Trabajo de grado
Sistema de conferencia de Chat con referencias

Universidad Nacional de La Plata
Facultad de Informática

Alumno: Ricardo Tesoriero
Legajo: 2562/3

Directora: Alicia Díaz
Codirector: Alejandro Fernández

Índice general

Índice de figuras

Agradecimientos

En esta oportunidad me gustaría agradecer a toda la gente que hizo posible que llega a esta instancia en la carrera.

Muchos años han pasado desde que comencé, y muchas personas han dejado su granito de arena para que pueda continuar mis estudios.

Hay gente que ya no está, a ellos mis recuerdos y mis más profundos sentimientos.

A los que están, en primer lugar quiero agradecer muy especialmente a toda mi familia y amigos, que me han apoyado incondicionalmente durante todos estos años.

No me puedo olvidar de la paciencia y la inspiración que me han brindado Alicia Díaz y Alejandro Fernández, así que a ellos, muchas gracias.

Además, no dejo de acordarme de toda la gente del LIFIA que me ha dado su apoyo en todo momento.

Por último, a los profesores de la Universidad Nacional de La Plata que me han formado como profesional.

Capítulo 1

Introducción

Los sistemas de Chat son uno de los medios de comunicación más utilizados y populares en la actualidad.

El mercado está inundado de productos para soportar el intercambio de texto entre ordenadores. La mayoría de ellos contemplan características similares que poco han evolucionado conceptualmente desde su concepción.

Uno de los objetivos de este trabajo es introducir algunos conceptos que permitan mejorar el desempeño de este tipo de herramientas.

Un concepto importante que trata de incorporarse es el de visualización compartida de documentos. Se intenta aplicar el mismo principio utilizado en los mensajes estándar[1] a los documentos [2], de esta forma es posible el intercambio de documentos para obtener una visualización compartida por los participantes de un sistema de intercambio de texto.

Otro concepto a introducir es el de referencia. Una referencia representa una marca asociada a una región de un documento, la cual está ligada a un mensaje estándar.

El objetivo de estas mejoras es permitir una mejor y más rica expresión en la las comunicaciones basadas en texto.

1.1. Estructura del trabajo de grado

El trabajo está documentado en ocho capítulos:

El primer capítulo introduce brevemente la intención del trabajo. Además expone la estructura del trabajo para obtener una visión general del mismo.

[1] Llamamos mensajes estándar a una unidad de información, usualmente textual, concebida de forma espontánea antes de ser enviada

[2] Un documento es una unidad de información más compleja y eleborada que un mensaje. Usualmente se conciben con mayor anterioridad que los mensajes

El segundo capítulo plantea las razones por las cuales se llevó a cabo el trabajo. El capítulo comienza con las características de los sistemas de intercambio de texto. Luego se explican las características del intercambio de documentos para visualización. También describe la referencia de los mensajes en torno a un documento. La explicación de estas características se ejemplifica mediante situaciones reales. La sección concluye con tres requerimientos generales que debe satisfacer el sistema para soportar los conceptos expuestos.

El tercer capítulo deriva del anterior un conjunto de requerimientos más detallados del sistema que se pretende obtener. La derivación se lleva a cabo refinando de los requerimientos generales expuestos en el capítulo anterior. La conclusión de este capítulo es una lista de requerimientos puntuales que debería cumplir la aplicación.

El cuarto capítulo pretende describir y evaluar un conjunto de aplicaciones relacionadas al intercambio de texto con referencias. La descripción de las aplicaciones se divide en tres conjuntos:

1. Los Chats
2. Los sistemas de escritorios compartidos
3. Los sistemas de Chats con referencias

Concluye con una evaluación de las aplicaciones respecto a los requerimientos expuestos en el capítulo anterior.

El quinto capítulo se explica conceptualmente una aplicación que cumple con los requerimientos del Capítulo 3. Se hace una descripción funcional del sistema y se lo descompone en sub-funciones.

El sexto capítulo esboza un diseño de la aplicación. Toma los puntos más destacados de la aplicación proponiendo varias soluciones de diseño para cada caso en particular. A partir de este conjunto de soluciones se adopta la que mejor soluciona el problema y se justifica la elección.

El séptimo capítulo plantea una implementación para cada diseño propuesto. La implementación trata la estructura e interacción de los principales componentes de software de la aplicación.

El octavo y último capítulo establece las conclusiones del trabajo. Comienza con una comparación del software obtenido con el estudiado en el Capítulo 4. Continúa con un resumen en pocas palabras del aporte del sistema a los sistemas de comunicaciones tradicionales. Finalmente presenta los puntos más importantes para continuar con un trabajo futuro.

Capítulo 2

Motivación

Este capítulo explica el móvil por el cual los sistemas de intercambio de texto son interesantes desde el punto de vista de la comunicación entre usuarios. Explora las características de los sistemas de intercambio de texto, documentos y referencias de mensajes a regiones de los documentos (referencias) mediante ejemplos. El capítulo concluye con los requerimientos generales que debe satisfacer el sistema para soportar los conceptos expuestos.

En la actualidad existen muchas formas de comunicación basadas en sistemas informáticos. Las comunicaciones de propósito general son llevadas a cabo con distintos medios. Entre los medios más populares se encuentran, voz, video y texto. Las comunicaciones de voz y video se difunden cada vez más a medida que transcurre el tiempo. Sin embargo, el intercambio de texto tiene algunas ventajas:

- No es necesario hardware adicional (aunque actualmente el costo del hardware específico tiende a bajar y es muy accesible).
- Menor uso del ancho de banda de la red informática. Las comunicaciones de voz y de video utilizan un mayor ancho de banda que el intercambio de texto para poder comunicar la misma información. Como consecuencia se consigue optimizar el uso del ancho de banda y economizar el intercambio de información.
- Fácil revisión. La revisión de comunicaciones realizadas a partir del intercambio textual es mucho más simple y rápida que la revisión de comunicaciones realizadas por medio de voz y video. Para revisar algún fragmento de la comunicación basada en texto, solo se requiere de una caracterización relacionada al contenido de dicho fragmento y realizar una búsqueda sobre la conversación para encontrarlo. El caso de

una conversación basada en voz y video requiere de una caracterización temporal para encontrar el fragmento a revisar. La caracterización temporal no tiene relación directa con el contenido de la información que se desea revisar. Por lo tanto, dicha caracterización no suele ser fácil de encontrar, si no se utiliza un método de señalización para la relación de contenido-tiempo. En la mayoría de los casos, la revisión de una conversación textual requiere de un "vistazo" para encontrar el segmento de información requerida (con la ayuda de potenciales buscadores). En cambio, la revisión de voz y video requiere tiempos más extensos de revisión. Encontrar un segmento de información en voz y video requiere observar durante cierto tiempo el medio. Es la única forma de encontrar exactamente el segmento de información buscado.

- "Seguimiento" simultáneo de conversaciones. La utilización de un sistema basado en texto permite al usuario poder participar en más de una conversación a la vez, sin mayores complicaciones. En cambio, la utilización de un sistema basado en voz y video no permite al usuario llevar ambas conversaciones al mismo tiempo. Esto se debe al nivel de "atención" que se necesita en el caso de utilizar este tipo de medio.

- Contextualizar Latecoming[1]. Contextualizar usuarios que "llegan tarde" a una conversación es más simple para los usuarios que utilizan medios textuales que para los que utilizan voz y video. Esto se debe a los puntos tratados anteriormente: *fácil revisión* y *"seguimiento" simultáneo* de conversaciones. Contextualizar para Latecoming se puede ver como un "mini caso" de revisión. Cuando un usuario se "une" a una conversación; existen ciertos temas que pudieron ser tratados anteriormente y por lo tanto se necesita realizar una revisión rápida de tales temas para que el usuario que "llega tarde" pueda contextualizar la conversación actual.

Así el intercambio de texto se convierte en una herramienta potente para comunicar ideas entre personas que están geográficamente separadas.

2.1. Características de los sistemas de intercambio de texto

[1]Llegada tarde en castellano

2.1.1. Canal

Los sistemas de intercambio de texto comunican a los usuarios mediante canales. Los canales son el medio por el cual la información fluye del emisor al receptor. Éstos pueden establecer la comunicación entre más de dos usuarios. La información enviada a un canal es recibida por todos los usuarios que lo comparten.

2.1.2. Densidad de información

Una característica de estos sistemas es la densidad de información a transmitir. La densidad depende de la unidad de transmisión de datos que se utilice para intercambiar la información. Básicamente se define cuanta información envía el emisor al receptor y cómo realiza el envío. Las unidades de información más utilizadas son el carácter (fina) y el mensaje (gruesa).

Carácter El carácter es la mínima unidad de transmisión de datos. El envío de información del emisor al receptor se efectúa por cada caráter escrito. Por lo tanto cada tecla presionada por el emisor será recibida por el receptor, incluso los caracteres de control como *suprimir carácter* y *borrar anterior*.

Mensaje Un mensaje puede comprender un conjunto de caracteres que representan un párrafo o conjunto de ellos. Así los usuarios para enviar un mensaje se valen de un control extra que indica el envío del mismo. Usualmente es un botón o combinación de teclas que representan la acción de enviar el mensaje.

2.1.3. Clasificación en función del tiempo

La clasificación de los sistemas Groupware en función del tiempo se basan en la matriz espacio-tiempo[4] que los divide en dos grandes grupos:

Asincrónicos Los sistemas asincrónicos son aquellos en los que la comunicación de la información no se establece en tiempo real. Ejemplos de sistemas de intercambio de texto asincrónicos son: el correo electrónico y los artículos de noticias.

Sincrónicos En los sistemas sincrónicos, la información es transmitida y recibida en tiempo real. Ejemplos de sistemas sincrónicos de intercambio de texto son el Chat o la conferencia de Chat.

2.1.4. Tipos de sistemas sincrónicos de intercambio de texto

Los tipos de sistemas sincrónicos de intercambio de texto más conocidos son el Chat y la conferencia de Chat.

La diferencia entre Chat y conferencia de Chat esta dada por la cantidad de participantes por canal; un Chat tiene exactamente dos usuarios por canal mientras la conferencia de Chat puede tener varios participantes por canal.

Conceptualmente un Chat se puede ver como una vía de comunicación persona a persona; en cambio la conferencia de Chat se puede ver como una sala a la cual acuden los participantes para comunicarse.

Un sistema del tipo Chat puede ser simulado mediante un sistema de conferencia de Chat en el cual participan como máximo dos personas. Un canal con estas características se lo denomina canal privado.

El objetivo de la próxima sección es encontrar las limitaciones del uso de conferencias de Chat para luego, plantear soluciones a estos problemas.

2.2. Conversaciones en torno a documentos

En general, los Chats restringen el tipo de información a transmitir a texto únicamente, acotando la potencia de expresión de este medio de comunicación.

Whittaker[1] identifica dos tipos de contribuciones que tienen lugar en una comunicación mediada por computadoras: artefactos y prosa.

Prosa La prosa es la expresión de una observación en forma textual.

Artefacto Los artefactos(o foco) son el centro de la conversación, del cual se desprenden las observaciones.

Usualmente una conversación gira en torno a elementos que van más a allá de texto simple. Por ejemplo, una conversación puede involucrar documentos como artefactos o foco.

La imposibilidad de tratar directamente este tipo de elementos trae como consecuencia la necesidad de describir los elementos de manera textual. El trabajo de transmitir la información del documento en algunos casos es trivial mientras que en otros puede llevar gran cantidad de tiempo o puede llevar a confusión en cuanto a la información a transmitir. Para explicar de manera más concreta la situación recurriremos a un ejemplo:

Sean dos ingenieros que desean discutir sobre la resistencia de un determinado material. Para ello se valen de varias radiografías del mismo.

Es probable que las radiografías no se encuentren en ambas estaciones de trabajo. Sería natural asumir que las radiografías se encuentran en la máquina de ingeniero que realizó el estudio. Así, la radiografía debería ser transportada a la máquina del otro ingeniero para poder comenzar la conversación.

Actualmente algunos sistemas Chat soportan el transporte de archivos entre usuarios. De esta forma, se pueden transportar las radiografías de la máquina del ingeniero que realizó el estudio al otro interesado. Sin embargo, al menos surgen dos problemas:

- ¿Que sucedería si el destinatario de las radiografías no posee la aplicación para visualizarlas o la aplicación es incompatible con el sistema operativo del ordenador?

- No es una situación excepcional manejar más de un documento al mismo tiempo en una conversación. Incluso es muy probable que los documentos no sean del mismo formato ya que el formato usualmente depende del aspecto a discutir. Como ejemplo podemos mencionar que el informe de la radiografía tiene un formato diferente a la radiografía en sí. Esto lleva a la interacción con diferentes aplicaciones(o instancias de las mismas con diferente contenido) que pueden descontextualizar la conversación fácilmente.

Por lo tanto, proponemos como requerimiento:

Una herramienta que posibilite el manejo de documentos sobre los cuales se pueda llevar a cabo una conversación de manera ordenada; minimizando los efectos de la descontextualización por el uso de múltiples documentos.

2.3. Conversaciones con referencias a documentos

Una situación común en las conversaciones entorno a documentos es hacer referencia a fragmentos de los documentos sobre los cuales se desarrolla la conversación. Los sistemas Chat tradicionales no contemplan la posibilidad de realizar referencias explícitas; sino por medio de texto. Para que la situación sea más clara, tomando el ejemplo planteado en la Sección 2.2, platearemos las siguientes situaciones:

1. Si dicha discusión debe llevarse a cabo mediante un sistema de Chat tradicional deberíamos asegurarnos que ambos ingenieros están visualizando el mismo conjunto de radiografías. Esto podría no ser trivial si existen varias versiones de la misma radiografía.

2. Para hacer un comentario sobre una sección de alguna radiografía es necesario describir exactamente el documento y la sección a la cual se hace referencia. Si el interés de la conversación se centra en diferentes secciones, la descripción de las mismas podría ser de una longitud comparable con el tamaño del comentario o incluso mayor. Esta situación aumenta considerablemente la probabilidad de incurrir en confusiones o ambigüedades en la transmisión de la idea que se necesita expresar.

Las situaciones descriptas anteriormente se aplican a una conversación llevada a cabo entre dos participantes. Como uno de los objetivos de transmitir una idea es que todos los participantes de una conversación tengan una interpretación similar de la idea a discutir; la complejidad del manejo de información cuando se celebra una conversación entre más participantes aumenta substancialmente. El factor determinante es que tanto la interpretación de la idea como de la referencia dependen de cada participante en particular, pudiendo originar diferentes interpretaciones.

Como conclusión del análisis del ejemplo podemos describir dos potenciales problemas:

- Sobrecarga de información y mayor probabilidad de error en la transferencia de la información debido a la descripción de la referencia.
- Introducción de ruido en el concepto a expresar. Como consecuencia de la mezcla del concepto a expresar con la descripción de la referencia.

A consecuencia de los puntos anteriormente mencionados la conversación pierde eficiencia y fluidez.

Por lo tanto, proponemos como requerimiento:

La posibilidad de brindar extensiones a las conferencias de Chat tradicionales que soporten:

1. Intercambio de documentos.
2. Presentación de referencias a fragmentos de contenido de los documentos.

2.4. Conclusión

Como conclusión de la definición del objetivo del trabajo se persigue:

- La obtención de conversaciones más claras y concisas.

- La disminución de malas interpretaciones de conceptos y de errores en la interpretación de los mismos.

Para poder cumplir con estos objetivos, se presentarán un conjunto de requerimientos generales que la herramienta debe contemplar:

1. Intercambio de mensajes.
2. Intercambio de documentos.
3. Intercambio de referencias.

Capítulo 3

Análisis detallado de requerimientos

La intención de este capítulo es derivar de la Sección 2.4 un conjunto de requerimientos más detallados del sistema que se pretende obtener.

3.1. Intercambio de mensajes

El mensaje es la unidad mínima de información que se intercambia entre los usuarios de la aplicación.

La estructura de un mensaje es un conjunto de palabras que forman una frase. Por lo tanto, un usuario escribe una frase y luego es enviada por el canal de comunicaciones.

La confección de la frase tiene alcance local únicamente, es decir que solo el autor tiene acceso al mensaje antes de ser enviado.

3.1.1. Formato de mensajes

Los mensajes deben soportar contenido en formato de texto enriquecido o RTF[5]. Es importante el soporte de este formato ya que el nivel de expresión que se le intenta atribuir al mensaje debe ser alto.

Como los atributos que RTF maneja son abundantes, se restringen al siguiente conjunto:

- Negrita.
- Cursiva.
- Fuente.

- Tamaño.
- Color.

3.1.2. Tipo de Chat

Una característica importante del sistema es el modo de distribución de los mensajes a utilizar. La distribución de mensajes define cómo fluirá la conversación entre los distintos participantes.

Como mencionamos anteriormente, los modos de distribución más difundidos son:

Chat tradicional La comunicación se establece entre pares de usuarios (ver Figura 3.1). Por lo tanto el receptor del mensajes es solamente un usuario. Usualmente, para identificar al receptor del mensaje se utiliza un identificador de usuario.

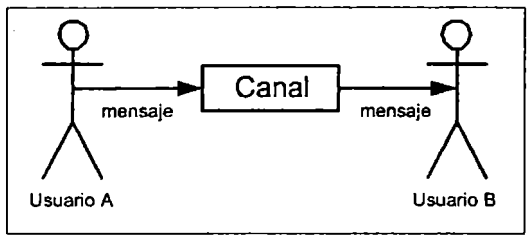


Figura 3.1: Chats tradicionales

Conferencia de Chat La conferencia de Chat (también conocida como Groupchat) establece una comunicación entre varios usuarios simultáneamente (ver Figura 3.2). Para llevar a cabo la comunicación se utiliza un canal de comunicación único para el grupo de usuarios que pretendan tomar parte de la conversación. Por lo tanto, en este caso no existe un único receptor para el mensaje, sino que es un grupo de usuarios el receptor del mismo. Usualmente, para identificar el receptor del mensaje se utiliza al canal de comunicación que comparten los usuarios, al que se lo denomina sala (también conocida como Room).

El modo de distribución de mensajes requerido debería ser el de conferencia de Chat porque es el más general. El esquema de Chat tradicional se puede implementar con una sala por cada par de usuarios.

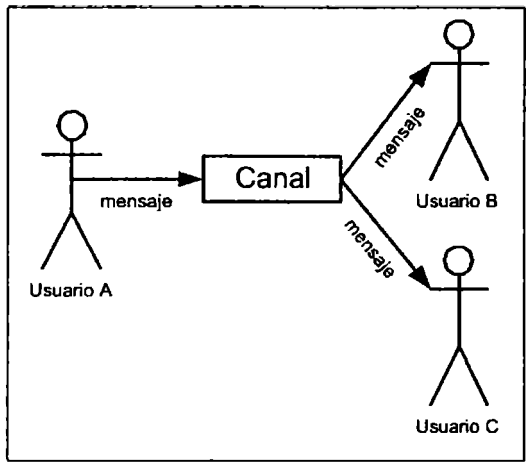


Figura 3.2: Conferencia de Chat

3.1.3. Características del intercambio de mensajes

1. Flexibilidad de acceso: El servicio de intercambio de mensajes debe ser lo suficientemente flexible para que se puedan utilizar servidores de acceso público y gratuito en Internet. Esta característica permite la reducción de costo y mantenimiento del servicio.

2. Flexibilidad de extensión: Otra característica deseable es la capacidad de ampliar el sistema de intercambio con alguna característica adicional. Por ejemplo, el soporte para el intercambio de referencias a documentos.

   La ampliación del sistema, dentro de los marcos generales de adaptación, no debería impactar en el servidor de mensajes para mantener la política anterior. Por ejemplo, añadir un nuevo formato de documento.

3. Flexibilidad de ejecución: El sistema debe correr en diferentes plataformas de sistemas operativos. Por ejemplo, Linux y Windows.

3.2. Intercambio de documentos

Esta sección trata los aspectos más importantes en relación al intercambio de documentos para visualización. Básicamente trata el modo de distribución, manipulación y formato de documentos.

3.2.1. Distribución de documentos

El sistema debe ser capaz de "compartir" documentos para su visualización. Para ello debe articular los medios necesarios para que un documento, al que solo puede acceder un participante, pueda ser accesible al resto. Además, el documento debe poder ser utilizado como referencia por los mensajes.

3.2.2. Manipulación de documentos

La manipulación de los documentos debe ser homogénea, el formato de los mismos debe ser transparente para el usuario. Por ejemplo, el modo en que un mensaje hace referencia a un documento en formato gráfico, debe ser muy similar al de un documento de texto. La consecuencia de esta forma de manipulación de documentos por el cliente minimiza la pérdida de contexto de la conversación.

3.2.3. Formato de documentos

Los medios que los documentos contemplarían serían:

1. Texto
2. Gráfico
3. Mixto (texto y gráfico "mezclados")

La voz y el video quedan excluidos como medios válidos de manipulación por la herramienta.

Como requerimiento de la herramienta restringimos el soporte de medios o formatos a los siguientes:

1. ANSI Plain Text[6] (TXT).
2. Graphics Interchange Format[7] (GIF).
3. Joint Photographic Experts Group [8] (JPEG, JPG).
4. Portable Network Graphic[9] (PNG).
5. Rich Text Format[5] (RTF).
6. Microsoft Word format[10] (DOC).
7. Microsoft Excel format[11] (XLS).

8. Microsoft PowerPoint format[12] (PPT).

9. Referencias a páginas Web [13] formato HTML Versión 3.2 o inferior (sin soporte Javascript) (HTML, HTM).

En cuanto a la flexibilidad, el sistema debería ser capaz de soportar extensiones en cuanto a los formatos de archivos, dentro de los medios mencionados anteriormente. Por lo tanto el mantenimiento en este sentido debe ser de bajo impacto en la arquitectura del sistema.

3.3. Intercambio de referencias

Las referencias son el elemento que "une" un mensaje con una sección o región perteneciente a un documento. Utilizando el ejemplo del capítulo anterior (análisis de radiografías), podemos ejemplificar la situación de la siguiente manera:

Un mensaje "Deberíamos revisar esto" asociado a una "marca" sobre una parte del material revela que dicha parte debe ser revisada.

En el caso de querer expresar el mismo concepto en un Chat tradicional, la descripción de la sección de la pieza sería considerablemente mayor que la idea en si.

Las secciones de un documento pueden presentarse en diferentes medios: gráficos, textuales o mixtos.

3.3.1. Medios de representación textuales

Las referencias a secciones en los medios textuales deberían ser representadas por medio de una selección de texto. como se indica en Figura 3.3.

La Figura 3.3 está mostrando que las palabras: **reactivos M2487 y 2369 en proporción 50-50** representan una referencia.

Si dicha referencia está asociada a un mensaje "Tener en cuenta", indicaría que los reactivos son importantes.

3.3.2. Medios de representación gráficos

Las referencias de secciones gráficas deberían ser representadas mediante áreas geométricas, identificando la sección deseada como se muestra en la Figura 3.4.

Como ejemplo; si a la marca de la Figura 3.4 está asociada con un mensaje "Frágil", está mostrando que la sección marcada del material es frágil.

Informe de resistencia de materiales (pieza 46/05)

Este informe forma parte de la radiografía 3001/05 que fué tomada el día 12 de enero de 2005 a las 15:45.
Los ingenieros Juan Acosta matriculado bajo el legajo 1547/79 y Felipe Morales matriculado bajo el legajo 978/62 son los actuantes en este estudio.

Materiales:

El estudio contempla la imagen de perfil del material ferroso (número de pieza 46/05) con los reactivos M2487 y Z369 en proporción 50-50.

Conclusión:
La imagen muestra una fatiga severa en la sección central del material.

Figura 3.3: Marca de texto

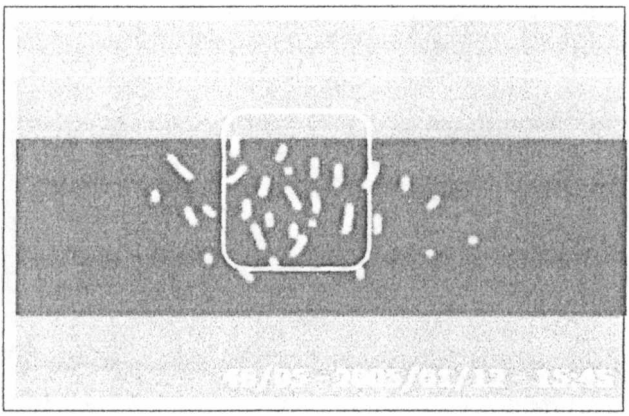


Figura 3.4: Marcas de imagen

3.3.3. Medios de representación mixtos

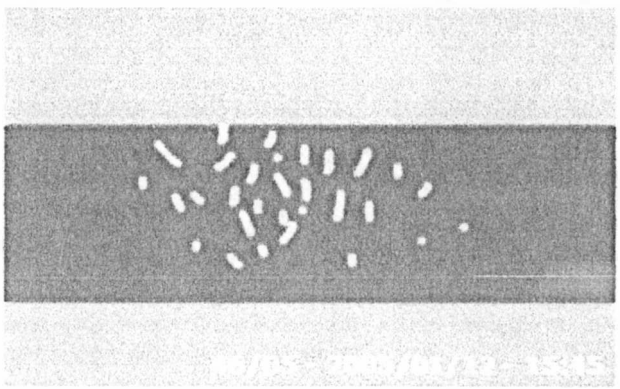
En caso de mezclas de medios, se opta por la forma de marca textual, que engloba a la imagen como parte integrante del texto. Un ejemplo se muestra en la Figura 3.5.

Informe de resistencia de materiales (pieza 46/05)

Este informe forma parte de la radiografía 3001/05 que fué tomada el día 12 de enero de 2005 a las 15:45.
Los ingenieros Juan Acosta matriculado bajo el legajo 1547/79 y Felipe Morales matriculado bajo el legajo 978/62 son los actuantes en este estudio.

Materiales:

El estudio contempla la imagen de perfil del material ferroso (número de pieza 46/05) con los reactivos M2487 y Z369 en proporción 50-50.



Conclusión:
La imagen muestra una fatiga severa en la sección central del material.

Figura 3.5: Marca mixta

3.3.4. Flexibilidad

Una de las características que debería tenerse en cuenta es la relacionada con el bajo impacto en la arquitectura del sistema de la funcionalidad de marcado de secciones. Por ejemplo, modificaciones como el cambio de la forma del área para referenciar una sección sobre un medio gráfico no debe influir en la arquitectura del sistema.

3.4. Listado de requerimientos

A modo de resumen de este capítulo se realizará un listado con los requerimientos más salientes de la herramienta.

Este listado tiene por objetivos:

1. Evaluar el trabajo relacionado.
2. Guiar el desarrollo de la herramienta.

Requerimientos:

1. Soporte de distribución de mensajes del tipo conferencia de Chat.
2. Contenido de mensaje en RTF (con las limitaciones mencionadas en la Sección 3.1.1).
3. Flexibilidad de acceso (posibilidad de utilizar servidores públicos).
4. Flexibilidad de ejecución (plataformas Windows y Linux).
5. Distribución de documentos (posibilidad de compartir documentos).
6. Manipulación homogénea de documentos (manipular los diferentes documentos sin descontextualizar la conversación).
7. Manejo de los siguientes formatos:
   a) ANSI Plain Text[6] (TXT).
   b) Graphics Interchange Format[7] (GIF).
   c) Joint Photographic Experts Group [8] (JPEG, JPG).
   d) Portable Network Graphic[9] (PNG).
   e) Rich Text Format[5] (RTF).
   f) Microsoft Word format[10] (DOC).
   g) Microsoft Excel format[11] (XLS).
   h) Microsoft PowerPoint format[12] (PPT).
   i) Referencias a páginas Web [13] formato HTML Versión 3.2 o inferior (sin soporte Javascript) (HTML, HTM).
8. Soporte de referencias
9. Adaptabilidad de las referencias al medio sobre el cual se necesita realizar la referencia.

Capítulo 4

Trabajo relacionado

Esta sección tiene como principales objetivos:

- Explorar el estado del arte en el campo de las aplicaciones relacionadas con el análisis anterior.
- Evaluar las soluciones existentes con respecto a los requerimientos presentados en la Sección 3.4.

