UNIVERSIDAD EUROPEA MIGUEL DE CERVANTES

FACULTAD / ESCUELA

ESCUELA POLITÉCNICA SUPERIOR

TITULACIÓN: INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

PLATAFORMA P2P DE INTERCAMBIO DE FICHEROS PARA WINDOWS

AUTOR

NURIA SANTOS SÁNCHEZ

DIRECTOR

SUSANA MARCOS MARTÍN

VALLADOLID, SEPTIEMBRE DE 2011

AGRADECIMIENTOS

Con motivo de la finalización de la carrera universitaria y con ello la realización del proyecto, me gustaría dar las gracias a todas esas personas que siempre estuvieron a mi lado.

Especialmente al apoyo de mis padres, por facilitarme el camino para lograr esta meta y muchas otras.

También me gustaría agradecer el apoyo a los profesores de la Universidad Europea Miguel de Cervantes, que me han hecho posible alcanzar un nivel de conocimientos técnicos necesarios para la realización del proyecto y en especial a Susana Marcos por confiar en mí para el desarrollo de dicho proyecto.

A todos ellos gracias.

RESUMEN

Los requerimientos de las grandes empresas y de los usuarios son cada vez mayores, y la tecnología, para satisfacer esta demanda, avanza a pasos agigantados día a día, como predijo, Gordon E. Moore, cofundador de *Intel*, en su famosa *Ley de Moore*, al expresar que “aproximadamente cada dos años se duplica el número de transistores en un computador”, cuyo cumplimiento se ha podido constatar hasta hoy, lo que es una clara muestra de este enorme desarrollo.

Pero aún con más intensidad que la potencia de los procesadores, está avanzando la capacidad de las redes de comunicación. Tanto es así, que si antes se consideraba como principal "cuello de botella" las redes de comunicación en el desarrollo de aplicaciones distribuidas o aplicaciones con comunicación a través de la red, ahora el problema empieza a ser los propios elementos físicos de la informática, el hardware.

George Gilder, director del *Media Lab* en Massachussets y reconocido tecnólogo, ya dejó entrever dicho crecimiento de la capacidad en las comunicaciones en su libro “Life after television”. Las aseveraciones de Gilder le llevaron a formular lo que se conoce como *Ley de Gilder* o *Ley de la Banda Ancha* que dice que “la capacidad de las comunicaciones se triplica cada doce meses”, y que es una muestra más del potencial de crecimiento de las TIC. Un claro ejemplo del planteamiento de Gilder que demuestra dicho potencial es que ya en 2001 era posible enviar más información por un solo cable de comunicaciones que toda la información que se enviaba por Internet en 1997.

Y si bien la necesidad de información de los usuarios de las redes de comunicación crece año a año, uno de los elementos determinantes para este espectacular crecimiento de las redes y su capacidad ha sido la popularización del software P2P, y la demanda de sus usuarios de capacidades de transmisión cada vez mayores.

Muestra de este increíble crecimiento es el crecimiento en España de las redes de comunicaciones. Cuando se empezó a popularizar el P2P gracias al famoso *Napster*, las conexiones a nivel usuario en España eran del orden de los 28 Kbps. Estas conexiones eran caras, muy inestables, y con una baja penetración a nivel doméstico. En 2007, siete años después, se ofrecían conexiones de 20 Mbps, del orden mil veces más veloces, y en el presente año 2011, de 50 Mbps, a pesar de las posiciones monopolísticas que han tenido algunas empresas suministradoras de Internet, que han dificultado una mayor competencia y han bloqueado desde hace años la capacidad de subida de datos en las líneas ADSL domésticas, impidiendo así un mayor crecimiento del que disponen otros países como por ejemplo Japón, con conexiones diez veces superiores a las nuestras, o incluso nuestro vecino Portugal, que ofrece conexiones de 1Gbps simétricos.

