacado es Alfaomega que se encarga de implementar una solución informática, la plataforma educativa Zera. La versión actual posee un *chat* implementado con la tecnología Ajax para ofrecer un servicio de mensajería aparentemente instantánea.

Este sistema posee servicios como saber los usuarios en línea o conectados, notificaciones en tiempo aparentemente real (depende de la capacidad de respuesta del sistema, rendimiento del servidor y tiempo de espera entre solicitudes), estas pueden ser gestos o acciones de los usuarios, ejemplo: si un usuario está escribiendo, si ha dejado de escribir, fecha y hora del mensaje y en dependencia de la implementación, manejo de lista de contactos, historial de conversaciones y notificaciones vía correo electrónico (20).

1.1.4.3 *Chat* del Entorno Virtual de Aprendizaje

Moodle es una plataforma de aprendizaje diseñada para proporcionar a educadores, administradores y estudiantes un sistema integrado único, robusto y seguro para crear ambientes de aprendizaje personalizados (21). El Entorno Virtual de Aprendizaje (EVA) de la UCI es una personalización de Moodle, tiene como objetivo la formación de los estudiantes, a través de cursos y foros. Esta herramienta permite la interacción entre el colectivo de profesores y los colegiales.

La versión empleada actualmente posee un *chat* como herramienta de comunicación entre los usuarios del sistema. Al igual que el *chat* de la plataforma Zera emplea la tecnología Ajax para garantizar la instantaneidad de los mensajes, así como un conjunto de notificaciones y características tales como, soporte de direcciones URL, emociones, integración de HTML e imágenes. Todas las sesiones quedan registradas para verlas posteriormente y logran ponerse a disposición de los estudiantes. Pueden programarse sesiones periódicas que aparecerán en el calendario (22).

1.1.5 Mensajería instantánea en el Chat Inteligente de Desarrollo Colaborativo

XMPP es una excelente opción para un sistema de MI, pero en el presente trabajo se desea realizar un sistema de MI propio e interno para el CIDC, que pueda integrarse fácilmente al resto de sus componentes y que no posea dependencia de servidores de terceros, para tener el control del flujo de información. Además, se desea definir los tipos de mensajes y notificaciones en tiempo real, no solo las propias del *chat*, sino de todo el sistema, para que funcione en su totalidad como un espacio en tiempo real. Se decide optar por Socket.IO unido a Node.js como medio de comunicación entre los clientes del *chat* y el servidor, asegurando así que todos los procesos sean en tiempo real y que el servidor pueda manejar múltiples conexiones concurrentes sin bloqueos o pérdida de información. Por otra parte, para lograr que el sistema sea un espacio de desarrollo colaborativo los espacios de *chat* deben ser en forma de salas, donde puedan conectarse múltiples usuarios y contribuir en conjunto al desarrollo de soluciones. ¿Cómo asegurar que el desarrollo sea colaborativo?

1.2 Sistema de desarrollo colaborativo

En las fuentes consultadas la mayoría de los autores definen como sistema colaborativo:

“Un conjunto de programas informáticos que integran el trabajo en un solo proyecto con muchos usuarios concurrentes que se encuentran en diversas estaciones de trabajo, conectadas a través de una red (Internet o Intranet)” (23).

“Groupware o sistemas colaborativos, son aplicación para grupos u organizaciones. Surge de la unión de computadoras, grandes bases de información y de tecnologías de comunicación. Los sistemas colaborativos están diseñados para facilitar el trabajo en grupo. Se puede usar para la comunicación, cooperación, coordinación, resolver problemas, competir o negociar” (24).

Los autores de la presente investigación asumen la definición emitida por el Dr. Abdiel E. Cáceres González debido a que es una de las más completas y acorde al trabajo en cuestión, en la cual se enuncia que los sistemas colaborativos son “*Los métodos y las herramientas de software que facilitan el trabajo en grupo, mejorando su rendimiento y ayudan a que personas que están localizadas en puntos geográficos diferentes puedan trabajar a la vez, ya sea directamente o de forma anónima, a través de las redes*” (25) .

1.2.1 Desarrollo colaborativo

La misma naturaleza de Internet que permite la creación de espacios virtuales para unir los interesados sin importar su situación geográfica, trajo una revolución en la colaboración al

dotarla de medios y prácticas más populares que facilitan la comunicación, para lograr metas comunes. Dentro de los elementos de colaboración que han evolucionado se encuentran entre otros: el correo electrónico, la MI, las salas de *chat*, los grupos de discusión y las wikis.

Dentro de la colaboración aparece el término trabajo colaborativo soportado por tecnologías (*Computer Supported Collaborative Work* en inglés) surgido en la década de los 80 (26), referido a las actividades coordinadas tales como la solución de tareas y la comunicación, llevadas a cabo entre individuos que colaboran entre sí apoyados por la tecnología. El término *groupware* es usado tanto en ambientes empresariales como en Internet, para designar al *software* que facilita compartir conocimiento e información entre los grupos geográficamente dispersos. El término equipo o grupo disperso geográficamente (GDT, por sus siglas en inglés *Geographically Disperse Team*) o equipo virtual es definido como un grupo de personas que trabajan a través del tiempo, espacio y límites de la organización con vínculos fortalecidos por redes de tecnología de la comunicación (27).

1.2.2 Ejemplos de comunidades colaborativas

Como ejemplo bandera se cita la comunidad de Linux (28). Dentro de las comunidades de expertos existentes en Internet, ocupan un lugar relevante las comunidades de desarrollo de *software* por los productos y servicios de altas prestaciones que han creado. Hoy en día es común el desarrollo de sistemas de código abierto por comunidades de acuerdo a los diversos intereses profesionales. Por solo mencionar ejemplos se pueden encontrar comunidades detrás de sistemas operativos (29), Debian (30), OpenSuse (31); navegadores de Internet como es el caso del Mozilla Firefox (32) y sistemas gestores de bases de datos de altas prestaciones como PosgreSQL (33).

1.2.3 Comunidades y blogs dedicados al desarrollo colaborativo en la Universidad

En la UCI existen varias comunidades de desarrollo de *software* como: “La comunidad de PHP en la UCI” cuyo objetivo como su nombre lo indica es promover y contribuir a desarrollar aplicaciones, empleando como lenguaje de programación PHP. El sitio posee un sistema de foros donde los usuarios comparten soluciones e intercambian dudas, opiniones, sus experiencias y colaboran entre sí, independientemente del nivel profesional. Se publican artículos, libros digitales, programas y recursos útiles para los desarrolladores, en su mayoría relacionados con PHP y tecnologías propiamente referentes a la plataforma *web*.

Otros ejemplos son “PostgreSQL, la comunidad técnica de desarrollo”, cuyo objetivo es promover el uso del sistema de base de datos de igual nombre, muy utilizado a nivel mundial

por su robustez. “Drupaleros” es otra comunidad orientada a la implementación de soluciones informáticas haciendo uso del CMS[11] Drupal, al igual que las anteriores, cuenta con foros, recursos compartidos y artículos referentes al propio sistema de gestión de contenidos. “RICE” es un blog dedicado a los desarrolladores de la Universidad, definido por su equipo de trabajo como:

“La comunidad pretende ser un punto de apoyo a los desarrolladores de la universidad, principalmente en tecnologías que se utilizan en la red de centros como Symfony, NodeJS, Django, Spring, jQuery y otras” (34).

“RICE” es una evolución natural del blog “echo”, cuya misión es compartir las experiencias que diariamente se adquieren en la producción, docencia e investigaciones. De forma que sus miembros no tengan que “reinventar la rueda” y encuentren fácilmente la solución a muchos de los problemas que ya otros miembros han solucionado (34).

1.2.3 Chat Inteligente de Desarrollo Colaborativo un espacio de desarrollo colaborativo

CIDC con el objetivo de ser una aplicación para la colaboración en tiempo real, debe ser estructurado de manera que permita el intercambio entre múltiples usuarios simultáneamente, un espacio virtual donde se pueda crear un ambiente comunitario. Donde la información sea almacenada para futuras referencias, pública y de fácil acceso para todos. Un sitio sin discriminación profesional, en el cual los usuarios independientemente de su nivel puedan compartir opiniones, soluciones y formular dudas que puedan ser útiles para otros. Debe incluir un sistema que le permita buscar información para los desarrolladores que se encuentren autenticados, pero ¿cómo localizar la información dispersa en la red de la UCI?

1.3 Sistema de Recuperación de Información

Los Sistemas de Recuperación de Información ó *Information Search and Retrieval (ISR)* en inglés, son una familia de sistemas de búsqueda entre los que se encuentran los buscadores, los meta-buscadores y los motores de búsqueda. Un sistema de recuperación de información se define como el “*proceso donde se accede a una información previamente almacenada, mediante herramientas informáticas que permiten establecer ecuaciones de búsquedas específicas. Dicha información ha debido ser estructurada previamente a su almacenamiento*” (35).

11 **CMS**: Sistema de Gestión de Contenidos en inglés *Content Management System*.

1.3.1 Los buscadores

Los buscadores, son sistemas de recuperación de información que permiten, dado un criterio de búsqueda introducido por el usuario, obtener un subconjunto de los documentos de mayor relevancia para dicho criterio de búsqueda, mediante la realización de operaciones sobre una base de datos. Estos sistemas consideran la *web* como una extensa base de datos lo que conlleva que se deban enfrentar a algunos obstáculos que imponen las propias características presentes en la *web*.

Algunos de estos obstáculos son (36):

- La información existente en la *web*, no sigue una estructura bien definida lo que implica que la información se encuentre desordenada.
- La información cambia continuamente.
- La información es redundante.

La mayoría de los buscadores emplean una arquitectura araña-indizador centralizada, es decir, la araña y el componente de indización se encuentran unidos (37). La araña es la encargada de realizar peticiones a servidores distantes en busca de la información contenida en los mismos. Estas han evolucionado a un punto tal que permiten realizar peticiones por distintos protocolos de la familia TCP/IP tales como: HTTP y FTP[12]. Por su parte, el componente encargado de la indización, recibe las páginas recuperadas por la araña, extrae una representación interna de la misma y la almacena en forma de índices en una base de datos (38).

1.3.2 Motor de búsqueda

El motor de búsqueda, por su parte, recibe la consulta de un usuario, que consiste en la introducción de un grupo de palabras claves sobre la información deseada. Estas palabras claves son convertidas por el sistema de formulación de consultas en un conjunto de incógnitas entendibles por el sistema, las que serán utilizadas por el subsistema de evaluación para devolver los documentos existentes en la base de datos, otorgando un orden de relevancia a dichos documentos en correspondencia con la consulta originalmente introducida por el usuario (39).

12 **FTP**: Protocolo de Transferencia de Archivos en inglés *File Transference Protocol*.

1.3.3 Los meta-buscadores

Los meta-buscadores poseen varias definiciones, en las cuales cada autor aporta su conceptualización de las características principales que deben cumplir estos sistemas informáticos. A continuación se muestran tres conceptos que describen en síntesis que es un meta-buscador.

"Un meta-buscador es un buscador de buscadores*. Una potente herramienta que realiza rastreos a diferentes bases de datos y proporciona una combinación de los mejores resultados. Comúnmente se les denomina robots, arañas o gusanos…"* (40).

Según el Diccionario de Informática el término meta-buscador es definido como "*un motor de búsqueda que envía una solicitud de búsqueda a otros múltiples buscadores o bases de datos, retornando un listado con los resultados de búsqueda o un listado de enlaces para acceder a los resultados individuales de cada buscador de forma fácil …*" (41).

Para el desarrollo de la presente investigación los autores consideraron la siguiente proposición. A diferencia de los buscadores, un meta-buscador no posee una base de datos propia, en su lugar utiliza la de los sitios encuestados para encontrar la información solicitada por el usuario. Su único trabajo consiste en combinar las mejores páginas que ha devuelto cada buscador, logrando así un mayor abanico de resultados con más calidad (42).

1.3.4 Buscador del Chat Inteligente de Desarrollo Colaborativo.

El buscador a incluir en CIDC debe permitir buscar al momento la información e indizarla en la base de datos, para futuras búsquedas. Debe funcionar como una fusión de un motor de búsqueda con un meta-buscador con el objetivo de optimizar los resultados y la cantidad de información que se almacena en la base de datos. A diferencia de un buscador convencional que indiza completamente el contenido de todos los sistemas publicados en la red en su base de datos, el componente de CIDC no debe ejecutar este procedimiento. Este realiza sus búsquedas usando los motores de búsquedas propios de los sitios a los cuales se dirige, comportamiento común de los meta-buscadores, pero a diferencia de estos que carecen de base de datos, el buscador de CIDC si tendrá una y almacenará en ella los resultados encontrados. Se desecha el uso de los buscadores y meta-buscadores existentes decidiéndose implementar un sistema de búsqueda propio que satisfaga las necesidades expuestas anteriormente. Para desarrollar este sistema es necesario identificar que características debe poseer el sistema donde será desplegado, en especial la plataforma óptima para su ejecución.

1.4 Plataforma de desarrollo

Las aplicaciones de escritorio tienen una serie de deficiencias con respecto a los sistemas web, tales como: disponibilidad, es difícil asegurar que estarán instaladas en todos los Sistemas Operativos (SO). De compatibilidad, no siempre son compatibles con todos los SO, lo que provoca que funcione incorrectamente parcialmente o en su totalidad. En ocasiones para su ejecución requieren otros *software* (dependencias) que deben estar previamente instalados, permisos de administración, emplear espacio en el disco duro para almacenar información y rendimiento para procesar grandes volúmenes de información.

La escalabilidad de la *web* supone la plataforma ideal para el desarrollo de un sistema como CIDC, porque los usuarios no deben instalar aplicaciones de terceros, para ejecutar el sistema, los requisitos del mismo solo los debe cumplir el servidor. Los usuarios solo deberán tener un navegador *web* en su SO que sea compatible con las nuevas tecnologías *web* y el sistema debe visualizarse y operar correctamente. Para guiar el proceso de desarrollo de la propuesta de solución es necesario identificar la metodología que se adecue a las características del equipo de trabajo.

1.5 Metodologías de desarrollo

Las metodologías de desarrollo son una estrategia, una guía orientada al proceso de desarrollo de *software*; están centradas en las personas o los equipos y orientadas hacia la funcionalidad y la entrega. Sus objetivos primordiales son elevar la calidad del *software* a través de un mayor control sobre el proceso y solucionar los problemas existentes en la elaboración de un producto de *software*, que cada vez son más complejos (43).

Estas abarcan procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de *software*. Las características de cada proyecto exigen que el proceso sea adaptable en cuanto a equipo y recursos. La comparación y/o clasificación de metodologías no es una tarea fácil debido a las diferentes propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas. En la actualidad, las metodologías de desarrollo de *software* se dividen en dos grandes grupos: ágiles y tradicionales (44).

Las metodologías tradicionales: son aquellas que están guiadas por una fuerte planificación durante todo el proceso de desarrollo, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema. Entre las más usadas se encuentran Proceso Unificado de Desarrollo (RUP) y *Microsoft Solutions Framework* (MSF), son utilizadas en grandes proyectos

donde se requiere una gran cantidad de documentación durante todo su ciclo de vida. Al usar este tipo de metodología se incurre que, en muchas ocasiones no se tiene un contacto directo con el cliente o el mismo no forma parte del equipo de trabajo.

Las metodologías ágiles: están más dirigidas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen énfasis en aspectos humanos asociados al trabajo en equipo. Entre las metodologías ágiles más usadas se encuentran SCRUM y Programación Extrema (XP), por las características que presentan cada una de ellas, se pueden adaptar a la aplicación que se desea desarrollar (45).

Para mejor entendimiento de las metodologías ágiles y tradicionales ver una breve comparación entre ellas (Ver anexo #1).

Por las características ya mencionadas sobre las metodologías ágiles y las características del equipo de desarrollo a continuación solo se analizarán las metodologías ágiles.

SCRUM

SCRUM es una metodología ágil, la cual está especialmente indicada para proyectos con entornos complejos, donde se necesita obtener resultados pronto, la innovación, la competitividad y la productividad son fundamentales. Entre sus principales características se encuentra que, el desarrollo de *software* se realiza mediante iteraciones, con una duración de 30 días, se destaca además la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (46).

Crystal Methodologies

Es 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. Han sido desarrolladas por Alistair Cockburn. El desarrollo de *software* se considera un juego cooperativo de invención y comunicación, limitada 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). Esta metodología es apropiada para entornos ligeros donde se tiene realimentación de los usuarios, en cada iteración se definen cuáles son los objetivos de la siguiente, tiene una planificación más

transparente para los clientes al estar diseñada para el cambio experimenta reducción de costo. Sin embrago, delimita el alcance del proyecto con el cliente (47).

Adaptive Software Development

Adaptive Software Development (ASD), su impulsor es Jim Highsmith, posee como principales características: iterativo, orientado a los componentes de *software* más que a las tareas y tolerante a los cambios. El ciclo de vida que propone esta metodología tiene tres fases esenciales: 1) especulación; donde se inicia el proyecto y se planifican las características del *software*, 2) colaboración; aquí se desarrollan las características y 3) aprendizaje; fase en la cual se revisa su calidad, y se entrega al cliente si no tiene errores. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo (48).

Desarrollo de *Software* Adaptable (*Adaptive Software Development* en inglés, abreviado ASD) según (49), puede ser usada para aprender de los errores e iniciar nuevamente el ciclo de desarrollo, aplica la información disponible acerca de todos los cambios para poder mejorar el comportamiento del *software* y proclama la colaboración y la interacción de personas. Sin embargo los errores y cambios que no son detectados con anterioridad afectan en gran medida la calidad del producto y su costo total.

Extreme Programming

*Extreme Programming***:** es una metodología ágil creada para dirigir las necesidades específicas del desarrollo de *software* conducido por equipos pequeños. Los miembros programan en parejas, a diferencia de SCRUM que trabajan de forma individual, además de que XP está especialmente definido para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. (50) Una de las ventajas es que precisa de poca documentación y planificación, lo que propicia así una agilización de la elaboración del producto. XP es considerada ligera, predecible, flexible y de bajo riesgo para la realización de un producto de *software* (51).