Para relacionar las aplicaciones con los conceptos expresados en la sección anterior, se escogen tres tipos de aplicaciones diferentes, las cuales serán analizadas por medio de aplicaciones concretas en cada caso. Los tipos de aplicaciones a analizar son:

- Sistemas de Chat tradicionales.
- Sistemas para compartir escritorios.
- Sistemas de Chat con referencias.

4.1. Sistemas de Chat tradicionales.

Debido a la popularidad de esta herramienta en el campo del entretenimiento existen numerosas aplicaciones de este tipo en el mercado. Sin embargo proveen una funcionalidad muy similar.

4.1.1. TALK

El TALK[14] era una herramienta que formaba parte del sistema operativo UNIX.

Es una de las aplicaciones más antiguas en el área de los Chat. Por lo tanto, carece de muchas características que las aplicaciones más modernas ya contemplan.

Esta aplicación soporta intercambio de mensajes dentro de un entorno solamente textual (consola), por lo tanto, no contempla texto enriquecido como contenido de mensajes.

La distribución de mensajes es soportada entre dos usuarios exclusivamente, por lo tanto no existe el concepto de conferencia de Chat.

La herramienta no soporta directamente el transporte de documentos. Aunque UNIX dispone de un sistema de archivos compartido (Network File System).

La interpretación de documentos está limitada a las aplicaciones UNIX que los soporten. Los formatos relacionados con Microsoft no son soportados directamente.

Las referencias no son soportadas de ningún modo.

4.1.2. ICQ

ICQ[15] es una de las aplicaciones más populares y difundidas últimamente.

Descripción

La Figura 4.1 muestra la interfaz de usuario de la aplicación. Está dividida en dos ventanas:

Lista de contactos El objetivo de lista contactos[1] es proveer awareness de presencia [2] de los contactos y una forma de iniciar una conversación. Para iniciar la una conversación se selecciona el contacto y se obtiene una ventana de comunicación.

Comunicación El objetivo de una ventana de conversación es intercambiar mensajes con un contacto. Básicamente está dividida en el área de recepción de datos (dónde se muestra la conversación) y el área de envío de datos (dónde se edita el mensaje).

[1]Un contacto es un usuario con el cual se puede establecer una conversación. En el caso, de conferencias de Chat, el contacto es un salón.

[2]El awareness de presencia representa la percepción de presencia de los usuarios. Básicamente muestra el estado de disponibilidad del mismo

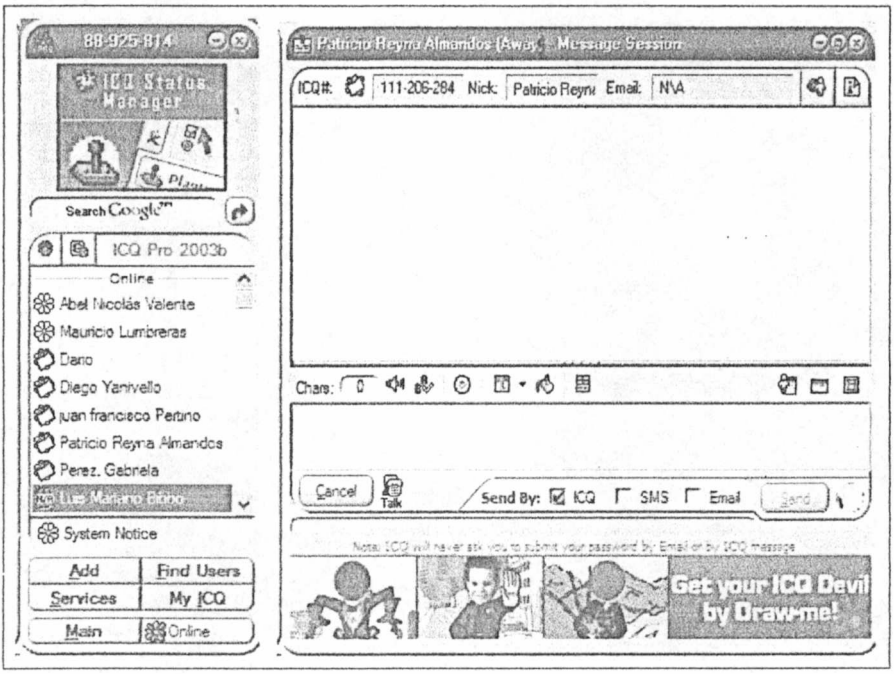


Figura 4.1: ICQ

Características

El contenido de los mensajes cumple con los requerimientos definidos en la Sección 3.1.1.

La distribución de mensajes puede realizarse tanto en la forma tradicional como en conferencia de Chat.

Una característica interesante es el soporte de percepción de presencia y estado de usuarios.

El transporte de documentos es soportado internamente, sin embargo la visualización se realiza por medio de visualizadores externos.

Esta herramienta está disponible en ambientes Windows solamente.

Al utilizar visualizadores externos no es posible la implementación de referencias.

4.1.3. MIRC

MIRC[16] es una implementación del protocolo de comunicación IRC[17].

Descripción

La Figura 4.2 muestra una conversación en una sala de Chat con la aplicación MIRC.

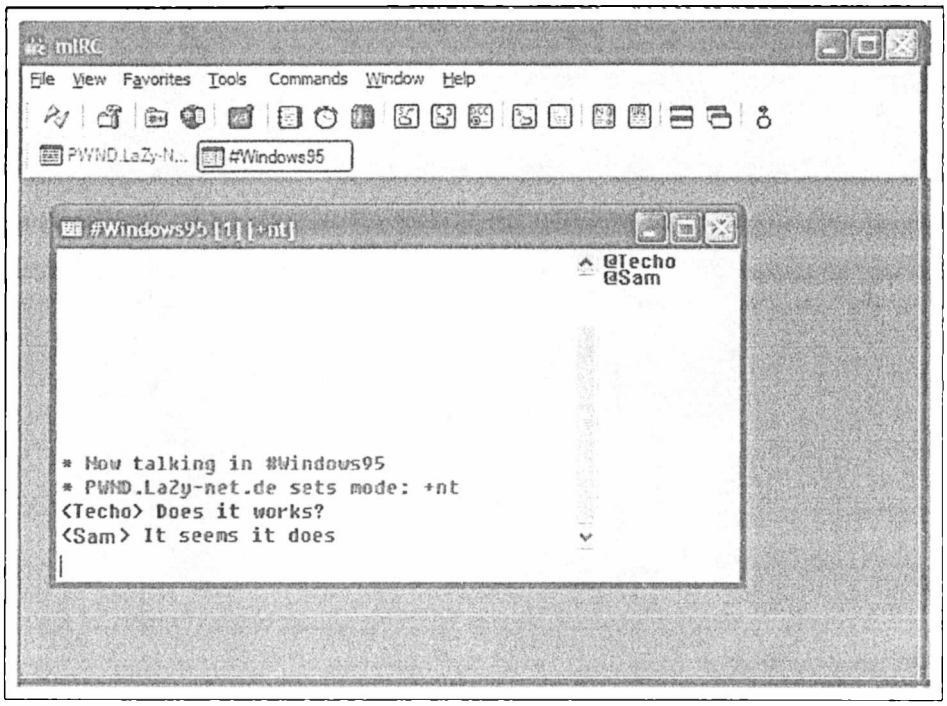


Figura 4.2: MIRC

La ventana está dividida en tres partes:

Lista de usuarios La lista de usuarios está a la derecha. Permite percibir la presencia de usuarios y su disponibilidad.

Recepción El área de recepción se encuentra a la izquierda en la parte superior. Éste área muestra la conversación.

Envío El área de envío se encuentra a la izquierda en la parte inferior. Permite editar un mensaje para ser enviado.

Características

El contenido de los mensajes cumple con las restricciones impuestas en la Sección 3.1.1.

Esta aplicación soporta ambos métodos de distribución de mensajes (tradicional y conferencia). La restricción del método tradicional radica en que sólo puede establecerse entre usuarios que comparten una sala. Este modo de Chat es conocido como Chat privado.

El pasaje de archivos no es soportado directamente por el protocolo IRC [17] que implementa esta herramienta. Sin embargo, existen otras aplicaciones distintas a MIRC dónde se soporta el intercambio de archivos por fuera del protocolo.

La visualización de los documentos se puede realizar por medio de aplicaciones externas únicamente.

Las referencias no son parte de la implementación del MIRC ya que no forman parte del protocolo IRC.

4.1.4. American Instant Messenger (AIM), Microsoft Messenger (MSM) y Yahoo Messenger

AIM[18], MSM[19] y Yahoo Messenger[20] son un conjunto de aplicaciones de diferentes empresas con características similares al ICQ. Junto con ICQ estos sistemas de Chat son los más utilizados.

La principal diferencia con ICQ es que estas aplicaciones no soportan la distribución de mensajes de conferencia de Chat (al menos en las versiones originales). Sin embargo, soportan percepción de presencia y estado de usuarios.

Este conjunto de herramientas tienen interfaces muy similares al ICQ y el mismo esquema de funcionamiento (ver Sección 4.1.2).

El intercambio de archivos es soportado por la mayoría de ellos.

La visualización de los mismos debe ser realizada mediante aplicaciones externas.

No existe soporte de referencias por ninguno de ellos.

4.2. Sistemas para compartir escritorios

Los sistemas para compartir escritorios permiten que dos o más usuarios tengan el control del mismo escritorio. La posibilidad de compartir el escritorio de un usuario tiene muchas aplicaciones. Como ejemplos, tenemos desde aplicaciones de aprendizaje hasta compartir un editor.

Generalmente se implementa mediante una ventana donde el escritorio del usuario que desea compartirlo, se visualiza en el ordenador del otro usuario.

Las acciones que se ejecutan en dicha ventana prácticamente tienen el mismo efecto que realizarlas localmente.

Varios sistemas operativos tienen esta herramienta disponible. El caso que analizaremos se basa en la que distribuye Microsoft.

4.2.1. Netmeeting

Netmeeting[21] es la herramienta básica de Groupware que provee Microsoft para los usuarios del sistema operativo Microsoft Windows.

Como se ha mencionado anteriormente, permite compartir el escritorio de un usuario entre varios. De esta forma los usuarios pueden comunicarse y sincronizar tareas.

Esta aplicación provee un conjunto de componentes de propósito general como Chat tradicionales, conferencias de Chat, pizarras compartidas, transmisión de voz y de video.

Los componentes de Chat que brinda la herramienta se pueden utilizar para intercambiar mensajes entre los usuarios. Dichos componentes soportan contenido en formato de texto enriquecido y distribución de mensajes en modo de conferencia de Chat.

En cuanto a la posibilidad de compartir documentos se utiliza el escritorio compartido del usuario que desea compartir el documento para visualizarlo. De esta forma, no hay necesidad de contar con la aplicación para visualizar dicho formato en todos los clientes. Es un modo muy flexible para tratar los documentos ya que la introducción del un nuevo formato de documento implica que solo el propietario del documento necesita contar con la herramienta de visualización para poder compartir la información contenida en el documento con los demás usuarios.

Sin embargo, encontramos las mismas limitaciones que con las aplicaciones anteriormente analizadas respecto a las referencias. No es posible unir de forma permanente un mensaje con una sección de un documento.

Netmeeting provee un entorno con información irrelevante para el objetivo que se persigue, permite compartir un escritorio completo y eso es mucho más de lo que deberíamos obtener, que es compartir solo un documento. Una de las consecuencias de compartir todo el escritorio es desperdiciar ancho de banda de comunicación (se transporta información irrelevante). Además fomentamos la descontextualización de la conversación.

Otra restricción radica en que los clientes deben ejecutar Netmeeting, eso acota las posibilidades de los mismos a correr sobre sistemas operativos Microsoft Windows únicamente.

4.3. Sistemas de Chat con referencias

Estos sistemas son los más similares a lo que se pretende desarrollar. La mayoría de las características que introducen estas aplicaciones se incorporan al sistema que se va a desarrollar. Permiten llevar a cabo una conversación teniendo como eje fundamental un documento. Uno de los puntos más destacables de estas aplicaciones es la inclusión de referencias.

4.3.1. XEROX Anchored conversations

XEROX Anchored conversations[22] permite realizar un Chat en el contexto de un documento. Esta aplicación permite realizar un Chat alrededor de un punto en un documento como muestra la Figura 4.3.

Presentaremos las limitaciones propias de un producto pionero en el campo de Chats con referencias. Sin embargo, esta aplicación fue una de las fuentes de inspiración para el producto que se intenta construir.

Descripción

Esta aplicación se basa en marcas (puntos) sobre un documento que comparten dos usuarios. De esta forma, una marca en el documento inicia un Chat tradicional. Allí se lleva a cabo una conversación y queda asociada al punto que se marcó. Por lo tanto, existe una sesión de Chat diferente por cada punto de discusión entre dos personas. El proceso se repite en cada punto de interés del documento. La Figura 4.3 muestra un documento con dos marcas(o referencias).

Características generales

El contenido de los mensajes se restringe a texto plano únicamente.

La distribución de mensajes se realiza entre dos usuarios exclusivamente.

No soporta manejo de percepción de presencia ni de estado de usuarios. Aunque la presencia de usuarios no es muy útil cuando la comunicación se realiza únicamente entre dos usuarios. Sin embargo, el awareness de presencia es muy útil para la sincronización de tareas.

Los Chat se desarrollan mediante la tecnología ActiveX[23] y por lo tanto son propietarios de la plataforma Microsoft Windows.

Referencias

Las referencias son contempladas en el sistema. Sin embargo, cuentan con una limitación importante. son puntos. No hay posibilidad de realizar

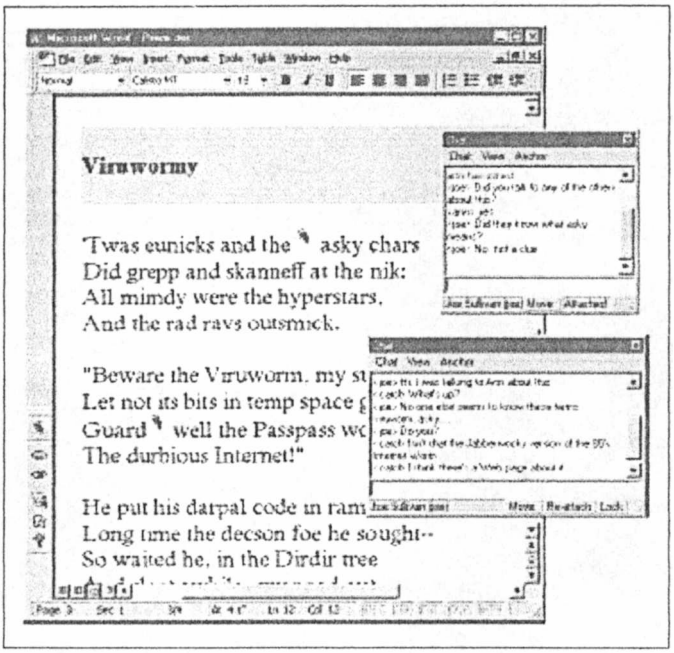


Figura 4.3: XEROX Anchored conversations

marcas sobre una región del documento, limitando el poder de expresión de la referencia.

Como ejemplo simple, supongamos que se realiza una marca puntual dentro de un párrafo de texto, como en la Figura 4.3.

¿Cuál es el alcance de la referencia?

1. El documento.
2. Un párrafo.
3. Una sección de un párrafo.
4. Una palabra.

Las marcas puntuales no distinguen la región del documento que se está marcando. El mismo problema respecto al alcance de las referencias que aparece sobre texto aparece sobre gráficos.

Supongamos que un documento gráfico tiene una marca puntual en algún lugar del documento.

¿Cuál es el alcance de la referencia?

1. ¿La imagen completa?

2. ¿Una región? ¿Dónde comienza? ¿Dónde termina? ¿Cuáles son sus límites?

Revisión de la conversación

Uno de los puntos débiles de esta aplicación es la pérdida del seguimiento temporal de la misma. El problema de tener los Chats en diferentes puntos del documento, en vez de centralizado, permiten que se pierda el orden de los mensajes.

Como consecuencia de ello, la conversación pierde el contexto temporal de las ideas y por lo tanto las revisiones se hacen más difíciles.

4.3.2. Snapchat

El objetivo de Snapchat[24] es ligar los mensajes con referencias a documentos. Este proyecto es el más cercano al objetivo de la herramienta pretendida. La Figura 4.4 muestra la interfaz de la herramienta.

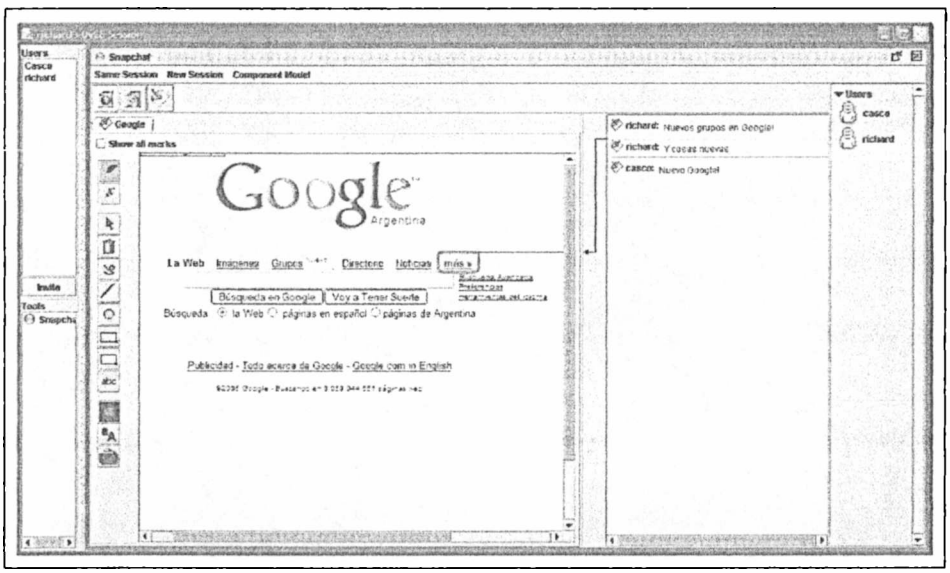


Figura 4.4: Snapchat

Descripción

La herramienta se divide su funcionalidad en tres grandes áreas:

Awareness de contactos Permite visualizar el estado de presencia de los usuarios en el Chat.

Intercambio de mensajes Los mensajes enviados a la sala son publicados en el área. Es posible editar, borrar y referenciar mensajes. Permite la manipulación de varios documentos por medio de solapas.

Espacio de referencias Las fuentes de referencias de los mensajes pertenecen están depositadas aquí. Dichas fuentes pueden ser páginas Web (limitadas a HTML 3.2), documentos de texto editados en el sistema (no se puede importar texto) y gráficos (también editados aquí).

Funcionamiento

Para enviar un mensaje, se necesita "marcar" una región del documento. Una vez realizada la "marca" se hace clic con el botón derecho sobre la marca y se selecciona "mark". Observaremos que aparece en la parte inferior de la pantalla una entrada para ingresar un mensaje. Al enviarlo es recibido por los participantes de la conferencia.

Cuando cualquier participante desea ver la referencia de dicho mensaje se selecciona de la lista y se muestra la referencia dentro del documento.

Implementación de Snapchat

Esta aplicación fue concebida a partir de un framework llamado DyCE[25] (Dynamic Coolaborative Environment). El objetivo de este framework es brindar soporte para la creación de aplicaciones colaborativas. DyCE es un framework que permite la manipulación de objetos distribuidos. El framework se encarga de preservar la consistencia de los datos, así como también de notificar los cambios producidos en el modelo. Dyce basa la construcción de las aplicaciones en dos componentes básicos:

1. El modelo compartido.
2. La componente de visualización.

El modelo compartido contiene los datos del modelo de la aplicación. La componente visual permite modificar y representar los cambios realizados a tal modelo.

En el caso del Snapchat, está compuesto por un modelo compartido que contiene a los mensajes y el estado de los usuarios. Además posee una componente visual(interfaz del Snapchat) que permite modificar los datos (por ejemplo, añadir un mensaje). La conferencia de Chat se implementa con la

creación de diferentes componentes de visualización (una por cada usuario) sobre el modelo compartido.

Características

El contenido de los mensajes es compatible con el requerido para el sistema.

Los medios que maneja son muy limitados, básicamente texto plano, páginas Web y gráficos propietarios.

Las referencias son soportadas y bien manipuladas, ya que permiten la marcación de texto y gráfico adecuadamente.

La posibilidad de extensión es muy limitada, haciendo difícil incluir nuevos formatos.

En síntesis, podemos afirmar que el sistema parece cumplir con el objetivo de nuestra aplicación, sin embargo no es del todo adecuado por las siguientes razones:

1. El sistema de comunicación es propietario y por lo tanto no es extensible.
2. Los formatos de documentos requeridos no son soportados.
3. No hay servidor público disponible.

4.4. Evaluación de sistemas

Esta sección es el resumen de la evaluación de las aplicaciones vistas en el capítulo respecto a los requerimientos expuestos en la Sección 3.4.

Presentaremos dos tablas que muestran la relación de cobertura de los requerimientos por las herramientas evaluadas. Cada columna representa una aplicación y cada fila un requerimiento de la lista de requerimientos de la Sección 3.4.

La Tabla 4.1 muestra la cobertura de los requerimientos generales de la aplicación respecto de la Sección 3.4 excluyendo los formatos de documentos que se estudian en la Tabla 4.2.

Se utilizarán las siguientes referencias para indicar la cobertura del requerimiento:

S Soportado

N No soportado

P Parcialmente soportado (no cumple con los requerimientos del todo).

	TALK	ICQ	MIRQ	AIM, MSM, YM	Netmeeting	XEROX	Snapchat
Conferencia de Chat	N	S	S	N	S	N	S
Mensaje en RTF	N	S	S	S	S	N	S
Servidores públicos	N	S	S	S	N	S	N
Multiplataforma SO	N	N	N	N	N	N	S
Documentos compartidos	P	P	P	P	P	S	N
Manipulación homogénea	N	N	N	N	N	N	S
Soporte de Referencias	N	N	N	N	N	P	S
Adaptabilidad de referencias	N	N	N	N	N	N	S

Cuadro 4.1: Tabla de requerimientos generales

	TALK	ICQ	MIRQ	AIM, MSM, YM	Netmeeting	XEROX	Snapchat
TXT	N	N	N	N	S	S	N
GIF	N	N	N	N	S	S	N
JPEG	N	N	N	N	S	S	N
PNG	N	N	N	N	S	S	N
RTF	N	N	N	N	S	S	N
DOC	N	N	N	N	S	S	N
XLS	N	N	N	N	S	S	N
PPT	N	N	N	N	S	S	N
HTML	N	N	N	N	S	S	S

Cuadro 4.2: Tabla requerimientos de formatos de documentos

Capítulo 5

Conceptualización

Este capítulo pretende introducir la funcionalidad de la aplicación a desarrollar.

Para describir el contenido de la aplicación hay que organizar la información que contendrá. La posible organización de la aplicación se plantea a nivel de funcionalidad y distribución de información.

Para orientar al lector, las funciones serán ilustradas con interfaces prototípicas en cada caso.

La estructura es la siguiente:

- Funciones orientadas al awareness de presencia
  - Información de contactos
  - Estado del usuario en sala
- Funciones de manipulación de información relacionada a la conversación
  - Envío de información
  - Recepción de información
    - Recepción de mensajes
    - Recepción de documentos
      - ◇ Referencias en un documento
    - Enlace de información

La aplicación está dividida en dos grandes grupos de funciones que la definen (ver Figura 5.1):

1. Funciones orientadas al awareness de presencia

2. Funciones de manipulación de información relacionada a la conversación

Figura 5.1: Funciones principales de la aplicación

5.1. Funciones orientadas al awareness de presencia

Las funciones orientadas al awareness de presencia de usuarios es un aspecto fundamental en toda aplicación Groupware. La percepción de presencia y estado de disponibilidad de usuarios permite establecer una relación entre un usuario y el resto de los participantes de la sesión[1]. Podemos dividir este conjunto de funciones dos subconjuntos. Uno de ellos es el conjunto de funciones de información de contactos y el otro es el conjunto de funciones relacionadas al estado del usuario en la sala, como se ve en la Figura 5.2).

[1]Se define sesión como lapso de tiempo en el que un conjunto de usuarios cooperan para cumplimentar un objetivo

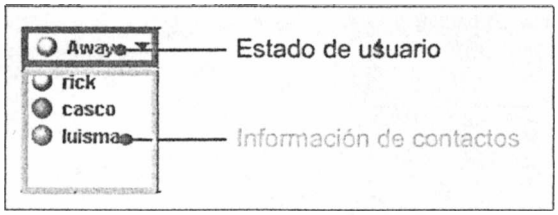


Figura 5.2: Awareness de presencia

5.1.1. Información de contactos

La información de contactos expresa el estado de disponibilidad de los usuarios conectados a la sala. Además, podría contar con la posibilidad de realizar actividades privadas relacionadas con un contacto en particular. Una función que usualmente se añade es el intercambio de mensajes privados entre usuarios.

5.1.2. Estado del usuario en sala

El estado de usuario en la sala establece la disponibilidad de presencia del usuario en la sala. De esta forma se pueden obtener diferentes estados de presencia en diferentes salas. El estado del usuario es una forma de expresar el nivel atención del mismo en la sala.

5.2. Funciones de manipulación de información relacionada a la conversación

La responsabilidad de las funciones de manipulación de información relacionada al desarrollo de la conversación es relacionar la información que se desprende de la comunicación entre los usuarios del sistema.

Los mensajes y los documentos son los principales tipos de información que se manipulan.

Los documentos pueden contener distintos medios de información como texto, gráfico o combinación de ambos.

Un aspecto importante es la relación que existe entre los documentos y los mensajes (referencias).

Tanto la funcionalidad de recepción como de envío de información forman parte del conjunto de funciones que se muestra en la Figura 5.3).

Comenzaremos describiendo las funcionalidades de envío y recepción por separado para luego relacionarlas.

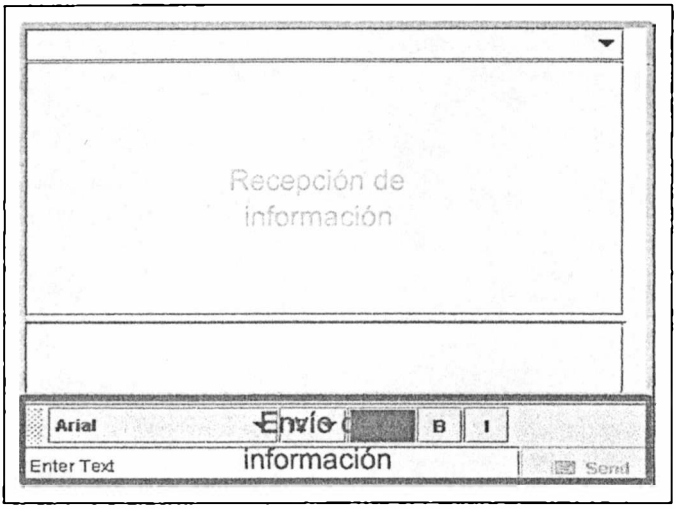


Figura 5.3: Funciones de manipulación de la información

5.2.1. Envío de información

El envío de información interpreta información desde la interfaz gráfica hacia los demás participantes de la conversación.

Los documentos y los mensajes son las principales formas de información.

Envío de documentos

Para poder mandar un documento debería ser posible escogerlo dentro de la máquina que ejecuta la aplicación y poder distribuirlo a los demás usuarios que son parte de la conferencia de Chat. Dichos usuarios deben ser capaces no sólo de recibir el documento sino también de poder interpretarlo para luego ser utilizado como parte de las referencias de los mensajes.

Envío de mensajes

El envío de mensajes debe contemplar dos aspectos: envío de texto enriquecido y una referencia.

Para poder realizar el envío de un mensaje con referencia deberían seguirse los siguientes pasos:

1. Elegir el documento al cual el mensaje hace referencia.
2. Marcar la zona que sea de interés.
3. Redactar el mensaje y enriquecerlo con los atributos que se consideren necesarios.

4. Enviar el mensaje.

De esta forma se establece una relación de cooperación entre la recepción de documentos y el envío de mensajes.