A pesar de ser una tecnología relativamente incipiente, el uso del P2P para compartir música, vídeo u otros archivos en formato digital ha popularizado tanto su uso que se estima que más de la mitad de ancho de banda consumido en el planeta lo consumen programas que hacen uso de dicha tecnología. Pero no solo se comparten archivos con contenidos audiovisuales gracias a esta tecnología, su desarrollo ha derivado en otros interesantes conceptos basados en P2P, como televisión a través de P2P, telefonía VOIP (el conocido *Skype*), compartir “ciclos de procesador” en aplicaciones distribuidas (*SETI@Home*),etc

Este proyecto pretende explorar las posibilidades que ofrecen el crecimiento de las redes de comunicaciones y la tecnología P2P mediante el desarrollo de una aplicación de ocio y entretenimiento que permita chatear, intercambiar archivos, visualizar fotos, escuchar música, reproducir vídeo y realizar descargas mediante P2P.

ÍNDICE

1. INTRODUCCIÓN 9

1.1. PEER-TO-PEER 9
1.2. FRIEND-TO-FRIEND 11
1.3. HISTORIA DE LAS APLICACIONES P2P 13
1.4. APLICACIONES DE LAS REDES P2P 17
1.5. CARACTERÍSTICAS DE LAS REDES P2P 21
1.6. CLASIFICACIÓN DE LAS REDES P2P 23
1.7. PROBLEMAS DE LAS REDES P2P 29
1.8. P4P, EL SIGUIENTE PASO EN EL DESARROLLO P2P 31

2. MODELO DE NEGOCIO 33

2.1. MODELO DE COMUNICACIÓN EN INTERNET 35
2.2. BARRERAS DE INTERNET: FIREWALL, NAT Y PROXY 37
2.3. PROTOCOLO DE COMUNICACIÓN EN REDES P2P 39
2.4. WINDOWS Y EL FUNCIONAMIENTO DE LAS P2P 41
2.5. SEGURIDAD 51
2.6. ANONIMATO 53
2.7. SITUACIÓN LEGAL DE LAS REDES P2P 55
2.8. SITUACIÓN LEGAL EN ESPAÑA 59

3. OBJETIVOS DEL PROYECTO 61

4. TECNOLOGÍAS EMPLEADAS 63

4.1. PROGRAMACIÓN 63
4.1.1. WINDOWS PRESENTATION FOUNDATION (WPF) 63
4.1.2. WINDOWS COMMUNICATION FOUNDATION (WCF) 65
4.1.3. C# 77
4.1.4. PLATAFORMA .NET 81
4.2. PLATAFORMAS DE DESARROLLO 85
4.2.1. GANTTPROJECT 85
4.2.2. VISUAL PARADIGM PARA UML 87
4.2.3. VISUAL STUDIO 2010 89

5. PROCESO DE DESARROLLO 91

5.1. METODOLOGÍAS ÁGILES DE DESARROLLO E INGENIERÍA DEL SOFTWARE (*MADIS*) 91
5.2. JUSTIFICACIÓN DEL PROCESO DE DESARROLLO ELEGIDO 97

6. METODOLOGÍA DE DESARROLLO 99

6.1. PLANIFICACIÓN INICIAL 99
6.1.1. DIAGRAMA DE GANTT INICIAL 99
6.2. CAPTURA DE REQUISITOS 101
6.2.1. REQUISITOS FUNCIONALES 103
6.2.2. REQUISITOS NO FUNCIONALES 107
6.3. PLANIFICACIÓN FINAL 109
6.3.1. DIAGRAMA DE GANTT FINAL 109
6.4. ANÁLISIS 111
6.4.1. ANÁLISIS DE REQUISITOS 113
6.4.2. DIAGRAMA DE PAQUETES 115
6.4.3. DIAGRAMA DE ACTORES 117
6.4.4. DIAGRAMAS DE CASOS DE USO 119
6.4.5. ESPECIFICACIÓN DE CASOS DE USO 125
6.4.6. DIAGRAMA DE CLASES EN LA FASE DE ANÁLISIS 137
6.4.7. DIAGRAMAS DE SECUENCIA EN LA FASE ANÁLISIS 139
6.5. DISEÑO 153
6.5.1. DISEÑO DE ARQUITECTURA BASADA EN EVENTOS 153
6.5.2. DIAGRAMAS DE CLASES EN LA FASE DE DISEÑO 157
6.5.3. DIAGRAMAS DE SECUENCIA EN LA FASE DE DISEÑO 165
6.5.4. DISEÑO DE INTERFAZ 179
6.6. PRUEBAS 183
6.6.1. PRUEBAS UNITARIAS 185
6.6.2. PRUEBAS DE INTEGRACIÓN 195