Centrada en las buenas prácticas de programación, especifica las prácticas de ingeniería de *software* tales como el desarrollo guiado por pruebas, la refactorización, la programación en parejas, la integración continua y el diseño incremental. Se beneficia de todas las técnicas de desarrollo ágil. Es flexible a cambios porque a diferencia de SCRUM se pueden hacer cambios durante las iteraciones y como máximo estas duran catorce días.

Metodología a utilizar

Una vez analizadas las metodologías de desarrollo de *software* ágiles más usadas se decidió optar por XP como la más adecuada para encaminar el desarrollo del *software*, debido a que se adapta con mayor facilidad al producto de *software* que se desea realizar, se ajusta a un proyecto con un pequeño equipo de trabajo, así como, un corto período de desarrollo, por lo que estas características son un punto de contacto con el desarrollo de esta investigación.

1.6 Tecnologías y herramientas

A continuación se describen las herramientas y tecnologías propuestas para el desarrollo de la aplicación. Especificando los aspectos más importantes de las mismas y las versiones empleadas debido a que estas determinan en gran parte la calidad de los productos elaborados, por lo que es necesario hacer una buena selección de las versiones.

***Framework* de PHP: Symfony 2.3.5**

Un *framework* facilita la programación de aplicaciones, encapsula operaciones complejas con instrucciones sencillas, suministra estructura al código fuente, esto ayuda al desarrollador que lo está utilizando a crear un código más legible y fácil de mantener. Symfony ha tomado las mejores ideas de *Rails*, incorporó ideas nuevas y el resultado es un *framework*, estable, productivo y bien documentado. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación *web* compleja. Entre sus principales características se encuentran (52).

- ✓ Su desarrollo es muy activo, puesto que incluye constantemente mejoras combinando la flexibilidad con la rapidez de ejecución.
- ✓ Tiene una amplia documentación.
- ✓ Reutilización muy activa de usuarios: la comunidad de usuarios crece cada día y ofrece un gran soporte de forma gratuita.
- ✓ Flexibilidad tanto en el diseño global del *framework*, como en su sistema de configuración y en los plugins.

Doctrine 2

Como Mapeador de Objeto Relacional (ORM[13]) por sus siglas en inglés. Empleado por defecto en Symfony 2, bastante completo y versátil. Una de las principales características de Doctrine es el bajo nivel de configuración que requiere para comenzar un proyecto. Puede generar clases a partir de una base de datos creada, y el programador puede especificar relaciones y agregar funcionalidades comunes para las clases generadas. No existe necesidad de generar o mantener esquemas complejos en lenguaje extensible de marcas (XML) por sus siglas en inglés. Otra característica es la habilidad de escribir consultas a la base de datos a partir de la programación orientada a objetos, llamada DQL (*Doctrine Query Language*) (53).

Sistema Gestor de Base de Datos: MySQL 5.5.34

MySQL es un servidor de base de datos muy rápido, sencillo de usar y es multi-usuario. SQL (*Structured Query Language*) se encuentra entre las consultas de base de datos de gran auge entre los programadores. Sus características principales son la velocidad, la robustez y la facilidad de uso. Provee sistemas de almacenamientos transaccionales y no transaccionales. Tiene un sistema de privilegios y contraseñas que es flexible y seguro. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor. Brinda soporte a grandes bases de datos con gran cantidad de registros. Además, de que tiene una excelente integración con PHP y Symfony 2 (54).

Cliente del gestor de base de datos: PhpMyadmin 4:3.5.8.1

Phpmyadmin es un cliente del gestor de base de datos MySQL, es gratuito escrito en PHP, permite gestionar el servidor de bases de datos MySQL desde un navegador *web*, está disponible para *Windows* y *Linux*. Proporciona herramientas de gestión de base de datos: edición, creación, modificación y eliminación de bases de datos, tablas, vistas, campos, relaciones e índices, mantenimiento de usuarios, privilegios y brinda soporte a procedimientos almacenados. Entre otras de sus funcionalidades se encuentran, que permite optimizar y reparar tablas, que son dos tareas de mantenimiento importantes en el desarrollo *web*. Además, brinda la posibilidad de realizar búsquedas en las bases de dato, así como poder escribir consultas SQL y ejecutarlas (55).

13 **ORM**: en inglés *Object Relational Mapping*, es un sistema que se encarga de transformar los datos de las tablas de una base de datos relacional en objetos, para que sea más fácil trabajar con sus campos al tratarlos como atributos de un objeto.

Servidor *web*: Apache 2.0

El objetivo de la fundación de *software* Apache es crear un servidor HTTP que cumpla los estándares, de código abierto, eficiente y extensible. Sus características principales son: poder usar, Protocolo Seguro de Transferencia de Hipertexto (HTTPS) por sus siglas en inglés, *hosts* virtuales, Interfaz de Entrada Común (CGI) por sus siglas en inglés, integración sencilla de scripts y base de datos, filtros de peticiones y respuestas, varios esquemas flexibles de autenticación. Funciona sobre la mayoría de los sistemas operativos incluyendo *UNIX, MS-Windows, Macintosh* y *NetWare*. Está basado en *software* libre y es distribuido bajo la licencia Apache de esta misma empresa (56). Apache ha sido el más popular servidor *web* en Internet desde abril de 1996. Este dispone de varios módulos que no están incluidos en el núcleo del servidor, algunos de estos facilitan las funcionalidades básicas para un servidor *web* y otras lo convierten en mucho más que eso, permitiéndole optimizar el rendimiento y la rapidez del código (57).

En la actualidad el 60% de los servidores que se encuentran trabajando es con Apache y esto es muestra de cuan factible es su uso. Cuando se tiene una duda es fácil obtener respuesta precisa de este servidor *web* lo que incurre en un ahorro de tiempo en busca de la información necesaria (58).

Nodejs

NodeJS es una tecnología que permite una comunicación bidireccional entre un cliente y un servidor, emplea como lenguaje de programación JavaScript y se ejecuta del lado del servidor. Node.js es una plataforma realizada con el *runtime*[14] JavaScript de Chrome (V8 Javascript *Engine* de Google) para construir fácilmente aplicaciones rápidas de red escalables. Node.js utiliza un modelo de E / S sin bloqueos, lo que lo hace ligero y eficiente, ideal para aplicaciones en tiempo real de datos intensivos que se ejecutan a través de dispositivos distribuidos. (59)

Socket.IO es un Interfaz de Programación de Aplicaciones (API) por sus siglas en inglés, de Node.js que permite la creación de múltiples *sockets* de comunicación bidireccional que hacen posible un flujo de información en tiempo real. Socket.IO es a su vez un API de *WebSocket* el cual tiene como objetivo proporcionar un mecanismo para las aplicaciones basadas en un navegador, estas aplicaciones necesitan una comunicación bidireccional con los servidores y

14 **Runtime**: en términos de computación se refiere a un traductor o conjunto de programas que interpretan las acciones del usuario y la transforman en eventos en un lenguaje de programación determinado.

que no dependan de la apertura de múltiples conexiones HTTP: por ejemplo, utilizando XMLHttpRequest o <iframe> y sondeo largo (60).

***Framework* para el lenguaje JavaScript: JQuery**

Es un producto con una aceptación por parte de los programadores muy buena y un grado de acierto en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones. Además, es un producto estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del *framework*, todas las ventajas que presenta el *framework* sin duda son de agradecer ya que se obtienen de manera gratuita, porque tiene licencia para uso en cualquier tipo de plataforma, personal o comercial. Implementa una serie de clases (de programación orientada a objetos) que permite programar sin preocupaciones del navegador con el que está trabajando el usuario, ya que funcionan de exacta forma en todas las plataformas más habituales. Para la implementación del sistema se utilizará JQuery en su versión 1.8.2 (61).

Entorno de Desarrollo Integrado (IDE)

En inglés (*Integrated Development Environment*), es un programa compuesto por un conjunto de herramientas que proporcionan un marco de trabajo incondicional para los lenguajes de programación como C++, Python, Java, C#, Delphi y PHP, todo para un mejor desempeño de los programadores (62). Entre los IDE utilizados para desarrollar aplicaciones *web* con el lenguaje de programación php y que se integren con el *framework* Symfony 2, se encuentran PhpStorm, PhpDesigner y NetBeans. Posteriormente se brindan algunas características de estos IDE.

PhpStorm 7

PhpStorm 7 es un IDE que soporta a Drupal, *Web Components* y TypeScript, se puede depurar fácilmente. Tiene soporte para la última versión de PHP 5.5 y posee una nueva forma de inspeccionar código, es una herramienta para crear ambientes de desarrollos reproducibles. Entre sus principales características se encuentran que: permite el completamiento código, ayuda a depurar y a probar unidades, permite que se ejecuten un conjunto de pruebas en cualquier momento. Entre sus agravantes se encuentra que guarda automáticamente y posee ajustes de interfaz de complejidad (63).

PhpDesigner 8

PhpDesigner 8 es un editor rápido y potente IDE de PHP con soporte completo para HTML5, CSS3 y JavaScript. IDE altamente personalizable con resaltado de sintaxis inteligente, soporte de depuración, análisis sintáctico, el apoyo a la codificación orientada a objetos, plantillas de código, fragmentos de código, de tareas pendientes y errores, trabajar con proyectos y *frameworks*, navegación de código intuitiva, formateadores de código y simplificaciones y todo envuelto en una agradable e intuitiva interfaz de usuario (64).

Vale destacar que ambos *software* son distribuidos bajo licencias privativas, por lo que no es recomendable su uso en proyecto de *software* libre.

NetBeans en su versión 7.4

NetBeans es un producto libre y sin restricciones de uso, está disponible para todos los programadores permitiéndoles escribir, compilar, depurar y ejecutar programas, brinda soporte a otros lenguajes de programación. A diferencia de PhpStorm este IDE es gratuito y está disponible en varios idiomas (65). Además, existe un número importante de módulos para extender las funcionalidades del NetBeans. Funciona en sistemas operativos compatibles con la máquina virtual *Java* (*Windows XP, Windows Vista, Windows 7, Ubuntu, Solaris, Mac OS*). Este IDE trabaja adecuadamente con Symfony 2 permite ejecutar sus comandos y tiene completamiento de código, Lenguaje de Marca de Hipertexto (HTML), JavaScript, Hojas de Estilo en Cascada (CSS) y lenguaje de programación PHP. Una vez anunciada la disponibilidad de la versión 7.4, una de las características esenciales es que extiende soporte a HTML5 (66).

PHP 5.3

El lenguaje PHP es gratuito independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Su seguridad se ve reforzada por el hecho de que el código original permanece oculto al usuario: el navegador lo ejecuta y lo muestra “traducido” a HTML. A diferencia de otros lenguajes (como ASP, de Microsoft) es gratuito y de código abierto. Gracias a una enorme comunidad de desarrolladores a nivel mundial cuenta con una amplia documentación (67).

HTML 5

Lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas *web*. Gracias a Internet y a los navegadores del tipo *Internet Explorer, Opera, Firefox* o *Netscape*, el HTML se ha convertido en uno de los formatos

más populares que existen para la construcción de documentos y también de los más fáciles de aprender (68).

Java 7

El lenguaje de programación más popular del 2013, lo cual se evidencia en el (Anexo # 5) es elegido para implementar el motor de búsqueda de CIDC. Este posee una vasta documentación y varias librerías útiles para el desarrollo de la propuesta de solución, de igual forma es fácil de integrar con el resto de los componentes. El equipo de desarrollo posee los conocimientos necesarios para la realización del motor de búsqueda utilizando Java. La selección de este lenguaje de programación fue respaldada por especialistas del proyecto de Alfaomega como una opción válida.

CCS 3

Las Hojas de Estilo en Cascada (*Cascading Style Sheets* en inglés, abreviado CSS) es una tecnología desarrollada con el objetivo de separar la estructura de una aplicación *web* de su presentación (69). En este sentido, HTML es la caja en que se muestran los contenidos y CSS es la manera en que se hace. Entre las principales ventajas de CCS se encuentran (70):

- Con una Hoja de Estilo es posible alterar la presentación de cada elemento sin tocar el código HTML. Haciendo más simple el mantenimiento del sitio *web* y minimizando las posibilidades de cometer errores o equivocaciones inesperadas.
- El lenguaje de las CSS posee y ofrece una gran variedad de herramientas de composición más potentes que HTML, permitiendo aplicar prácticamente todas las propiedades a cualquier elemento de la página.
- El lenguaje CSS, aunque muy potente, es relativamente sencillo y fácil de aprender.

***Framework* CSS: Bootstrap**

Es un *framework* diseñado para facilitar la creación de sitios *web*. Es un tipo de técnica de muestreo de datos que permite resolver problemas relacionados con la estimación de los intervalos de confianza o la prueba de significación estadística. Resulta más accesible y simple de comprender. Para la implementación del sistema se escogió Bootstrap en su versión 2.0 (71).

Conclusiones parciales

El estudio de sistemas similares permitió obtener características y funcionalidades útiles para el desarrollo de la propuesta solución, decidiéndose optar por Socket.IO unido a Node como medio de comunicación entre los clientes del *chat* y el servidor. Permitiendo así que todos los procesos sean en tiempo real. El análisis realizado de las metodologías, herramientas y tecnologías de desarrollo posibilitó seleccionar la metodología XP para encaminar el desarrollo del *software*, debido a que se adapta con mayor facilidad al producto de *software* que se desea realizar, ajustándose a un proyecto con un pequeño equipo de trabajo, así como, con un corto período de desarrollo. Se seleccionó como *framework* de desarrollo Symfony 2, como ORM Doctrine 2, Sistema Gestor de Base de Datos MySQL en su versión 5.5.34, como lenguajes de programación, HTML 5, Java, CCS 3 y JavaScript, cliente del gestor de base de datos PhpMyadmin versión 4:3.5.8.1, Sistema Gestor de Base de Datos MySQl, el IDE a utilizar NetBeans versión 7.4, servidor *web* Apache en su versión 2.0, *framework* de CSS Bootstrap 2.0 y como *framework* de JavaScript JQuery en su versión 1.8.2.

Capítulo 2: Modelo de diseño de la propuesta de solución.

Introducción

Para el diseño del CIDC se propone un marco de proceso de desarrollo de *software* basado en el estudio de la metodología seleccionada. Se describe la propuesta de solución del sistema. Se detallan las Historias de Usuario (HU), donde se ha estimado un tiempo para su realización y establecido el orden en que serán implementadas atendiendo a su prioridad. Como parte del diseño de la propuesta de solución se muestran las tarjetas Clase o Cargo, Responsabilidad, Colaboración (CRC) y los prototipos de interfaz de usuario más importantes del sistema.

2.1. Propuesta de solución

CIDC por consiguiente puede ser conceptualizado como un sistema que permite la comunicación escrita entre dos o más personas de manera instantánea a través de la red. Emplea los buscadores de una serie de sitios, almacenados en su base de datos, a los cuales les envía la información que busca el usuario, de forma tal que cada sitio aporta un gran número de resultados. Facilita el trabajo en grupo, porque las charlas funcionan como salas de *chat*, varios usuarios pueden intercambiar mensajes simultáneamente a través de la red desde puntos distantes. Estos mensajes pueden ser bibliografía, código o información textual, útil para el resto de los usuarios.

Una vez autenticado el usuario se le mostrarán todas las conversaciones activas, usuarios en línea, las noticias de la red y una zona donde puede realizar sus preguntas. Cada vez que realiza una pregunta se crea una charla, lo que le permite conversar a través de un *chat* en tiempo real. El cual le posibilita invitar cualquier usuario conectado a unirse a la conversación, resaltar el código en los mensajes, cambiar su estado a invisible para el resto de los usuarios que participan en la charla, buscar información almacenada en el sistema, mostrar el historial y guardarlo. Un usuario puede crear simultáneamente cuantas charlas (preguntas o dudas) estime conveniente, pero solo puede chatear en una a la vez, porque el sistema solo visualiza una sola área para realizar esta operación. El resto de las conversaciones en la cual esté participando se le muestran en un listado, que le permite tener un acceso rápido a cada una o cerrarla directamente. El buscador realiza una búsqueda en la red con el texto insertado al crear una charla y almacena en la base de datos del sistema los resultados, el contenido queda indexado para futuras búsquedas y le envía una solicitud al API de node.js para que le actualice la información al usuario, mostrándole los resultados como sugerencias. La pregunta queda almacenada, para ser sugerida a otros usuarios con la misma duda, junto a las sugerencias

asociadas. Cada pregunta realizada crea una conversación y esta a la vez se convierte en una sugerencia que se visualiza en el área de notificación del resto de los usuarios en línea.

2.2 Usuarios del sistema

Los usuarios del sistema son las personas o sistemas que interactúan con el mismo (72). En el CIDC los usuarios involucrados son los (estudiantes y profesores) y el administrador que aparte de ser usuario más del sistema es el de mayor privilegio.

Tabla1 Usuario del sistema

Usuarios del sistema	Descripción
Administrador	Persona con todos los permisos para la gestión del sistema. Lleva a cabo las funciones administrativas para darle soporte y mantenimiento a la aplicación y de esta manera garantiza la seguridad completa del *software*.
Usuario	Es la persona que accede al sistema, ya sea estudiante o profesor de la UCI.

Aspectos funcionales del sistema

Según la metodología seleccionada los aspectos funcionales son características que el sistema debe poseer, estos contienen los procedimientos que el usuario podrá ejecutar en dependencia de sus privilegios. El sistema debe ser capaz de cumplir con los siguientes aspectos funcionales (AF) para la aceptación de los usuarios:

AF1: Registrar usuario.

AF2: Autenticar usuario.

AF3: Gestionar usuario.

AF3.1: Modificar usuario.

AF3.2: Listar usuarios.

AF3.3: Eliminar usuario.

AF4: Gestionar sugerencia

AF4.1: Adicionar sugerencia.

AF4.2: Listar sugerencia.

AF4.3: Eliminar sugerencia.

AF5: Crear charla.

AF6: Gestionar charla.

AF6.1: Actualizar charla.

AF6.2: Listar charlas activas.

AF6.3: Listar charlas almacenadas.

AF6.4: Eliminar charla.

AF7: Gestionar mensaje.

AF7.1: Adicionar mensaje.

AF7.2: Listar mensaje.

AF7.3: Eliminar mensaje.

AF8: Listar mensajes del historial.

AF9: Contenido texto plano.

AF10: Buscar charla.

AF11: Buscar usuario.

AF12: Gestionar proveedor.

AF12.1: Adicionar proveedor.