Se tiene por premisa que para enviar un mensaje con una referencia a un documento, primero se debe enviar el documento al que hace referencia el mensaje.

Recepción información

La responsabilidad del área de recepción información se reparte en tres conjuntos de funciones (ver Figura 5.4).

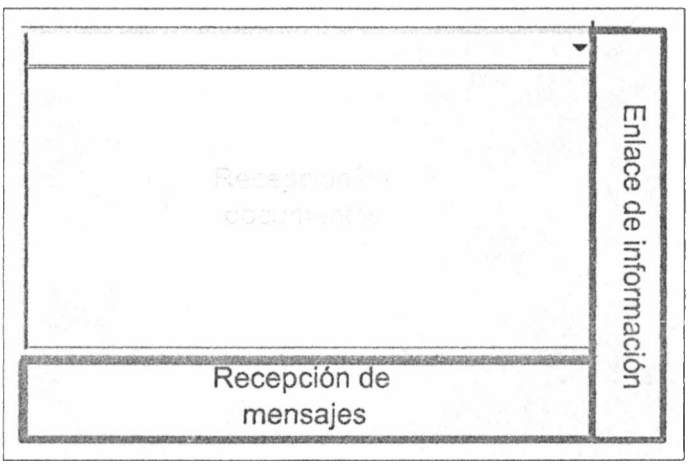


Figura 5.4: Recepción de la información

5.2.2. Recepción de mensajes

La recepción de mensajes se ocupa de que los mensajes recibidos estén disponibles al usuario. Estos mensajes contemplan el uso de referencias, por lo cual debe ser posible seleccionar algún mensaje y recuperar la información relacionada al mismo.

Los mensajes, al menos deben presentar la información del autor del mensaje (fundamental en entornos donde los participantes son más de dos) y alguna información con relación al tipo de referencia.

Recepción de documentos

La recepción de documentos tiene tres aspectos fundamentales:

- Visualización de los documentos recibidos.
- Visualización de las referencias pertenecientes a los mensajes.
- Marcar sobre el documento para enviar referencias como parte de los mensajes.

La recepción debe soportar manejo de múltiples documentos. Además debe ser capaz de manipular diferentes formas de referencias, ya que los medios de representación que deben ser soportados son variados y no exclusivos de un formato. Por ejemplo, HTML mezcla varios tipos de medios como pueden ser tablas, gráficos, texto o links en un solo documento.

Referencias en un documento

Uno de los puntos más importantes en la definición de la aplicación son las referencias asociadas a los mensajes. A partir de los mensajes con referencia se debe obtener la información del documento al cual pertenecen y la región dentro del documento.

Como mínimo son necesarias tres tipos de marcas:

Imagen Una referencia de imagen señala una región de una imagen. Dado que una imagen puede estar embebida en un documento mixto; para definir una marca en una imagen se necesita:

1. Las coordenadas absolutas de la región dentro de la imagen donde se ubica la referencia.
2. El desplazamiento dentro del documento donde está insertada la imagen. El desplazamiento es necesario ya que dentro de un documento puede convivir más de un gráfico. Indicando el desplazamiento dentro del documento queda determinado unívocamente el gráfico. Si el documento es una imagen solamente, entonces dicha posición es 0.

Textuales Una referencia textual limita un conjunto de caracteres consecutivos. Para poder definir la marca son necesarias las posiciones de inicio y fin dentro del documento.

Global Una referencia global asigna a un mensaje un documento completo.

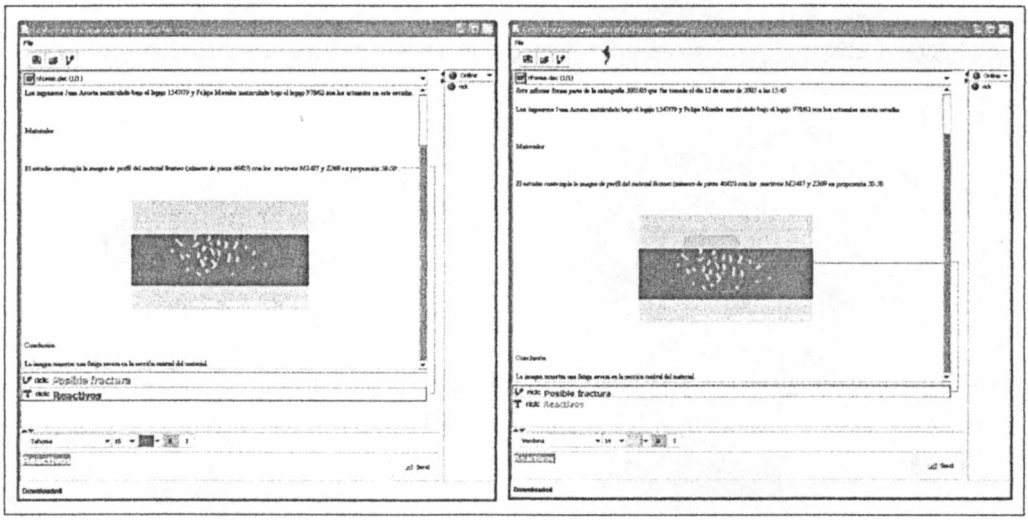


Figura 5.5: Ejemplos de marcas en documentos

Enlace de información

El enlace de información es el nexo entre las funciones de recepción de mensajes y documentos. La responsabilidad de relacionar un mensaje con su referencia es parte de su función.

La funcionalidad que tiene asignada este conjunto de funciones es la de orientar a los usuarios sobre la ubicación de una referencia dentro del documento.

Capítulo 6

Diseño

En esta sección se presentarán los diseños para la implementación de los requerimientos expuestos en el Capítulo 3. Toma los puntos más destacados de la aplicación y propone varias soluciones de diseño para cada caso en particular. A partir de este conjunto de soluciones se adopta la que mejor resuelve el problema y se justifica la elección.

6.1. Intercambio de mensajes

El primer problema que enfrentaremos es el de la comunicación. El principal objetivo del sistema es el intercambio de mensajes, por lo que es lógico pensar en un protocolo de basado en mensajes para implementar la aplicación.

6.1.1. Arquitectura

Esta sección establecer un diseño de arquitectura para este tipo de aplicación.

Dadas las características de la aplicación existen básicamente dos arquitecturas: por pares (Peer-To-Peer) y Cliente-Servidor para el intercambiar mensajes .

Arquitectura Peer to Peer

Las arquitecturas por pares tienen como objetivo establecer un canal de comunicación entre cada par de usuarios.

Un requerimiento de la aplicación es implementar un Chat con distribución de mensajes del tipo de conferencia de Chat. Este concepto implica que varios usuarios se comunican por medio de un canal de comunicación

común (sala). Por lo tanto, la idea de plantear una arquitectura de pares no parecería la manera más natural para implementar la solución.

Arquitectura Cliente-Servidor

La arquitectura Cliente-Servidor se basa en un esquema diferente. El servidor provee servicios, los cuales son requeridos por los clientes. Dichos servicios son procesados por el servidor y el resultado es retornado hacia el cliente. Una arquitectura de comunicación basada en cliente servidor centraliza la comunicación en un único nodo. Ésta parece ser una arquitectura más adecuada para modelar la solución.

Arquitectura bi-direccional

La desventaja de la arquitectura Cliente-Servidor en su estado más puro es que el cliente debe realizar un pedido para que el servidor envíe datos al cliente. Existen situaciones, por ejemplo, el estado de percepción de presencia de usuarios, que requiere que el servidor envíe datos a los clientes para actualizar el estado de los distintos contactos.

Por lo tanto, se puede definir un esquema "extendido" en el cual la comunicación puede ser establecida en ambos sentidos. Este esquema permitirá que el servidor reciba información de los clientes y los clientes del servidor.

Así queda definida la arquitectura de comunicación a utilizar.

6.1.2. Contenido de mensajes

Se debe definir la información que contendrán los mensajes dentro de la arquitectura definida. El contenido de los mensajes deberá especificar como mínimo la siguiente información:

- Identificación de un usuario en una sala de Chat.
- Especificación de la información necesaria para llevar el control del Chat.
- Especificar la información de los mensajes que intercambian los usuarios.

Un protocolo define en que forma se intercambiará la información entre los usuarios. Los requisitos de un protocolo que implemente la funcionalidad de una conferencia de Chat son:

1. Registro, autenticación y autorización de usuarios.

2. Soporte de percepción de presencia y estado de usuarios.

3. Creación dinámica de salas de conferencia de Chat.

4. Soporte de texto enriquecido para los mensajes.

5. Posibilidad de extensión del protocolo para:

   a) Envío y recepción de información de propósito general. El propósito, en este caso es poder enviar información relacionada a los documentos.

   b) Ligar información extra a los mensajes. Para enviar información relacionada a la referencia del mensaje.

Las alternativas ante la situación planteada anteriormente son:

1. Implementar un protocolo propio.

2. Utilizar protocolo existente.

Implementación un protocolo propietario

Implementar un protocolo propio implica un trabajo de diseño complicado, ya deben contemplar las características que mencionamos anteriormente.

Utilizando un formato propietario no podríamos valernos de software existente. Por lo tanto, habría que implementar tanto un cliente como un servidor para el intercambio de mensajes. Implementar un protocolo propietario trae las siguientes desventajas:

- Difícil integración con sistemas existentes.
- Bajo nivel de re-usabilidad.
- Altos requisitos de tiempo para puesta a punto y prueba.
- Implementación de clientes y servidores completos.

Utilización de un protocolo existente

La utilización de un protocolo existente trae una cantidad de beneficios, a saber:

1. Posibilidad de reutilización de código existente tanto en el cliente como en el servidor.

2. Existe cierta certeza que el protocolo es fiable.

3. Reducción en el tiempo de implementación y diseño.

De las alternativas anteriormente expuestas se optó por la utilización de un protocolo de mensajería llamado Jabber [26]. Este protocolo abarca todas las características que planteamos al principio de la sección.

Arquitectura del protocolo de comunicación Jabber

La arquitectura de este protocolo se basa en un servidor y varios clientes que se conectan al mismo. La comunicación entre los clientes siempre se lleva a cabo mediante el servidor, cuya función principal es distribuir los mensajes a los respectivos clientes(ver Figura 6.1 caso superior). En consecuencia, la intervención del servidor es transparente desde el punto de vista de los clientes (ver Figura 6.1 caso inferior).

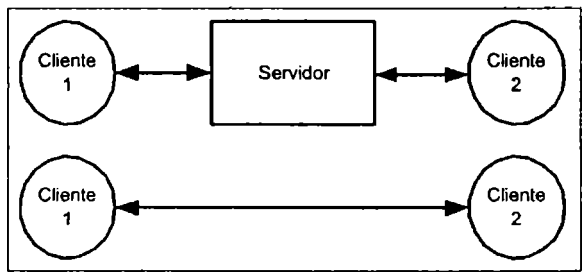


Figura 6.1: Comunicación Jabber

Jabber soporta varios protocolos de intercambio de mensajes, entre los que se encuentra el de conferencia de Chat que será implementado.

Una de las características más salientes del protocolo es la posibilidad de definir extensiones a los mensajes. De esta forma, el protocolo se puede modificar y adecuar a nuestras necesidades. Esto se logra básicamente cuando el servidor ignora la información que no puede interpretar y es reenviada a los clientes sin modificación alguna.

La filosofía del protocolo Jabber es delegar toda la carga de procesamiento que sea posible en el servidor y que los clientes tengan una carga mínima. Esto llevaría a pensar que las modificaciones o extensiones al protocolo deberían incurrir en modificaciones en el servidor. Sin embargo, se optó por delegar en el servidor el trabajo de manejo de percepción de presencia y estado de usuarios, así como también la seguridad y distribución de mensajes. La nueva funcionalidad, posiblemente propensa a modificaciones, se delega en el cliente. De esa manera, es el servidor el que no sufre modificaciones.

Para usar servidores estándar (públicos y gratuitos) se delega la funcionalidad extra en el cliente.

6.2. Intercambio de documentos

El intercambio de documentos es un punto central en el desarrollo de la aplicación. Citando la Sección 3.2 la visualización de un documento debe poder ser compartida con otros usuarios, de manera similar a la que lo hace un mensaje.

Existen algunos problemas relacionados con la posibilidad de compartir documentos.

1. Manejar diferentes formatos de documentos.

2. Transportar una representación del documento para ser visualizada. Debe ser posible transportar un documento de una máquina determinada al resto, para que los usuarios puedan utilizar dicha información como referencia en sus mensajes.

6.2.1. Formato de documentos

Primero se abordará el problema del manejo de diferentes formatos de documentos. Se define el manejo de documentos como la habilidad de cada cliente para visualizar un documento dentro del contexto de la aplicación e interpretarlo de manera tal que pueda reconocer los distintos tipos de zonas de marcado y referencia.

Dependiendo del medio de información del documento los tipos de marcas pueden variar, por ejemplo una marca de tipo texto es diferente a una marca de tipo gráfico. Ver Figuras 3.3, 3.4 y 3.5.

Un conjunto de alternativas para solucionar este problema son:

1. Diseñar un visualizador propio, interpretando cada formato

2. Diseñar un visualizador genérico que interprete un formato propietario, al que todos los demás formatos deben convertirse

3. Utilizar una componente que interprete HTML, y convertir todos los formatos a HTML

Alternativa utilizando un visualizador propietario

Tiene como principal desventaja la codificación para cada formato de un visualizador y por lo tanto la interpretación y procesamiento de cada formato por separado.

Como consecuencia, se tiene un alto impacto en la adaptación del software para el soporte de nuevos formatos o versiones de los formatos actualmente soportados. Un cambio de formato lleva a la modificación del visualizador.

La Figura 6.2 muestra la situación. Por cada formato se debe realizar un visualizador apropiado. Las componentes que están encerradas en el recuadro indican que deben ser re codificadas en caso de agregar un nuevo formato.

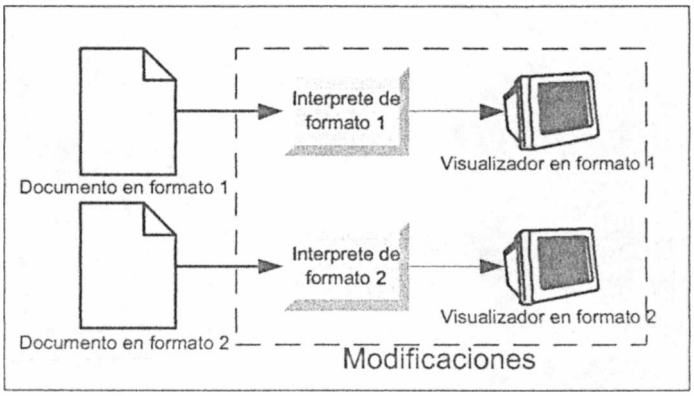


Figura 6.2: Impacto de modificaciones en caso de uso de un visualizador propietario

Alternativa utilizando un formato de interpretación propietario

La idea es diseñar un formato intermedio, propietario y genérico que sea interpretable por un visualizador único.

El formato intermedio debe contemplar los posibles medios de información de los distintos formatos. Además, se debe diseñar e implementar un visualizador para ese formato. La Figura 6.3 muestra el impacto en la aplicación en caso de añadir nuevos formatos.

Se debe proveer un conversor de todos los formatos a este formato propietario. De esta forma se heredan algunas de las desventajas del modelo anterior, por ejemplo, el cambio de versiones y la adición de nuevos formatos lleva a un cambio importante en el conversor. Sin embargo, el impacto es bastante inferior, ya que el visualizador no debería sufrir modificaciones.

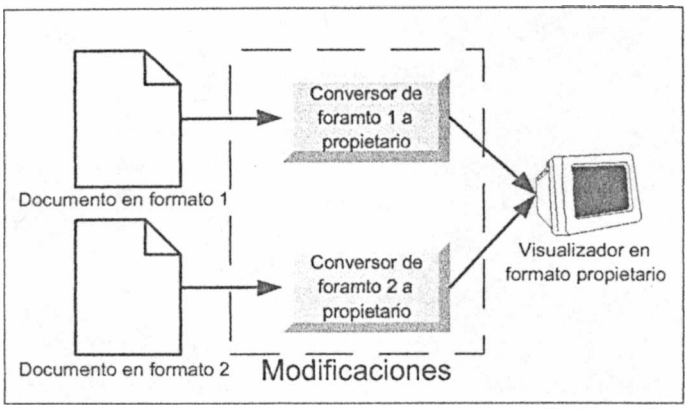


Figura 6.3: Impacto de modificaciones en caso de uso de un conversor propietario

Alternativa HTML como formato común

Esta alternativa, pretende convertir los formatos de los documentos a un formato estándar HTML[13]. HTML abarca de manera muy rica un conjunto suficientemente grande de elementos para la interpretación de distintos medios. Así el diseño del formato propietario planteado en las alternativas anteriores no existiría porque se utiliza este formato de presentación.

Algunas ventajas extras de utilizar un formato estándar como HTML son:

1. Usar las aplicaciones relacionadas a cada formato de documento para generar el formato HTML respectivo, eliminando la posibilidad de incompatibilidades entre versiones.

2. Usar componentes en los lenguajes existentes que interpreten HTML, evitando el diseño del visualizador.

Si bien necesita re codificación en caso de un nuevo formato a soportar, sería mínima ya que en el caso del visualizador se puede contar con componentes que ya están disponibles en los lenguajes de programación y por lo tanto no deben ser ni siquiera codificadas una vez. En el caso del conversor, la utilización del programa de aplicación desde el lenguaje de programación es una solución que provee una conversión independiente de las versiones del formato para la conversión. De esta forma, se evita la re implementación. La Figura 6.4 esquematiza la solución a implementar.

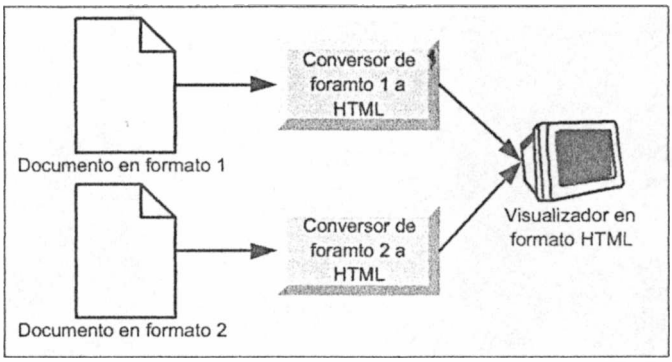


Figura 6.4: Impacto de modificaciones en caso de uso de HTML como formato para documentos

Conclusión sobre el visualizador de documentos

Podemos afirmar que la última solución es la más adecuada por las siguientes razones:

1. Gran independencia de las aplicaciones a soportar, casi todas las aplicaciones tienen forma de exportar sus documentos a un formato HTML.

2. Independencia de las versiones, si se tiene disponible la versión de la aplicación con la que fue confeccionado el documento, el diseño no sufriría modificaciones.

3. HTML como formato estándar presenta la posibilidad de uso de componentes del lenguaje para su interpretación e implementación, independientemente del formato. Sin embargo, se debería tener disponible la versión de la aplicación con la que se crea el documento a transportar en todos los clientes y eso puede no ser posible, un ejemplo es la posibilidad de que la aplicación no corra en plataformas de sistema operativo distintas.

6.2.2. Transporte de documentos

El problema planteado al final de la sección anterior fue:

HTML como formato estándar presenta la posibilidad de uso de componentes del lenguaje para su interpretación e implementación, independientemente del formato. Sin embargo, se debería tener disponible la versión de la aplicación con la que se crea el documento a transportar en todos los clientes y eso puede no ser posible, un ejemplo es la posibilidad de que la aplicación no corra en plataformas de sistema operativo distintas.

Existen diversas formas de transportar la información del documento para que sea interpretada por los diferentes clientes. Entre las posibles soluciones analizaremos:

Un documento como parte un mensaje

Se analiza la posibilidad de enviar el contenido del documento en formato HTML al resto de los usuarios como parte de un mensaje.

Un problema de este tipo de transporte es: ¿cómo distinguir un mensaje de un documento?.

Además hay que abordar el tema de Latecomming. El Latecoming es una situación que es bastante usual en entornos Groupware como el que estamos tratando y se refiere al problema que ocurre cuando existen usuarios que se unen a una sesión de trabajo cuando esta ya está en desarrollo. Para ejemplificar la situación enunciaremos el siguiente ejemplo:

Sean A y B dos usuarios que llevan una conversación sobre un documento enviado por A. Un usuario C se quiere unir a la conversación, sin embargo el documento no está disponible ya que el mensaje fue enviado y no fue recibido por C.

Existen dos soluciones para este problema:

1. Reenviar el documento para que el nuevo usuario pueda utilizar el documento. Esta es la solución más simple; sin embargo, el resto de los clientes recargarían sus documentos nuevamente y si se deben manejar varios documentos al mismo tiempo se tendrían varias versiones de los mismos documentos en una misma sesión llevando a un exceso de información en todos los clientes. Además, si varios usuarios desean unirse a la sesión, existiría una cantidad proporcional de versiones de documentos a la cantidad de usuarios que se unan a la sesión en desarrollo. La Figura 6.5 muestra un esquema de la situación.

2. Guardar toda la conversación y cuando un usuario "llega tarde" se carga toda la información de la sesión en el cliente y de esa forma queda sincronizado.

   La solución parece prometedora; sin embargo, no se condice con la realidad. Cuando un usuario se "une" a una conversación no es natural "escuchar" lo anteriormente mencionado. A pesar de ello, es muy corriente que los documentos tratados en una conversación si estén disponibles.

Como conclusión, deberíamos tener el documento disponible pero no la conversación completa. Para ello se recomienda un servidor separado dónde

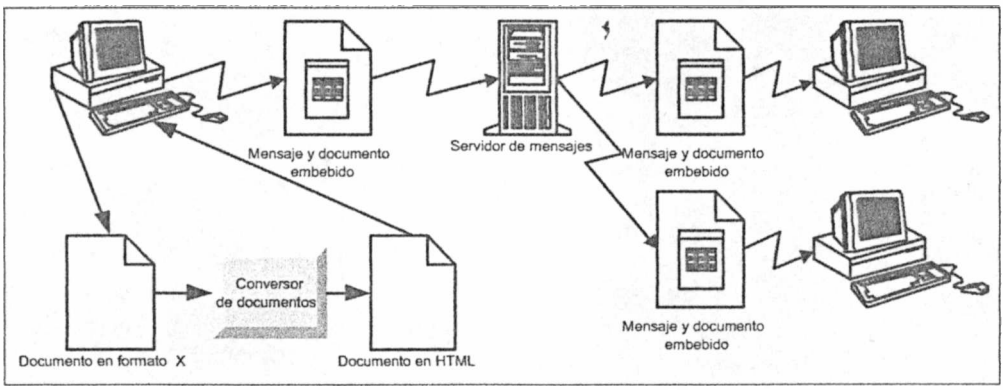


Figura 6.5: Documento como contenido de un mensaje

todos los documentos estén disponibles.

Servidor de documentos con conversión local

Esta solución se basa en que el documento permanezca en un servidor mientras dure la sesión. De esta forma cuando un usuario se "une" a una sesión, puede recuperar el documento sin necesidad de reenvío de información. La secuencia de acciones sería la siguiente:

1. El usuario poseedor del documento a compartir lo "transporta" a un servidor de archivos.

2. Luego envía la dirección dónde lo depositó al resto de los usuarios.

3. Los usuarios que reciben el mensaje acceden al archivo, lo convierten a HTML y luego lo visualizan.

De esta forma, cualquier referencia en los mensajes puede cargar el documento desde el servidor, convertirlo y utilizarlo.

La Figura 6.6 describe la situación en los distintos pasos.

El principal problema de esta estrategia reside en la necesidad de que los usuarios deben tener las aplicaciones en sus estaciones de trabajo para poder convertir los formatos a HTML. Diferentes de configuraciones en las distintas estaciones de trabajo pueden no coincidir en la transformación del archivo en formato HTML, trayendo consecuencias en las referencias hacia ese documento.

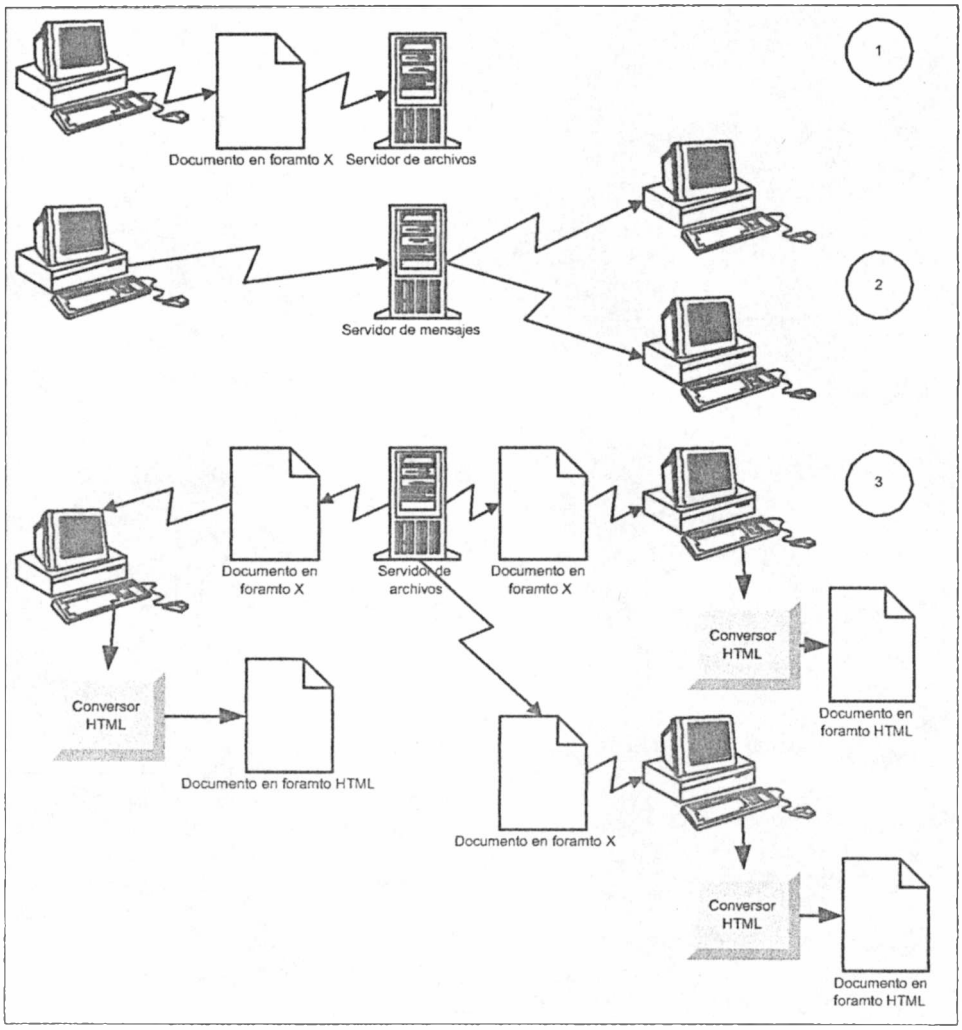


Figura 6.6: Servidor de documentos con conversión local

Servidor de documentos con conversión remota

La última solución al problema de transferencia de archivos que planteamos acerca del servidor de documentos, se basa en la anterior, con una leve modificación para evitar las conversiones locales.

La arquitectura se basa en el esquema presentado en la Figura 6.7.

Un cliente interactúa con dos servidores para llevar a cabo el transporte de documentos. El servidor de documentos, está relacionado con el sistema de almacenamiento, conversión y publicación de documentos, mientras que el servidor de mensajes se encarga de la sincronización del transporte.