7. MANUALES 201

7.1. MANUAL DE INSTALACIÓN 201
7.2. MANUAL DE USUARIO 209

8. CONCLUSIONES 241

8.1. CONCLUSIONES 241
8.2. LÍNEAS FUTURAS 243

9. REFERENCIAS 245

9.1. BIBLIOGRÁFICAS 245
9.2. ELECTRÓNICAS 247

1. INTRODUCCIÓN

1.1. PEER-TO-PEER

Una red *peer-to-peer*, también llamada red de pares, red entre iguales, red entre pares o red punto a punto (*P2P*), es una red de computadoras en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos; son una serie de nodos que se comportan como iguales entre sí, es decir, actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red. Las redes P2P permiten el intercambio directo de información, en cualquier formato, entre los ordenadores interconectados.

El hecho de que sirvan para compartir e intercambiar información de forma directa entre dos o más usuarios ha propiciado que parte de los usuarios lo utilicen para enviar archivos cuyo contenido está sujeto a las leyes de copyright, lo que ha generado una gran polémica entre defensores y detractores de estos sistemas.

Las redes *peer-to-peer* emplean recursos distribuidos para desarrollar distintas funciones de una manera descentralizada, aprovechando, administrando y optimizando el uso del ancho de banda de los demás usuarios de la red por medio de la conectividad entre los mismos, y obteniendo así más rendimiento en las conexiones y transferencias que con algunos métodos centralizados convencionales, donde una cantidad relativamente pequeña de servidores provee el total del ancho de banda y recursos compartidos para un servicio o aplicación.

La eficacia de los nodos en el enlace y transmisión de datos puede variar según su configuración local (cortafuegos, NAT, ruteadores, etc.), velocidad de proceso, disponibilidad de ancho de banda de su conexión a la red y capacidad de almacenamiento en disco.

La computación P2P no es un concepto nuevo. Aunque es en estos últimos años cuando está adquiriendo popularidad, es posible encontrar algunas aplicaciones de esta filosofía en la historia de Internet. De esta forma, la red Internet originalmente concebida en la década de 1960 por encargo del Departamento de Defensa de los Estados Unidos (conocida como *ARPANET*[1]), era un sistema P2P que permitía el cálculo distribuido en cualquier lugar de Estados Unidos. El desafío consistía en integrar diferentes tipos de redes existentes así como permitir la integración de aquellas que pudieran surgir en un futuro.

Por otra parte, el sistema de nombres de dominio (*DNS*[2]) puede considerarse, en ciertos aspectos, un sistema P2P, pues la resolución de nombres es llevada a cabo de una manera distribuida por un número de máquinas cooperantes, resultando una fiabilidad y eficacia altas.

Muchos factores hacen al P2P muy práctico para una gran cantidad de aplicaciones. Entre estos factores se incluyen el incremento de dispositivos conectados a la red, el rápido incremento del ancho de banda, velocidad y potencia de las computadoras y el aumento de la capacidad de almacenamiento, sin olvidar, por supuesto, el incremento de la información disponible en la red.