AF12.2: Listar proveedor.

AF12.3: Modificar proveedor.

AF12.4: Eliminar proveedor.

Características del sistema

"*Las características especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, facilidad de mantenimiento, extensibilidad y fiabilidad*" (73). Estas ayudan a que el sistema que se esté implementando cumpla un conjunto de parámetros esenciales para que el producto posea una mayor aceptación. Seguidamente se hace alusión a estas características:

Apariencia o interfaz externa

- Se debe emplear un diseño sencillo e intuitivo para el usuario. Mantener homogeneidad en la estructura visual del sistema ayuda a mejor la apariencia. Este diseño debe ser sin muchas animaciones ni imágenes pesadas de forma tal que no afecten la rapidez de la aplicación.
- **Usabilidad**

  CIDC debe ser fácil de manejar por todas las personas asociadas a la universidad, su interfaz debe ser similar a la de los *chats* estudiados, para facilitar la interacción con el mismo. La aplicación debe ser diseñada de forma que los usuarios que hagan uso de la misma obtengan los conocimientos necesarios en el menor tiempo posible aunque sea la primera vez que interactúen con el sistema.

- **Disponibilidad**

  Se debe contar con la disponibilidad del *software* todo el tiempo para hacer de él un uso exhaustivo.

- **Portabilidad**

  El *software* debe ser multiplataforma, compatible con *Linux, Mac OS, Windows, Solaris*, también para dispositivos móviles *smartphones, tablets* y cualquier otra plataforma con soporte para navegadores *web*.

- **Seguridad**

  Debe garantizarse que el acceso a la información se efectúe de acuerdo al rol que desempeñan los usuarios. Impedir que la información pueda ser modificada por usuarios que no tengan permiso para hacerlo. Symfony 2 posee un sistema de seguridad que permite definir roles y los permisos correspondientes a cada uno, a través de una lista de control de acceso, en la cual se plasman las rutas del sistema y los roles que pueden acceder a la misma. Emplea algoritmos de codificación o encriptación para darle seguridad a las contraseñas y Doctrine 2 con el lenguaje para realizar consultas DQL que mitiga los ataques a la base de datos.

- ***Software***

  Navegadores *web* con soporte para HTML5 y CSS3, (*Google Chrome/Chromium, Firefox, Opera, Safari, Maxthon, Midori, Internet Explorer* 8 o superior). Se debe tener instalado preferentemente el sistema operativo *Linux (Ubuntu* y derivados), aunque también se podrá utilizar en el sistema operativo *Mac OS X* y *Windows XP SP2* o superior y otras plataformas que soporten los siguientes requisitos:

JVM (máquina virtual de Java instalada, versión 7 o superior)

Node.js

PHP5

MYSQL-Server 5.5 o MariaDB

- ***Hardware***

Para poder ejecutar CIDC es recomendable que el sistema posea un 1GB RAM o superior.

2.3 Exploración

En esta primera fase de la metodología XP es donde se definen las historias de usuario, que son la descripción de las funcionalidades con que debe cumplir el sistema. Cada plantilla describe las características que deben ser adicionadas al programa. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas y las tecnologías que utilizarán en el proyecto (74).

2.3.1 Historias de usuarios

Las historias de usuario (HU) son descripciones cortas de una necesidad que el *software* debe compensar, su propósito no es describir en detalle la funcionalidad. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. En muchas ocasiones se piensa que las historias de usuario en XP son similares a los casos de uso en otras metodologías, pero no tienen similitud. Las HU solo muestran el perfil de una tarea a realizar (75).

¿Para qué se utilizan?

El propósito de las HU es la estimación del esfuerzo necesario para implementar una nueva funcionalidad en un *software*. Para estas estimaciones, se le asignan unidades de medida a las historias que representen magnitud y no días reales de duración, tales como: puntos de estimación, días ideales u otra unidad de medida (76).

Seguidamente se detallan los campos que componen las HU en función:

- ✓ Número de HU: este campo contiene el número que le es asignado a las HU consecutivamente.
- ✓ Nombre de HU: identifica la HU a la que se hace alusión.
- ✓ Usuario: usuario del sistema que interactúa o realiza la funcionalidad.

- ✓ Prioridad en el negocio: prioridad que le es asignada a la HU en el negocio de acuerdo a las necesidades del usuario.
- ✓ Riesgo en el desarrollo: riesgo que se corre en el desarrollo de la aplicación en caso de no realizarse la HU.
- ✓ Puntos estimados: esta estimación se realizó en semanas trabajando 8 horas diarias de lunes a viernes, la misma es formalizada por el equipo de desarrollo evaluando el tiempo que incurre en la realización de esta HU.
- ✓ Iteración asignada: este campo es en dependencia de la prioridad que se establece, lo cual se determina en la iteración que se desarrolla la HU.
- ✓ Descripción: como su nombre lo indica, se realiza una pequeña descripción de la función de la HU.
- ✓ Observación: datos adicionales de la HU en cuestión, resalta datos que son necesarios cumplir, para una mejor utilización y entendimiento de la funcionalidad que se brinda.
- ✓ Seguidamente se muestran cuatro HU una por cada iteración a desarrollar, estas son las más críticas y representan un gran peso en el desarrollo. Para visualizar las demás HU (Ver anexo # 2).

Tabla 2 HU Autenticar usuario

Historia de usuario	
Nombre: Autenticar usuario.	**Número:** 2
Usuario: todos	**Iteración asignada:** 1
Prioridad en el negocio: alta	**Puntos estimados:** 1
Riesgo en el desarrollo: alto	
Descripción: el usuario ingresa su usuario y contraseña, el sistema valida la veracidad de los datos, seguidamente se le muestra la interfaz principal del sistema.	
Observaciones: responde al requisito funcional número 2. Si es primera vez que se accede al sistema, debe registrarse previamente y entrar con la URL que es enviada al correo electrónico.	

Tabla 3 HU Crear charla

Historia de usuario

Nombre: Crear charla.	**Número:** 9
Usuario: todos	**Iteración asignada:** 2
Prioridad en el negocio: alta	**Puntos estimados:** 1
Riesgo en el desarrollo: alto	
Descripción: en la vista de las entradas el usuario escribe su duda o pregunta en la caja de texto y selecciona crear.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad número 5.	

Tabla 4 HU Adicionar mensaje

Historia de usuario	
Nombre: Adicionar mensaje.	**Número:** 11
Usuario: todos	**Iteración asignada:** 3
Prioridad en el negocio: alta	**Puntos estimados:** 1
Riesgo en el desarrollo: bajo	
Descripción: el usuario redacta un argumento en el área de texto y selecciona la opción enviar.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad 7.1 que pertenece al gestionar mensaje con número 7.	

Tabla 5 HU Adicionar sugerencia

Historia de usuario	
Nombre: Adicionar sugerencia.	**Número:** 18
Usuario: todos	**Iteración asignada:** 4
Prioridad en el negocio: alta	**Puntos estimados:** 1
Riesgo en el desarrollo: alto	

Descripción: el usuario realiza una pregunta y el buscador le muestra un listado de las posibles sugerencias del tema especificado.
Observaciones: esta HU responde a la funcionalidad número 4.1 que está contenida en gestionar sugerencia con número 4. Se necesita estar autenticado para poder llevar a cabo esta HU.

2.4. Planificación

En esta segunda fase se realiza una estimación del esfuerzo que costará implementar todas las historias de usuario, teniendo en cuenta además que a cada una se le asignó en la fase de exploración un tiempo para desarrollarla. Luego se procede a organizarlas en las iteraciones correspondientes, en dependencia de la prioridad especificada por el cliente y del tiempo de desarrollo de cada una de las HU. Toda técnica de planificación de *software* debe tratar de crear visibilidad, de tal manera que los involucrados en el proyecto puedan realmente ver cuán largo es. Esto ayuda a concebir metas claras, que no sean dudosas y representen un proceso de avance (77).

2.4.1 Estimación de esfuerzo por historia de usuario

Seguidamente se muestra una tabla con el esfuerzo estimado por cada historia de usuario realizada, la misma refleja el número, nombre y estimación que le fue atribuida a cada una.

Tabla 6 Estimación de esfuerzos de acuerdo a las HU

No	Historia de usuario	Estimación (semanas)
1	Registrar usuario	1
2	Autenticar usuario	1
3	Modificar usuario	0,5
4	Listar usuarios	0,5
5	Eliminar usuario	0,5
6	Adicionar sugerencia	1
7	Listar sugerencias	0,5
8	Eliminar sugerencia	0,1

9	Crear charla	1
10	Actualizar charla	1
11	Listar charlas activas	0,6
12	Listar charlas almacenadas	0,3
13	Eliminar charla	0,1
14	Adicionar mensaje	1
15	Listar mensaje	0,5
16	Eliminar mensaje	0,5
17	Listar mensajes del historial	0,5
18	Contenido texto plano	0,2
19	Buscar charla	0,1
20	Buscar usuario	0,1
21	Adicionar proveedor	0,5
22	Modificar proveedor	0,2
23	Listar proveedor	0,2
24	Eliminar proveedor	0,2

2.4.2 Plan de iteraciones

Iteración se le puede llamar a las entregas pequeñas de un proyecto, donde a medida que se realizan las iteraciones avanza la realización del producto y se reflejan los nuevos cambios realizados en él. Al finalizar todas las mini-entregas se obtiene como resultado el *software* con la calidad requerida. Con una buena planificación de las iteraciones el cliente quedará totalmente satisfecho con el producto que fue acordado inicialmente (78).

A continuación se describen las cuatro iteraciones correspondientes al desarrollo:

Iteración 1: en la primera iteración se pretende dar cumplimiento al desarrollo de las HU número 1, 2, 3, 4 y 5. Estas HU posibilitan que el usuario pueda registrarse, autenticarse y gestionar su información a mostrar.

Iteración 2: el principal objetivo de la segunda iteración es desarrollar las HU número 9, 10, 11, 12 y 13, en la que se maneja toda la información referente al gestionar charla.

Iteración 3: en esta tercera iteración se implementaran las HU número 14, 15, 16, 17, 18, 19 y 20.

Iteración 4: en esta última iteración se persigue implementar las HU número 6, 7, 8, 21, 22, 23 y 24.

Luego de realizar las iteraciones antes mencionadas el sistema se encontrará en su versión 1.0.

Para la confección del plan de duración de las iteraciones se colocaron primero las HU que pertenecen a la primera iteración y las demás ubicadas en las siguientes iteraciones, de forma tal que la implementación de cada una de ellas se relacione con la implementación de las otras que están en la misma iteración. A continuación se muestra la tabla con el plan de duración de las iteraciones.

Tabla 7 Plan de duración de las iteraciones

Iteración	Orden de las HU	Duración total
Iteración 1	Registrar usuario Autenticar usuario Modificar usuario Eliminar usuario Listar usuario	3,5 semanas

Iteración 2	Crear charla Actualizar charla Listar charlas activas Eliminar charla Listar charlas almacenadas	2,6 semanas
Iteración 3	Adicionar mensaje Listar mensaje Eliminar mensaje Listar mensaje del historial Contenido en texto plano Buscar charla Buscar usuario	2,9 semanas

Iteración 4	Adicionar proveedor Eliminar proveedor Modificar proveedor Listar proveedores Adicionar sugerencia Eliminar sugerencia Listar sugerencia	2,7 semanas

2.4.3 Plan de entregas

El objetivo del plan de entregas es establecer las HU que serán agrupadas para conformar una entrega, teniendo en cuenta la prioridad que se estableció para cada una de ellas (79), a continuación la tabla que da cumplimiento a dicho plan de entregas.

Tabla 8 Plan de entregas

Historia de usuario	Iteración 1	Iteración 2	Iteración 3	Iteración 4
Registrar usuario	V 1.0	Finalizado	Finalizado	Finalizado
Autenticar usuario	V 1.0	Finalizado	Finalizado	Finalizado
Modificar usuario	V 1.0	Finalizado	Finalizado	Finalizado
Eliminar usuario	V 1.0	Finalizado	Finalizado	Finalizado
Listar usuarios	V 1.0	Finalizado	Finalizado	Finalizado
Crear charla	-	V 1.0	Finalizado	Finalizado

Actualizar charla	-	V 1.0	Finalizado	Finalizado
Listar charlas activas	-	V 1.0	Finalizado	Finalizado
Listar charlas almacenadas	-	V 1.0	Finalizado	Finalizado
Eliminar charla	-	V 1.0	Finalizado	Finalizado
Adicionar mensaje	-	-	V 1.0	Finalizado
Listar mensaje	-	-	V 1.0	Finalizado
Eliminar mensaje	-	-	V 1.0	Finalizado
Listar mensaje del historial	-	-	V 1.0	Finalizado
Contenido en texto plano	-	-	V 1.0	Finalizado
Buscar charla	-	-	V 1.0	Finalizado
Buscar usuario	-	-	V 1.0	Finalizado
Adicionar proveedor	-	-	-	V 1.0
Eliminar proveedor	-	-	-	V 1.0
Modificar proveedor	-	-	-	V 1.0
Listar proveedor	-	-	-	V 1.0
Adicionar sugerencia	-	-	-	V 1.0
Eliminar sugerencia	-	-	-	V 1.0
Listar sugerencia	-	-	-	V 1.0

2.5. Diseño

El diseño en XP sigue rigurosamente el principio MS (mantenlo sencillo). Un diseño sencillo siempre se prefiere sobre una representación más compleja. Esta etapa toma como su aportación inicial las necesidades identificadas en el proceso de definición de los aspectos funcionales. Los elementos que componen esta fase describen las características de *software*

que se desea (80). Para la presente aplicación se decidió que su diseño fuera lo más ligero y entendible posible.

2.5.1 Modelo de datos

EL presente modelo almacena toda la información que contienen las tablas (*user, message, chatuser, chat taxonomy, chattaxonomy, suggestion, suggestiontaxonomy* y *provider)* y la forma en que se relacionan.