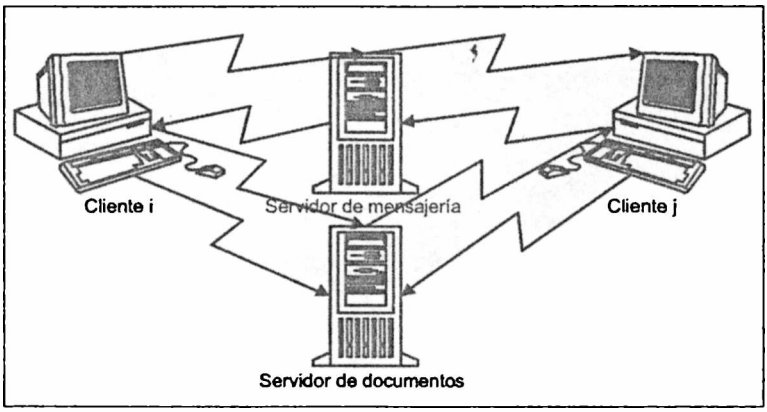


Figura 6.7: Servidor de documentos con conversión remota

1. El servidor de mensajería será el encargado de llevar a cabo todas las tareas relacionadas con el servicio de distribución de mensajes, percepción de presencia de usuarios, registro de usuarios, autenticación de usuarios, manejo de salas de conferencia, entre otras.

2. El servidor de documentos tiene dos servicios a su cargo:

   a) El servicio de conversión se encarga de obtener la información del documento en los formatos soportados y convertirlos en HTML.

   b) El servicio de publicación de documentos brinda acceso a los documentos HTML para todos los clientes.

La estrategia de diseño que usamos sigue los siguientes pasos:

1. El usuario poseedor del documento hace una copia del archivo a compartir en el servidor de archivos.

2. El servidor guarda el archivo y lo convierte en una serie de documentos HTML que aloja en un directorio público. Notar que un archivo puede dar origen a más de un documento HTML. Por ejemplo, un documento en formato PowerPoint puede convertir cada una de sus diapositivas en un documento distinto.

3. El servidor retorna un conjunto de direcciones donde aloja al documento al usuario que envió el documento.

4. El usuario, propietario del documento, recibe y reenvía las direcciones a los demás usuarios.

5. Cada usuario carga las direcciones enviadas.

Para clarificar la situación se desarrollará un ejemplo de transferencia de documentos.

Sean los clientes, Cliente i, Cliente j y Cliente k usuarios del sistema que comparten una sala de conferencia de Chat. La situación describe la posibilidad de que un documento, Documento 1, que reside en la estación de trabajo del Cliente i, sea compartido con los clientes j y k.

Para poder compartir un documento, primero debe ser transportado desde la estación de trabajo del Cliente i hacia el servidor de conversión y publicación (ver Figura 6.8).

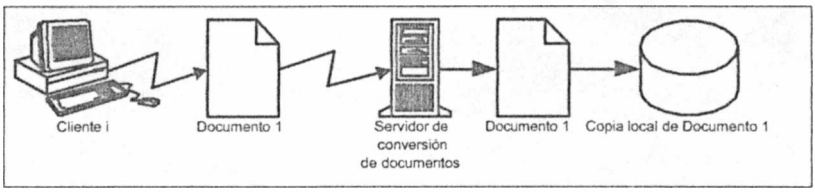


Figura 6.8: Primera etapa de conversión

Una vez que el Documento 1(en formato original) está disponible en el servidor; se procede a convertirlo al formato HTML y generar una versión del mismo para publicarlo. De esta forma está disponible al resto de los clientes (ver Figura 6.9).

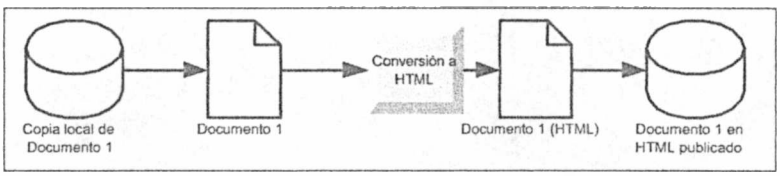


Figura 6.9: Segunda etapa de conversión

Una vez que la versión del documento en formato HTML es publicada por el servicio de conversión, el servidor responderá al Cliente i con la o las direcciones para acceder al documento. Más de una dirección puede ser generada, ya que un documento, como una presentación en Microsoft PowerPoint, está dividido lógicamente en diapositivas y cada una puede ser una página de HTML diferente (ver Figura 6.10).

Luego de recibir la colección de direcciones de acceso al documento convertido, el Cliente i envía un mensaje al servidor de mensajería. Este mensaje contiene la o las direcciones de acceso al documento (ver Figura 6.11).

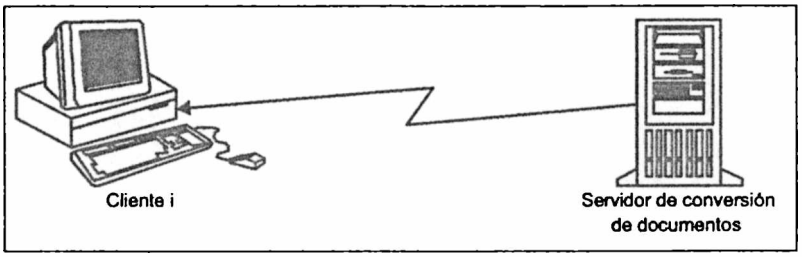


Figura 6.10: Tercera etapa de conversión

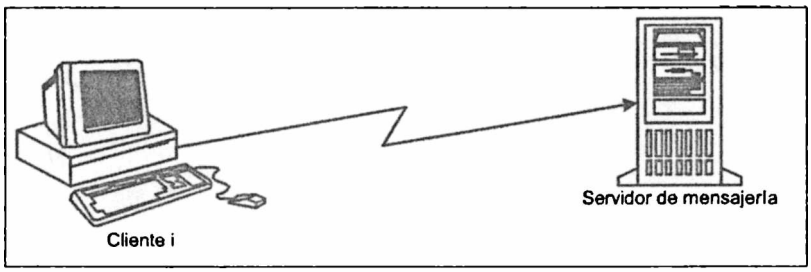


Figura 6.11: Cuarta etapa de conversión

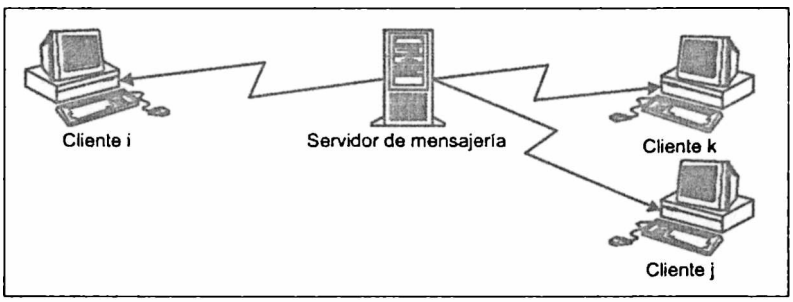


Figura 6.12: Quinta etapa de conversión

El mensaje es propagado al resto de los clientes en esa sala de conferencia para que dicho documento sea accedido (ver Figura 6.12).

Una vez que el mensaje es recibido por los Clientes el acceso se realiza por medio del servicio de publicación de documentos (ver Figura 6.13).

Las dos ventajas más importantes de este diseño son:

1. Uniformidad de la conversión. El servidor de documentos es la única fuente de información recibida por los usuarios.

2. Independencia de la plataforma para los usuarios. El servidor de documentos es la única estación que debe convertir los documentos, y por lo tanto manejar aplicaciones propietarias.

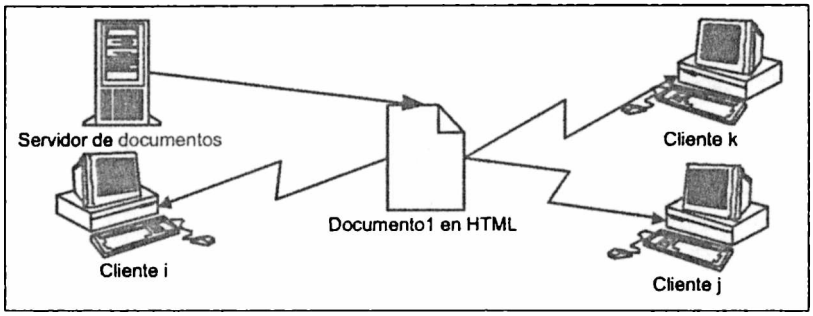


Figura 6.13: Sexta etapa de conversión

6.3. Intercambio de referencias

Una vez terminado el diseño del intercambio de mensajes y documentos, hay que representar las referencias de los mensajes a los documentos. La manera más simple de hacerlo es mediante extensiones al mensaje. La extensión contendrá la información de la "marca" o referencia al documento.

En cuanto a la arquitectura relacionada con el transporte de referencias, las mismas son enviadas a través del servidor de mensajes como parte de los mismos.

La estructura de un mensaje estará compuesta de un mensaje en formato HTML (para brindar la posibilidad de texto enriquecido) y un adicional de información que representa la referencia en un documento.

Un mensaje contiene la siguiente información:

1. Mensaje en HTML que se desean enviar al resto de los usuarios.
2. Referencia al documento referenciado.
3. Referencia a la sección relacionada con el texto del mensaje.

Un mensaje que contiene una referencia tiene la estructura que se muestra en la Figura 6.14.

Todas las marcas, independientemente del tipo, tienen una estructura común y se puede dividir en tres partes.

La referencia: Especifica el documento al cual el mensaje está haciendo referencia

La selección: Especifica la forma de la marca de la referencia

La unión: Especifica el color con el que la marca se unirá al mensaje

Las Figuras 6.15, 6.16 y 6.17 dan una visión más concreta del asunto, mostrando cómo las marcas formarán parte de los mensajes.

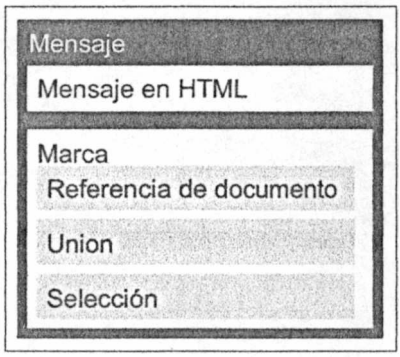


Figura 6.14: Estructura de un mensaje

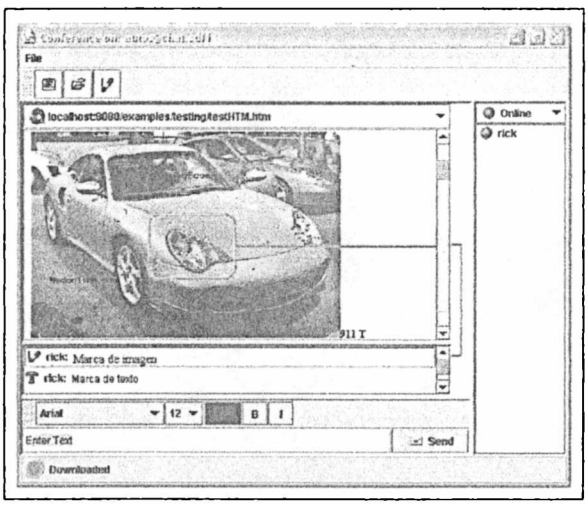


Figura 6.15: Marca imagen

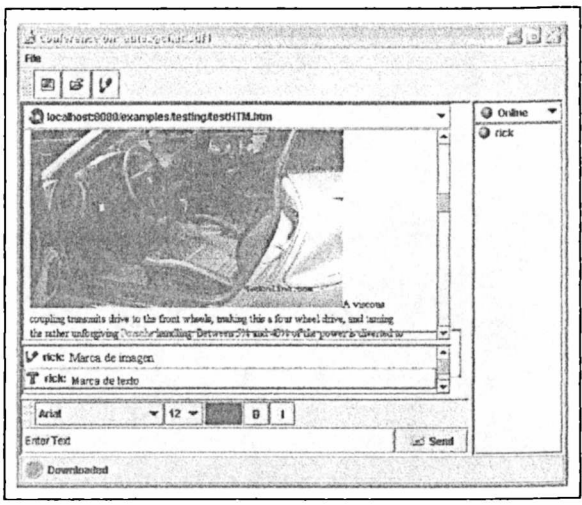


Figura 6.16: Marca texto

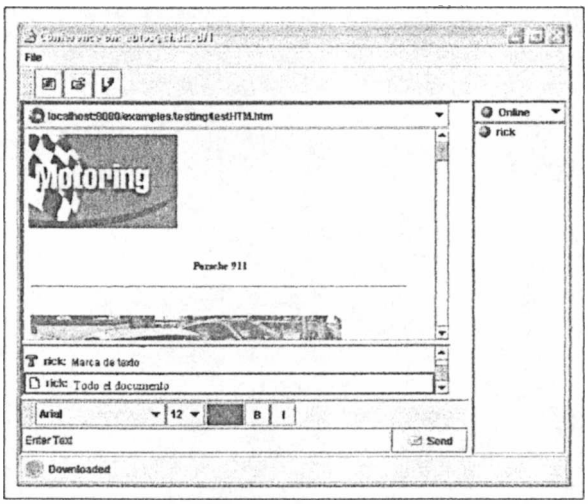


Figura 6.17: Marca global

Capítulo 7

Implementación

Se tratan los puntos más importantes de la implementación del diseño propuesto.

La implementación es descripta a alto nivel y esboza la estructura e interacción de los principales componentes de software de la aplicación.

Comienza con la descripción de la representación de la información necesaria en función de una implementación basada en el protocolo Jabber.

Luego continúa con los aspectos de implementación de los servidores de documentos y mensajes.

Una vez descriptos los servidores, se describen los puntos esenciales del cliente. El cliente se implementa en tres capas transporte, modelo e interfaz. Para describir el cliente se utilizan diagramas UML de clases e interacción.

Por último se describe el conversor de documentos, tanto desde el punto de vista del cliente como del servidor.

7.1. Representación de la información

La implementación de la comunicación de mensajes y sincronización se basará en el protocolo de mensajería Jabber. Presentaremos una breve reseña del modelo de mensajería Jabber basado en la bibliografía[28].

Elementos del protocolo

Existen cuatro elementos importantes en cualquier intercambio de mensajes usando Jabber.

Servidor El servidor Jabber maneja y participa en toda comunicación Jabber. Tiene como responsabilidad principal proveer servicios a los clien-

tes. Los servicios más importantes que ofrece son el de distribuir los mensajes y de manejar las cuentas de usuarios.

Cliente Los clientes Jabber actúan como agentes que muestran información (mensajes) al usuario final. Además, responden a las entradas de los usuarios. Los clientes Jabber pueden ofrecer servicios autónomos en la red Jabber (chatbots).

Flujos Conceptualmente la red de comunicación entre el cliente y el servidor son un par de flujos de una dirección. Veremos la comunicación desde 2 puntos de vista:

- Desde el punto de vista de XML, el flujo Jabber es un flujo de documento XML[27] encerrado entre marcas `<stream:stream>` .
- Desde un punto de vista lógico el flujo forma una sesión de meta información de contexto sobre el flujo de la conversación.

La meta-información referida anteriormente se basa en el JABBERID. El JABBERID representa una identificación para los elementos del sistema.

Paquetes Los paquetes son fragmentos de XML enviados a través del flujo entre el cliente y el servidor. Cada paquete es un sub-documento de XML válido y auto contenido. Los protocolos Jabber especifican el formato de los paquetes y el procedimiento correcto para intercambiarlos. Por lo tanto se intercambiarán paquetes para todas las operaciones que permite el protocolo.

Flujo de información del protocolo

Descriptos los elementos que componen el protocolo se presenta el flujo de la información en la Figura 7.1.

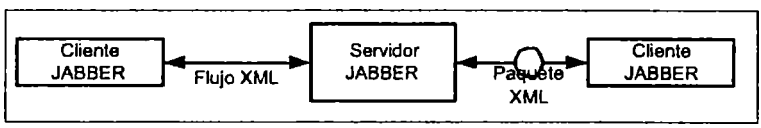


Figura 7.1: Flujo de información Jabber

Identificación de los elementos del protocolo

Para identificar tanto a los clientes como al servidor se utilizan identificadores llamados JABBERID. Los identificadores constan de tres partes fundamentales:

1. El nombre del usuario.
2. El dominio al cual pertenece el usuario, es decir en el dominio que está registrada la cuenta.
3. El recurso (dispositivo usado para la comunicación).

Así una típica dirección puede verse como **user@domain/resource**.

En el caso de Chats tradicionales (dónde la comunicación se establece entre un par de usuarios) la dirección de destino de los mensajes será la del usuario que recibirá el mensaje. Sin embargo, en el caso de conferencia de Chat, la dirección destino será la de una sala. Como es de esperar la dirección de una sala de conferencia de Chat no especifica recurso ya que no tiene sentido especificar el dispositivo destino.

Paquetes del protocolo

Existen tres tipos principales de paquetes:

Paquetes de presencia Este tipo de paquetes tienen por función principal manejar el estado de percepción de presencia de los usuarios o contactos. Un ejemplo de un paquete de presencia es:

```
<presence type='available'/>
```

De esta forma se indica que el usuario que envía dicho paquete está disponible. La presencia de usuarios tiene distintos niveles, por lo que pueden variar de acuerdo al grado de disponibilidad del usuario. Más información sobre estos paquetes puede encontrarse en [26].

Paquetes de información y consulta Este tipo de paquetes es muy importante para el manejo del estado de los usuarios, el registro y autenticación de los mismos. Para detalles sobre este tipo de paquetes y los de presencia referirse a [26].

Paquetes de mensajes Nos detendremos en este tipo de paquetes ya que será el que modificaremos para que nuestra aplicación pueda enviar y recibir la información necesaria para llevar a cabo la transferencia de documentos y el intercambio de mensajes con referencias.

7.1.1. Intercambio de mensajes

Los formatos descriptos en esta sección se refieren a mensajes intercambiados mediante la distribución de mensajes en forma de conferencia de Chat. Este tipo de paquetes es el indicado para transferir datos entre los usuarios. El paquete tiene el siguiente formato:

```
<message to='sala@servidor'>
<body> Esto es un mensaje </body>
</message>
```

El XML anterior, es el correspondiente a un mensaje que tiene por destino *sala@servidor* y como contenido *Esto es un mensaje*. El texto dentro del sub-paquete **body** especifica un mensaje en texto plano. Sin embargo un mensaje con HTML puede ser enviado como cuerpo de un mensaje. Este cuerpo se especifica mediante un sub-paquete **html**; en simultáneo con el mensaje en texto plano. Un ejemplo que representa un mensaje con un cuerpo en HTML se muestra a continuación.

```
<message to='sala@servidor'>
<body> Esto es un mensaje </body>
<html>
<body>
<p>Esto es un <b>mensaje</b></p>
</body>
</html>
</message>
```

Se utilizan las dos alternativas (texto y HTML para el mensaje) porque el cliente puede no soportar HTML. La recepción de un mensaje por un cliente tiene el siguiente formato:

```
<message to='receptor@server.com' from='sala@server'>
<body> Esto es un mensaje </body>
<html>
<body>
<p>Esto es un <b>mensaje</b></p>
</body>
</html>
</message>
```

De esta forma, el servidor completa la información necesaria para que el mensaje contenga los datos para identificar el receptor del mismo. Entre los datos se encuentra el JABBERID de la sala a la cual fue enviado el mensaje.

Existe la posibilidad de manejar extensiones en los mensajes. Las extensiones tienen por objeto brindar un medio de comunicación flexible, que permita la inserción de información extra en los mensajes. Esta información no es procesada por el servidor de mensajes, por lo que el contenido de dicho paquete pasa a los clientes, siempre y cuando sea XML válido.

Uno de los aspectos más importantes de las extensiones es el espacio de nombres, etiquetado como **xmlns**. El espacio de nombres se usa como un identificador para interpretar el mensaje. El siguiente código ejemplifica la recepción de un mensaje mientras el usuario estuvo offline.

```
<message from='server.com' to='user@server.com'>
<subject>Bienvenido!</subject>
<body>Bienvenido a Jabber! </body>
<x xmlns='jabber:x:delay' from='source@server.com'
stamp='20011206T18:22:09'>
Offline Storage
</x>
</message>
```

subject Indica el motivo del mensaje.

body Indica el contenido del mensaje en texto plano.

x Indica un sub-paquete de extensión en un espacio de nombres *jabber:x:delay*. Contiene información relacionada al origen del mensaje, la fecha en que fue enviado y el motivo de la demora en la recepción.

Como conclusión de la representación de información mediante el modelo de mensajería Jabber, se puede asegurar que tanto la implementación del intercambio de documentos como del intercambio de mensajes con referencias se puede llevar a cabo mediante extensiones a los mensajes.

7.1.2. Intercambio de documentos

Como se mencionó en la Sección 7.1.2 el intercambio de documentos se efectúa en dos etapas:

1. La primera es la etapa de conversión.

2. La segunda es la etapa de propagación de la información relacionada con la locación del documento.

Una vez realizada la conversión del documento comienza la etapa de propagación de información sobre la locación del mismo. Así, el cliente propietario del documento a compartir tiene la información de la locación del documento en el servidor. Teniendo esta dirección, se procede a enviar un mensaje a la sala de conferencia de Chat en la que se desea compartir el documento. Para que los participantes de la conferencia obtengan la locación del documento, el propietario envía un mensaje a la sala deseada.

El mensaje tiene la siguiente estructura:

```
<message type='groupchat'
to='richard@sdf1/home'
from='java@chat.sdf1/richard'
id='' xmlns='jabber:client'>
<body xmlns='jabber:client'>
http://localhost:8080/ConverterWEB/files/3/3/3.html received
</body>
<x xmlns='http://www.richard.org/docMessage'>
<url xmlns='http://www.richard.org/docMessage'>
http://localhost:8080/ConverterWEB/files/3/3/3.html
</url>
<description xmlns='http://www.richard.org/docMessage'>
Testing.doc (1/1)
</description>
<action xmlns='http://www.richard.org/docMessage'>
add
</action>
</x>
</message>
```

Se puede observar que el mensaje contiene una extensión con nombre de espacio:

http://www.richard.org/docMessage

El nombre de espacio está relacionado con la entidad que define el contenido del mismo. Este formato de XML fue diseñado para contener básicamente tres sub-paquetes:

url Este paquete contiene la dirección dónde se aloja el documento.

description Se utiliza para darle un nombre lógico al documento. Se toma por convención, dar como descripción del documento, el nombre del archivo del documento. Como se mencionó anteriormente, un documento puede derivar en una colección de direcciones, así que se añade el número de página al nombre del documento para identificar una hoja del mismo y la cantidad total de páginas.

action Representa la acción que lleva a cabo este paquete. En este caso, es un paquete que indica la adición de un mensaje a la conversación.

7.1.3. Intercambio de referencias

Para llevar a cabo el intercambio de referencias se utiliza otra estructura de mensaje. En este caso, el mensaje contiene dos aspectos:

- El contenido del mensaje en HTML que incluye el sub-paquete HTML.
- La extensión para la referencia implementada mediante un sub-paquete de extensión identificado con el nombre de espacio *jabber:x:*Mark*. El asterisco denota un tipo de marca dependiente del medio y forma de marcado.

La estructura del mensaje (descripta en la Sección 6.3) puede representarse como se ve en la Figura 7.2

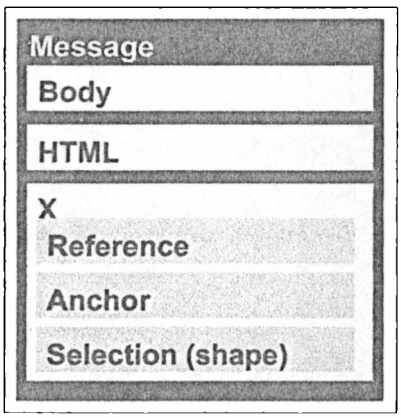


Figura 7.2: Estructura de un paquete de mensajes Jabber

Las marcas usualmente especifican colores. Estos colores son descriptos en sub-paquetes que se explicarán para cada tipo de marca. Sin embargo, para todos ellos se utiliza la paleta de colores basada en RGB para definir el color. Se utilizarán las etiquetas R para rojo, G para verde y B para el azul.

La implementación de las marcas se describe a continuación:

Marca de imagen

El objetivo de esta marca es limitar la región dentro de un gráfico que se encuentra dentro de un documento. Para poder ubicar una marca se debe obtener la siguiente información.

1. El documento a la cual se refiere la marca.
2. La imagen dentro del documento, ya que puede darse la posibilidad de que exista más de una imagen dentro de un documento.
3. La región dentro de un documento.

Para transportar la información se utilizan las extensiones Jabber a los mensajes, dicha extensión cuenta con tres sub-paquetes fundamentales:

El sub-paquete "reference" El sub paquete "reference" contiene una etiqueta indicando la **url** a la cual hace referencia la marca.

El sub-paquete "anchor" El sub-paquete "anchor" describe el color de la unión que existe entre el mensaje y la marca.

El sub-paquete "selection" El sub-paquete "selection" describe la marca. La marca está descripta en base a cuatro sub-paquetes:

El sub-paquete "background" El sub-paquete "background" describe el color del fondo de la marca. Usualmente se lo transparenta para que se pueda observar el contenido de la marca.

El sub-paquete "foreground" El sub-paquete "foreground" describe el color del borde de la marca. Usualmente este color se sólido ya que denota los límites de la marca.

El sub-paquete de "offset" El sub-paquete de "offset" cumple la función de identificar una imagen dentro de un documento de contenido mixto. Usualmente todos los elementos dentro de un documento HTML tienen un orden, por lo tanto, cada elemento tiene asignado un número que identifica el "orden" dentro del documento. De esta forma se puede identificar cualquier elemento dentro de un documento HTML. El número de orden se describe como **offset** y ubica la imagen dentro del documento HTML.

El sub-paquete de forma El sub-paquete de forma contiene información relacionada con la forma geométrica de la marca. Por lo tanto, este paquete describe el tipo de dibujo que se usará para marcar. La implementación actual utiliza de rectángulos, así que el sub-paquete utilizado es el **rectangle**.

El manejo de otros tipos de figuras geométricas no es complejo, basta con cambiar la descripción y adecuar el contenido a la marca deseada (con su debida implementación). En el caso de la implementación del rectángulo como medio de marca, se especifican las coordenadas "x" e "y" del extremo superior izquierdo del rectángulo relativas a la imagen y además, el ancho y alto de la marca en "width" y "height" respectivamente.

Presentaremos como ejemplo:

```
<message type='groupchat' to='richard@sdf1/home'
from='java@chat.sdf1/richard' id='' xmlns='jabber:client'>
<html xmlns='http://www.w3.org/1999/xhtml'>
   <head xmlns='http://www.w3.org/1999/xhtml'>
     <font color='#000000' size='3' face='Arial'
      xmlns='http://www.w3.org/1999/xhtml'>
     </font>
    </head>
   <body xmlns='http://www.w3.org/1999/xhtml'>
     Graph mark
   </body>
  </html>
  <x xmlns='jabber:x:imageMark'>
<reference
url='http://localhost:8080/ConverterWEB/files/1/1/1.html'
xmlns='jabber:x:imageMark'/>
    <anchor xmlns='jabber:x:imageMark'>
     <color b='0' r='255' g='0' xmlns='jabber:x:imageMark'/>
    </anchor>
    <selection xmlns='jabber:x:imageMark'>
     <background b='0' r='255' g='0'
     xmlns='jabber:x:imageMark'/>
      <foreground b='0' r='255' g='0'
       xmlns='jabber:x:imageMark'/>
      <offset xmlns='jabber:x:imageMark'>23207</offset>
      <rectangle xmlns='jabber:x:imageMark'>
       <x xmlns='jabber:x:imageMark'>25</x>
        <y xmlns='jabber:x:imageMark'>159</y>
        <width xmlns='jabber:x:imageMark'>287</width>
        <height xmlns='jabber:x:imageMark'>34</height>
      </rectangle>
</selection>
  </x>
```

```
</message>
```

Descripción:

- La marca referencia un documento ubicado en la URL *http://localhost:8080/ConverterWEB/files/1/1/1.html* dentro del servidor de documentos.
- El color de la unión entre la marca y el mensaje es rojo.
- El número de orden de la imagen dentro del documento es 23207.
- La selección se representa por medio de un rectángulo de coordenadas x = 25 e y = 159 con un ancho de 287 y un alto de 34 píxeles. El color del borde y del contenido es rojo.

Marca de texto

El contenido de una marca de texto es muy similar al de una marca de imagen. La principal diferencia radica en la información necesaria para poder definir la marca.