P2P ofrece una alternativa a la arquitectura tradicional de Internet, el modelo cliente-servidor, que basa su efectividad en la existencia de un servidor que ofrece servicios a posibles clientes. Ambos modelos, sin embargo, se complementan y no suponen una alternativa entre ellos. La red del presente y del futuro tomará características de ambas arquitecturas y su evolución irá de la mano de la evolución de ambos modelos.

[1] Advanced Research Projects Agency Network - Red de Organismos de Proyectos de Investigación

[2] Domain Name System - Sistema de Nombres de Dominio

1.2. FRIEND-TO-FRIEND

El término *friend-to-friend* fue introducido por Dan Bricklin (creador, junto con Bob Frankston, del que es considerado como el primer programa de hoja de cálculo moderno: *VisiCalc*) en Agosto del 2000.

Una red de computadoras friend-to-friend (*F2F* o *Amigo a Amigo*) es un tipo particular de P2P anónimo en donde la gente se conecta directamente con sus "amigos". Los programas F2F solamente permiten personas en las cuales uno confía, usando direcciones IP o firmas digitales para hacer los intercambios de archivos. De esta manera los amigos de sus amigos (y así sucesivamente) podrán descargar indirectamente archivos de una persona de manera anónima.

En estas redes, solamente los amigos de un usuario pueden saber que su dirección está siendo usada para enviar archivos. Además estas redes permiten controlar qué tipo de ficheros pueden los “amigos” intercambiar con un nodo, evitando la posibilidad de intercambiar ficheros que el usuario desapruebe.

Una de las mayores ventajas de este tipo de redes es que pueden crecer en tamaño sin comprometer el anonimato del usuario.

Ejemplos de éste tipo de redes son: *Waste*, *Turtle F2F*, *ANts P2P*, *GNUnet*, *MUTE* y *Napshare*.

La aplicación desarrollada queda enmarcada dentro de este tipo de redes, ya que cada usuario establece un canal autentificado y seguro con un número limitado de nodos controlados por sus amigos, donde todas las transferencias funcionan sin un servidor central.

1.3. HISTORIA DE LAS APLICACIONES P2P

La historia de los sistemas P2P se remonta a finales de la década de 1970. El servicio de noticias de Internet o *Usenet* (1979) y el servicio de resolución de nombres de Internet o *DNS* (1983) son consideradas sus madres. No obstante, el P2P no llegaría a ser conocido por los usuarios hasta la aparición de *Napster* a mediados de 1999, a quien erróneamente se atribuye la invención del P2P.

La primera aplicación P2P, sin embargo, fue *Hotline Connect*, desarrollada en 1996 para el sistema operativo *Mac OS* por el joven programador australiano Adam Hinkley. *Hotline Connect*, distribuido por *Hotline Communications*, pretendía ser una plataforma de distribución de archivos destinada a empresas y universidades, pero no tardó en servir de intercambio de archivos de casi todo tipo, especialmente de contenido ilegal y en gran parte pornográfico. Sin embargo, también se podían compartir archivos de contenido libre de distribución.

El sistema *Hotline Connect* estaba descentralizado, puesto que no utilizaba servidores centrales, sino completamente autónomos: los archivos se almacenaban en los ordenadores de los usuarios que deseaban funcionar como servidores, y permitían, restringían o condicionaban la entrada al resto de usuarios, los clientes. En caso de que un servidor se cerrase, no existía ningún otro lugar del cual seguir descargando ese mismo archivo, y no quedaba más remedio que cancelar la descarga y empezar desde cero en otro servidor.

Este sistema bien planteado, en el que cada usuario dependía de un único servidor, no tardó en quedar obsoleto. Por otra parte, al ser una aplicación desarrollada fundamentalmente para una plataforma minoritaria como *Mac* no atrajo la atención de la prensa generalista.

No fue hasta mayo de 1999 cuando el uso de las redes P2P se hizo masivo.