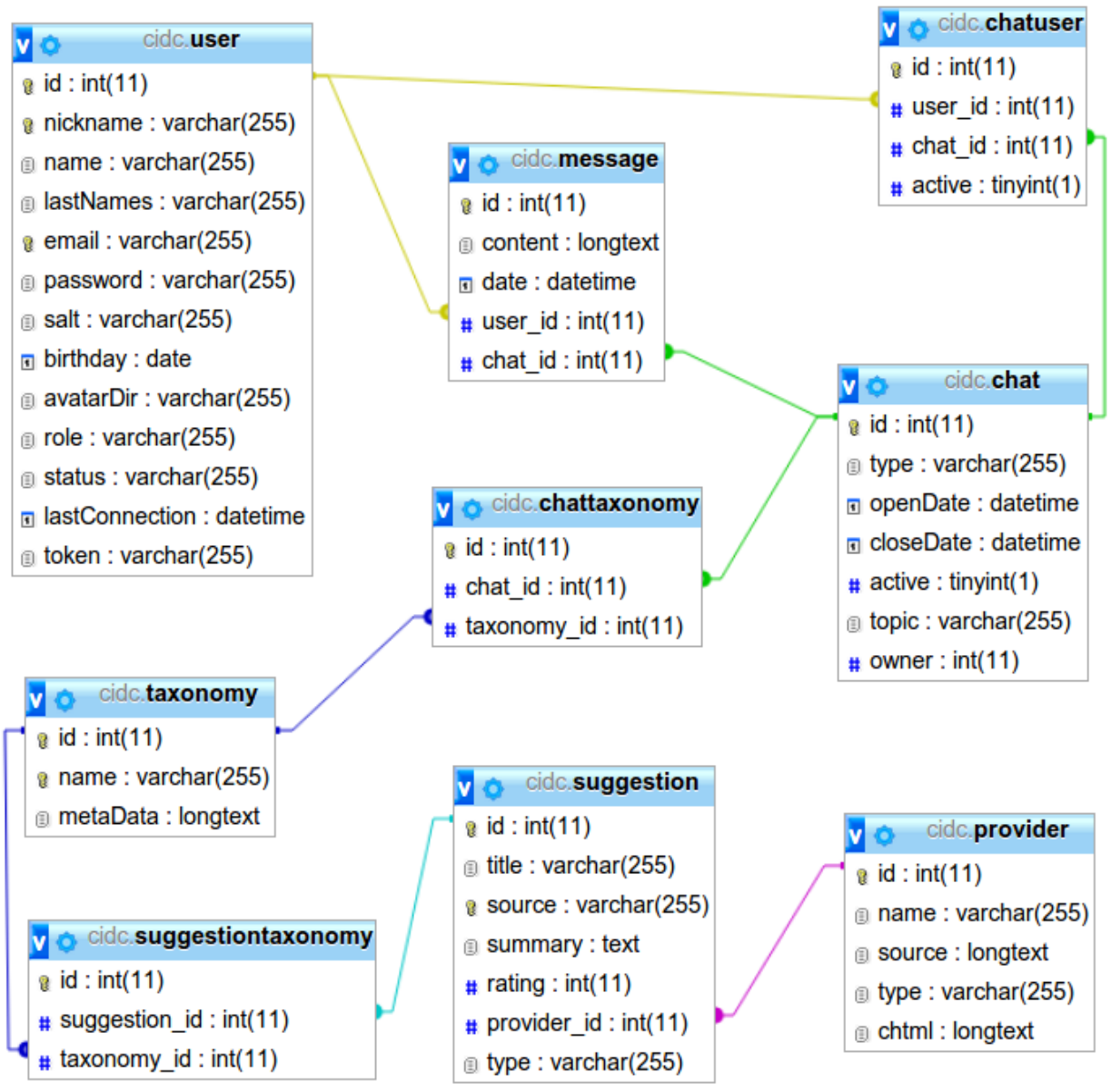

Figura 2: Modelo de datos

2.5.2 Interfaz de usuario

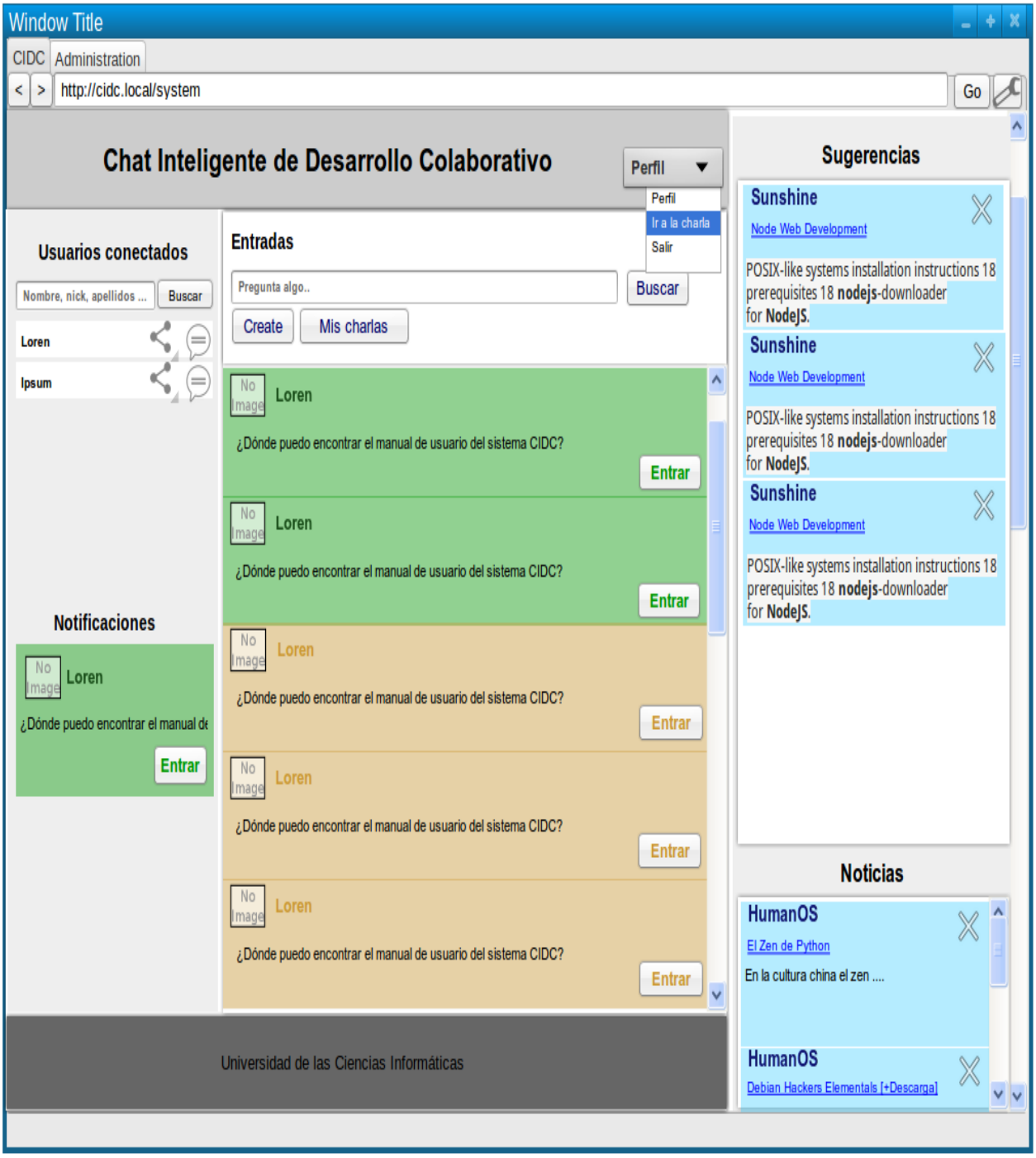

Figura 3: Prototipo de interfaz de usuario de entrada

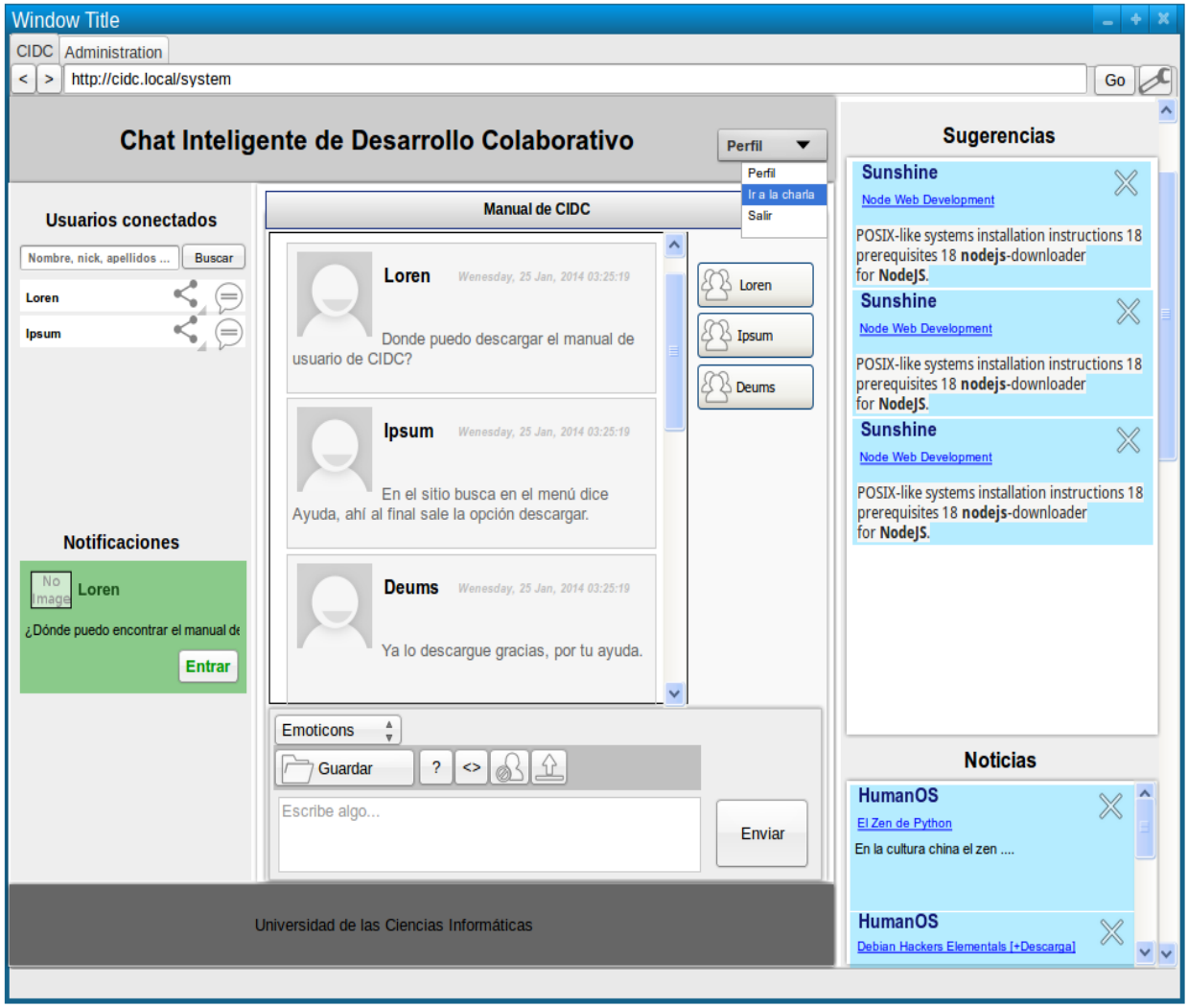

Figura 4: Prototipo de interfaz de usuario de charla

2.5.3 Tarjetas CRC

Las tarjetas CRC (Contenido, Responsabilidad y Colaboración) se elaboran con el objetivo de realizar un mejor diseño de la propuesta de solución. Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores (81). Generalmente en el borde superior en forma de título se plasma el nombre de la clase o contenido, las responsabilidades se colocan en el extremo derecho y las colaboraciones implicadas por estas responsabilidades se colocan en el extremo izquierdo. A continuación se muestra la colaboración que existe entre los módulos: *User, Chat, Suggestion y SearchEngine.*

Tabla 9 Tarjeta CRC *User*

Módulo: User	
Responsabilidades	**Colaboraciones**
Adicionar usuario Eliminar usuario Modificar usuario Visualizar usuario	*user*

Tabla 10 Tarjeta CRC *Chat*

Módulo: Chat	
Responsabilidades	**Colaboraciones**
Adicionar charla Eliminar charla Modificar charla Visualizar charla Adicionar mensaje Eliminar mensaje Visualizar mensaje	*user* *message* *chatuser*

Tabla 11 Tarjeta CRC *SearchEngine*

Módulo: SearchEngine	
Responsabilidades	**Colaboraciones**
Adicionar proveedor Eliminar proveedor Modificar proveedor Listar proveedor	*user* *chat*

Tabla 12 Tarjeta CRC *Suggestion*

Módulo: Suggestion	
Responsabilidades	**Colaboraciones**
Adicionar sugerencia Eliminar sugerencia Visualizar sugerencia	*user* *SearchEngine*

2.5.4 Patrón arquitectónico empleado

En el CIDC se usó el patrón arquitectónico Modelo-Vista-Controlador (MVC), que tiene definido el *framework* Symfony 2 para separar las distintas partes que forman la aplicación *web.* Esta arquitectura que organiza y guía la implementación de la lógica del negocio se divide en tres componentes (82):

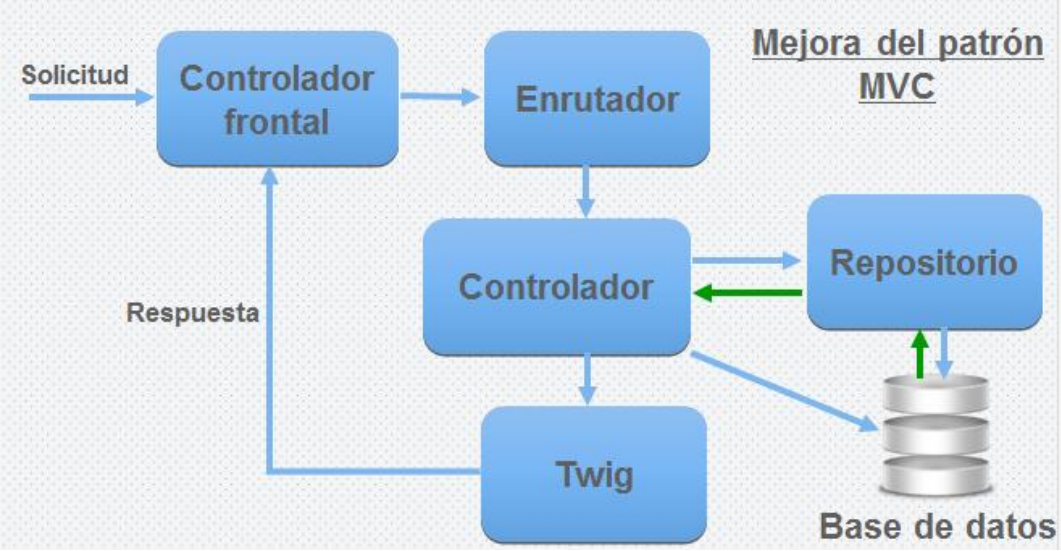

Figura 5: Patrón arquitectónico MVC

Modelo

Tiene la responsabilidad de gestionar todo el acceso a la base de datos, en Symfony 2 se emplean las clases EntityRepository.php para realizar todas las operaciones de acceso a los datos y otros métodos propios del *framework* que facilitan el manejo de la información contenida en la base de datos, con el uso del lenguaje DQL de Doctrine para las consultas.

Vista

Es la clase encargada de visualizar la información del usuario en un formato que sea adecuado para interactuar, usualmente la interfaz de usuario es sencilla. Symfony 2 gestiona las vistas a través del motor de plantillas *twig*, aunque de igual forma permite el uso de documentos HTML en la creación de las interfaces de usuario.

Controlador

Se encarga de darle respuesta a los distintos eventos que produce el usuario en su interacción con las vistas e invoca peticiones al modelo y luego envía los resultados a las vistas correspondientes para satisfacer las solicitudes del cliente.

2.5.5 Patrones diseño

Un patrón es una solución reutilizable de problemas recurrentes que ocurren durante el desarrollo del *software*. Estos ayudan a entender la solución del problema de investigación, formando un vocabulario común entre los diseñadores, permitiéndoles un mejor entendimiento del producto que se está realizando. Primeramente un patrón tiene que cumplir ciertas características como: efectividad y reusabilidad. La primera se refiere a resolver problemas similares a anteriores y la segunda es que sea aplicable a diferentes problemas de diseño en distintas circunstancias. Un patrón de diseño identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades, además de que ayuda a construir clases y a estructurar sistemas de clases (83).

Patrones GRASP utilizados

Los Patrones Generales de *Software* para Asignación de Responsabilidades a objetos (GRASP) describen los principios fundamentales del diseño de objetos para la asignación de responsabilidades y constituyen la base del cómo se diseñará el sistema (84).

Creador

El patrón creador ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan

bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización.

Se evidencia en la clase SearchEngine.java del motor de búsqueda, en la misma se crea una instancia de la clase jarvisAIS, la cual está encargada de crear todas las clases necesarias del sistema.

```
public class SearchEngine {

    public static void main(String[] args) {
        JarvisAIS jarvisAIS = new JarvisAIS();
        String action = args[0];
        switch (action) {
            case "cron":
                jarvisAIS.consumeRSSFeeds();
                break;
            case "search":
                jarvisAIS.activeDaemon("", null);
                break;
            case "cron/search":
                jarvisAIS.consumeRSSFeeds();
                jarvisAIS.activeDaemon("", null);
                break;
        }
    }
}
```

Figura 6: Utilización del patrón creador

Controlador frontal

Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que recibe los datos del usuario y los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, para aumentar la reutilización del código y a la vez tener un mayor control. Recomienda dividir los eventos del sistema en el mayor número de controladores posibles para poder aumentar la cohesión y disminuir el acoplamiento.

En el sistema todas las peticiones *web* son manipuladas por un solo controlador frontal (app.php o app_dev.php) donde se evidencia este patrón, Symfony 2 crea una instancia del *kernel* para manipular las peticiones y por cada una devuelve la respuesta correspondiente.

Este patrón se ve reflejado de esta manera, Controlador app.php.

```
require_once __DIR__.'/../app/AppKernel.php';
//require_once __DIR__.'/../app/AppCache.php';

$kernel = new AppKernel('prod', false);
$kernel->loadClassCache();
```

Figura 7: Utilización del patrón controlador frontal

Bajo acoplamiento

Este patrón de diseño mantiene las clases lo menos relacionadas entre sí, de forma tal que si se produce una modificación en alguna de ellas, exista la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre ellas.

Symfony 2 emplea el patrón arquitectónico MVC, por lo que las clases que implementan la lógica del negocio y el acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Este patrón se ve reflejado en las carpetas. A continuación se especifica datos que son almacenados en estas:

- DefaultController.php: es donde se procesa los datos enviados desde las vistas. Accede a la base de datos a través de los EntityRepository, ejecuta el *action* definido por el *routing* y devuelve una respuesta a la vista.
- *Entity*: modelo de patrón MVC.
- user.php es una entidad y representa una tabla de la base de datos.
- userRepository.php: es donde se definen todas las consultas a la base de datos. Hace uso de los atributos de la entidad.
- *Resources*: vista de patrón MVC.
- routing.yml: es donde se definen todas las rutas correspondientes al *bundle* y sus respectivos parámetros, así como el controlador frontal que debe procesar cada ruta y la función que debe ejecutar.
- index.html.twig: se visualiza la información enviada por el controlador.

```
UserBundle
    Controller
        DefaultController.php
    Entity
        user.php
        userRepository.php
    Resources
        config
            routing.yml
        public
            css
            js
            images
        views
            Default
                index.html.twig
```

Figura 8: Utilización del patrón bajo acoplamiento

Experto

La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase se considera experta si contiene toda la información necesaria para realizar la labor que tiene encomendada.

Este patrón se utiliza cuando se trabaja con Symfony 2, en las entidades, las cuales representan una tabla en la base de datos y poseen todos lo métodos y atributos relacionados con el objeto que representan. Su uso se pone de manifiesto en el siguiente fragmento de código:

```
use CIDC\system\UserBundle\Entity\user;
use CIDC\system\UserBundle\Form\userType;
 $user = new user();
 $myform = $this->createForm(new userType(), $user);
 $myform->handleRequest($req);

 $tmp_password = $user->getPassword();
 $user->setPassword($passwordCodify);
 $user->setRole('ROLE_STANDARD');
 $user->setStatus('BLOCK');
```

Figura 9: Utilización del patrón experto

Alta cohesión

Cada elemento del diseño debe realizar una labor única dentro del sistema, lo cual expresa que la información que almacena una clase debe de ser coherente y estar relacionada con ella. En

la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con alta cohesión posee poca complejidad.

Symfony 2 posee una alta interrelación entre sus distintos componentes, existe una comunicación o un flujo constante de datos de una clase a otra durante su ejecución, por lo que las aplicaciones que son desarrolladas con este *framework* poseen una alta cohesión.

Patrones GoF empleados

Los patrones de diseño GoF se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento desglosados en veintitrés patrones. Symfony utiliza muchos patrones en la estructura que concibe por defecto, como constancia de ello a continuación se nombran algunos de estos: *Command, Chain of Responsibility, Prototype, Visitor, Interpreter, Mediator, Bridge, Strategy, Factory Method* y *Template Method* (85).

Decorator

La utilización de este patrón se evidencia en métodos que pertenecen a los archivos layout.html.twig y admin_layout.html.twig, padres de todas las vistas, guardan el código html que es común en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido se integra a la plantilla padre o global mediante el motor de platillas *twig,* que permite la herencia entre las mismas, por lo que se construye un contenido dinámicamente y se visualiza a través de los archivos app.php o app_dev.php en dependencia del contexto en el que se esté empleando, desde otro punto de vista, las plantillas son decoradas con el *layout* global. Este procedimiento es una implementación del patrón decorador y a continuación se evidencia en los siguientes fragmentos de código.

Plantilla global layout.html.twig:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />

        <!-- This block allow developers select a title for each page -->
        <title>{% block title %}CIDC{% endblock %}</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
                                (... código intermedio ...)
        <!-- In this block childs pages can add they own javascript files -->
{% block add_js %}{% endblock %}
{% endblock %}
```

Figura 10: Utilización del patrón *decorator*

Plantilla index.html.twig:

```
{% extends 'UserBundle::Interface/layout.html.twig' %}

{% block add_js %}
<script src="{{ asset('bundles/user/js/site_actions.js') }}" type="text/javascript"
></script>
{% endblock %}
```

Figura 11: Utilización del patrón *decorator*

Builder

El patrón *builder* es utilizado en las clases *ManagerController* y *RegisterController* en la función *RegisterAction* y en la clase *SessionController* en la función *passwordLostAction*, donde se hace alusión a la clase MailBuilder.php la cual es la encargada de construir los correos electrónicos del sistema. En el siguiente código ejemplifica el uso del mismo.