La marca se expresa por medio de una extensión. La extensión tiene el mismo formato anterior y se divide en tres partes. La función de cada parte es la misma, la diferencia se encuentra en el sub-paquete de **selection**. En el caso de una marca de texto la información para definir la región dentro del documento se basa en la posición de inicio y final del documento. En este caso definidas en las etiquetas **start** y **end**. Los sub-paquetes **background** y **foreground** definen los colores del fondo de la selección y de la fuente de la marca. Un ejemplo de este tipo de marca sería:

```
<message type='groupchat' to='richard@sdf1/home'
from='java@chat.sdf1/richard' id=''
xmlns='jabber:client'>
  <html xmlns='http://www.w3.org/1999/xhtml'>
<head xmlns='http://www.w3.org/1999/xhtml'></head>
   <body xmlns='http://www.w3.org/1999/xhtml'>
     Text mark
   </body>
  </html>
  <x xmlns='jabber:x:textMark'>
   <reference
   url='http://localhost:8080/ConverterWEB/files/1/1/1.html'
```

```
    xmlns='jabber:x:textMark'/>
    <anchor xmlns='jabber:x:textMark'>
     <color b='0' r='255' g='0'
     xmlns='jabber:x:textMark'/>
    </anchor>
    <selection xmlns='jabber:x:textMark'>
     <background b='0' r='255' g='0'
     xmlns='jabber:x:textMark'/>
      <foreground b='0' r='255' g='0'
       xmlns='jabber:x:textMark'/>
      <start xmlns='jabber:x:textMark'>9932</start>
      <end xmlns='jabber:x:textMark'>9954</end>
    </selection>
  </x>
</message>
```

Descripción:

- La marca referencia un documento ubicado en la URL *http://localhost:8080/ConverterWEB/files/1/1/1.html* dentro del servidor de documentos.
- El color de la unión entre la marca y el mensaje es rojo.
- El número de orden de la imagen dentro del documento es 23207.
- La selección comienza en el carácter 9932 y termina en 9954. El color del borde y del contenido es rojo.

Marca global

La última forma de expresión implementada por la aplicación es la referencia a documentos completos. La única información necesaria para que una marca de este tipo sea definida es la ubicación del documento. Dicha información se encuentra en el sub-paquete de **reference**. Los colores definidos en las etiquetas **background** y **foreground** tienen las mismas características e interpretación que los definidos para las marcas anteriores. El siguiente segmento representa una marca a un documento completo:

```
<message type='groupchat' to='richard@sdf1/home'
from='java@chat.sdf1/richard' id='' xmlns='jabber:client'>
<html xmlns='http://www.w3.org/1999/xhtml'>
```

```
   <head xmlns='http://www.w3.org/1999/xhtml'>
     <font color='#000000' size='3' face='Arial'
     xmlns='http://www.w3.org/1999/xhtml'></font>
</head>
   <body xmlns='http://www.w3.org/1999/xhtml'>
     <font color='#000000' size='3' face='Arial'
     xmlns='http://www.w3.org/1999/xhtml'>
     Enter Text
     </font>
   </body>
  </html>
  <x xmlns='jabber:x:globalMark'>
   <reference
   url='http://localhost:8080/ConverterWEB/files/2/2/2.html'
   xmlns='jabber:x:globalMark'/>
    <anchor xmlns='jabber:x:globalMark'>
      <color b='255' r='0' g='0'
      xmlns='jabber:x:globalMark'/>
     </anchor>
     <selection xmlns='jabber:x:globalMark'>
      <background b='255' r='0' g='0'
      xmlns='jabber:x:globalMark'/>
      <foreground b='255' r='0' g='0'
       xmlns='jabber:x:globalMark'/>
     </selection>
  </x>
</message>
```

Descripción:

- La marca referencia un documento ubicado en la URL *http://localhost:8080/ConverterWEB/files/1/1/1.html* dentro del servidor de documentos.
- El color de la unión entre la marca y el mensaje es rojo.
- El número de orden de la imagen dentro del documento es 23207.
- El color del borde y del contenido es rojo.

7.2. Los servidores

Como se mencionó en la Sección 5, se utilizan dos servidores para poder llevar a cabo la transferencia de información.

Repasaremos los objetivos de los tres componentes principales del software (servidor de documentos, servidor de mensajes y cliente).

El servidor de documentos tiene por objetivos:

- Brindar un espacio público para compartir documentos.
- Proveer un servicio de conversión de documentos a HTML.

El servidor de mensajes se basa en el protocolo de comunicación de mensajes Jabber. Este servidor provee las funciones báscas de un servidor de intercambio de mensajes.

- Servicio de percepción de presencia y estado de usuarios.
- Autorización, registro y autenticación de usuarios.
- Creación de salas de conferencia .
- Distribución de mensajes.

El cliente debe ser capaz de interactuar con los dos servidores e interpretar la información que recibe de ellos. Las funciones del cliente básicamente son:

- Enviar y recibir documentos.
- Enviar y recibir mensajes con referencia.
- Mostrar el contenido de los mensajes
- Permitir las operaciones necesarias sobre ellos para generar referencias.

7.2.1. El servidor de mensajes

El servidor de mensajes no fue implementado. El motivo es que no se necesitan modificaciones sobre este elemento de software para que la aplicación pueda llevar a cabo su objetivo; por lo tanto el servidor puede ser público, freeware o propietario.

Existen ciertos requerimientos que debe cumplir el servidor para que sea compatible con la aplicación cliente.

1. Compatibilidad con el protocolo de intercambio de mensajes Jabber.

2. Soporte para extensiones de mensajes. Esta característica es estándar de Jabber, pero pueden existir implementaciones muy primitivas que no la soporten porque no es excluyente con el servicio de mensajería básico (Chat tradicional).

3. Soporte para el envío de HTML como contenido de mensaje. Se debe destacar que este no es un estándar del protocolo; sin embargo, está propuesto como tal y no está lejos de ser aprobado.

4. Soporte para el protocolo de conferencias de Chat. Existen muchas implementaciones de servidores Jabber que no implementan este tipo de protocolo de comunicación. Pero es un estándar fundamental para el funcionamiento de la aplicación.

Como consecuencia de las características apuntadas anteriormente, se ha elegido para mostrar el prototipo JIVE Messager[29](para instalar este servidor se dirigirse al Apéndice C.2). Este servidor propietario permite su uso en forma gratuita con la restricción de no conectar más de diez usuarios al mismo tiempo. Esta restricción no es obstáculo para mostrar la potencialidad de la herramienta.

7.2.2. El servidor de documentos

Debe brindar al menos tres servicios:

1. Transporte.

2. Conversión.

3. Publicación.

Como servidor de documentos se utiliza un servidor HTTP[42]. En este caso se optó por Apache Tomcat[30] por las siguientes razones:

- Este servidor es gratis y posee un buen respaldo en cuanto a documentación y soporte ya que es parte del Jakarta Project
- Es open source, por lo tanto, están disponibles los códigos fuente del servidor
- Provee la capacidad de generar servicios bajo el protocolo de comunicación HTTP. Dichos servicios son codificados mediante la implementación de Servlets[31] en JAVA[32]

- Capacidad de publicación de documentos bajo el protocolo HTTP
- Existen implementaciones tanto para plataformas Windows[33] como para UNIX[34]

Las razones expuestas anteriormente son solo las más salientes de Apache Tomcat respecto al contexto de la aplicación; sin embargo, cuenta con muchas ventajas más.

La publicación de documentos se implementa a partir de un servicio Web el cual está implementado como un Servlet. El Servlet recibe el contenido de un documento como parámetro de un requerimiento de servicio HTTP. La respuesta a ese requerimiento es un conjunto de direcciones HTTP que representan las ubicaciones del documento convertido en la red. De esta forma el cliente conoce la ubicación del documento que ha transferido.

El Servlet ejecuta la siguiente secuencia de acciones:

- Establece una ubicación dentro del sistema de archivos donde el documento será depositado para ser publicado.
- Transfiere el archivo desde el cliente al servidor. El servicio de transferencia se basa en la implementación del protocolo HTTP en su versión 1.1. Para ello se vale de la implementación del método POST de RFC 2616[35].
- Se procede a la conversión del documento del formato original a HTML. Este procedimiento será explicado de manera más detallada al final de la sección.
- Publicación del documento en formato HTML. La publicación es prácticamente automática ya que la ubicación del documento en formato HTML se especifica dentro del directorio dónde el contexto de TOMCAT está ejecutándose.
- Notificar al cliente la ubicación del mismo. Se recoge la ubicación de los documentos que la conversión dio como resultado. Luego se genera una respuesta al requerimiento, basada en el método RESPONSE RFC 2616[36] del protocolo HTTP. La ubicación de los documentos forma parte del cuerpo de la respuesta del requerimiento del cliente.

La Figura 7.3 representa el transporte y publicación de documentos:

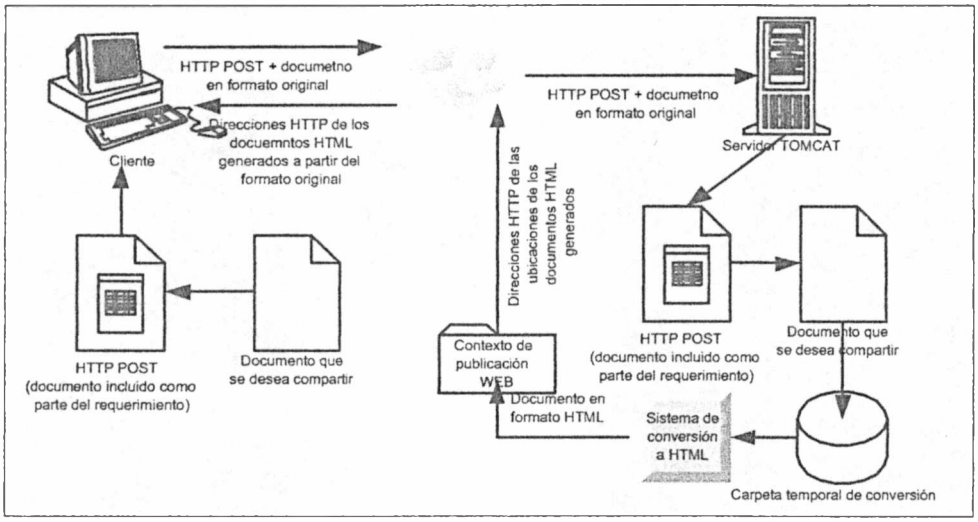


Figura 7.3: Transporte y publicación de documentos

7.2.3. Reseña de servidores

Como se ha explicado en la sección de diseño, un documento se transporta con la colaboración de los dos servidores.

- Para compartir un documento el cliente hace un requerimiento HTTP al servidor de documentos con el contenido del documento como parámetro.
- El servidor de documentos acepta este pedido y se lleva a cabo la transferencia del documento.
- A partir de ese momento se efectúa la conversión del documento a formato HTML y luego se publica.
- Como respuesta del requerimiento HTTP el servidor envía como contenido de la respuesta una colección de URLs que representan el contenido del documento en formato HTML.
- El cliente que recibe la respuesta genera por cada URL un mensaje que será enviado al servidor de mensajes. El mensaje tiene por destino una sala de conferencia de Chat.
- De esta forma el mensaje es recibido por los usuarios ligados a la sala, incluyendo al usuario que envió el documento.

- Todos los clientes acceden al mismo documento en el servidor de documentos.

7.3. El cliente

El cliente está basado en un modelo por capas. LAs capas principales son tres:

- Transporte
- Modelo
- Interfaz

7.3.1. Capa de transporte

La capa de transporte es una capa ligada a la transferencia de mensajes entre el cliente y el servidor Jabber y tiene como responsabilidades:

1. Establecer la comunicación entre el cliente y el servidor.
2. Proveer una capa de abstracción de la conexión al resto de la aplicación.

La clase Connection

Para establecer la comunicación entre el cliente y el servidor se utiliza la clase **Connection** (ver Figura 7.4). El mecanismo de comunicación que utiliza esta clase son **Sockets**[1]. Un socket permite establecer una conexión bi-direccional a partir de una dirección de HOST y un puerto de comunicaciones. La comunicación fluirá a través de los flujos de entrada y salida, **InputStream**[2] y **OutputStream**[3] respectivamente. **Connection** clase encapsula la siguiente funcionalidad de conexión:

- Conexión al servidor Jabber: Mediante el método *open* se indica el host y el puerto con el cual se establece la comunicación.
- Desconexión del servidor Jabber.
- Envío de datos: El método *getOutputStream* devuelve un **OutputStream** que permite enviar bytes.

[1]http://java.sun.com/j2se/1.4.2/docs/api/java/net/Socket.html
[2]http://java.sun.com/j2se/1.4.2/docs/api/java/io/InputStream.html
[3]http://java.sun.com/j2se/1.4.2/docs/api/java/io/OutputStream.html

- Recepción de datos: El método *getInputStream* devuelve un **InputStream** que permite recibir bytes.

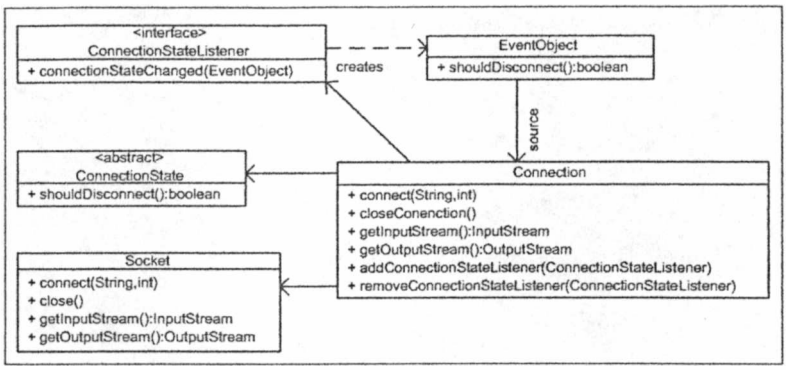


Figura 7.4: Diagrama de clases de la etapa de conexión

Para modelar el estado de la conexión se construyó el patrón de diseño State [2] en el que la clase **Connection** y la clase abstracta **ConnectionState** toman los roles de contexto y estado respectivamente. Esta clase también posee un mecanismo de notificación de cambio de estado mediante la utilización del patrón de diseño Observer[2] dónde la clase **Connection** representa al sujeto y la interfaz **ConnectionStateListener** representa al observador.

La clase Transporter

Esta capa permite al programador enviar y recibir unidades más manejables que XML en su forma primitiva (bytes). Para ello utiliza la clase **Transporter** (ver Figura 7.5).

Las unidades de transporte están representadas por la clase **Packet** cuya única función es esconder la complejidad del armado de los paquetes XML al programador. De esta forma, para el programador sólo existe el envío y recepción de objetos **Packet**.

La responsabilidad de la clase **Transporter** es proveer una interfaz de comunicación en la que se envían y reciben **Packet**, además de los métodos de conexión y desconexión respectivos.

Para enviar un **Packet**, mediante la capa **Transporter**, existe el método *send* que delega en la clase **Sender** la responsabilidad de enviar el XML representado por el **Packet** al servidor.

La recepción es un poco más complicada que el envío de mensajes, ya que se presenta un problema de concurrencia denominado Productor-Consumidor (ver Figura 7.6).

Los participantes fundamentales de la recepción son tres:

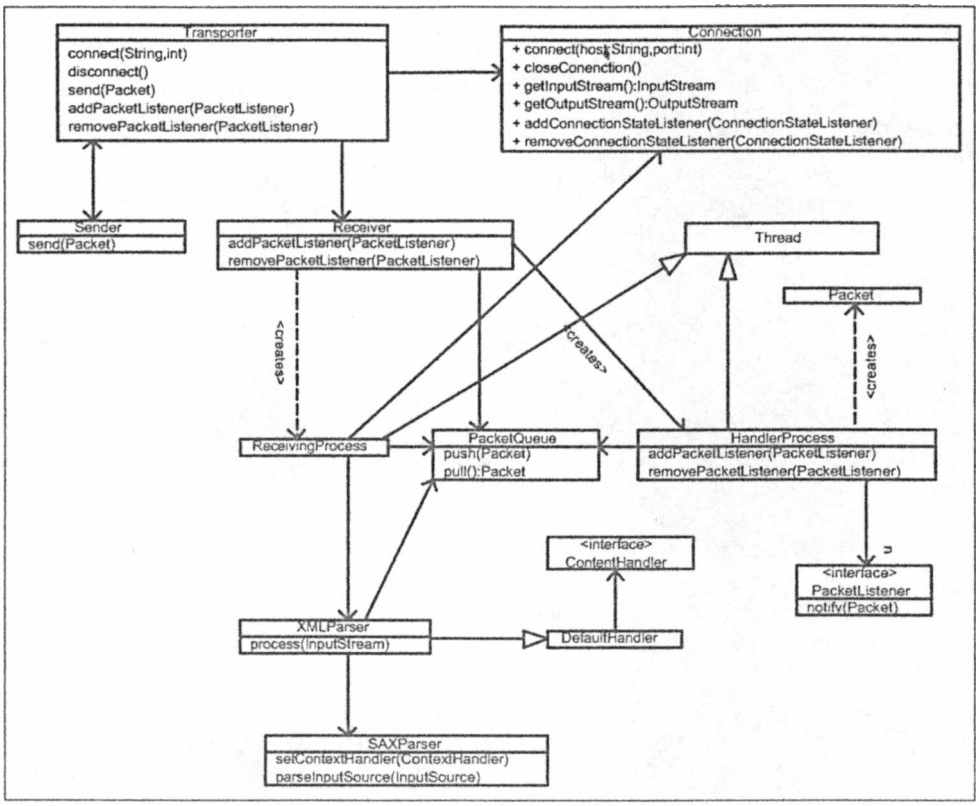


Figura 7.5: Diagrama de clases la etapa de transporte

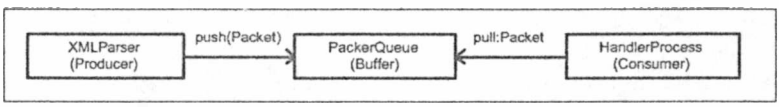


Figura 7.6: Esquema Productor-Consumidor

1. El productor. representado por un proceso **ReceivingProcess** que se encarga de recibir XML del **InputStream** y depositar un **Packet** en el **PacketQueue**

2. El buffer, representado por la clase **PacketQueue** permite independizar el proceso de recepción del proceso de manejo del **Packet**.

3. El consumidor, representado por un proceso **HandlerProcess** obtendrá los **Packet** de **PacketQueue** y los procesará.

ReceivingProcess utiliza un parser de XML incremental el cual está implementado por las clases **XMLParser** y **SAXParser**, estas clases son es-

pecializaciones de la librería de clases XERCES[37].

XERCES permite el parsing de XML mediante la especialización de algunas clases y la construcción de objetos a partir de XML.

Se basa en el patrón de diseño Builder[2]. La clase **DefaultHandler** cumple el rol de Builder, **XMLParser** de ConcreteBuilder, **ReceivingProcess** de Director y **Packet** de Product.

HandlerProcess toma un **Packet** desde el **PacketQueue** y luego notifica de su recepción a los **PacketListeners** que tiene registrados. Para cumplir esta función se implementó el patrón de diseño Observer[2]. **HandlerProcess** cumple el rol de sujeto mientras que **PacketListener** cumple el rol de observador.

Los **PacketListener** son registrados en el **HandlerProcess** a través de la interfaz de **Transporter**.

El objetivo de los **PacketListener** es procesar un tipo determinado de **Packet** y de esa forma ejecutar las acciones pertinentes al **Packet** recibido. La relación establecida entre cada **PacketListener** y un **Packet** puede verse como una modificación del patrón de diseño Visitor[2] dónde el visitador sería el **Packet**, el elemento sería **PacketListener** y la estructura objeto sería el **HandlerProcess**.

De esa forma, la aplicación recibe los **Packet** desde el servidor.

7.3.2. Capa de modelo de la aplicación

Para poder continuar con la explicación de los distintos procesos involucrados debemos definir el modelo de la aplicación.

El modelo de la aplicación tiene la estructura mostrada en el diagrama de clases de la Figura 7.7.

La clase **PapaloteGroupchatClientApp** es un intermediario entre el modelo de la aplicación y la interfaz. Para implementarlo se aplicó el patrón Mediator[2].

PapaloteGroupchatClientApp consta de dos partes:

1. La sesión
2. El modelo de Chat

La sesión (representada por la clase **Session**) maneja toda la información relacionada con la sesión entre el cliente y el servidor Jabber. Por ejemplo: registrar un usuario, conectar y desconectar al usuario del servidor de mensajes, unirse a una sala, etc. Para cumplir con su función conoce la cuanta que representa al usuario en el servidor (representada por la clase **Account**)

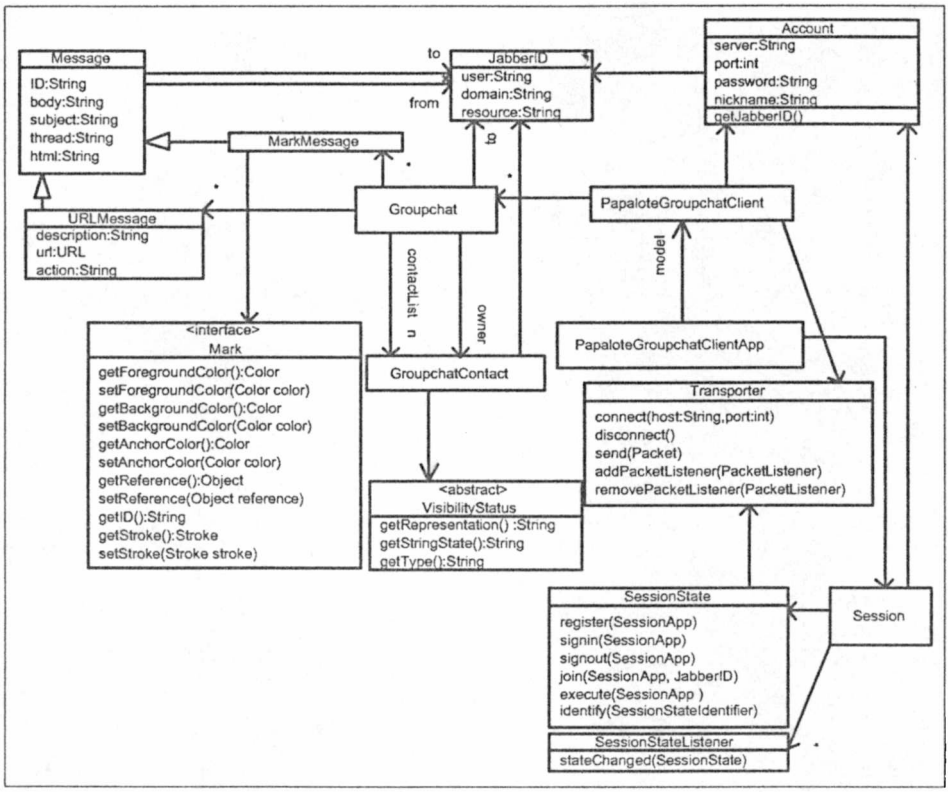


Figura 7.7: Diagrama de clases del modelo de la aplicación

y a la clase **Transporter** para poder llevar a cabo la comunicación con el servidor. La implementación de **Session** está basada en el patrón de diseño State[2]. dónde el estado se representa con la clase **SessionState**.

El modelo de Chat (representado por la clase **PapaloteGroupchatClient**) es responsable de manejar toda la información relacionada al desarrollo de las conversaciones en las diferentes salas de Chat.

La clase Groupchat

Cada sala de Chat está representada por una instancia de la clase **Groupchat**. La responsabilidad de esta clase es manipular la información relacionada con la sala. Dicha información comprende tanto la recepción como el envío de información desde y hacia la sala como también el awareness de presencia de los contactos en la sala. Cada instancia de **Groupchat** es identificada como un contacto (representado por la clase **GroupchatContact**) a la cual se le envían y del cual se reciben mensajes.

Groupchat además conoce:

Una colección de documentos Esta colección representa a los documentos que se utilizan en la sala. Cada documento está representado por una instancia de la clase **URLMessage**).

Una colección de mensajes con referencias Esta colección representa a los mensajes intercambiados por los usuarios conectados a la sala. Cada mensaje está representado por una instancia de la clase **MarkMessage**.

Una colección de contactos Esta colección representa los usuarios conectados a la sala.

La clase Message

Tanto **URLMessage** como **MarkMessage** son subclases de **Message** que presenta un mensaje. El mensaje es la unidad de información que se intercambian los usuarios para comunicarse.

La clase **URLMessage** representa el envío de un documento y contienen la URL donde se publica el documento.

La clase **MarkMessage** representa un mensaje con referencia. Contiene una URL que identifica al documento al que se refiere y una marca (representada por la interfaz **Mark**).

La marca es representada por la interfaz **Mark** que define los métodos necesarios para poder definir una marca dentro de un documento. Las implementaciones se basan en las marcas de imágenes, texto o globales a un documento.

La clase GroupchatContact

Un **GroupchatContact** representa una entidad a la cual se le pueden enviar o de la cual se pueden recibir mensajes. Puede ser una sala o un usuario conectado a una determinada sala.

En el caso que manejamos, servicio de conferencia de Chat solamente, no ha implementado el Chat privado por lo tanto los contactos conectados a una sala no pueden intercambiar mensajes entre ellos. Fue una decisión de arbitraria, pero no por ello se altera la definición de contacto.

En el caso de un usuario la clase **GroupchatContact** permite identificar un usuario y conocer su estado de disponibilidad (awareness de presencia). La solución para implementar el awareness de presencia de los usuarios se basa en el patrón de diseño State[2].

La estructura de la aplicación quedó expuesta en esta sección, la próxima sección describirá la interacción de los componentes de software para los casos de envío y recepción de mensajes.

7.3.3. Envío de mensajes

El envío de mensajes se basa en tres pasos fundamentales:

1. Recopilación de información
2. Codificación del mensaje (que se delega en la subclase de **Codec** que corresponda)
3. Envío del mensaje (que se delega en la clase **Transporter**)

Recopilación de información

La recopilación de información depende del tipo de mensaje que deseamos enviar:

Documentos Si se desea enviar una URL, simplemente se crea una instancia de **URLMessage** y se completa el atributo **reference** con la URL deseada. Si por el contrario es un archivo, entonces se tiene pasar por el proceso de conversión primero ver Sección 7.4.

Mensajes con referencias En el caso de los mensajes existen dos elementos a tener en cuenta:

El contenido del mensaje El contenido del mensaje es texto HTML. Dicho texto puede ser directamente obtenido de la componente **JEditorPane**. Mediante la utilización de una barra de herramientas, se pueden cambiar los atributos del texto de la componente mencionada anteriormente y luego obtener su representación en HTML. Así esa información formará parte de un **MarkMessage**.

La referencia Como se ha mencionado, un **MarkMessage** tiene una referencia a **Mark** que representa la marca dentro del documento. Mediante el uso de una componente especialmente diseñada (**MarkableEditorPane** ver Sección 7.3.5) se puede realizar una marca sobre un documento y obtener su representación.

Envío del mensaje

Luego de recopilar la información (contenido del mensaje y referencia) la instancia de **Groupchat** recibe el mensaje

sendGroupchat<tipo>MarkMessage(GroupchatMarkMessage message)

, dónde *<tipo>* indica el tipo referencia del mensaje a enviar. Dicho mensaje es delegado en el estado del Groupchat (representado por la clase **GroupchatState**) que codifica el mensaje en un paquete (ver Codificación del mensaje) y recupera la instancia de **Transporter** para enviar el mensaje llamando al método *send(Packet packet).*

Codificación del mensaje

Cada tipo de mensaje es representado por una clase concreta diferente. Cada clase consta de un **Codec** representado por la clase **Groupchat <tipo>MarkMessageCodec** dónde *<tipo>* depende de la referencia que se está tratando.

Codec encapsula los procesos de codificación y decodificación de Mensajes a Paquetes y viceversa. Para poder codificar un mensaje basta con invocar el método *Packet codify(Object entity)* de la clase indicada, pasando como parámetro el mensaje y devolverá el paquete que lo representa.

7.3.4. Recepción de mensajes y documentos

Esta sección describirá la implementación de la recepción de mensajes y documentos.

La recepción de información funciona a través de la capa de transporte, detallada en la sección 7.3.1.