Shawn Fanning y Sean Parker, estudiantes de la Northeastern University (Boston, E.U.A.), crearon entonces *Napster*, una red cuyo objetivo principal era el intercambio de archivos musicales. Aunque ya existían aplicaciones que permitían el intercambio de archivos entre los usuarios, como *IRC* o *Usenet*, *Napster* se presentó como la primera aplicación para PC especializada en los archivos de música mp3. El programa se hizo inmediatamente famoso entre los internautas, dado el gran abanico de posibilidades que proporcionaba.

La red se valía de un cliente (programa) que se podía descargar desde cualquier parte del mundo a través de la World Wide Web.

Aunque las transferencias de los archivos tenían lugar directamente entre dos equipos, *Napster* utilizaba servidores centrales para almacenar la lista de equipos y los archivos que proporcionaba cada uno, con lo que no era una aplicación perfectamente P2P.

El hecho de que *Napster* fuera un servicio centralizado resultó su perdición. En diciembre de 1999, varias discográficas estadounidenses demandaron a *Napster*, y también músicos reconocidos como Lars Ulrich, batería del grupo Metallica, reclamaron su cierre. La demanda, lejos de asustar a los usuarios, dio publicidad al servicio, de forma que en febrero de 2001 *Napster* había llegado a su cima con 13,6 millones de usuarios en todo el mundo.

Muchos argumentaron que el cierre de *Napster* sólo llevaría al surgimiento de otras aplicaciones similares de intercambio de archivos. El juez dictó el cierre de Napster en julio de 2001. Después de esa fecha, *Napster* se transformó en un servicio de pago, a costa de ser prácticamente olvidado por la comunidad internauta.

Durante un tiempo, el intercambio de archivos fue a la deriva. Existían ya bastantes alternativas. Al principio se seguía usando *Napster* mediante servidores no oficiales (usando *OpenNap*, por ejemplo) a los que se podía acceder gracias a un programa llamado *Napigator*. También surgieron programas como *Winmx* (cerrado en 2005 por amenazas de la *RIAA*[3]), e *iMesh*.

[3] Recording Industry Association of America - Asociación Estadounidense de Discográficas

Después se estableció como líder P2P *Audiogalaxy*, otra aplicación centralizada de intercambio, centrada en el contenido musical, que acabó siendo clausurada por orden judicial. La *RIAA* tomó estas resoluciones judiciales como victorias importantes encaminadas a acabar con la llamada piratería, y emprendió una cruzada para acabar con ésta, que continúa a día de hoy con escaso éxito.

Acabar con las redes centralizadas era relativamente sencillo, pues bastaba con cerrar el servidor que almacena las listas de usuarios y archivos compartidos. Pero tras el cierre de cada servidor surgieron otras aplicaciones más modernas, y particularmente como gran logro fue la creación de redes descentralizadas, que no dependen de un servidor central, y por tanto no tienen constancia de los archivos intercambiados.

Clientes nuevos y la aparición de la red *Gnutella*, fueron sustituyendo a *Napster* y *Audiogalaxy*, entre otros. Luego, en el año 2002, se dio un éxodo masivo de usuarios hacia las redes descentralizadas, como *Kazaa*, *Grokster*, *Piolet* y *Morpheus*. También aparecieron *Ares* y *Ares Lite*, libres de spyware y que usan la red *Ares Galaxy*.

La *RIAA* intentó, también por la vía judicial, acabar con los nuevos servicios descentralizados, y que permitían compartir varios tipos de archivos (no sólo mp3), pero Grokster y Morpheus ganaron sus juicios en abril de 2003.

Luego apareció *eDonkey 2000* (ya existía en el 2001 pero no era popular), aplicación que se mantuvo junto a *Kazaa* como líder del movimiento P2P. Más tarde, la aparición de otros clientes basados en el protocolo de *eDonkey 2000*, como *Lphant*, *Shareaza*, *eMule* y sus Mods, y otros menos conocidos como *aMule* y *MLDonkey* para Linux, causó el progresivo declive del programa original *eDonkey 2000*. Más bien se dejó de usar porque lo reemplazó el *eMule* y sus Mods.