```
use CIDC\system\UserBundle\Core\MailBuilder;
use CIDC\system\UserBundle\Core\Utilities;

        $mail_builder = new MailBuilder();
        $utility = new Utilities();

            $account_activation_msg = $mail_builder->createMailStructure(
                'http://'.$req->getHost().$req->getBaseUrl(),
                $utility->getClientIP(),
                'register',
                $user,
                $tmp_password
            );
```

Figura 12: Utilización del patrón *builder*

Conclusiones parciales

La realización del modelo de diseño de la propuesta de solución permitió concebir las funcionalidades y características que debe cumplir el sistema. Especificándose la audiencia a la que va dirigida la solución, lo cual arrojó como resultado que existen dos tipos de usuarios que interactúan con el sistema, el administrador y como usuario común estudiantes y profesores de la UCI. Las fases de exploración y planificación permitieron la elaboración de veinticuatro HU, dando lugar a la realización de cuatro iteraciones además de permitir la realización del plan de entregas que arrojó como resultado la realización de cuatro entregables de la versión 1.0 del producto.

Capítulo 3: Implementación y prueba

Introducción

En el presente capítulo se da cumplimento a las cuatro iteraciones definidas en el capítulo anterior, desplegándose las fases de implementación y prueba. Se puntualizan los estándares de codificación a utilizar para el desarrollo de la propuesta de solución. El proceso de implementación del sistema será guiado por las descripciones de las tareas de ingeniería. La fase de prueba de la metodología seleccionada se divide en dos grupos las pruebas unitarias, desarrolladas por los programadores y las pruebas de aceptación que se realizarán en compañía del cliente.

3.1 Estándar de codificación

Los estándares de codificación, también llamados estilos de programación o convenciones de código, son convenios para escribir código fuente en ciertos lenguajes de programación. Estos estándares facilitan el mantenimiento del código, sirven como punto de referencia para los programadores, mantienen un estilo de programación y ayudan a mejorar el proceso de codificación, haciéndolo más eficiente (86).

Los estándares de codificación utilizados para la implementación son los que emplea Symfony 2, que se rige por las normas de Recomendación de Estándar de PHP (PSR) por sus siglas en inglés. El *framework* utiliza las normas PSR0, PSR1 y PSR2. A continuación se propone la estructura y nomenclatura a emplear (87).

3.1.1 Estructura

- Añadir un solo espacio después de cada delimitador (,).
- Añadir un solo espacio alrededor de los operadores (==, &&, ...).
- Añadir una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último.
- Añadir una línea en blanco antes de las declaraciones *return*, a menos que el valor devuelto sea dentro de un grupo de declaraciones.
- Usar llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.
- Declarar las propiedades de clase antes que los métodos.
- Declarar métodos públicos primero, luego los protegidos y finalmente los privados.

- Utilizar paréntesis al crear instancias de clases, independientemente del número de argumentos que el constructor tenga.
- Las cadenas de mensajes de excepción deben ser concatenadas usando *sprintf*.

3.1.2 Nomenclaturas

- Utilizar mayúsculas intercaladas sin guiones bajos en nombres de variables, funciones, métodos o argumentos.
- Usar guiones bajos para nombres de opciones y nombres de parámetros.
- Prefijar las clases *abstractas* con *Abstract*.
- Sufijar las interfaces con Interface.
- Sufijar las características con *Trait*.
- Sufijar las excepciones con *Exception*.
- Utilizar caracteres alfanuméricos y guiones bajos para los nombres de archivos.

3.2 Implementación

La implementación se hace ateniendo los estándares de codificación ya creados. La implementación es la fase en la cual los desarrolladores escriben el código fuente de las aplicaciones informáticas y no se termina hasta que todas las funcionalidades hayan sido implementadas y probadas. En esta fase se propone tener en cuenta aspectos muy importantes como la programación en equipo y la disponibilidad del cliente para lograr mayores resultados en la implementación del *software* (44).

Disponibilidad del cliente: formó parte del equipo de desarrollo, describió las HU, guío la toma de decisiones, aprobó las versiones del producto y verificó el cumplimiento de los objetivos trazados.

Desarrollo en pareja: toda la implementación fue realizada por dos personas que trabajaron de forma conjunta para llevar a cabo la realización del producto de *software* en tiempo y forma.

A continuación se detallan las tareas de ingeniaría pertenecientes a la fase de implementación.

3.2.1 Tareas de ingeniería

Para llevar a cabo la correcta implementación de las HU se definen por parte del equipo de desarrollo las Tareas de Ingeniería (TI), que se realizarán en cada una de las iteraciones. Las TI también conocidas como tareas de implementación permiten a los desarrolladores obtener un nivel de detalle más avanzado que el que propician las HU, permitiéndole la representación gráfica de las responsabilidades asignadas a cada miembro del equipo de desarrollo (88).

A continuación se muestran algunas de estas TI. Para obtener más información (Ver anexo # 3).

Tabla 13 TI Autenticar usuario

Tarea de ingeniería	
No. de la tarea: 2	**No. de la HU:** 2
Nombre de la tarea: Autenticar usuario.	
Tipo de tarea: desarrollo	**Puntos estimados:** 2
Programador responsable: Rosalia Martínez García.	
Descripción: Implementar la funcionalidad autenticar usuario, asegurando la seguridad del sistema.	

Tabla 14 TI Crear charla

Tarea de ingeniería	
No. de la tarea: 9	**No. de la HU:** 9
Nombre de la tarea: Crear charla.	
Tipo de tarea: desarrollo	**Puntos estimados:** 2
Programador responsable: Rosalia Martínez García.	
Descripción: Implementar la funcionalidad crear charla haciendo uso del servidor nodejs, el motor de búsqueda para buscar información relacionada con el tema de la misma y el *framework* de JavaScript jQuery para visualizarla.	

Tabla 15 TI Adicionar mensaje

Tarea de ingeniería	
No. de la tarea: 11	**No. de la HU:** 11
Nombre de la tarea: Adicionar mensaje.	
Tipo de tarea: desarrollo	**Puntos estimados:** 2

Programador responsable: Leandro Campos Rojas.
Descripción: Implementar la funcionalidad adicionar mensaje empleando el módulo Socket.IO para enviarlo de un cliente a otro, el módulo mysql del servidor nodejs para almacenarlo en la base de datos y el *framework* de JavaScript jQuery para visualizar dinámicamente los mismos en el *chat*.

Tabla 16 TI Adicionar proveedor

Tarea de ingeniería	
No. de la tarea: 18	**No. de la HU:** 18
Nombre de la tarea: Adicionar proveedor.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5
Programador responsable: Leandro Campos Rojas.	
Descripción: implementar la funcionalidad adicionar proveedor haciendo uso de un *crud* de la entidad *provider*.	

3.3 Prueba

Las pruebas constituyen una de las fases de la metodología XP, perfiladas con el objetivo de comprobar el funcionamiento del código que se ha implementado. Esto permite aumentar la calidad de los sistemas, reduciendo el número de errores detectados. También, aumenta la seguridad y evita efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. En XP se definen dos tipos de pruebas (48):

Pruebas unitarias: encargadas de verificar el código y diseñadas por los programadores.

Pruebas de aceptación o pruebas funcionales: destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente.

3.3.1 Pruebas de aceptación

Las pruebas de aceptación fueron creadas basándose en las HU. El cliente especificó uno o diversos escenarios para comprobar que cada historia de usuario fue correctamente implementada. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. En caso de cualquier problema en la realización de estas se indica el orden de prioridad para darle resolución. Este período de prueba se conoce también como período de

caja negra, donde se definen las entradas al sistema y los resultados esperados de estas entradas. Una HU puede tener todas las pruebas de caja negra que necesite para asegurar su correcto funcionamiento. El objetivo final es garantizar que los requerimientos se cumplieron y que el sistema es aceptable (89).

A continuación se muestran algunos casos de pruebas de aceptación. Para más información ver (Ver anexo # 4).

Tabla 17 Caso de prueba de aceptación Autenticar usuario

Caso de prueba de aceptación	
Código: 2	**HU:** 2
Nombre: Autenticar usuario.	
Descripción: prueba para la funcionalidad que permite al usuario autenticarse, asegurando la seguridad del sistema.	
Condiciones de ejecución: verificar que el usuario esté previamente registrado en el sistema.	
Entrada/Pasos de ejecución: el usuario ingresa su usuario y contraseña, el sistema valida la veracidad de los datos y seguidamente se le muestra la interfaz principal del sistema.	
Resultado esperado: que el usuario pueda autenticarse satisfactoriamente.	
Evaluación de la prueba: satisfactoria.	

Tabla 18 Caso de prueba de aceptación Crear charla

Caso de prueba aceptación	
Código: 9	**HU:** 9
Nombre: Crear charla.	
Descripción: prueba para la funcionalidad que permite al usuario crear una charla.	
Condiciones de ejecución: el usuario debe estar autenticado en el sistema para poder crear una charla.	
Entrada/Pasos de ejecución: el usuario escribe su duda o pregunta en la caja de texto y selecciona la opción crear.	
Resultado esperado: que el usuario pueda crear una charla.	
Evaluación de la prueba: satisfactoria.	

Tabla 19 Caso de prueba de aceptación Adicionar mensaje

Caso de prueba de aceptación	
Código: 11	**HU:** 11
Nombre: Adicionar mensaje.	
Descripción: prueba para la funcionalidad que permite al usuario adicionar un mensaje.	
Condiciones de ejecución: el usuario debe estar autenticado y posteriormente seleccionar a quien quiere enviar el mensaje.	
Entrada/Pasos de ejecución: el usuario redacta un argumento en el área de texto y selecciona la opción enviar.	
Resultado esperado: que el usuario pueda adicionar un mensaje y enviarlo.	
Evaluación de la prueba: satisfactoria.	

Tabla 20 Caso de prueba de aceptación Adicionar proveedor

Caso de prueba de aceptación	
Código: 21	**HU:** 21
Nombre: Adicionar proveedor.	
Descripción: prueba para la funcionalidad que permite al usuario adicionar un proveedor.	
Condiciones de ejecución: el usuario debe estar autenticado como administrador para poder llevar a cabo esta funcionalidad.	
Entrada/Pasos de ejecución: el sistema muestra al administrador la lista de los proveedores existentes y le posibilita crear una nueva entrada con los datos del proveedor a adicionar (nombre, fuente, tipo y Chtml) seguidamente opta por la opción crear. Una vez adicionado se mostrará como un resultado más en la lista de proveedores.	
Resultado esperado: que el administrador pueda adicionar correctamente un proveedor siguiendo las pautas definidas.	
Evaluación de la prueba: satisfactoria.	

3.3.2 Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos y componentes deben de pasar por estas antes de ser liberados o publicados. En este sentido, el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por

otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo. Estas pruebas no generan artefactos y no son directamente palpables para el cliente (90).

Las pruebas unitarias realizadas al código en Symfony 2, empleando los pasos que define el *framework* para verificar la correctitud del mismo y la librería PHPUnit, que satisface dichas necesidades.

La siguiente imagen muestra la clase del núcleo del sistema Share, la cual es la encargada de obtener el contenido de los archivos de configuración del API de nodejs, el motor de búsqueda en Java y el archivo parameters.yml que contiene la configuración Symfony 2.

```php
<?php

namespace CIDC\system\UserBundle\Core;

use CIDC\system\UserBundle\Core\File\FileReader;
use CIDC\system\UserBundle\Core\File\FileWriter;

class Share {

    public function getInfo($type) {
        $COMPONENTS_PATH = __DIR__ . '/../../../components/';
        $SHARE_FILE_PATH = $COMPONENTS_PATH . '/ChatBundle/API/share.json';
        $reader = new FileReader();
        return $reader->read($SHARE_FILE_PATH, $type);
    }
```

Figura 13: Clase Share del núcleo del sistema

La clase *ShareTest* es la encargada de probar todo el código de la clase *Share*, donde se definen distintos juegos de datos para ver la respuesta de la misma ante disímiles situaciones.

```php
<?php

namespace CIDC\system\UserBundle\Tests\Core;

use Symfony\Component\Yaml\Yaml;
use CIDC\system\UserBundle\Core\Share;
use CIDC\system\UserBundle\Core\File\FileReader;
use CIDC\system\UserBundle\Core\File\FileWriter;

class ShareTest extends \PHPUnit_Framework_TestCase
{
    public function testGetInfo_Type_Valid()
    {
        $share = new Share();
        $result = $share->getInfo('text');
        $this->assertTrue($result != false);
        $result = $share->getInfo('json');
        $this->assertTrue($result != false);
        $result = $share->getInfo('array');
        $this->assertTrue($result != false);
    }

    public function testGetInfo_Type_Invalid()
    {
        $share = new Share();
        $result = $share->getInfo('invalid');
        $this->assertEquals(false, $result);
    }
```

Figura 14: Clase *ShareTest* destinada a probar la clase *Share*.

3.3.3 Análisis de los resultados de las pruebas

Resultado de las pruebas de aceptación

Luego de ejecutadas las pruebas de aceptación, se detectaron algunas no conformidades, las cuales se dividen en significativas, no significativas y recomendaciones. Entre las no significativas se detectaron, errores ortográficos, de interfaz y apariencia del sistema. Como parte de las no conformidades significativas se encontraron funciones incorrectas o ausentes y errores de validación. A continuación se muestra una gráfica en la que se representan la cantidad de no conformidades detectadas, a las que se le dio solución una vez terminada cada iteración de prueba.

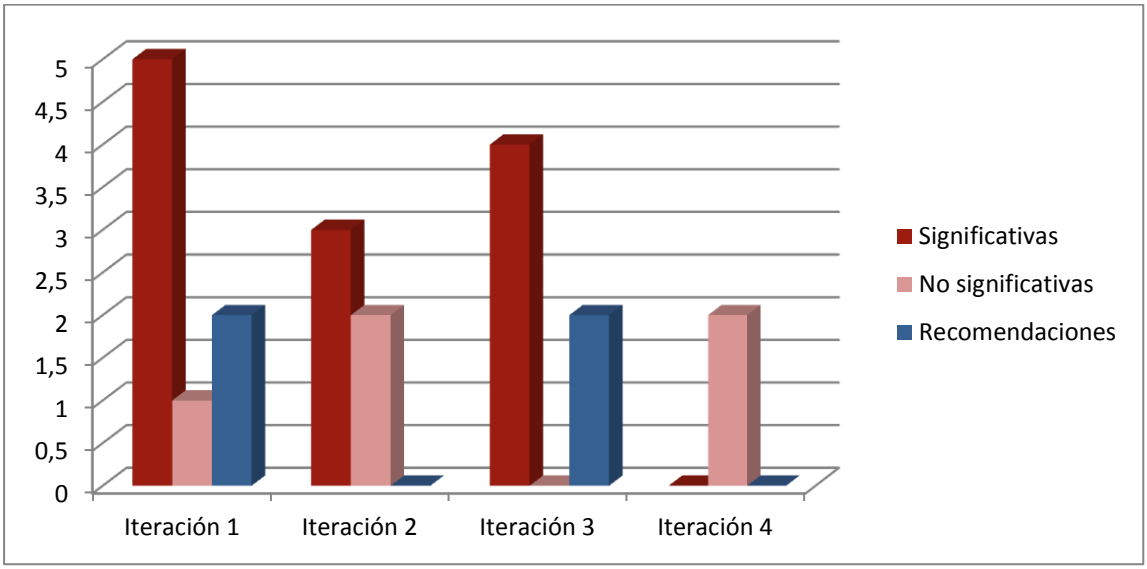

Figura 15: No conformidades por iteración

Resultado de las pruebas unitarias

Las pruebas realizadas permitieron validar el código implementado, se encontraron 9 errores en una primera iteración de las pruebas, los cuales fueron mitigados parcialmente. En la segunda iteración se lograron eliminar tanto los errores pendientes como 5 nuevos errores encontrados tras finalizar la adicción de las nuevas funcionalidades. En una tercera iteración solo se detectaron 2 errores que fueron erradicados inmediatamente. Al realizar la cuarta iteración ya se encontraban sentadas las bases para codificar de una forma más eficiente en el equipo de desarrollo, por lo que satisfactoriamente solo se detectó un error, de menor envergadura que fue rápidamente eliminado.

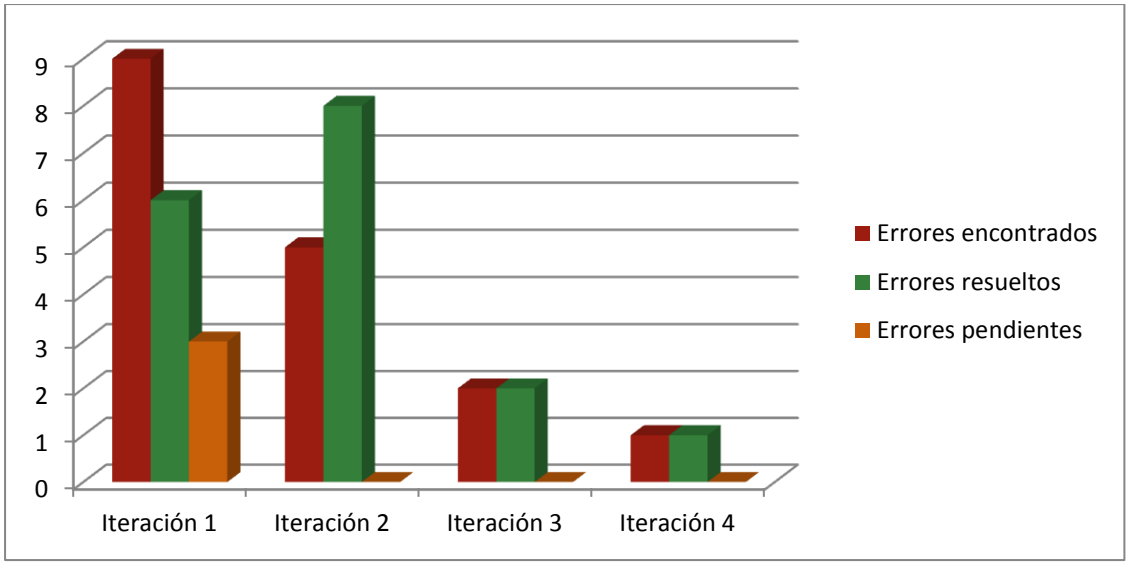

Figura 16: No conformidades por iteración

Esta imagen muestra los resultados obtenidos una vez ejecutas las pruebas unitarias a la clase *ShareTest* donde se muestra como resultado que se hicieron siete pruebas, en una de estas se realizan 3 comprobaciones por lo que se obtuvieron 9 resultados satisfactorios y ningún error.