Para recibir datos se registra una instancia de la clase **PacketListener** en **Transporter**. Cuando un **Packet** es recibido por el cliente todos los **PacketListener** registrados son notificados.

Cada **Packet** tiene asignado un **PacketListener** específico que se encarga de ejecutar las acciones correspondientes a ese **Packet**. De esta forma, para procesar mensajes o documentos se utiliza un **PacketListener** determinado, denominado **PacketParser**.

PacketParser utiliza el mismo principio que **PacketListener** y delega en las clases que implementan **PacketProcessor** la responsabilidad de procesar los **Packet** relacionados a cada tipo de mensaje. De esta forma **PacketProcessor** se encarga de procesar el **Packet** relacionado con la llegada de mensajes al cliente y ejecutar la acción correcta en el **PapaloteGroupchatClientApp** (encargado de modificar el modelo de la aplicación y notificar a

la interfaz el cambio).

Cada **PacketProcessor** ejecutará un método relacionado con una acción ligada al **Packet**. Esta acción se traduce en la implementación del patrón de diseño Action[2]. Los diagramas de la Figura 7.8 muestran la implementación estructural de los **PacketListeners** y los **PacketProcessor**.

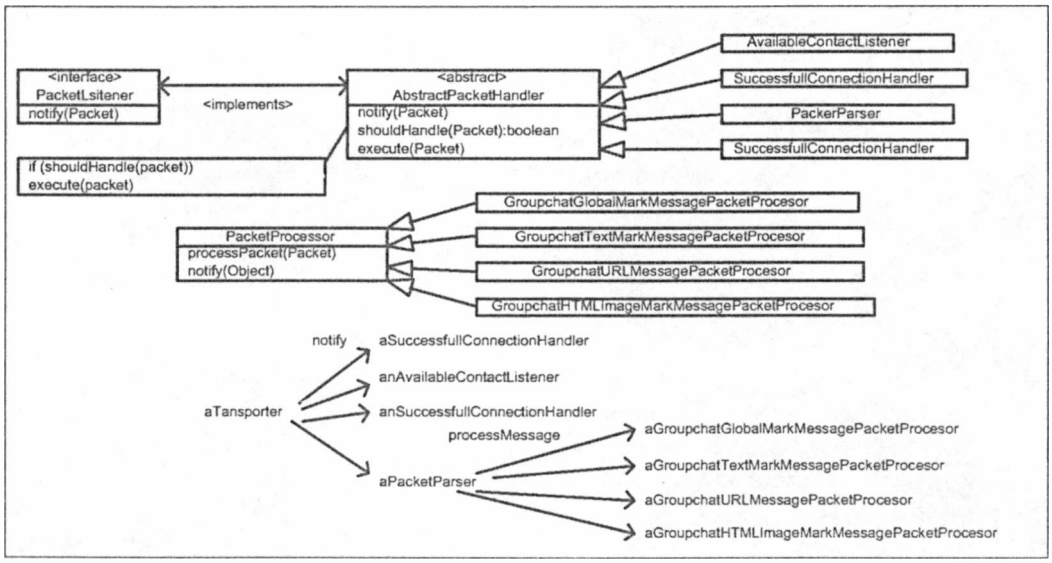


Figura 7.8: Diagrama estructural de recepción

Cuando un **Packet** llega al **PacketParser** notifica a los **PacketProcessor**(mediante el método *processPacket* que recibe como parámetro un **Packet**) la llegada de un nuevo paquete. Este método es el cargado de decidir si el paquete corresponde con el procesamiento de la acción que representa el **PacketProcessor**. De ser así, decodifica el **Packet** y notifica al **PapaloteGroupchatApp** sobre la acción a ejecutar y como parámetro pasa el objeto resultante de la decodificación del **Packet**.

Por ejemplo, si el **Packet** que recibe el **PacketParser** está relacionado con la llegada de un documento al Chat, se ejecutará el *urlMessageReceived(URLMessage)* en **PapaloteGroupchatClientApp**.

Para poder decodificar el **Packet** el **PacketProcessor** utiliza un **Codec**. **Codec** se encarga de:

1. Codificar un **Object** en **Packet**.
2. Decodificar un **Packet** en el **Object** asociado.
3. Permite saber si un determinado **Packet** puede ser decodificado por el **Codec**.

Por lo tanto existirá un **Codec** asociado a cada tipo de mensaje.

Como cada **PacketProcessor** tiene un Codec asociado; el **PacketProcessor** no solo actúa como observador en la aplicación del patrón de diseño Observer[2] en conjunto con la clase **PacketParser** que actúa como observado, sino que también forma parte del patrón de diseño Template[2] cuyo método "template" sería *processPacket* que puede escribirse en pseudo código de la siguiente forma:

```
if(codec.canProcess(packet)){
notify(codec.decode(Packet))
}
```

De esta forma, cada subclase de **PacketProcessor** consultará a su **Codec** asociado si puede decodificar el **Packet** que ha llegado. En caso de ser posible, decodifica el **Packet** y ejecuta en **PapaloteGroupchatClientApp** la acción asociada.

La ventaja de utilizar un **PacketProcessor** en vez de un **PacketListener** se basa en la posibilidad de proveer distintos **Codec** para procesar la misma acción y de esa forma desacoplar la codificación y decodificación de las acciones a ejecutar. Además, para agregar nuevos tipos de mensajes, el impacto en la arquitectura es menor.

Para completar la explicación se mostrará un diagrama de clases (Figura 7.9) y de interacción (Figura 7.10) de la recepción de mensajes.

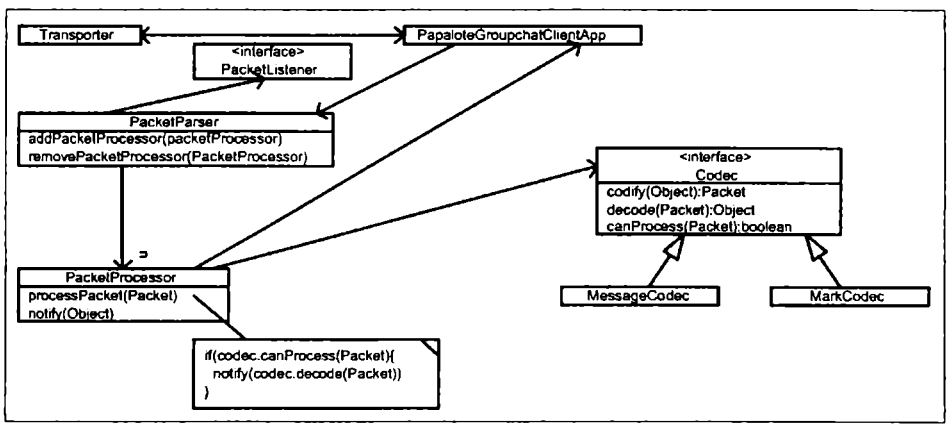


Figura 7.9: Diagrama de clases de recepción

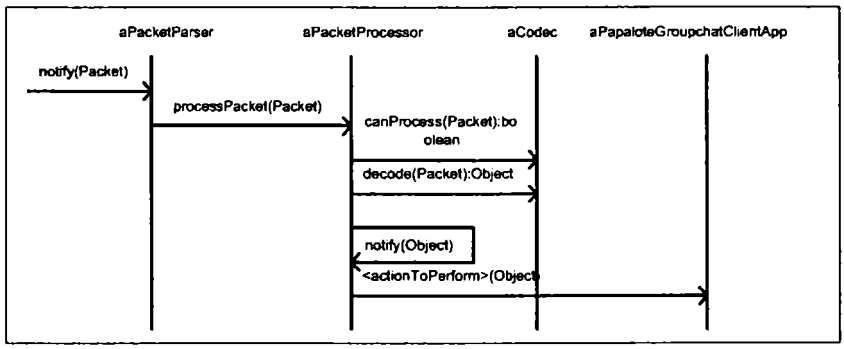


Figura 7.10: Diagrama de interacción de recepción

7.3.5. El MarkableEditorPane

Un documento HTML puede ser mostrado al usuario por medio de un **JEditorPane**. Para mostrar un documento con esta componente basta con la dirección del documento.

El **MarkableEditorPane** no sólo debe ser capaz de mostrar HTML sino también debe proveer los medios necesarios para poder realizar marcas sobre el documento en los distintos medios que muestra.

Existen tres estados distintos dentro del componente de marcado:

1. Marcar sobre el documento
2. Mostrar una marca
3. No mostrar marca

La diferencia entre los estados radica en la forma que se muestra una marca.

Los objetos responsables de mostrar las marcas son **GlobalMarkRenderer**, **TextMarkRenderer** y **HTMLImageMarkRenderer**. Estas clases cumplen funciones de "rendering" en el método *paintComponent(Graphics g)* que dibuja la marca.

Para que las marcas se puedan mostrar cuando se están creando, se necesita alguna forma de identificar la acción del usuario sobre el documento. Para ello se cuenta con clase **MarkCapturer** que se utiliza para identificar cuando el usuario ha marcado sobre el documento. Cada **MarkCapturer** cuenta con una subclase por medio de marca. Tiene la habilidad de reconocer si la acción efectuada por el usuario le corresponde o no. Por lo tanto para agregar un nuevo tipo de marca hay que codificar un **MarkCapturer**.

El **MarkingProcessor** indica al **MarkCapturer** la captura de la marca y se obtiene la mark.

Al producirse esta captura se notifica a todos los **MarkListener** registrados en la componente **JHTMLMarkablePane**. Para ello se invoca el método *marked* que toma como parámetro la marca efectuada. Aquí se vuelve a utilizar el patrón de diseño Observer[2] entre **JHTMLMarkablePane** (Observable) y **MarkListener** (Observer).

Otra acción a ser registrada es la de "marcando". Funciona de la misma manera que la anterior, pero en vez de notificar cuando una marca fue realizada, notifica cada vez que la marca se modifica mientras se está realizando. El método que se invoca es el *marking* que tiene como parámetro la marca actual.

Para visualizar la marca actual se utilizan los métodos *setMatk(Mark)* y *getMark():Mark*.

El diagrama de clases de la Figura 7.11 describe la estructura de la solución:

Para que el esquema sea flexible y pueda ser reutilizado o extendido se deben implementar de las siguientes interfaces: MarkCapturer, Mark y MarkRenderer. Estas interfaces definen como se obtiene la marca, cuál es la información que define la marca y cómo se muestra la marca respectivamente.

7.4. Conversión de documentos

Esta sección expone aspectos relacionados con etapa de conversión de documentos de algún formato determinado a HTML. Está compuesta de dos partes: servidor y cliente.

7.4.1. El servidor de conversión de documentos

Existen dos tipos de procesos de conversión. Varían de acuerdo al formato del archivo. Los formatos de archivos pueden dividirse entre los que necesitan la intervención de una aplicación Microsoft para poder ser convertidos y los que no.

Los formatos de archivos que no necesitan la intervención de una aplicación Microsoft son convertidos por la clase **HTMLConverter** en forma directa (ver Figura 7.13). Los formatos soportados que cumplen con esta condición son JPG, JPEG, PNG, GIF y TXT. Dichos documentos son embebidos directamente en HTML.

El caso de los formatos que dependen de una aplicación Microsoft para poder ser convertidos, como RTF, DOC, XLS y PPT utilizan un esquema

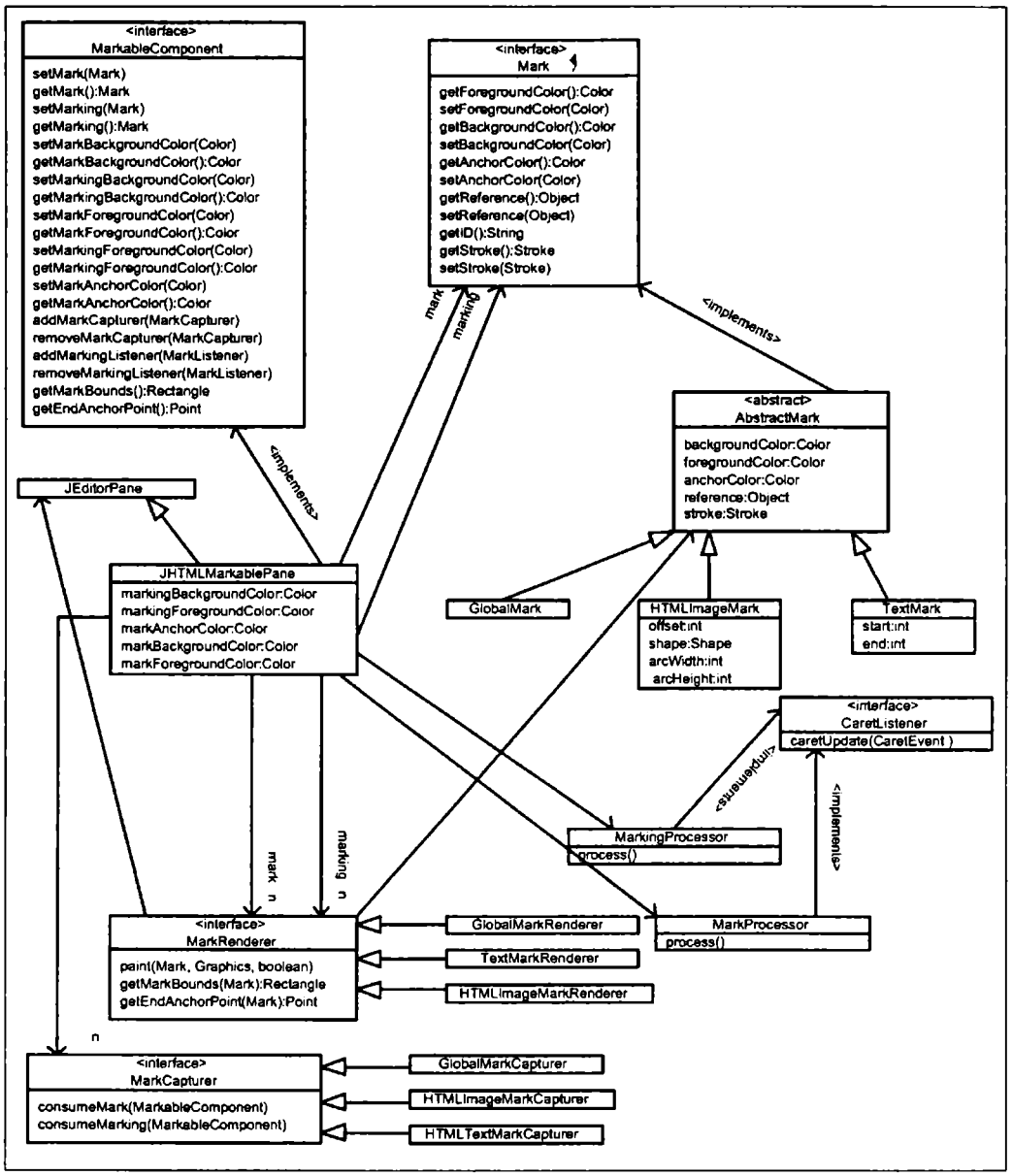


Figura 7.11: Diagrama de clases de la componente de marcado

diferente. La Figura 7.12 esquematiza las tecnologías que intervienen en la conversión.

Todo comienza a partir de una llamada al **Servlet** que se activa a partir de un requerimiento HTTP desde el cliente al servidor. Luego, se examina la extensión del archivo y así se reconoce su formato. Una vez realizado el

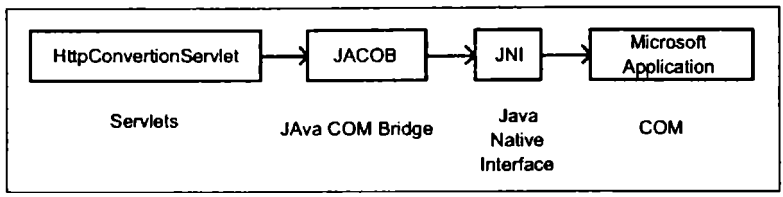


Figura 7.12: Esquema de tecnologías utilizadas en el proceso de conversión

reconocimiento del formato, se llama al procedimiento asociado al formato para:

1. Ejecutar la aplicación relacionada al formato para poder iniciar la conversión.

2. Abrir el archivo (previamente transportado del cliente al servidor en carácter temporal).

3. Establecer el formato de salida. En este caso HTML para versión 3.2 sin Javascript[38] que es el formato soportado por el **JEditorPane** de Java.

4. Guardar el archivo en una locación adecuada para su publicación. En nuestro caso, dentro de un contexto de ejecución de Apache Tomcat.

Para ejecutar las acciones anteriormente mencionadas se utiliza JACOB. JACOB permite hacer llamadas a componentes COM desde Java.

JACOB[39] significa *JAva COm Bridge* (en castellano, puente de aplicaciones COM[40] a Java). Es una forma de acceder a las aplicaciones que cumplen con la interfaz COM desde un ambiente JAVA. Para que las llamadas desde Java a los componentes COM se puedan efectuar, Java provee un mecanismo para ejecutar procedimientos fuera del entorno, denominado JNI. De esta forma se pueden "manejar" las aplicaciones Microsoft.

JNI[41] significa Java Native Interface (en castellano, interfaz nativa de Java). Permite la comunicación desde Java al exterior del ambiente. Se basa en llamadas a métodos que se definen en una librería dependiente del sistema operativo mediante una interfaz determinada.

La implementación de la conversión de documentos se basa en unas pocas clases. La Figura 7.13 representa el diagrama de clases de la estructura de solución implementada:

La clase **HttpConvertionServlet** es subclase de **HttpServlet**. De esa forma, el servidor Apache Tomcat hace una llamada al método *post* de la clase

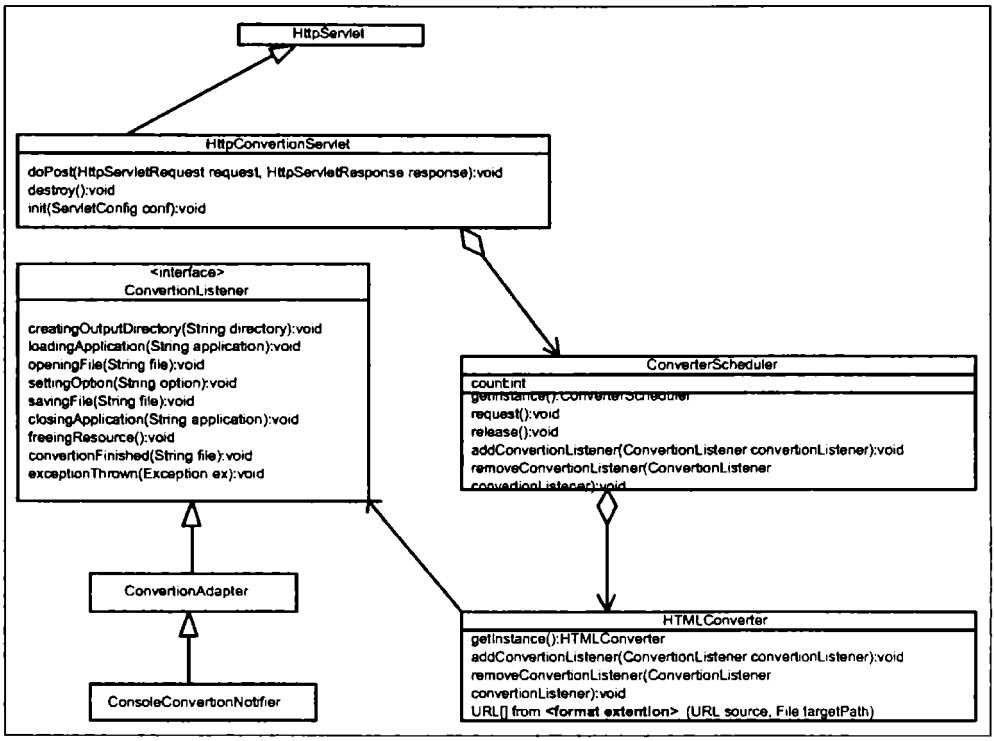


Figura 7.13: Diagrama de clases del proceso de conversión

HttpServlet cada vez que un cliente requiere el servicio que el **Servlet** implementa. Los dos parámetros **HttpServletRequest** y **HttpServletResponse** representan los parámetros que provienen de la llamada del cliente y la respuesta que será devuelta como resultado del servicio respectivamente.

Como consecuencia de la concurrencia en la ejecución de los servicios Web y la necesidad de poner en secuencia los pedidos a la conversión de archivos (ya que no es posible corren en paralelo la tarea) se necesita implementar un monitor que controle el acceso al conversor de archivos. El monitor es implementado en la clase **ConverterScheduler**. Esta clase implementa el patrón Singleton[2] que controla el acceso a **HTMLConverter**.

HTMLConverter es la clase que se encarga de convertir los archivos. Esta clase también implementa el patrón de diseño Singleton[2]. Provee un par de métodos de acceso a una colección de notificadores que implementan la interfaz **ConvertionListener**. Los objetos registrados por **HTMLConverter** que implementen esta interfaz serán los utilizados para mostrar el estado de la conversión de los documentos. Se observa la implementación del patrón de diseño Observer[2] cumpliendo del rol de observable se encuentra la

clase **HTMLConverter** y de observadores las clases que implementen **ConvertionListener**. Como consecuencia de este patrón se pueden implementar diferentes formas de notificar el progreso de la conversión de documentos. La única implementación que está disponible es la clase **ConsoleConvertionNotifier** que registra en la consola los eventos relacionados con el proceso de conversión.

HTMLConverter implementa un método por cada formato a convertir. Cada método toma como parámetro la URL del archivo origen y el directorio destino. Como resultado se obtiene un arreglo de URL que representan las direcciones que representan al documento en el nuevo formato.

El directorio destino está ubicado dentro del contexto en dónde corre el conversor y es accesible por todos los usuarios.

7.4.2. El cliente de conversión de documentos

La conversión de documentos desde el punto de vista del cliente se realiza en dos etapas:

1. Conversión de documentos y publicación de documentos
2. Propagación de las direcciones que contienen la información del documento

La conversión y publicación de documentos se resume en la llamada a un servicio Web configurado desde el cliente.

El cliente ejecuta una acción que envía los datos en formato HTTP bajo RFC2616[42] mediante el método POST[35] utilizando el formato MULTIPART para el transporte de los datos del archivo que se desea compartir. Como resultado obtiene una respuesta RESPONSE[36], HTTP bajo RFC2616 con una colección de URLs representando a los documentos del resultado de la convesión en HTML del documento original.

La propagación de las direcciones obtenidas en el paso anterior se realiza mediante el envío de una serie de mensajes URLMessage que contienen como atributo *reference* la URL que representan. De esa forma se notifica al resto de los participantes de la sesión que dicho documento está disponible.

Capítulo 8

Conclusiones

El objetivo del capítulo es destacar los puntos más importantes de la contribución de este trabajo en el área de comunicaciones que utilizan como medio un ordenador.

8.1. Comparación con trabajo relacionado

Esta sección realizará una comparación de la herramienta obtenida con respecto a las aplicaciones evaluadas en el Capítulo 4.

Respecto a los sistemas de Chat tradicionales tratados en la Sección 4.1 la aplicación añade tres características fundamentales:

1. Transporte de documentos para visualización.
2. Manipulación de documentos integrada al Chat.
3. La posibilidad de ligar mensajes a regiones de documentos (referencias).
4. Flexibilidad de ejecución (el sistema corre en Windows y Linux).

En relación a los sistemas para compartir escritorios se obtienen las siguientes ventajas:

1. Manipulación de documentos integrada al Chat con la posibilidad de ligar mensajes a regiones de dichos documentos (referencias).
2. Flexibilidad de ejecución (el sistema corre en Windows y Linux).
3. Posibilidad de manejo de documentos independientemente de las aplicaciones disponibles.

Si comparamos la aplicación con otros sistemas de Chat con referencias actuales, las mejoras son:

1. Manipulación de documentos integrada al Chat para todos los formatos requeridos.

2. Los mensajes tienen la posibilidad de hacer referencias a dichos documentos en distintos medios de información (texto, gráfico o mezcla de ambos).

3. Posibilidad de manejo de documentos independientemente de las aplicaciones disponibles.

4. Uso de servidores públicos y gratuitos.

5. Extensibilidad en función de los tipos de marcas y formatos de documentos con poco impacto en la arquitectura del sistema.

8.2. Resumen de conclusiones

Se ha implementado una herramienta capaz de compartir la visualización de documentos e intercambiar mensajes con referencias a regiones de dichos documentos.

Como se mencionó en la sección anterior, las ventajas obtenidas respecto a otras herramientas existentes en el mercado son muchas.

Mencionaremos ciertas características que posee esta herramienta a la hora de evaluar la comunicación por medio del intercambio de texto como modo de expresión.

Intercambio de documentos

Intuitivamente, un sistema de Chat con referencias permite agilizar la transmisión del "foco" de la conversación en una conversación. Además, al incluir menos prosa en la transmisión de información, podríamos decir que existiría una menor probabilidad de transmitir información errónea (malas interpretaciones, ambigüedades, etc.) en la identificación del foco de la conversación.

Como ejemplo de las características anteriormente expresadas, establecemos la situación:

"Dos personas discutirán entorno a un documento del cual existen dos versiones".

Es conocido que hay documentos que evolucionan a través del tiempo. Esta evolución se manifiesta en modificaciones. Un error común que se produce en conversaciones que tienen como foco un documento, es la posibilidad que diferentes personas se estén refiriendo a documentos de diferentes versiones.

Una consecuencia inmediata de obtener una versión común del documento es una posible disminución del tiempo en la duración de la conversación.

Visualización de documentos multiplataforma

Dada la arquitectura de la aplicación se pueden visualizar documentos en estaciones de trabajo en las cuales no se encuentra instalada la aplicación asociada al formato del documento. De esta forma se obtiene una visualización de documentos con ordenadores que corren en diferentes plataformas de Sistema Operativo (con las limitaciones que se muestran en los requerimientos). Por ejemplo, ver un archivo en formato Microsoft Word en un ordenador que corre con Linux y además poder lo usar como referencia en los mensajes que mande.

Intercambio de mensajes con referencias

El intercambio de referencias trae una cantidad bastante considerable de ventajas directas e indirectas.

En principio la posibilidad de ligar mensajes con regiones de documentos permite una expresión más clara y concreta del foco de la prosa escrita en un mensaje. Además, la expresión de una marca en texto (en modo de selección) o en gráfico (en modo de rectángulo) es posiblemente más rápida y menos ambigua que en los casos dónde se describen textualmente dichas marcas.

La posibilidad de expresión más rápida y menos ambigua presume un mejor flujo de información entre los usuarios.

De esta forma se contribuye a la mejora de las comunicaciones de intercambio de mensajes mediadas por computadoras.

8.3. Trabajo futuro

Existen muchas líneas de exploración sobre el trabajo presentado Entre las más interesantes, se encuentran:

1. Revisión de conversaciones off-line: Actualmente el sistema no posee forma de "recordar" las conversaciones y por supuesto tampoco de recordarlas. Esta característica podría ser de gran utilidad cuando una conversación puede llevar más de una sesión de duración.

2. Referencias a mensajes: Funcionalidad para evitar marcar más de una vez una zona sin necesidad. El sistema no permite hacer referencia a un mensaje, por lo tanto, es necesario "volver a marcar" la zona que tiene por foco un mensaje anterior relacionado.

3. Mejorar el JEditorPane: Esta componente Swing de Java es muy limitada y tiene muchas limitaciones. Esto atenta contra la calidad de visualización del documento generado.

4. Adición de formatos de documentos: Existen muchos formatos de documento que perfectamente pueden ser añadidos al sistema. Los formatos de los documentos Microsoft son fácilmente agregadas al sistema (escritura de un par de métodos en las clases de conversión) gracias a la interacción de Java con las interfaces COM. Una extensión muy útil sería los archivos VSD. generados por el programa Visio. Otro formato, aunque no es de la familia de productos Microsoft, muy útil serían los generados por las aplicaciones CAD como el Autocad. Autocad tiene interfaz COM así que podría utilizarse la misma solución que se utilizó con las aplicaciones Microsoft. Por último, los archivos en formato PDF son muy populares y serían una buena extensión al programa, sin duda.