Otro paso importante lo marcó el protocolo *BitTorrent*, que pese a tener muchas similitudes con *eDonkey 2000* proporciona, según los desarrolladores, una mayor velocidad de descarga, pero a costa de una menor variedad y longevidad de archivos en la red.

Otros importantes hitos han sido el uso de *Kademlia* y del *Webcaché*.

Los nuevos protocolos, cada vez más potentes, proporcionaban mayor velocidad de descarga, menor porcentaje de fallos, descentralización total (por ejemplo *Kademlia*, usado por *eMule*), e incluso anonimato. Pronto el uso de los P2P para descargar archivos se había convertido en una actividad cotidiana de la gente, al mismo tiempo que un verdadero quebradero de cabeza para la industria audiovisual.

1.4. APLICACIONES DE LAS REDES P2P

Actualmente, en Internet, el ancho de banda o las capacidades de almacenamiento y cómputo son recursos caros. En aquellas aplicaciones y servicios que requieran una enorme cantidad de recursos pueden ser muy eficaces las redes P2P.

Algunos ejemplos de aplicación de las redes P2P son los siguientes:

- Intercambio y búsqueda de ficheros. Quizás sea la aplicación más extendida de este tipo de redes. Algunos ejemplos son *BitTorrent* o la red *eDonkey2000*.
- Sistemas de ficheros distribuidos, como *CFS* o *Freenet*.
- Sistemas de telefonía por Internet, como *Skype*.
- A partir del año 2006, cada vez más compañías europeas y norteamericanas, como *Warner Bros* o la *BBC*, empezaron a ver el P2P como una alternativa a la distribución convencional de películas y programas de televisión, y ofrecen parte de sus contenidos a través de tecnologías como la de *BitTorrent*.
- Cálculos científicos que procesan enormes bases de datos, como los procedimientos bioinformáticos.
- Monedas virtuales para transacciones entre partes, como *BitCoin*.
- Las redes P2P pueden usarse también para hacer funcionar grandes sistemas de software diseñados para realizar pruebas que identifiquen la presencia de posibles fármacos. El primer sistema diseñado con tal propósito se desarrolló en el 2001, en el *Centro Computacional para el Descubrimiento de Medicamentos* (*Centre for Computational Drug Discovery*) en la prestigiosa Universidad de Oxford y con la cooperación de la *Fundación Nacional para la Investigación del Cáncer* (*National Foundation for Cancer Research*) de los Estados Unidos.

  Actualmente existen varios sistemas software similares que se desarrollan bajo el auspicio de proyectos como el proyecto de *Dispositivos Unidos en la Investigación del Cáncer* (*United Devices Cancer Research Project*).

- En una escala más pequeña, existen sistemas de administración autónoma para los biólogos computacionales, como el *Chinook*, que se unen para ejecutar y hacer comparaciones de datos bioinformáticos con los más de 25 diferentes servicios de análisis que ofrece. Uno de sus propósitos consiste en facilitar el intercambio de técnicas de análisis dentro de una comunidad local.
- Las instituciones académicas también han comenzado la experimentación con compartición de archivos, como es el caso de *LionShare*.

A continuación se muestran los protocolos de red y las aplicaciones multired más destacables.