```
leandro@d5-l207-pc27:/var/www/Leandro/CIDC$ phpunit -c app src/CIDC/system/UserBundle/Tests/Core/ShareTest.php
PHPUnit 3.6.10 by Sebastian Bergmann.

Configuration read from /var/www/Leandro/CIDC/app/phpunit.xml.dist

.......

Time: 0 seconds, Memory: 2.50Mb

OK (7 tests, 9 assertions)
leandro@d5-l207-pc27:/var/www/Leandro/CIDC$
```

Figura 18: Resultados de las pruebas realizadas a la clase *Share*

Conclusiones parciales

El estudio realizado en el presente capítulo permitió seleccionar el estándar de codificación a utilizar para que el código tenga una misma estructura en todo momento y siga las nomenclaturas definidas a utilizar en Symfony 2. Los métodos de pruebas permitieron validar el correcto desempeño de la aplicación, tanto a nivel de código como en las funcionalidades presentes en la interfaz de usuario donde se comprobó que dichas funcionalidades fueron las esperadas por el cliente. El desglose de las historias de usuario en tareas de ingenierías facilitó la implementación haciéndola menos complejas, estas posibilitaron a los programadores obtener en cada iteración, una versión funcional del producto.

Conclusiones Generales

Conclusiones generales

Después de desarrollar el presente trabajo y analizar los resultados obtenidos, se arriba a las siguientes conclusiones:

- Los métodos científicos, permitieron desarrollar la teoría que sustenta la investigación.
- La selección de la metodología de desarrollo, las herramientas, lenguajes de programación y tecnologías más apropiadas para dar cumplimiento al objetivo general, es un resultado del análisis del marco teórico realizado.
- El diseño desarrollado para la aplicación, permitió la implementación de funcionalidades que dieron solución a la problemática descrita.
- Las pruebas unitarias y de aceptación (caja blanca y caja negra) realizadas a la aplicación permitieron corregir los errores encontrados en cada iteración para obtener un producto con la calidad requerida.

Recomendaciones

Recomendaciones

- Tomar el presente trabajo como material de estudio para el desarrollo de aplicaciones similares.
- Generalizar el presente trabajo en el resto de las facultades de la UCI y a otras universidades donde se encuentren desarrolladores.
- Vincular el sistema con buscadores existentes en otras universidades del país.
- Se propone continuar con el desarrollo del componente *SuggestionBundle*, el cual necesita un sistema recomendador que optimice los resultados encontrados por el motor de búsqueda.
- Mejorar la administración del sistema, así como aumentar sus capacidades, en cuanto a notificaciones en tiempo real, monto de información visible simultáneamente y las capacidades del servidor del API del *chat*.

Referencias bibliográficas

Referencias bibliográficas

1. **Calderon, Anita.** Historia de la Informática. [En línea] [Citado el: 20 de Noviembre de 2013.] https://sites.google.com/site/gredossndiegocom/evolucion.

2. Real Academia Española. *Real Academia Española.* [En línea] [Citado el: 21 de Noviembre de 2013.] http://lema.rae.es/drae/srv/search?id=Adyw1Z7m32x5i5ngcfZ.

3. **Delgado, Jorge García.** Servicios de Red e Internet. [En línea] [Citado el: 10 de Diciembre de 2014.] http://jgdasir.files.wordpress.com/2012/02/ud-07-instalacion-y-administracion-de-servicios-de-mensajeria.pdf.

4. [En línea] [Citado el: 26 de Febrero de 2014.] http://definicion.de/protocolo-de-comunicacion.

5. **Rosas, Juan Eladio Sánchez.** El Mundo es Open Source. [En línea] 10 de Febrero de 2009. [Citado el: 27 de Febrero de 2014.] http://blogs.antartec.com/opensource/2009/02/xmpp/.

6. XMPP Standards Foundation. [En línea] [Citado el: 28 de Febrero de 2014.] http://xmpp.org/about-xmpp/technology-overview/muc/.

7. MSN Messenger Protocol. [En línea] 2003. [Citado el: 28 de Febrero de 2014.] http://www.hypothetic.org/docs/msn/general/overview.php.

8. SYSDTOR. [En línea] [Citado el: 22 de Febrero de 2014.] http://sisdtor.wordpress.com/2012/02/08/protocolos-para-la-mensajeria-instantanea/.

9. Comunicación unidireccional, bidireccional y multidireccional (redes de comunicación). [En línea] 30 de Octubre de 2011. [Citado el: 26 de Febrero de 2014.] http://claseuvaq.wordpress.com/2011/10/30/comunicacion-unidireccional-bidireccional-y-multidireccional-redes-de-comunicacion/.

10. **Exposito, Dino.** MSDN Magazine. [En línea] 2014. [Citado el: 13 de Enero de 2014.] http://msdn.microsoft.com/es-es/magazine/hh975342.aspx.

11. **López, Fransisco.** Node Hispano. [En línea] 2014. [Citado el: 20 de Marzo de 2014.] http://www.nodehispano.com/2012/09/introduccion-a-socket-io-nodejs/.

12. Donflopez. [En línea] 2 de Septiembre de 2012. [Citado el: 18 de Marzo de 2014.] http://donflopez.tumblr.com/post/30737985045/introduccion-a-socket-io.

Referencias bibliográficas

13. **Abernethy, Michael.** developerWorks. [En línea] 14 de Junio de 2011. [Citado el: 26 de Febrero de 2014.] http://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/.

14. Real Academia Española. *Real Academia Española.* [En línea] [Citado el: 3 de Diciembre de 2013.] http://lema.rae.es/drae/?val=ciber-charla.

15. definición abc. [En línea] [Citado el: 2 de Diciembre de 2013.] http://www.definicionabc.com/?s=Chat.

16. Definición.de. [En línea] 208. [Citado el: 4 de Diciembre de 2013.] http://definicion.de/chat/.

17. **Vanessa Mesa, Ana Ortega, Lydia FErnández y Juani Fernández.** En Clave de TIC. [En línea] [Citado el: 6 de Diciembre de 2013.] http://tice.wikispaces.com/Chat.

18. Social. [En línea] 2014. [Citado el: 23 de Marzo de 2014.] http://www.hisocial.com/esp/blog/cuantos-usuarios-tiene-facebook.

19. undernews. [En línea] 26 de Octubre de 2011. [Citado el: 12 de Marzo de 2014.] http://www.undernews.com/2011/10/26/facebook-chat-alert-te-avisa-cuando-tus-amigos-se-conectan/.

20. LibrosWeb. [En línea] [Citado el: 5 de Febrero de 2014.] http://librosweb.es/ajax/capitulo_1.html.

21. Moodle. [En línea] [Citado el: 05 de Enero de 2014.] http://docs.moodle.org/all/es/Acerca_de_Moodle.

22. **SANCHO, JESÚS BAÑOS.** *MANUAL DE CONSULTA PARA EL PROFESORADO .* Getafe : IES Satafi (Getafe), 2007.

23. **Torres, karla.** Prezi. *Prezi.* [En línea] [Citado el: 7 de Diciembre de 2013.] http://prezi.com/sr_mixyevmdh/software-colaborativo-o-groupware/.

24. **Bibb, Luis Mariano.** Postgrado Facultad de Informática. [En línea] [Citado el: 5 de Diciembre de 2013.] http://postgrado.info.unlp.edu.ar/Carreras/Magisters/Ingenieria_de_Software/Investigacion/Bibbo_Mariano.pdf.

25. *Revista Ciencias Básicas UJAT.* **González, Dr. Abdiel E. Cáceres.** 2, Diciembre 2005, Vol. 4.

Referencias bibliográficas

26. **Baecker, Ronald M.** Readings in Groupware and Computer-Supported Cooperative Work. [En línea] 1992. [Citado el: 5 de Marzo de 2014.] http://books.google.com/books?id=_z5d7iuh8IAC&lpg=PR11&ots=PlWujtprwC&dq=groupware%20concept&lr&pg=PP1#v=onepage&q=groupware%20concept&f=false.

27. **Anne Powell Ives, Gabriele Piccoli, Blake.** Virtual Teams. [En línea] 2004. [Citado el: 5 de Marzo de 2014.] https://wiki.cs.columbia.edu/download/attachments/1979/p6-powell.pdf?version=1&modificationDate=1173925425000.

28. Linux Community. [En línea] 2008. [Citado el: 8 de Marzo de 2014.] http://www.linux.org/.

29. Ubuntu. [En línea] 2008. [Citado el: 8 de Marzo de 2014.] http://www.ubuntu.com/community/conduct.

30. Debian. [En línea] 2009. [Citado el: 8 de Marzo de 2014.] http://www.debian.org.

31. OpenSuse. [En línea] 2010. [Citado el: 8 de Marzo de 2014.] http://www.opensuse.org.

32. Mozilla Firefox. [En línea] 2010. [Citado el: 8 de Marzo de 2014.] http://www.mozilla.com.

33. PostgreSQL. [En línea] 2010. [Citado el: 10 de Marzo de 2014.] http://www.postgresql.org.

34. RICE. [En línea] 2014. [Citado el: 11 de Marzo de 2014.] https://rice.uci.cu/?page_id=2098.

35. **Pinto, María.** Busqueda y Recuperación de Información. [En línea] 2004. [Citado el: 5 de Febrero de 2014.] http://www.mariapinto.es/e-coms/recu_infor.htm#ri1.

36. **Almaguer, José A. Cruz.** *Buscador Web.* 2004.

37. **Méndez, Francisco J. Martínez.** *Propuesta y desarrollo de un modelo para la evaluación de la recuperación de información en internet.* 2002.

38. **Aguirre, Jorge D.** Arquitectura de un buscador. [En línea] 2007. [Citado el: 15 de Febrero de 2014.] http://buscadores.fullblog.com.ar/post/arquitectura_de_un_buscador_531191953898/.

39. **Herrera, Antonio G. López.** *Modelos de Sistemas de Recuperación de Información Lingüística.* 206.

Referencias bibliográficas

40. Universia. *Universia.* [En línea] [Citado el: 3 de Diciembre de 2013.] http://biblio.universia.es/catalogos-recursos/metabuscadores/.

41. **Carla de Angelis, Sara Arrieta.** Slideshare. *Slideshare.* [En línea] [Citado el: 29 de Diciembre de 2013.] http://www.slideshare.net/cholicamisari/manejo-de-informacin.

42. Consoft. [En línea] 2002. [Citado el: 20 de Febrero de 2014.] http://www.consoft.es/noticias/news_text.asp?id=33219.

43. **Estrada, Yelena Hernández.** *Análisis del Módulo Proceso Confiscatorio de Bienes del proyecto Sistema de Gestión .* La Habana : s.n., 2009.

44. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* **Letelier, Patricio.** 26, Buenos Aires : s.n., 2006, Vol. 05.

45. **Wilancis.** wikispaces. [En línea] [Citado el: 23 de Enero de 2014.] http://procesosdesoftware.wikispaces.com/METODOLOGIAS+PARA+DESARROLLO+DE+SOFTWARE.

46. **Chaves, Débora Galvin.** slideshare. [En línea] [Citado el: 24 de Enero de 2014.] http://www.slideshare.net/deborahgal/diferencias-entre-scrum-y-xp-12219336.

47. **José H. Canós, Patricio Letelier y Carmen Penadés.** Metodologías Ágiles en el Desarrollo de Software. [En línea] [Citado el: 15 de 1 de 2014.] http://www.willydev.net/descargas/prev/TodoAgil.pdf.

48. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** Departamento de Lenguajes y Sitemas Informáticos. [En línea] [Citado el: 28 de Abril de 2014.] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

49. **Iván Andres Vidal, Diego Luis, Lasso, Alexander Pitto Medina.** ASD (Adaptive Software Development). [En línea] [Citado el: 26 de Enero de 2014.] http://www.slideshare.net/urumisama/metodologia-agil-asd.

50. **Chaffer, Jonathan. Drupal.** Drupal programming from an object-oriented perspective. [En línea] [Citado el: 28 de 2 de 2014.] http://drupal.org/node/547518.

51. **Torres, Ramor.** slideshare. [En línea] 6 de 2009. [Citado el: 20 de Enero de 2014.] http://www.slideshare.net/rtorres462003/metologa-agiles-desarrollo-software-xp-1709082.

52. Symfony.es. [En línea] [Citado el: 21 de Enero de 2013.] http://symfony.es/libro/.

Referencias bibliográficas

53. **Carrero, Angel.** Programación en castellano. [En línea] [Citado el: 6 de Febrero de 2014.] http://www.programacion.com/articulo/conceptos_basicos_de_orm_object_relational_mapping_349.

54. MySQL. [En línea] [Citado el: 29 de Febrero de 2014.] http://dev.mysql.com/doc/refman/5.0/es/features.html#.

55. PhpMyAdmin. [En línea] [Citado el: 27 de Febrero de 2014.] http://es.opensuse.org/PhpMyAdmin#Caracter.C3.ADsticas.

56. The Apache Software Foundation. [En línea] [Citado el: 10 de Enero de 2014.] http://www.apache.org/licenses/LICENSE-2.0.html.

57. Observatorio Tegnológico. [En línea] 21 de 4 de 208. [Citado el: 20 de Enero de 2014.] http://recursostic.educacion.es/observatorio/web/es/software/servidores/580-elvira-mifsud.

58. [En línea] [Citado el: 8 de Febrero de 2014.] http://www.masquecodificar.es/2012/06/lighttpd-como-alternativa-apache.html.

59. Node.js. [En línea] 2014. [Citado el: 15 de Marzo de 2014.] http://www.nodejs.org.

60. **Fette, Ian.** Internet Engineering Task Force (IETF). [En línea] 12 de 2011. [Citado el: 10 de Febrero de 2014.] http://tools.ietf.org/html/rfc6455.

61. **Maqueira Gonzáalez, Jeyke.** *Cliente JQuery para la integración del Servidor de Inteligencia de Negocio con el Sistema Informativo (SI) de la Administracion Provincial de Artemisa.* Artemisa : s.n., 2012.

62. **García, Fernadez.** wordpress. [En línea] 25 de 1 de 2013. [Citado el: 28 de Enero de 2003.] http://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

63. **Dan, Michelangelo Van.** Slideshare. [En línea] 25 de 4 de 2013. [Citado el: 27 de Enero de 2014.] http://www.slideshare.net/DragonBe/quality-assurance-for-php-projects-with-phpstorm.

64. mpsoftware. [En línea] 2014. [Citado el: 16 de Marzo de 2014.] http://www.mpsoftware.dk/phpdesigner.php .

65. **Muñoz, Vicente Eslava.** Conceptos Avanzados. 208.

Referencias bibliográficas

66. NetBeans. [En línea] 2013. [Citado el: 19 de Febrero de 2014.] https://netbeans.org/index_es.html.

67. 4r Blog. [En línea] 12 de Julio de 2012. [Citado el: 26 de Marzo de 2014.] http://www.4rsoluciones.com/las-ventajas-de-php-para-el-desarrollo-de-aplicaciones-y-sitios-web/.

68. **Cruz, Dayron Iñigo.** *Cuidad Habana.* s.l. : 2006-2007.

69. CCS3 HTML5. [En línea] [Citado el: 27 de Marzo de 2014.] http://html5.dwebapps.com/que-es-css3/.

70. eduglogs. [En línea] 2014. [Citado el: 2 de Marzo de 2014.] http://timanco.edublogs.org/ventajas-y-desventajas-del-css/.

71. **Ledesma, Rubén.** *Introducción al Bootstrap.Desarrollo de un ejemplo acompañado de software de aplicación.* Argentina, Mar del Plata : s.n.

72. **Hernandez, Fernanda.** Microsoft SQL Server. [En línea] 5 de Febrero de 2012. [Citado el: 10 de Abril de 2014.] http://technet.microsoft.com/es-es/library/ms155949(v=sql.105).aspx.

73. **Ivar JacobSon, Grady Booch , James Rumbaugh.** *El Proceso Unificado de Software.* 2000.

74. **Joskowicz, José.** *Reglas y Prácticas en eXtreme Programming.* España : Universidad de Vigo : s.n., 2008.

75. WebAcademia. [En línea] 20013. [Citado el: 21 de Marzo de 2014.] http://centrodeartigos.com/articulos-educativos/article_11461.html.

76. PMOinformática.com. *La Oficina de Proyectos de Informática.* [En línea] [Citado el: 22 de Marzo de 2014.] http://www.pmoinformatica.com/2013/04/que-son-las-historias-de-usuario-7.html.

77. ProcesosdeSoftware. *ProcesosdeSoftware.* [En línea] [Citado el: 25 de Marzo de 2014.] http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP.

78. **Joskowicz, Ing. José.** *Reglas y Prácticas en eXtreme Programming.* s.l. : Collection Openlibra, 2008.

79. Tripod.com. [En línea] [Citado el: 5 de Mayo de 2014.] http://programacionextrema.tripod.com/fases.htm.

Referencias bibliográficas

80. Metodología XP. *Metodología XP.* [En línea] [Citado el: 3 de Abril de 2014.] https://sites.google.com/site/xpmetodologia/marco-teorico/funcionamiento.

81. **Cervantes, DR Humberto.** Universidad Autónoma Metropolitana. *Universidad Autónoma Metropolitana.* [En línea] [Citado el: 24 de Abril de 2014.] http://www.humbertocervantes.net/homepage/itzamna/DOCUMENTACION/Doc2.html..

82. **Fabien Potencier, Ryan Weaver.** Libros Web. [En línea] [Citado el: 29 de Abril de 2014.] http://librosweb.es/symfony_1_2/capitulo_2/el_patron_mvc.html.

83. **Alises, David Villa.** Arco. [En línea] [Citado el: 29 de Abril de 2014.] http://arco.esi.uclm.es/~david.villa/pensar_en_C++/vol2/C10.html.

84. **Astudillo, Marcello Visconti y Hermás.** Departamento de Informática. [En línea] [Citado el: 26 de Abril de 2014.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

85. **Debrauwer, Laurent.** *Patrones de diseño para C#.* Barcelona : Editions ENI ISBN: 978-2-7460-7260-2., febrero,2012.

86. **Carmen, Arias Calleja y Mauel.** Estándares de codificación. [En línea] [Citado el: 03 de Mayo de 2014.] http://www.chubut.gov.ar/informatica/docs/EstandaresCodificacion.pdf.

87. **Pacheco, Nacho.** Manual de Symfony en Español. [En línea] [Citado el: 4 de Mayo de 2014.] http://gitnacho.github.io/symfony-docs-es/contributing/code/standards.html.

88. **Ariel Erlijman Piwen, Alejandro Goyén Fros.** Problemas y Soluciones en la Implementacion de Extreme Programming. [En línea] Abril de 2001. [Citado el: 12 de Mayo de 2014.] http://www.alejandrogoyen.com/MemoriaDeGradoXP.pdf.

89. INFORMATION SYSTEMs GROUP. [En línea] [Citado el: 28 de Abril de 2014.] http://indalog.ual.es/mtorres/LP/Prueba.pdf.

90. Metodología XP. [En línea] [Citado el: 05 de 04 de 2014.] https://sites.google.com/site/xpmetodologia/marco-teorico/funcionamiento.

91. Request For Comments. [En línea] [Citado el: 29 de Noviembre de 2013.] http://www.rfc-es.org/rfc/rfc2810-es.txt.

Anexos

Anexo # 1: Metodologías de desarrollo de *software*

Comparación entre metodologías tradicionales y metodologías ágiles (44).

Metodología Ágil	Metodología Tradicional
Pocos Artefactos. El modelado es prescindible, modelos desechables.	Más Artefactos. El modelado es esencial, mantenimiento de modelos