5. Nuevos tipos de marcas: Sería interesante agregar al sistema diferentes formas de marcado. Por ejemplo, proveer una forma directa de selección de una tabla, enlace o lista. La extensión en este sentido no es compleja y podría ser de gran utilidad a la hora de agilizar las selecciones.

6. Carga directa de enlaces: Una alternativa interesante es la posibilidad de compartir un documento con sólo hacer clic en el enlace de otro documento. Un ejemplo de utilización podría ser el browsing de documentos Javadoc.

Apéndice A

JDK ver. 1.4.1 rev. 1

A.1. Instalación

Al intentar instalar el JDK, se realiza un chequeo de existencia de versiones instaladas.

En el caso que la misma versión de JDK esté instalada, el programa emite una confirmación para borrar y volver a instalar el mismo (ver Figura A.1).

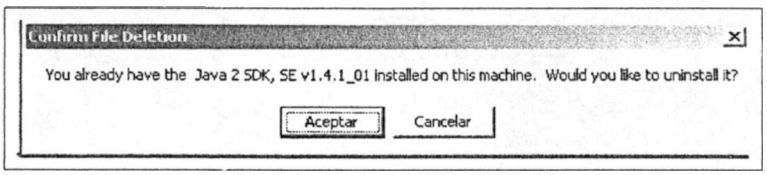


Figura A.1: Opción de re instalación

Hacer clic en **Cancel** para abortar la instalación.

En caso que no exista una versión adecuada permite instalarlo; mostrando la pantalla de la Figura A.2).

Hacer clic en **Next** para continuar con la instalación.

A continuación se deben aceptar los términos de la licencia (ver Figura A.3).

Hacer clic en **Next** para aceptar la licencia.

Luego se debe configurar el directorio de instalación (ver Figura A.4).

Para cambiar el directorio de instalación, clic en Browse. Usualmente se instala en el directorio por omisión.

Para continuar clic en **Next**.

Para seleccionar los componentes que se instalarán seleccionarlos en la pantalla de la Figura A.5. Usualmente, y por omisión, todos los componentes

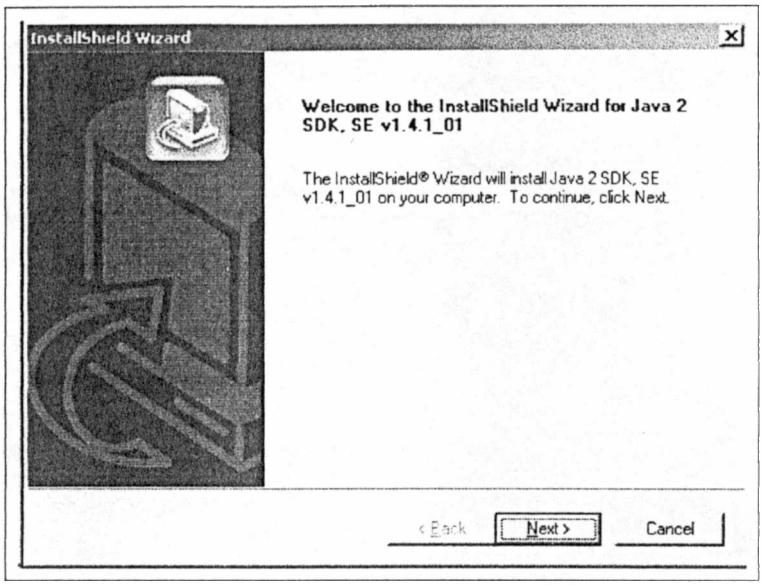


Figura A.2: Bienvenida a la instalación

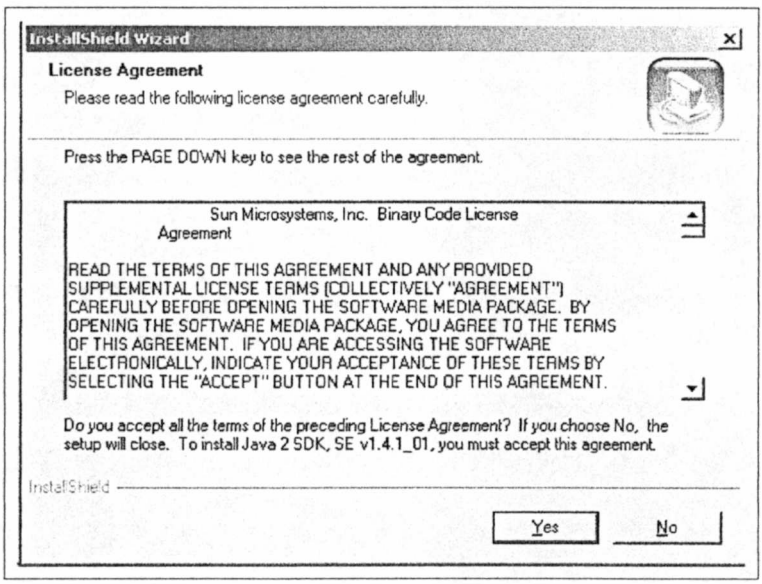


Figura A.3: Licencia de JDK

están seleccionados Sin embargo se pueden evitar algunos, como los archivos fuentes (última opción) y demostraciones (segunda opción).

Cuando la selección esté completa hacer clic en **Next**.

Luego se pregunta acerca de la integración con el explorador de Internet

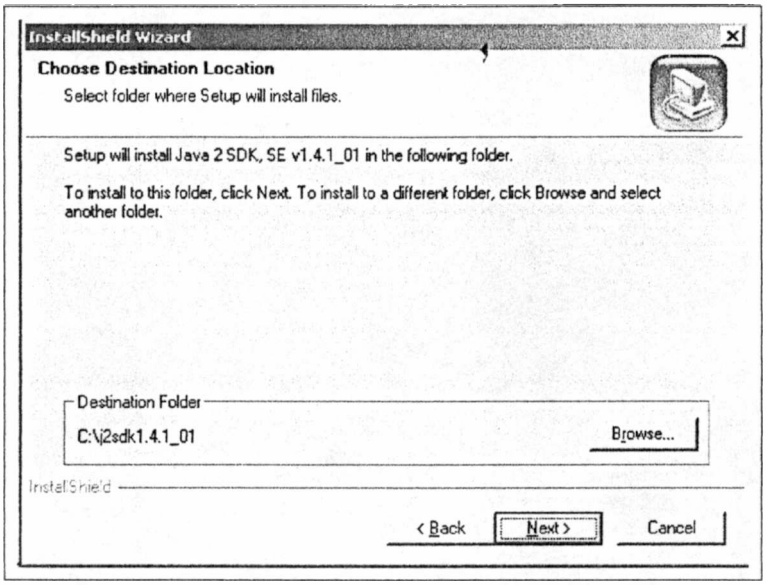


Figura A.4: Directorio de instalación

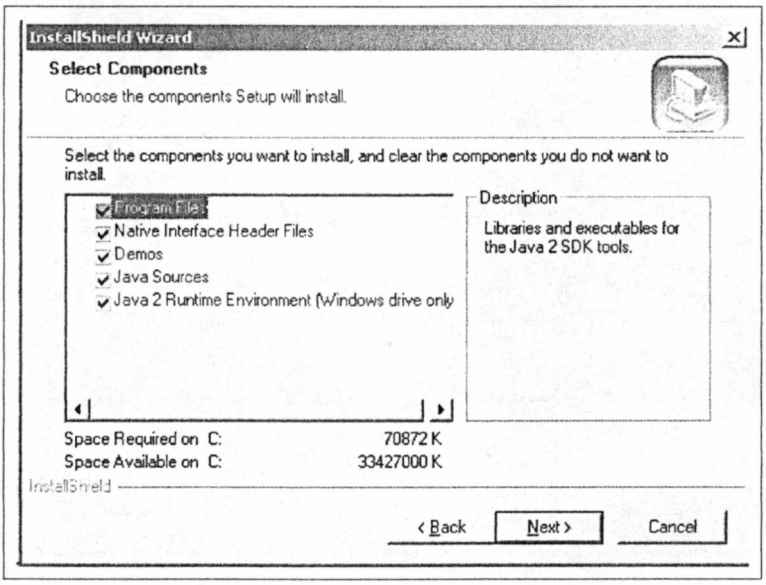


Figura A.5: Selección de componentes

(ver Figura A.6).

La integración no es necesaria. Pero la opción no interfiere el correcto funcionamiento del JDK, así que puede estar incluida en el explorador sin mayores inconvenientes.

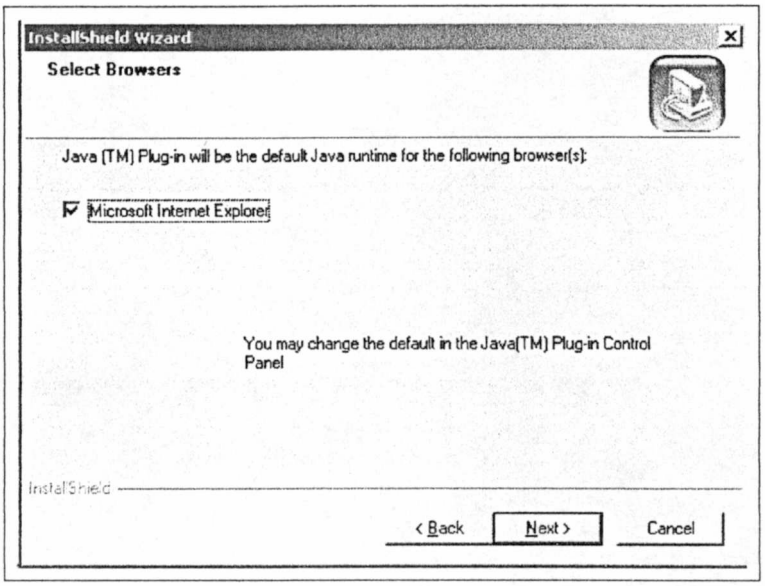


Figura A.6: Integración al explorador de Internet

Hacer clic en **Next** para continuar.

Al finalizar la instalación aparecerá esta ventana de notificación indicando el fin de la instalación del JDK (ver Figura A.7).

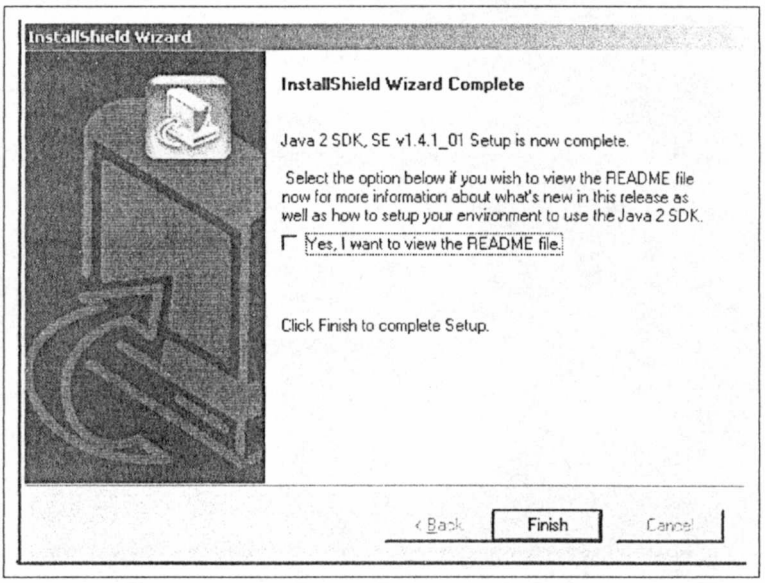


Figura A.7: Fin de la instalación

Nota: Cualquier duda dirigirse a la documentación de instalación de Sun.

Apéndice B

Apache Tomcat

El objetivo de este capítulo es proveer una guía rápida y visual para la instalación y configuración del servidor HTTP Apache Tomcat ver. 4.1 rev. 32.

Tomcat es la implementación del protocolo HTTP en el cual se basó el servico de conversión de archivos del desarrollo del trabajo de grado.

Los test cases incluidos en el "Sistema de conferencias con referencias" asumen la configuración que se introduce en este texto.

B.1. Instalación

El servidor tiene por pre-requisito la instalación del JDK ver. 1.4.1 rev. 1 (ver Capítulo A).

El programa de instalación chequea que el JDK esté instalado. Si lo está, aparecerá el diálogo de la Figura B.1 con el JDK seleccionado.

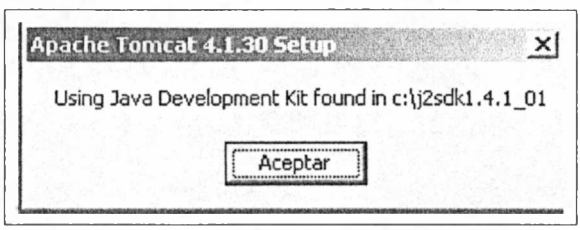


Figura B.1: Chequeo de JDK

Se debe verificar que sea la versión 1.4.1 ya que con otras versiones puede no funcionar. En el caso que tenga varias versiones de JDK instaladas en el sistema consultar la documentación.

En caso, contrario acusará error y no podrá ser instalado. Para instalarlo ver el Capítulo A.

Para continuar clic en **Aceptar**.

Luego hay que aceptar los términos de la licencia que se muestran en la Figura B.2.

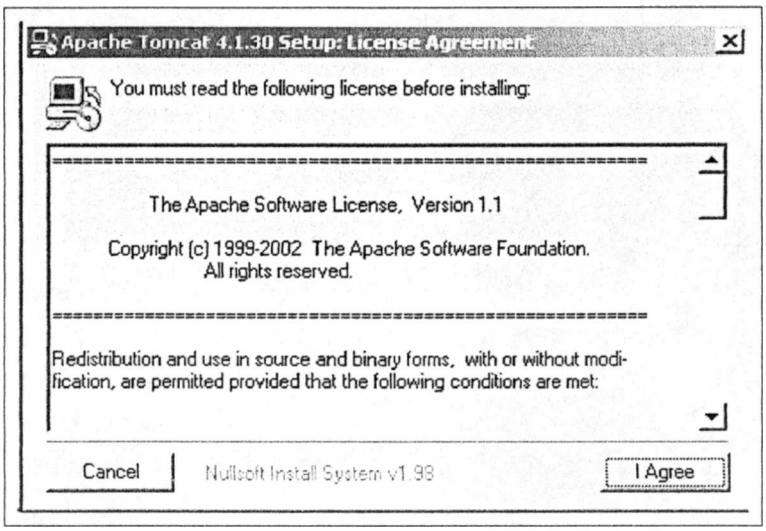


Figura B.2: Licencia Apache

Para aceptar clic en **I Agree**.

La selección de componentes queda a criterio del usuario.

La instalación típica es suficiente para la ejecución del servidor de conversión (ver Figura B.3).

Para más datos ver la documentación del producto.

Hacer clic en **Next** para continuar con la instalación.

La elección del directorio de instalación se realiza en este paso. Dicha elección queda a criterio del usuario. Las opciones por omisión son suficientes para que el producto funcione correctamente (ver Figura B.4).

Hacer clic en **Install** para continuar.

El último ítem a configurar es el acceso al servicio de administración del servidor vía Web (ver Figura B.5).

Para los fines de prueba es suficiente con los datos por omisión, que es el usuario admin y sin password.

Hacer clic en **Next** para continuar.

El último diálogo notifica que la instalación ha concluido (ver Figura B.6).

Hacer clic en **Close** para terminar.

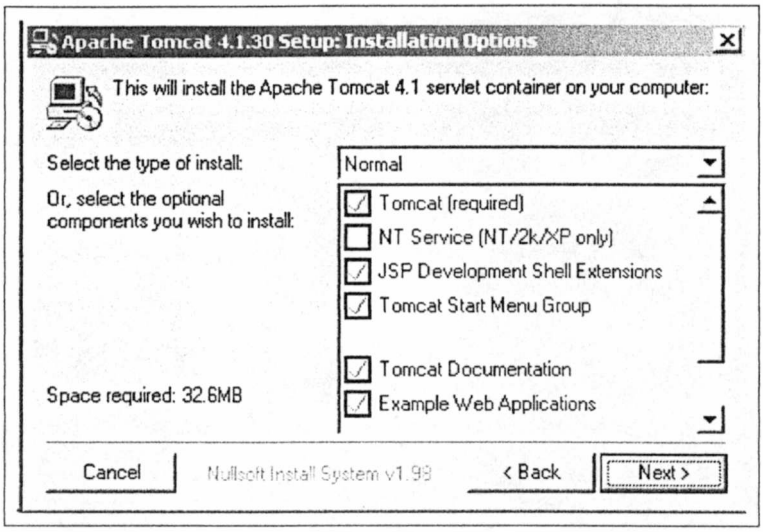


Figura B.3: Lista de componentes instalables

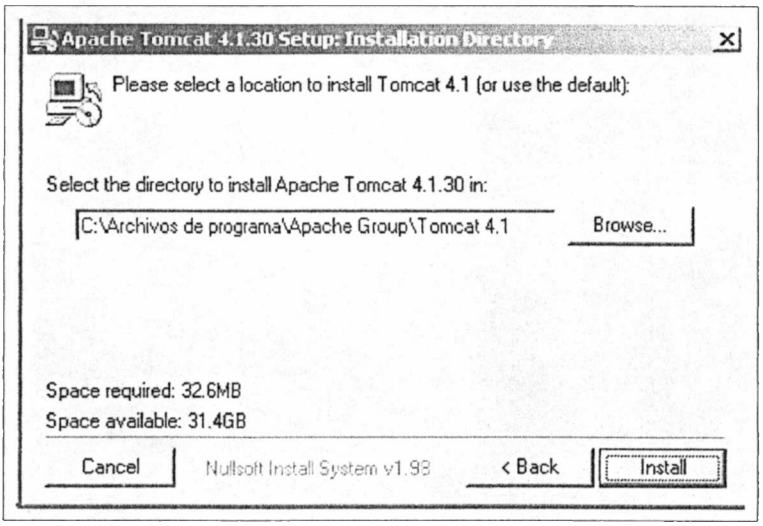


Figura B.4: Directorio de instalación

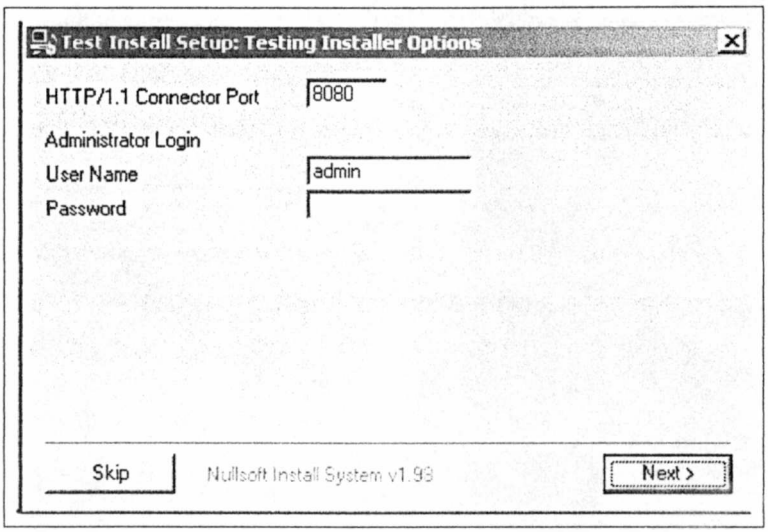


Figura B.5: Configuración de seguridad de administración

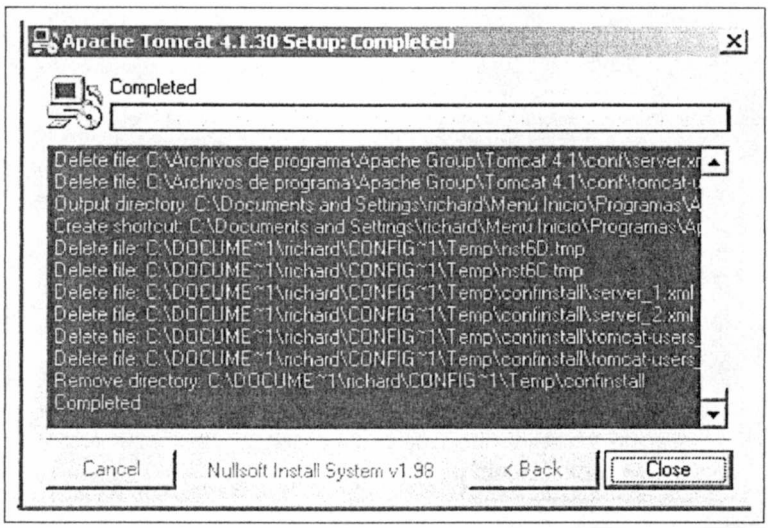


Figura B.6: Finalización de la instalación

Apéndice C

Jive Messenger ver. 1.0.8

C.1. Introducción

El objetivo de este capítulo es proveer una guía rápida y visual para la instalación y configuración del servidor JiveMessenger versión 1.0.8. JiveMessenger es la implementación del protocolo de mensajería instantánea Jabber en el cual se basó el desarrollo del trabajo de grado. Los test cases incluidos en el "Sistema de conferencias con referencias" asumen la configuración que se introduce en este texto. La configuración aquí expuesta está relacionada con la configuración del cliente (ver Sección E.1).

C.2. Instalación

La primera pantalla nos advierte de la instalación y nos da la bienvenida (ver Figura C.1).

Clic en **Next** para continuar.

La próxima pantalla nos muestra información acerca de la licencia del producto (ver Figura C.2).

Para aceptarla hacer clic en el botón de radio **I accept the terms of the License Agreement** y luego en **Next**.

Luego hay que configurar el JRE o JDK que se utilizará. Debemos escoger el JDK 1.4.1 como se indica en la Figura C.3.

Para aceptar la elección **Next**.

Luego se configura el directorio de instalación del Servidor Jabber (ver Figura C.4). Se recomienda el que se indica por omisión.

Hacer clic en **Next** para establecerlo.

Para acceder fácilmente al servidor se seleccionan los accesos directos al servidor (ver Figura C.5).

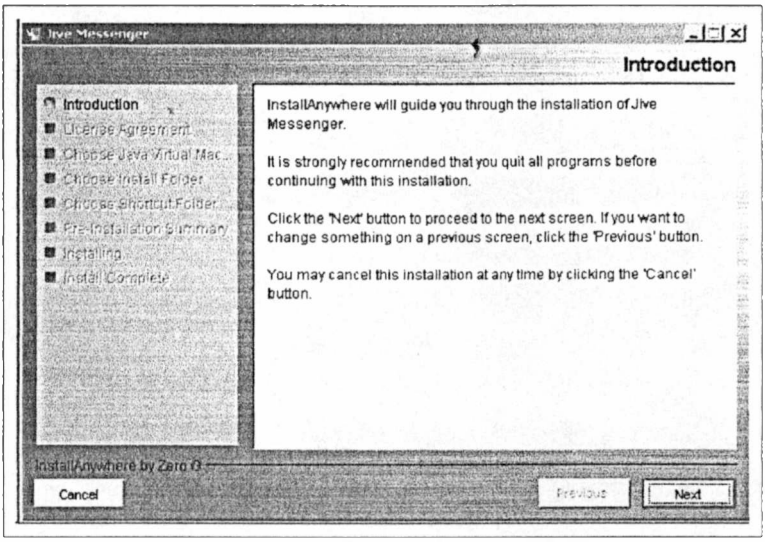


Figura C.1: Pantalla de bienvenida de instalación

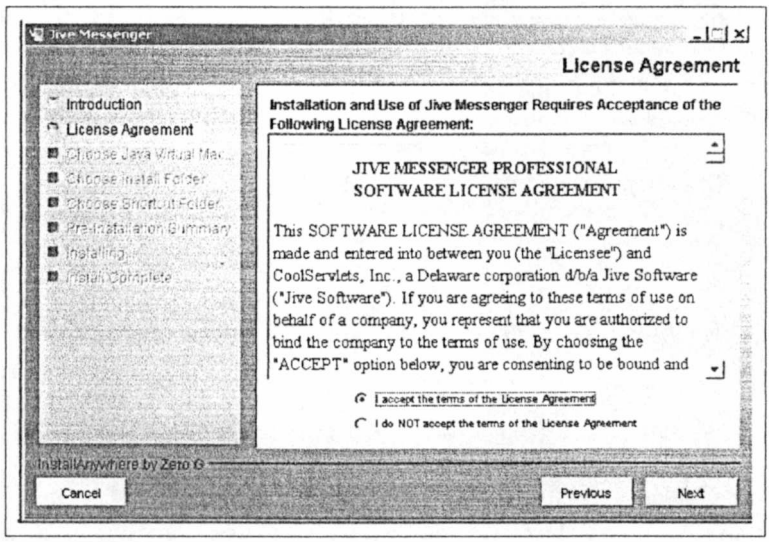


Figura C.2: Aceptación de licencia

Es suficiente con la opción indicada; sin embargo, se puede optar por otras formas de acceso.

Para continuar hacer clic en **Next**.

Se procede a la instalación mediante la confirmación de los datos en la pantalla de la Figura C.6.

Para aceptar los parámetros clic en **Install**.

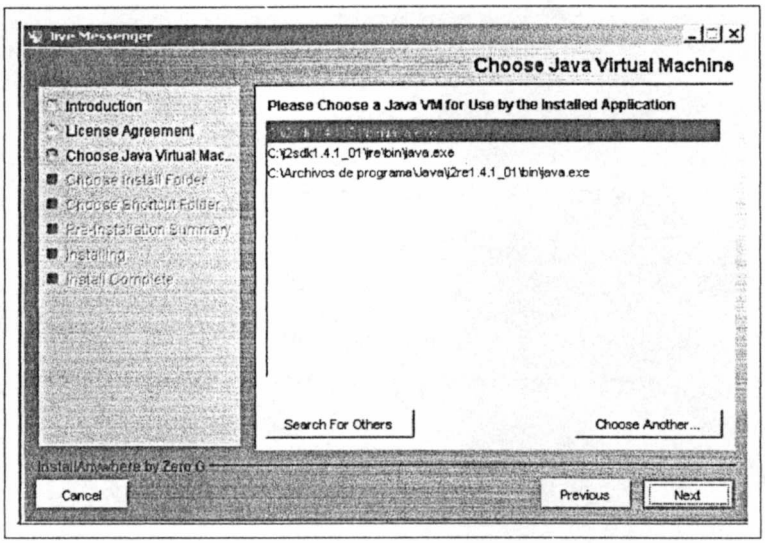


Figura C.3: Elección del JDK

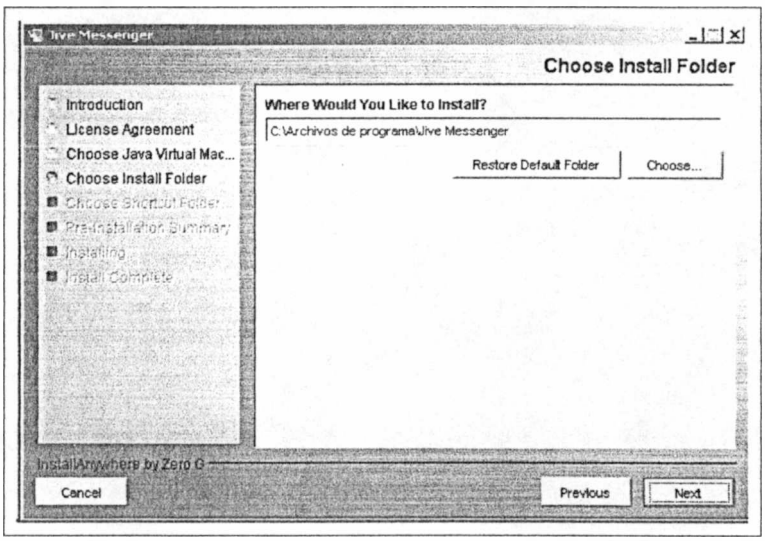


Figura C.4: Directorio de instalación

Notificación de instalación finalizada (ver Figura C.7).

Hacer clic en **Done** para continuar.

Con este paso culmina la instalación del servidor. Para poder utilizar el servidor correctamente hay que configurarlo. La configuración del servidor está descripta en el Capítulo D.