Pocos Roles, más genéricos y flexibles	Más Roles, más especificos
No existe un contrato tradicional, debe ser bastante flexible	Existe un contrato prefijado
Cliente es parte del equipo de desarrollo (además in-situ)	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 que la arquitectura se defina tempranamente en el proyecto
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo	Énfasis en la definición del proceso: roles, actividades y artefactos
Basadas en heuristicas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Se esperan cambios durante el proyecto	Se espera que no ocurran cambios de gran impacto durante el proyecto

Tabla 2. Diferencias entre metodologías ágiles y no ágiles

Anexo # 2: Historias de usuarios

Tabla 21 HU Autenticar usuario

Historia de usuario	
Nombre: Registrar usuario.	**Número:** 1
Usuario: todos	**Iteración asignada:** 1
Prioridad en el negocio: alta	**Puntos estimados:** 1
Riesgo en el desarrollo: alto	
Descripción: el usuario selecciona la opción registrar usuario, el sistema muestra el formulario con los datos que el usuario debe ingresar al sistema (usuario, nombre, apellidos, correo electrónico, contraseña, cumpleaños y avatar). Una vez insertada correctamente la información se acciona sobre el botón confirmar para registrar el usuario.	
Observaciones: da cumplimiento al requisito funcional número 1. Para un usuario poder realizar cualquier acción en el *chat*, debe estar previamente registrado.	

Tabla 22 HU Modificar usuario

Historia de usuario	
Nombre: Modificar usuario.	**Número:** 3
Usuario: todos	**Iteración asignada:** 1
Prioridad en el negocio: media	**Puntos estimados:** 0.5
Riesgo en el desarrollo: media	
Descripción: el usuario elige la opción editar cuenta, que contiene los datos a modificar (usuario, nombre, apellidos, correo electrónico, contraseña, cumpleaños y avatar) luego elige actualizar para guardar los cambios o cancelar en caso contrario.	
Observaciones: se necesita estar autenticado. Esta HU responde a la funcionalidad número 3.1 que está contenida en el gestionar usuario.	

Tabla 23 HU Listar usuario

Historia de usuario	
Nombre: Listar usuarios.	**Número:** 4
Usuario: todos	**Iteración asignada:** 1
Prioridad en el negocio: alta	**Puntos estimados:** 0.5
Riesgo en el desarrollo: alto	
Descripción: el sistema le permite al usuario visualizar los contactos que están conectados, en tiempo real.	
Observaciones: esta HU responde a la funcionalidad número 3.2 que está contenida en el gestionar usuario. Se necesita estar autenticado para poder llevar a cabo esta HU.	

Tabla 24 HU Eliminar usuario

Historia de usuario	
Nombre: Eliminar usuario.	**Número:** 5
Usuario: administrador	**Iteración asignada:** 1
Prioridad en el negocio: alta	**Puntos estimados:** 0.5
Riesgo en el desarrollo: bajo	
Descripción: el administrador elige la opción ver lista. A continuación se muestra una lista con todos los usuarios registrados en el sistema y selecciona la opción mostrar, donde se visualizan los datos del usuario seleccionado y elige la opción eliminar usuario.	
Observaciones: da cumplimiento al requisito funcional número 3.3, para un usuario poder realizar cualquier acción en el *chat*, debe estar previamente autenticado.	

Tabla 25 HU Actualizar charla

Historia de usuario	
Nombre: Actualizar charla.	**Número:** 6

Usuario: todos	**Iteración asignada:** 2
Prioridad en el negocio: alta	**Puntos estimados:** 0.6
Riesgo en el desarrollo: alto	
Descripción: el usuario escribe un mensaje y lo envía a todos los usuarios que estén conectados a la conversación, la ventana del *chat* se actualiza automáticamente visualizando el nuevo mensaje.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad 6.1 que pertenece al gestionar charla con número 6.	

Tabla 26 HU Listar charlas activas

Historia de usuario	
Nombre: Listar charlas activas.	**Número:** 7
Usuario: todos	**Iteración asignada:** 2
Prioridad en el negocio: alta	**Puntos estimados:** 0.6
Riesgo en el desarrollo: alto	
Descripción: el sistema muestra al usuario todas las charlas activas en ese momento.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad 6.2 que pertenece al gestionar charla con número 6.	

Tabla 27 HU Listar charlas almacenadas

Historia de usuario	
Nombre: Listar charlas almacenadas.	**Número:** 8
Usuario: administrador	**Iteración asignada:** 2
Prioridad en el negocio: alta	**Puntos estimados:** 0.3
Riesgo en el desarrollo: alto	
Descripción: el sistema muestra al administrador la lista de todas las charlas almacenadas de los	

usuarios.
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad 6.3 que pertenece al gestionar charla con número 6.

Tabla 28 HU Eliminar charla

Historia de usuario	
Nombre: Eliminar charla.	**Número:** 10
Usuario: administrador	**Iteración asignada:** 2
Prioridad en el negocio: alta	**Puntos estimados:** 0.1
Riesgo en el desarrollo: alto	
Descripción: el sistema muestra al administrador la lista de todas las charlas almacenadas de los usuarios y se elige editar, se mostrarán los datos de la charla y se opta por la opción eliminar charla y se elimina la charla deseada.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad 6.4 que pertenece al gestionar charla con número 6.	

Tabla 29 HU Listar mensaje

Historia de usuario	
Nombre: Listar mensaje.	**Número:** 12
Usuario: administrador	**Iteración asignada:** 3
Prioridad en el negocio: alta	**Puntos estimados:** 0.5
Riesgo en el desarrollo: medio	
Descripción: el sistema muestra al administrador la lista de todos los mensajes escritos por los usuarios.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad 7.2 que pertenece al gestionar mensaje con número 7.	

Tabla 30 HU Eliminar mensaje

Historia de usuario	
Nombre: Eliminar mensaje.	**Número:** 13
Usuario: administrador	**Iteración asignada:** 3
Prioridad en el negocio: alta	**Puntos estimados:** 0.5
Riesgo en el desarrollo: alto	
Descripción: el sistema muestra al administrador la lista de todos los mensajes escritos por los usuarios y se elige la opción eliminar para eliminar el mensaje deseado.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad 7.3 que pertenece al gestionar charla con número 7.	

Tabla 31 HU Listar Mensajes del historial

Historia de usuario	
Nombre: Listar mensajes del historial.	**Número:** 14
Usuario: todos	**Iteración asignada:** 3
Prioridad en el negocio: media	**Puntos estimados:** 0.5
Riesgo en el desarrollo: bajo	
Descripción: el usuario puede visualizar los mensajes anteriores de una conversación seleccionando la opción ver historial.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad número 8.	

Tabla 32 HU Contenido en texto plano

Historia de usuario	
Nombre: Contenido en texto plano.	**Número:** 15

Usuario: todos	**Iteración asignada:** 3
Prioridad en el negocio: alta	**Puntos estimados:** 0.2
Riesgo en el desarrollo: alto	
Descripción: le permite al usuario escribir mensajes en texto plano y código embebido.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad número 9.	

Tabla 33 HU Buscar charla

Historia de usuario	
Nombre: Buscar charla.	**Número:** 16
Usuario: todos	**Iteración asignada:** 3
Prioridad en el negocio: alta	**Puntos estimados:** 0.1
Riesgo en el Desarrollo: alto	
Descripción: el usuario escribe un tema o texto, selecciona buscar y se le muestra un listado de las charlas que estén relacionadas. El usuario selecciona la opción mis charlas y se le muestra un listado de todas las charlas que ha creado.	
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad número 10.	

Tabla 34 HU Buscar usuario

Historia de usuario	
Nombre: Buscar usuario.	**Número:** 17
Usuario: todos	**Iteración asignada:** 3
Prioridad en el negocio: alta	**Puntos estimados:** 0.1
Riesgo en el desarrollo: alto	

Descripción: el usuario escribe el nombre, el apodo o los apellidos de la persona a buscar, luego selecciona la opción buscar y se le muestra un listado con las coincidencias de la cadena introducida.
Observaciones: el usuario debe estar autenticado en el sistema. Esta HU responde a la funcionalidad número 11.

Tabla 35 HU Listar sugerencia

Historia de usuario	
Nombre: Listar sugerencias.	**Número:** 19
Usuario: todos	**Iteración asignada:** 4
Prioridad en el negocio: alta	**Puntos estimados:** 0.5
Riesgo en el desarrollo: alto	
Descripción: el sistema muestra al usuario todas las sugerencias existentes de un tema introducido por el usuario.	
Observaciones: esta HU responde a la funcionalidad número 4.2 que está contenida en gestionar sugerencia con número 4. Se necesita estar autenticado para poder llevar a cabo esta HU.	

Tabla 36 HU Eliminar sugerencia

Historia de usuario	
Nombre: Eliminar sugerencia.	**Número:** 20
Usuario: todos	**Iteración asignada:** 4
Prioridad en el negocio: alta	**Puntos estimados:** 0.1
Riesgo en el desarrollo: alto	
Descripción: el usuario selecciona la sugerencia a eliminar y elige la opción eliminar.	
Observaciones: esta HU responde a la funcionalidad número 4.3 que está contenida en gestionar sugerencia con número 4. Se necesita estar autenticado para poder llevar a cabo esta HU.	

Tabla 37 HU Adicionar proveedor

Historia de usuario	
Nombre: Adicionar proveedor.	**Número:** 21
Usuario: administrador	**Iteración asignada:** 4
Prioridad en el negocio: alta	**Puntos estimados:** 0.5
Riesgo en el desarrollo: alto	
Descripción: el sistema muestra al administrador una lista de los proveedores existentes, le posibilita crear una nueva entrada con los datos del proveedor a adicionar (nombre, fuente, tipo y Chtml) seguidamente opta por la opción crear. Una vez adicionado se mostrará como un resultado más en la lista de proveedor.	
Observaciones: el usuario debe estar autenticado. Esta HU responde a la funcionalidad 12.1 que pertenece al gestionar mensaje con número 12.	

Tabla 38 HU Modificar proveedor

Historia de usuario	
Nombre: Modificar proveedor.	**Número:** 22
Usuario: administrador	**Iteración asignada:** 4
Prioridad en el negocio: alta	**Puntos estimados:** 0.2
Riesgo en el desarrollo: alto	
Descripción: el sistema muestra al administrador la lista de los proveedores existentes y la opción editar proveedor, mostrándole así los atributos del mismo (nombre, fuente, tipo y Chtml) seguidamente opta por la opción modificar y se visualizaran los datos modificados en la lista de proveedores existentes.	
Observaciones: el usuario debe estar autenticado. Esta HU responde a la funcionalidad 12.2 que pertenece al gestionar mensaje con número 12.	

Tabla 39 HU Listar proveedor

Historia de usuario	
Nombre: Listar proveedor.	**Número:** 23
Usuario: administrador	**Iteración asignada:** 4
Prioridad en el negocio: alta	**Puntos estimados:** 0.2
Riesgo en el Desarrollo: alto	
Descripción: el sistema muestra al administrador la lista de los proveedores existentes.	
Observaciones: el usuario debe estar autenticado. Esta HU responde a la funcionalidad 12.3 que pertenece al gestionar mensaje con número 12.	

Tabla 40 HU Eliminar proveedor

Historia de usuario	
Nombre: Eliminar proveedor.	**Número:** 24
Usuario: administrador	**Iteración asignada:** 4
Prioridad en el negocio: alta	**Puntos estimados:** 0.2
Riesgo en el desarrollo: alto	
Descripción: el sistema muestra al administrador una lista de los proveedores existentes, le posibilita editar los datos de un proveedor especificado, opta por la opción eliminar proveedor y seguidamente el sistema muestra la lista de los restantes proveedores.	
Observaciones: el usuario debe estar autenticado. Esta HU responde a la funcionalidad 12.4 que pertenece al gestionar mensaje con número 12.	

Anexo # 3: Tareas de ingeniería

TI de la primera iteración

Tabla 41 TI Registrar usuario

Tarea de ingeniería	
No. de la tarea: 1	**No. de la HU:** 1
Nombre de la tarea: Registrar usuario.	
Tipo de tarea: desarrollo	**Puntos estimados:** 3
Programador responsable: Rosalia Martínez García.	
Descripción: Implementar la funcionalidad registrar usuario, asegurando la seguridad del sistema y la veracidad de los datos usuario.	

Tabla 42 TI Modificar usuario

Tarea de ingeniería	
No. de la tarea: 3	**No. de la HU:** 3
Nombre de la tarea: Modificar usuario.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5
Programador responsable: Rosalia Martínez García.	
Descripción: Implementar la funcionalidad modificar usuario empleando la creación de un *crud* (Gestionar de Symfony 2) de la entidad *user*.	

Tabla 43 TI Listar usuario

Tarea de ingeniería	
No. de la tarea: 4	**No. de la HU:** 4
Nombre de la tarea: Listar usuarios.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5

Programador responsable: Rosalia Martínez García.
Descripción: Implementar la funcionalidad listar usuario empleando la creación de un *crud* (Gestionar de Symfony 2) de la entidad *user*.

Tabla 44 TI Eliminar usuario

Tarea de ingeniería	
No. de la tarea: 5	**No. de la HU:** 5
Nombre de la tarea: Eliminar usuario.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5
Programador responsable: Rosalia Martínez García.	
Descripción: Implementar la funcionalidad eliminar usuario empleando la creación de un *crud* (Gestionar de Symfony 2) de la entidad *user*.	

TI de la segunda iteración

Tabla 45 TI Actualizar charla

Tarea de ingeniería	
No. de la tarea: 6	**No. de la HU:** 6
Nombre de la tarea: Actualizar charla.	
Tipo de tarea: desarrollo	**Puntos estimados:** 1
Programador responsable: Rosalia Martínez García.	
Descripción: Implementar la funcionalidad actualizar charla haciendo uso del servidor nodejs y el *framework* de javascript jQuery para visualizar los mensajes dinámicamente, así como usuarios en la misma y restantes notificaciones referentes a la charla.	

Tabla 46 TI Listar charlas activas

Tarea de ingeniería

No. de la tarea: 7	**No. de la HU:** 7
Nombre de la tarea: Listar charlas activas.	
Tipo de tarea: desarrollo	**Puntos estimados:** 1
Programador responsable: Rosalia Martínez García.	
Descripción: Implementar la funcionalidad listar charlas activas haciendo uso de Symfony 2 y la tecnología Ajax para visualizar dinámicamente las mismas. Así como el servidor nodejs para mostrar en tiempo real las nuevas charlas activas.	

Tabla 47 TI Listar charlas almacenadas

Tarea de ingeniería	
No. de la tarea: 8	**No. de la HU:** 8
Nombre de la tarea: Listar charlas almacenadas.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.3
Programador responsable: Rosalia Martínez García.	
Descripción: Implementar la funcionalidad listar charlas almacenadas haciendo uso de Symfony 2 y la tecnología Ajax para visualizar dinámicamente las mismas.	

Tabla 48 TI Eliminar charla

Tarea de ingeniería	
No. de la tarea: 10	**No. de la HU:** 10
Nombre de la tarea: Eliminar charla.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.1
Programador responsable: Rosalia Martínez García.	
Descripción: Implementar la funcionalidad eliminar charla haciendo uso de Symfony 2 y la generación de un *crud* de la entidad *chat*.	

TI de la tercera iteración

Tabla 49 TI Listar mensaje

Tarea de ingeniería	
No. de la tarea: 12	**No. de la HU:** 12
Nombre de la tarea: Listar mensaje.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5
Programador responsable: Leandro Campos Rojas.	
Descripción: Implementar la funcionalidad listar mensaje empleando la creación de un *crud* (Gestionar de Symfony 2) de la entidad *message*.	

Tabla 50 TI Eliminar mensaje

Tarea de ingeniería	
No. de la tarea: 13	**No. de la HU:** 13
Nombre de la tarea: Eliminar mensaje.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5
Programador responsable: Leandro Campos Rojas.	
Descripción: Implementar la funcionalidad eliminar mensaje empleando la creación de un *crud* (Gestionar de Symfony 2) de la entidad *message*.	

Tabla 51 TI Listar mensajes del historial

Tarea de ingeniería	
No. de la tarea: 14	**No. de la HU:** 14
Nombre de la tarea: Listar mensajes del historial.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5

Programador responsable: Leandro Campos Rojas.
Descripción: implementar la funcionalidad listar mensajes del historial haciendo uso de la tecnología Ajax y los beneficios del motor de plantillas *Twig* incluido en Symfony 2.

Tabla 52 TI Contenido en texto plano

Tarea de ingeniería	
No. de la tarea: 15	**No. de la HU:** 15
Nombre de la tarea: Contenido en texto plano.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.2
Programador responsable: Leandro Campos Rojas.	
Descripción: implementar la funcionalidad Contenido en texto plano, filtrando los mensajes para asegurar la integridad de la información almacenada en la base de datos y que el mensaje sea almacenado y visualizado como texto plano independientemente de su contenido.	

Tabla 53 TI Buscar charla

Tarea de ingeniería	
No. de la tarea: 16	**No. de la HU:** 16
Nombre de la tarea: Buscar charla.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.1
Programador responsable: Leandro Campos Rojas	
Descripción: implementar la funcionalidad buscar charla haciendo uso de la tecnología Ajax y el motor de plantillas *Twig* incluido en Symfony 2 para listar los resultados.	

Tabla 54 TI Buscar usuario

Tarea de ingeniería	
No. de la tarea: 17	**No. de la HU:** 17
Nombre de la tarea: Buscar usuario.	

Tipo de tarea: desarrollo	**Puntos estimados:** 0.1
Programador responsable: Leandro Campos Rojas	
Descripción: implementar la funcionalidad buscar usuario haciendo uso de la tecnología Ajax y el motor de plantillas *Twig* incluido en Symfony 2 para listar los resultados.	

TI de la cuarta iteración

Tabla 55 TI Adicionar sugerencia

Tarea de ingeniería	
No. de la tarea: 18	**No. de la HU:** 18
Nombre de la tarea: Adicionar sugerencias.	
Tipo de tarea: desarrollo	**Puntos estimados:**
Programador responsable: Leandro Campos Rojas	
Descripción: implementar la funcionalidad adicionar sugerencias haciendo uso de un motor de búsqueda implementado en java, el servidor nodejs para inicializarlo y notificar al cliente cuando existan nuevos resultados, así como la tecnología Ajax para mostrar los mismos dinámicamente.	

Tabla 56 TI Listar sugerencias

Tarea de ingeniería	
No. de la tarea: 19	**No. de la HU:** 19
Nombre de la tarea: Listar sugerencias.	
Tipo de tarea: desarrollo	**Puntos estimados:** 1
Programador responsable: Leandro Campos Rojas.	
Descripción: implementar la funcionalidad listar sugerencias haciendo uso de la tecnología Ajax y el motor de plantillas *Twig* incluido en Symfony 2 para listar los resultados.	

Tabla 57 TI Eliminar sugerencia

Tarea de ingeniería

No. de la tarea: 20	**No. de la HU:** 20
Nombre de la tarea: Eliminar sugerencia.	
Tipo de tarea: desarrollo	**Puntos estimados:** 1
Programador responsable: Leandro Campos Rojas.	
Descripción: implementar la funcionalidad eliminar sugerencia haciendo uso del *framework* de javascript JQuery.	

Tabla 58 TI Modificar proveedor

Tarea de ingeniería	
No. de la tarea: 22	**No. de la HU:** 22
Nombre de la tarea: Modificar proveedor.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5
Programador responsable: Leandro Campos Rojas.	
Descripción: implementar la funcionalidad modificar proveedor haciendo uso de un *crud* de la entidad *provider*.	

Tabla 59 TI Listar proveedor

Tarea de ingeniería	
No. de la tarea: 23	**No. de la HU:** 23
Nombre de la tarea: Listar proveedor.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5
Programador responsable: Leandro Campos Rojas.	
Descripción: implementar la funcionalidad listar proveedor haciendo uso de un *crud* de la entidad provider y de la tecnología Ajax para visualizar los resultados con paginado.	

Tabla 60 TI Eliminar proveedor

Tarea de ingeniería	
No. de la tarea: 24	**No. de la HU:** 24
Nombre de la tarea: Eliminar proveedor.	
Tipo de tarea: desarrollo	**Puntos estimados:** 0.5
Programador responsable: Leandro Campos Rojas.	
Descripción: implementar la funcionalidad eliminar proveedor haciendo uso de un *crud* de la entidad *provider*.	

Anexo # 4 Casos de prueba de aceptación

Tabla 61 Caso de prueba de aceptación Registrar usuario

Caso de prueba de aceptación	
Código: 1	**HU:** 1
Nombre: Registrar usuario.	
Descripción: prueba para la funcionalidad que permite al usuario ingresar los datos al sistema, para registrarse correctamente.	
Condiciones de ejecución: el usuario debe llenar todos los campos, completándolos con sus datos personales.	
Entrada/Pasos de ejecución: el usuario selecciona la opción registrar usuario, el sistema muestra el formulario con los datos que el usuario debe ingresar (usuario, nombre, apellidos, correo electrónico, contraseña, cumpleaños y avatar). Una vez insertada correctamente la información, selecciona el botón confirmar para registrar el usuario.	
Resultado esperado: que el usuario pueda registrarse favorablemente.	
Evaluación de la prueba: satisfactoria.	

Tabla 62 Caso de prueba de aceptación Modificar usuario

Caso de prueba de aceptación	
Código: 3	**HU:** 3
Nombre: Modificar usuario.	
Descripción: prueba para la funcionalidad que permite al usuario modificar los datos de su perfil.	
Condiciones de ejecución: el usuario debe estar autenticado, para poder realizar cambio en sus datos.	
Entrada/Pasos de ejecución: el usuario elige la opción editar cuenta, que contiene los datos a modificar (usuario, nombre, apellidos, correo electrónico, contraseña, cumpleaños y avatar) luego elige actualizar para guardar los cambios o cancelar en caso de no querer modificar sus datos.	
Resultado esperado: que el usuario pueda modificar los datos que desee de su perfil.	
Evaluación de la prueba: satisfactoria.	

Tabla 63 Caso de prueba de aceptación Listar usuarios

Caso de prueba de aceptación	
Código: 4	**HU:** 4
Nombre: Listar usuarios.	
Descripción: prueba para la funcionalidad que permite al usuario ver los demás contactos el línea.	
Condiciones de ejecución: el usuario debe estar autenticado en el sistema. Debe haber usuarios conectados para que se pueda visualizar la lista.	
Entrada/Pasos de ejecución: ninguna.	
Resultado esperado: que el sistema permita ver la lista de usuarios conectados.	
Evaluación de la prueba: satisfactoria.	

Tabla 64 Caso de prueba de aceptación Eliminar usuario

Caso de prueba de aceptación	
Código: 5	**HU:** 5
Nombre: Eliminar usuario.	
Descripción: prueba para la funcionalidad que permite eliminar un usuario.	
Condiciones de ejecución: el usuario debe estar autenticado como administrador para poner llevar a cabo esta funcionalidad.	
Entrada/Pasos de ejecución: el administrador elige la opción ver lista. A continuación se muestra una lista con todos los usuarios registrados en el sistema y selecciona la opción mostrar donde se visualizarán los datos del usuario seleccionado y finalmente se elige la opción eliminar usuario.	
Resultado esperado: que el usuario que se desea eliminar, no se encuentre una vez eliminado, en la lista de usuarios.	
Evaluación de la prueba: satisfactoria.	

Tabla 65 Caso de prueba de aceptación Actualizar charla

Caso de prueba de aceptación	
Código: 6	**HU:** 6

Nombre: Actualizar charla.
Descripción: prueba para la funcionalidad que permite al usuario crear un nuevo mensaje y enviarlo a los usuarios conectados.
Condiciones de ejecución: el usuario debe estar autenticado para poder actualizar una charla.
Entrada/Pasos de ejecución: el usuario escribe un mensaje y lo enviado a todos los usuarios que estén conectados a la conversación, la ventana del *chat* se actualiza automáticamente visualizando el nuevo mensaje.
Resultado esperado: que el usuario pueda enviar el mensaje a los contactos que estén conectados a esa charla y estos puedan visualizar dicho mensaje.
Evaluación de la prueba: satisfactoria.

Tabla 66 Caso de prueba de aceptación Listar charlas activas

Caso de prueba de aceptación	
Código: 7	**HU:** 7
Nombre: Listar charlas activas.	
Descripción: prueba para la funcionalidad que permite al usuario listar las charlas activas.	
Condiciones de ejecución: el usuario debe estar autenticado.	
Entrada/Pasos de ejecución: Ninguna.	
Resultado esperado: que el sistema muestra al usuario todas las charlas que estén activas en ese momento.	
Evaluación de la prueba: satisfactoria.	

Tabla 67 Caso de prueba de aceptación Listar charlas almacenadas

Caso de prueba de aceptación	
Código: 8	**HU:** 8
Nombre: Listar charlas almacenadas.	
Descripción: prueba para la funcionalidad que permite al usuario listar las charlas almacenadas.	

Condiciones de ejecución: el usuario debe estar autenticado como administrador para poner llevar a cabo esta funcionalidad.
Entrada/Pasos de ejecución: ninguna.
Resultado esperado: que el sistema le muestre al administrador la lista de las charlas
Evaluación de la prueba: satisfactoria.

Tabla 68 Caso de prueba de aceptación Eliminar charla

Caso de prueba de aceptación	
Código: 10	**HU:** 10
Nombre: Eliminar charla.	
Descripción: prueba para la funcionalidad que permite al usuario eliminar una charla.	
Condiciones de ejecución: el usuario debe estar autenticado como administrador para poner llevar a cabo esta funcionalidad.	
Entrada/Pasos de ejecución: el sistema muestra al administrador la interfaz que contiene la lista de todas las charlas almacenadas de los usuarios y se elige editar, que mostrará los datos de la charla y se opta por la opción de eliminar charla para eliminar la charla deseada.	
Resultado esperado: que el administrador pueda eliminar la charla seleccionada.	
Evaluación de la prueba: satisfactoria.	

Tabla 69 Caso de prueba de aceptación Listar mensaje

Caso de prueba de aceptación	
Código: 12	**HU:** 12
Nombre: Listar mensaje.	
Descripción: prueba para la funcionalidad que permite al usuario ver la lista de mensajes.	
Condiciones de ejecución: el usuario debe estar autenticado como administrador para poner llevar a cabo esta funcionalidad.	
Entrada/Pasos de ejecución: ninguna.	

Resultado esperado: que el sistema muestre al administrador la lista de todos los mensajes escritos por los usuarios.
Evaluación de la prueba: satisfactoria.

Tabla 70 Caso de prueba de aceptación Eliminar mensaje

Caso de prueba de aceptación	
Código: 13	**HU:** 13
Nombre: Eliminar mensaje.	
Descripción: prueba para la funcionalidad que permite al administrador eliminar un mensaje.	
Condiciones de ejecución: el usuario debe estar autenticado como administrador para poner llevar a cabo esta funcionalidad.	
Entrada/Pasos de ejecución: el sistema muestra al administrador la lista de todos los mensajes escritos por los usuarios y se elige la opción eliminar para eliminar el mensaje deseado.	
Resultado esperado: que el administrador pueda eliminar el mensaje deseado.	
Evaluación de la prueba: satisfactoria.	

Tabla 71 Caso de prueba de aceptación Listar mensajes del historial

Caso de prueba de aceptación	
Código: 14	**HU:** 14
Nombre: Listar mensajes del historial.	
Descripción: prueba para la funcionalidad que permite al usuario ver la lista de mensajes del historial.	
Condiciones de ejecución: el usuario debe estar autenticado.	
Entrada/Pasos de ejecución: se visualizan los mensajes anteriores de una conversación seleccionando la opción ver historial.	
Resultado esperado: que el usuario pueda ver el historial de los mensajes escritos anteriormente.	
Evaluación de la prueba: satisfactoria.	

Tabla 72 Caso de prueba de aceptación Contenido en texto plano

Caso de prueba de aceptación	
Código: 15	**HU:** 15
Nombre: Contenido en texto plano.	
Descripción: prueba para la funcionalidad que permite al usuario escribir Contenido en texto plano.	
Condiciones de ejecución: el usuario debe estar autenticado.	
Entrada/Pasos de ejecución: le permite al usuario escribir mensajes en texto plano y código embebido.	
Resultado esperado: que el usuario pueda escribir un texto.	
Evaluación de la prueba: satisfactoria.	

Tabla 73 Caso de prueba de aceptación Buscar charla

Caso de prueba de aceptación	
Código: 16	**HU:** 16
Nombre: Buscar charla.	
Descripción: prueba para la funcionalidad que permite al usuario buscar una charla.	
Condiciones de ejecución: el usuario debe estar autenticado.	
Entrada/Pasos de ejecución: el usuario escribe un tema o texto, selecciona buscar y se le muestra un listado de las charlas que estén relacionadas con el mismo. El usuario selecciona la opción mis charlas y se le muestra un listado de todas las charlas que ha creado.	
Resultado esperado: que el usuario encuentre la charla deseada.	
Evaluación de la prueba: satisfactoria.	

Tabla 74 Caso de prueba de aceptación Buscar usuario

Caso de prueba de aceptación	
Código: 17	**HU:** 17
Nombre: Buscar usuario.	

Descripción: prueba para la funcionalidad que permite al usuario buscar un contacto.
Condiciones de ejecución: el usuario debe estar autenticado.
Entrada/Pasos de ejecución: el usuario escribe el nombre, el apodo o los apellidos de la persona a buscar, luego selecciona la opción buscar y se le muestra un listado con las coincidencias de la cadena introducida.
Resultado esperado: que el usuario encuentre el contacto deseado.
Evaluación de la prueba: satisfactoria.

Tabla 75 Caso de prueba de aceptación Adicionar sugerencias

Caso de prueba de aceptación	
Código: 18	**HU:** 18
Nombre: Adicionar sugerencias.	
Descripción: prueba para la funcionalidad que permite al usuario adicionar una sugerencia.	
Condiciones de ejecución: el usuario debe estar autenticado.	
Entrada/Pasos de ejecución: el usuario realiza una pregunta y el buscador le muestra un listado de las posibles sugerencias del tema especificado.	
Resultado esperado: que el usuario pueda obtener una sugerencia asociada a la pregunta realizada.	
Evaluación de la prueba: satisfactoria.	

Tabla 76 Caso de prueba de aceptación Listar sugerencias

Caso de prueba de aceptación	
Código: 19	**HU:** 19
Nombre: Listar sugerencias.	
Descripción: prueba para la funcionalidad que permite al usuario visualizar la lista de sugerencia.	
Condiciones de ejecución: el usuario debe estar autenticado.	
Entrada/Pasos de ejecución: ninguna.	

Resultado esperado: que el sistema le muestre al usuario todas las sugerencias existentes de un tema introducido por él.
Evaluación de la prueba: satisfactoria.

Tabla 77 Caso de prueba de aceptación Eliminar sugerencias

Caso de prueba de aceptación	
Código: 20	**HU:** 20
Nombre: Eliminar sugerencias.	
Descripción: prueba para la funcionalidad que permite al usuario eliminar una sugerencia.	
Condiciones de ejecución: el usuario debe estar autenticado.	
Entrada/Pasos de ejecución: el usuario selecciona la sugerencia a eliminar y elige la opción eliminar.	
Resultado esperado: que el usuario pueda eliminar la sugerencia deseada.	
Evaluación de la prueba: satisfactoria.	

Tabla 78 Caso de prueba de aceptación Modificar proveedor

Caso de prueba de aceptación	
Código: 22	**HU:** 22
Nombre: Modificar proveedor.	
Descripción: prueba para la funcionalidad que permite al administrador modificar una sugerencia.	
Condiciones de ejecución: el usuario debe estar autenticado como administrador para poner llevar a cabo esta funcionalidad.	
Entrada/Pasos de ejecución: el sistema muestra al administrador una lista de los proveedores existentes, le posibilita editar los datos de un proveedor especificado por él, mostrándole así los campos que puede modificar (nombre, fuente, tipo y Chtml) seguidamente opta por la opción modificar y se visualizaran los datos modificados en la lista de proveedores existentes.	
Resultado esperado: que el administrador pueda modificar los campos que desee del proveedor seleccionado.	

Evaluación de la prueba: satisfactoria.

Tabla 79 Caso de prueba de aceptación Listar proveedor

Caso de prueba de aceptación	
Código: 23	**HU:** 23
Nombre: Listar proveedor.	
Descripción: prueba para la funcionalidad que permite al usuario ver el listado de los proveedores.	
Condiciones de ejecución: el usuario debe estar autenticado como administrador para poner llevar a cabo esta funcionalidad.	
Entrada/Pasos de ejecución: ninguna.	
Resultado esperado: que el sistema le muestre al administrador una lista de los proveedores existentes.	
Evaluación de la prueba: satisfactoria.	

Tabla 76 Caso de prueba de aceptación Eliminar proveedor

Caso de prueba de aceptación	
Código: 24	**HU:** 24
Nombre: Eliminar proveedor.	
Descripción: prueba para la funcionalidad que permite al administrador eliminar un proveedor.	
Condiciones de ejecución: el usuario debe estar autenticado como administrador para poner llevar a cabo esta funcionalidad.	
Entrada/Pasos de ejecución: el sistema muestra al administrador una lista de los proveedores existentes, le posibilita editar los datos de un proveedor especificado, opta por la opción eliminar proveedor y seguidamente el sistema muestra la lista de los restantes proveedores.	
Resultado esperado: que el usuario pueda eliminar el proveedor que desee.	
Evaluación de la prueba: satisfactoria.	

Anexo # 3: Ranking de lenguajes de programación

Tabla 77 Ranking de lenguajes de programación PYPL (*PopularitY of Programming Language*) (91).

Posición 04 2014	Posición 04 2013	Lenguaje de programación	Compartir en abril 2014	Doce meses tendencias
1	1	Java	26.9%	- 0.5%
2	2	PHP	13.5%	- 1.5%
3	4	Pitón	10.6%	0.6%
4	3	C#	10.5%	- 0.1%
5	5	C++	8.8%	- 0.5%
6	6	C	8.1%	0.2%
7	7	Javascript	8.0%	0%
8	8	Objetive-C	6.4%	1.6%
9	9	Visual Basic	3.1%	1.6%
10	10	Rubi	2.5%	0%