Nota: Cualquier duda dirigirse a la documentación de instalación de Jive

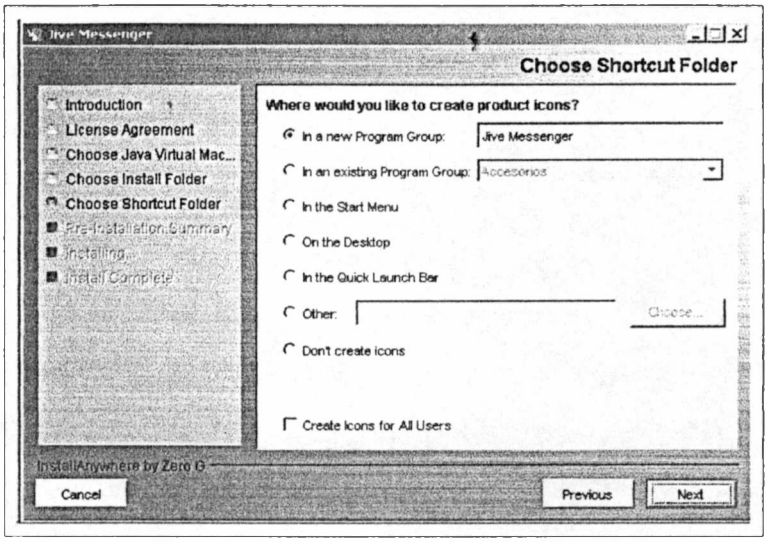


Figura C.5: Accesos directos

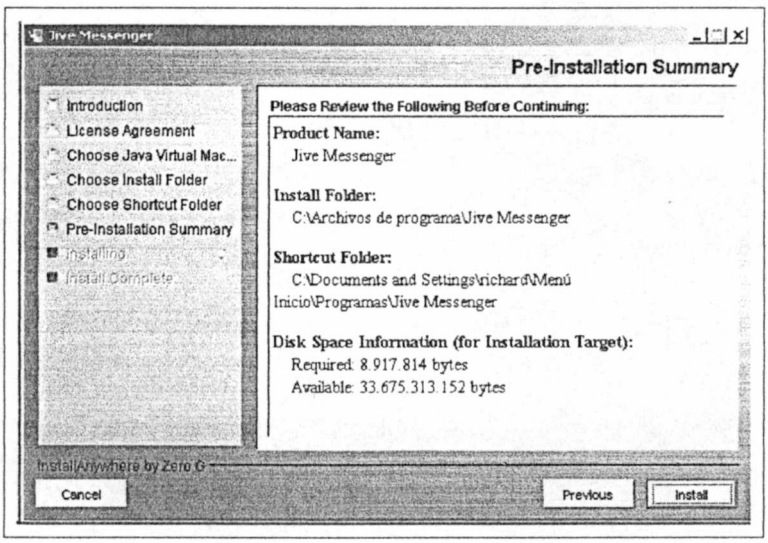


Figura C.6: Confirmación de parámetros

Messenger.

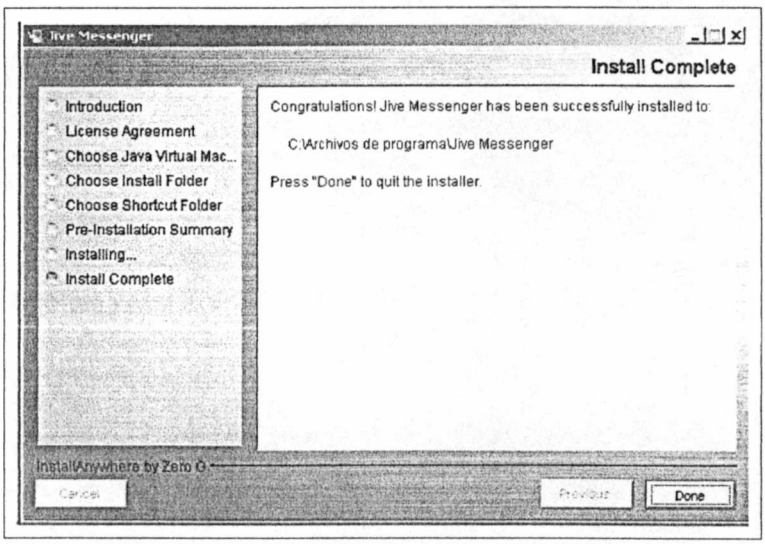


Figura C.7: Finalización

Apéndice D

Configuración de Jive Messager 1.0.8

Se describen los pasos a seguir en una instalación típica. Los parámetros que se utilizan son los indicados para la ejecución de los casos de prueba del servidor.

Para configurar el servidor hay que ejecutarlo. Se debe buscar el acceso directo en inicio como se indica en la Figura D.1.

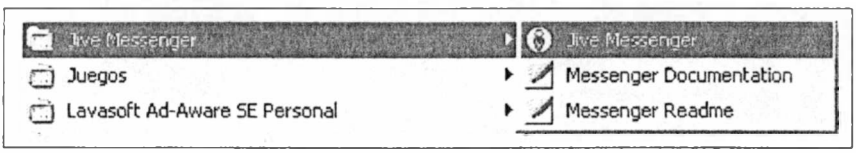


Figura D.1: Ejecución del servidor

Hacer clic en **JiveMessenger**.

Una vez que el servidor termina de cargarse, se habilitará el botón **Launch Admin** como en la Figura D.2.

Una vez habilitado, se hace clic y se abre el explorador de Internet predeterminado con la página para configurar el servidor.

El contenido de la página nos muestra una lista de los componentes y su estado de instalación (ver Figura D.3).

Hacer clic en **Continue**.

Éste es el paso más importante de la configuración. Por omisión los datos son establecidos mediante la configuración de red actual. Dichos datos **deben** ser tenidos en cuenta, ya que forman parte de la configuración del cliente.

El caso de ejemplo (ver Figura D.4 toma los siguientes datos:

- Nombre de la máquina / servidor: **sdf1**

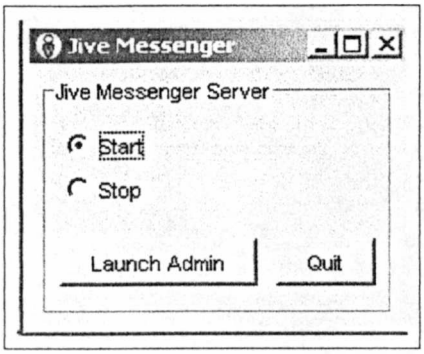


Figura D.2: Apertura del administrador del servidor

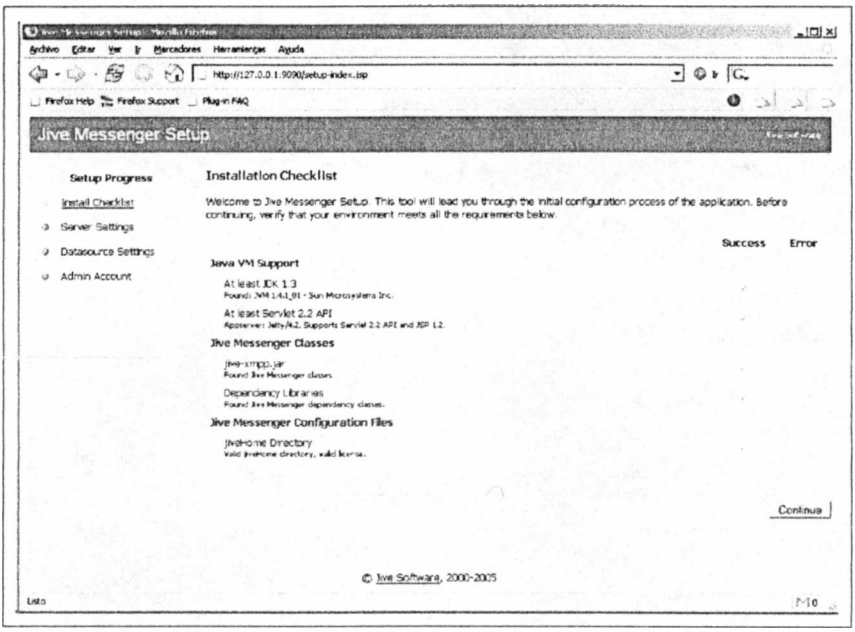


Figura D.3: Verificación de la instalación

- Nombre del servidor de Chat: **chat.sdf1**
- Puerto de Chat: **5222**
- Puerto Web (para el administrador): **9090**
- SSL: **Habilitado**.
- Puerto SSL: **5223**

Hacer clic en **Continue**.

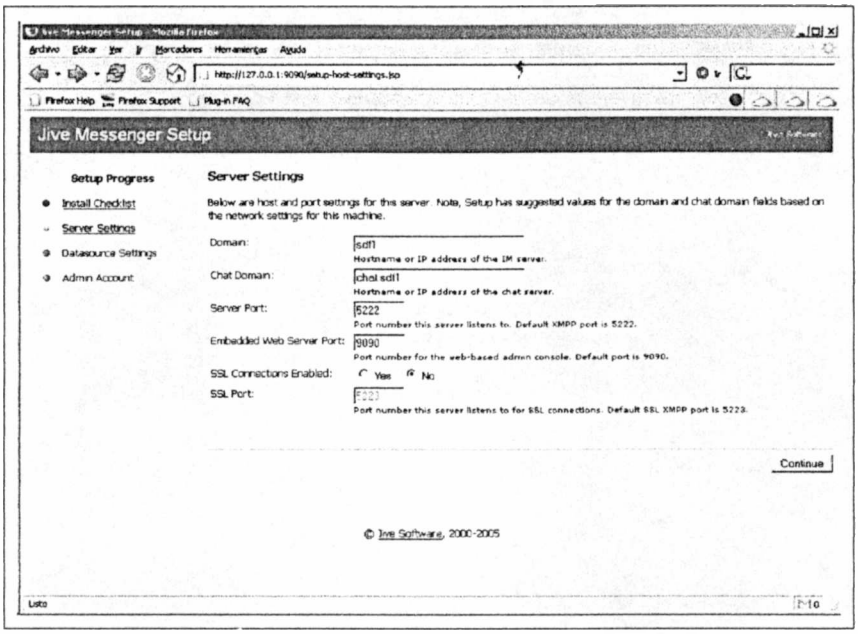


Figura D.4: Configuración del servidor

A continuación se selecciona la base de datos para el servidor. La configuración por omisión indica el uso de una base de datos externa; sin embargo, a modo de prueba iniciaremos el servicio con una base de datos embebida en el servidor (ver Figura D.5). La base de datos embebida no es muy eficiente, pero es suficiente para el trabajo que se le va a dar.

Hacer clic en **Continue**.

En este paso habría que configurar la administración de seguridad. Vamos a omitir dicha configuración para un acceso más fácil como se ve en la Figura D.6. De esta forma el password es *admin*.

Hacer clic en **Skip this step**.

Si la configuración fue exitosa aparecerá la pantalla de la Figura D.7.

Puede cerrar el explorador.

La configuración ha concluido.

Para modificaciones puede acceder desde el diálogo de la Figura D.1; introduciendo el password *admin*.

Nota: Cualquier duda dirigirse a la documentación de instalación de Jive Messenger.

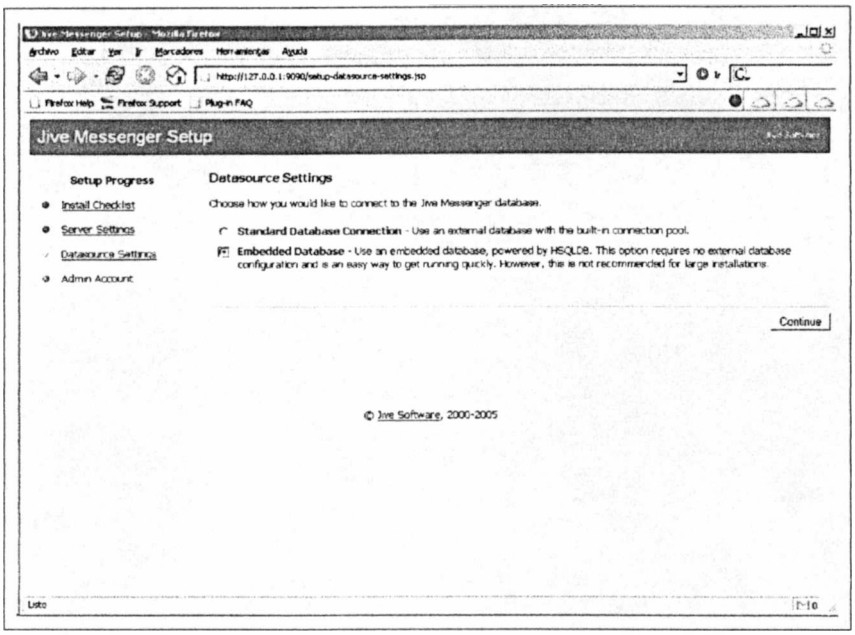


Figura D.5: Configuración de la base de datos

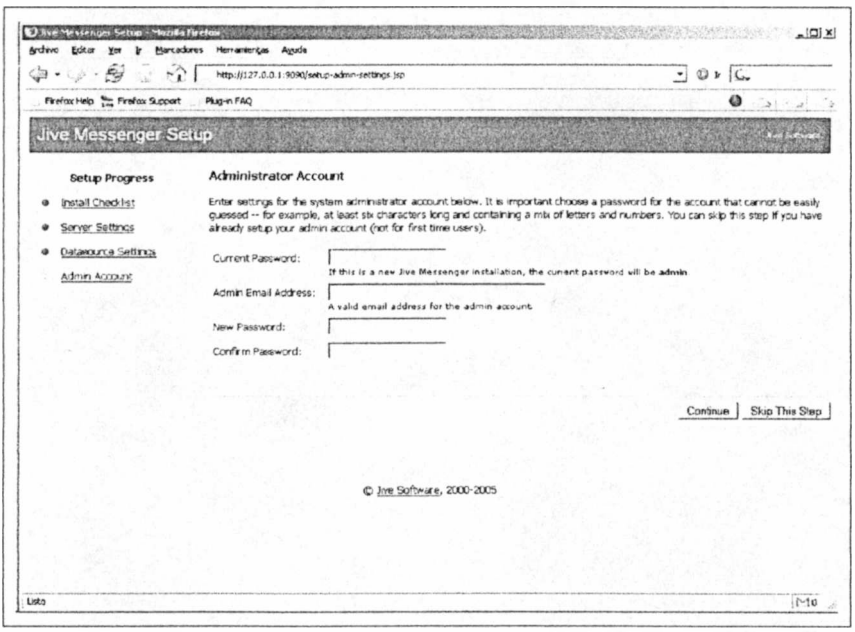


Figura D.6: Configuración de seguridad

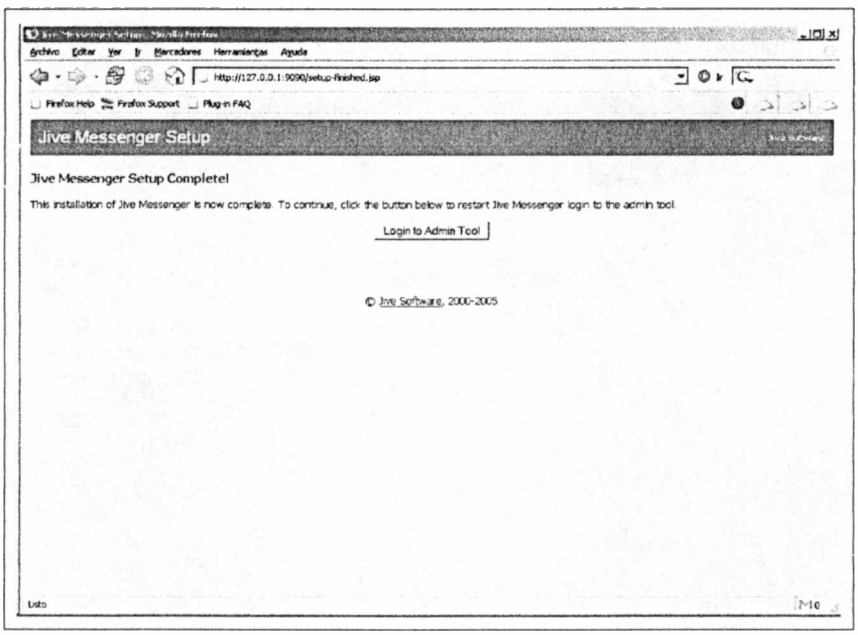


Figura D.7: Fin de configuración

Apéndice E

Configuración del cliente

E.1. Configuración del servidor Jabber

Al iniciar el cliente observamos la pantalla de la Figura E.1.

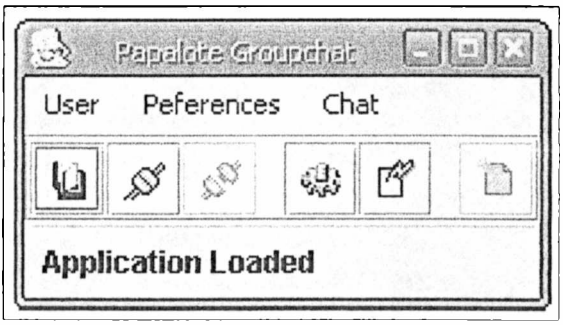


Figura E.1: Pantalla de bienvenida

Vamos a la opción indicada en la Figura E.2.

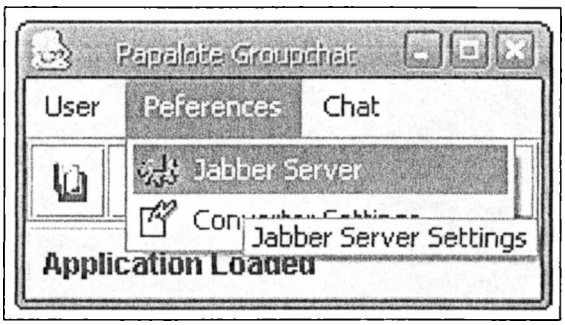


Figura E.2: Menú de configuración del servidor Jabber

Obtendremos la pantalla de la Figura E.3. Allí describiremos los datos de configuración del servidor Jabber.

User Describe el Jabber id del usuario para registrar o registrado en el servidor Jabber

Password Password para establecer (en el caso del registro del usuario) o utilizar (caso de conexión del usuario).

Nickname Nombre a utilizar en la sala de Chat

Server Dirección del servidor Jabber

Port Puerto de comunicaciones del servidor Jabber

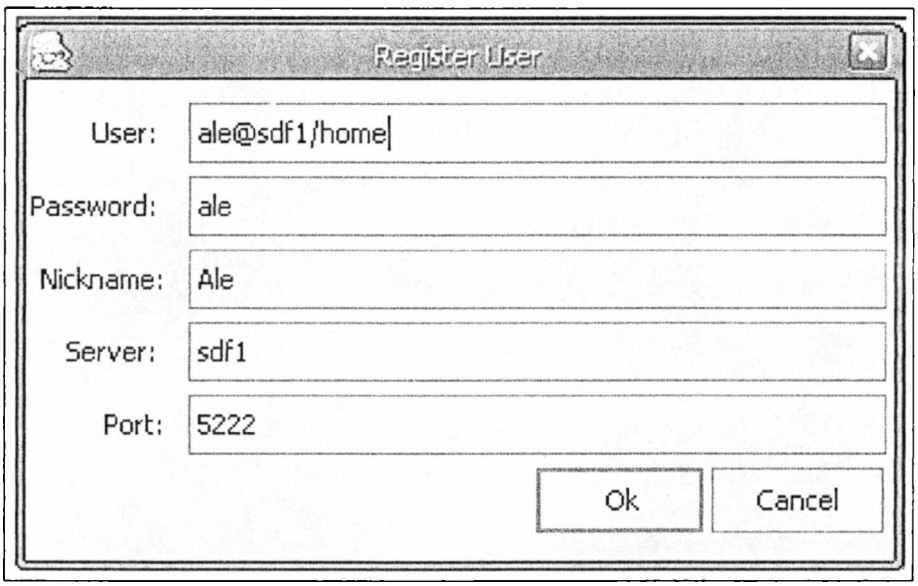


Figura E.3: Configuración del servidor Jabber

E.2. Configuración del servidor de conversión de documentos

Ahora es el turno de configurar el servidor de conversión de documentos. Para lograrlo hay que ejecutar la opción que se indica en la Figura E.4.

Al hacerlo obtendremos la pantalla de la Figura E.5. Allí se configuran:

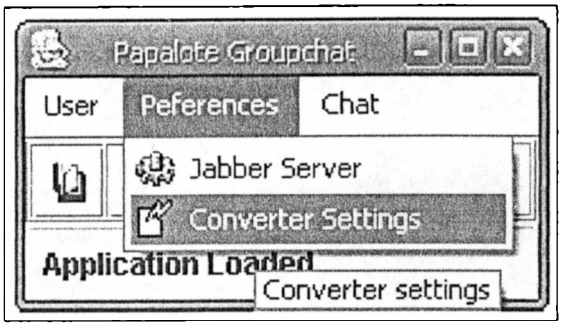


Figura E.4: Menú de configuración del servidor conversión de documentos

Server Dirección del servidor HTTP para conversión de documentos.

Port Puerto de comunicaciones del servidor HTTP para conversión de documentos.

Context Contexto de la aplicación de conversión. No es necesario alterarlo.

Service Servicio de la aplicación de conversión. No es necesario alterarlo.

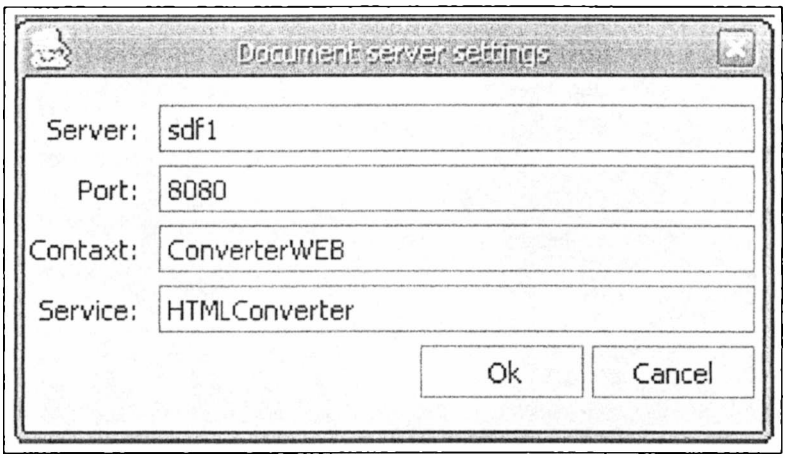


Figura E.5: Configuración del conversor de documentos

E.3. Procedimiento para entrar a una sala

E.3.1. Registro

Para poder acceder al servidor Jabber hay que registrarse. Para registrarse hay que configurar los servidores(indicado anteriormente) y hacer clic sobre el primer botón de izquierda a derecha (ver FiguraE.1).

Nota: Esta operación sólo es requerida por única vez.

E.3.2. Login

Para poder intercambiar mensajes hay que conectarse al servidor. Para ello, hay que configurar los datos con los cuales se registró en el servidor(como se indica en la sección anterior) y hacer clic en el segundo botón de izquierda a derecha(ver Figura E.1).

Nota: Para desconectarse del servidor hacer clic en el tercer botón de izquierda a derecha(ver Figura E.1).

E.3.3. Unirse a una sala

Para elegir una sala hay que hacer clic en el último botón, de izquierda a derecha(ver Figura E.1).

Aparecerá la pantalla de la Figura E.6:

Figura E.6: Lista de salas

Dentro de esa pantalla se pueden:

Agregar salas Mediante el botón Add.

Eliminar salas Para eliminar una sala hay que seleccionar la sala y presionar el botón Remove.

Unirse a una sala Para unirse a una sala hay que seleccionar la sala y presionar el botón Join o Enter.

Bibliografía

[1] Whittaker, S., Brennan, S., and Clark, H (1991). *Co-ordinating activity: an analysis of computer supported cooperative work.* In Proceedings of CHI'91 Human Factors in Computing Systems, New Orleans, USA, Eds., S. Robertson, J. Olson & G. Olson, NY: ACM Press, 360-367.

[2] Gamma, Eric; Helm, Richars; Johnson, Ralph; Vlissides, John *Design patterns. Elements of Reusable Object-Oriented Software.* Addison Wesley Publishing Company. ISBN 0-201-63361-2.

[3] Henri Ter Hofte, *Working Apart Together. Fundation for component groupware*, Telematica Instituut ISBN-90-75176-14-7.

[4] Johansen, R., *Current user approches to groupware*, En R.Johansen (ed.), Groupware: computer support for bussiness teams. Free Press New York, 1988,p. 12 - 44. Referencia de [3].

[5] RTF Specification.
http://www.biblioscape.com/rtf15_spec.htm.

[6] Unicode 3.0 specification.
http://www.unicode.org/.

[7] Gif Graphic Interchange Format.
http://www.w3.org/Graphics/GIF/spec-gif89a.txt.

[8] JPEG Graphic Interchange Format.
http://www.w3.org/Graphics/JPEG/

[9] PNG Prtable Network Graphic.
http://www.w3.org/TR/PNG/

[10] Microsoft Word.
http://office.microsoft.com/en-us/FX010857991033.aspx

[11] Microsoft Excel.
http://office.microsoft.com/en-us/FX010858001033.aspx

[12] Microsoft Power Point.
http://office.microsoft.com/en-us/FX010857971033.aspx

[13] HTML 3.2 Specification.
http://www.w3.org/TR/REC-html32

[14] TALK from Unix operating system.
http://unixhelp.ed.ac.uk/CGI/man-cgi?talk

[15] ICQ.
http://www.icq.com/

[16] MiRC.
http://www.mirces.com/

[17] IRC protocol specification.
http://www.rfc-es.org/rfc/rfc1459-es.txt

[18] AIM American Instant Messenger.
http://www.aim.com/

[19] Microsoft Messenger.
http://messenger.msn.com/

[20] Yahoo menssenger.
http://messenger.yahoo.com/

[21] Microsoft netmeeting
http://www.microsoft.com/windows/netmeeting/

[22] Churchill, Elizabeth F.; Trevor, Jonathan; Bly, Sara; Nelson, Les; Cubranic, Davor). *AnchoredConversations: Chatting in the context of a document(1991).*

[23] ActiveX COM objects.
http://www.microsoft.com/com/default.mspx

[24] Hans-Rüdiger Pfister (GMD-IPSI). Psychological Aspects of Collaboration in Multimedia Environments. Dagstuhl Seminar 00241 "Multimedia for Multimedia: Learning & Teaching in the next decade". Dagstuhl - June 11-16, 2000.
www.ipsi.gmd.de/concert.

http://elara.tk.informatik.tu-darmstadt.de/Publications/2000/ct-dagstuhl-seminar-00241.pdf.

[25] Dr. Daniel A. Tietze, Jessica Rubart. DyCE - A Framework for Component-Based Groupware.
http://www.go4teams.com/papers/dyce_gt.pdf

[26] Jabber messaging protocol.
http://www.jabber.org/about/overview.shtml

[27] XML specification.
http://www.w3.org/XML/

[28] Shigeoka, Iain. *Instant Messaging in Java The Jabber Protocols.*

[29] Jive Messenger.
http://www.jivesoftware.com/products/messenger/

[30] Jakarta Apache Tomcat Web Server.
http://jakarta.apache.org/tomcat/

[31] JSR-000154 JavaTM Servlet 2.4 Specification.
http://www.jcp.org/aboutJava/communityprocess/final/jsr154/

[32] Java Sun implementation.
http://java.sun.com/

[33] Windows operating system.
http://www.microsoft.com/

[34]]Unix operating system.
http://www.unix.org/

[35] RFC 2616 POST HTTP 1.1 method.
http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

[36] RFC 2616 RESPONSE HTTP 1.1 method.
http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

[37] XERCES Java parser.
http://xml.apache.org/xerces-j/

[38] Javascript specification.
http://www.w3.org/TR/REC-html40/interact/scripts.html

[39] Jacob - JAva COm Bridge.
http://danadler.com/jacob/

[40] COM - Component Object Model.
http://www.microsoft.com/com/default.mspx

[41] JNI - Java Native interface
http://java.sun.com/j2se/1.3/docs/guide/jni/

[42] HTTP Protocol specification
http://www.w3.org/Protocols/rfc2616/rfc2616.html