⇨ REDES Y PROTOCOLOS

Red o Protocolo	Uso	Programas
ANts P2P	Intercambio de ficheros/Distribución de software/Distribución de multimedia	ANts P2P
Ares	Intercambio de ficheros	Ares Galaxy, Warez P2P, KCeasy, jAres P2P
BitTorrent	Intercambio de ficheros/Distribución de software/Distribución de multimedia	Ver Cliente BitTorrent para una lista detallada.
Direct Connect	Intercambio de ficheros, chat	DC++, NeoModus Direct Connect, SababaDC, BCDC++, RevConnect, fulDC, LDC++, CzDC, McDC++, DCDM++, DDC++, iDC++, IceDC++, Zion++, R2++, rmDC++, LinuxDC++, LanDC++, ApexDC++, StrongDC++
eDonkey	Intercambio de ficheros	aMule, eDonkey2000 (extinguido), eMule, eMule Plus, FlashGet, iMesh, Jubster, IMule, MLDonkey, Morpheus, Pruna, Shareaza, xMule
FastTrack	Intercambio de ficheros	giFT, Grokster, iMesh (y sus variantes como iMesh Light), Kazaa (y sus variantes como Kazaa Lite), KCeasy, Mammoth, MLDonkey, Poisoned
Freenet	Almacenamiento distribuido	Freenet, Entropy (red separada de Freenet)
GNUnet	Intercambio de ficheros, chat	GNUnet, (GNUnet-gtk)
Gnutella	Intercambio de ficheros	BearShare, Cabos, FilesWire, FrostWire, Gnucleus, Grokster, gtk-gnutella, iMesh, Kiwi Alpha, LimeWire (extinguido), MLDonkey, Morpheus, MP3 Rocket, Poisoned, Shareaza, Swapper, XoloX, KCEasy
Gnutella2	Intercambio de ficheros	Adagio, Gnucleus, Kiwi Alpha, MLDonkey, Morpheus, Shareaza, TrustyFiles
JXTA	Aplicaciones distribuidas	Collanos Workplace (Software colaborativo), Sixearch
Kad	Intercambio de ficheros	aMule, eMule, MLDonkey
Napster	Intercambio de ficheros	Napigator, Napster
OpenNap	Intercambio de ficheros	WinMX, Utatane, XNap, Napster
Osiris sps	creación de portales web anónimos	Osiris (Serverless Portal System)
P2PTV	Streaming de video	TVUPlayer, Joost, CoolStreaming, Cybersky-TV, TVants, PPLive, LiveStation,Sopcast
PDTP	Streaming e intercambio de ficheros	PDTP
Peercasting	Streaming	PeerCast, IceShare, FreeCast, Rawflow
Pichat	Chat e intercambio de información	Pichat, Pidgin, Moonchat, C4
Usenet	Grupos de noticias	
WPNP	Intercambio de ficheros	WinMX
Windows Peer-to-Peer	Desarrollo de aplicaciones distribuidas, colaboración	Incluido en el *Advanced Networking Pack* para Windows XP, Windows XP SP2, Windows Vista.

Figura 1: Redes y protocolos P2P

→ Otras redes: *Applejuice, Audiogalaxy, Avalanche, CAKE, Chord, The Circle, Coral, Dijjer, FileTopia, Groove, Hamachi, iFolder, konspire2b, Madster/Aimster, MUTE, NeoRouter, OpenFT, P-Grid, IRC, MojoNation, Mnet, Octoshape, Omemo, Overnet, Peersites, Perfect Dark, Scour, SharingZone, Skype, Solipsis, soribada, Soulseek, SPIN, Swarmcast, WASTE, Winny, Wippien.*

⇨ APLICACIONES MULTIRED

Aplicación	Red o Protocolo	Sistema operativo	Software libre
aMule	eDonkey, Kad	Multiplataforma	Sí
eMule	eDonkey, Kad	Windows	Sí
FilesWire	Gnutella, G3	Multiplataforma	No
giFT	eDonkey, FastTrack, Gnutella	Multiplataforma	Sí
Gnucleus	Gnutella, Gnutella2	Windows	Sí
iMesh	FastTrack, eDonkey, Gnutella, Gnutella2 (versiones anteriores a la 6.0)	Windows	No
KCeasy	Ares Galaxy, FastTrack, Gnutella, OpenFT	Windows	Sí
Kiwi Alpha	Gnutella, Gnutella2	Windows	No
MLDonkey	BitTorrent, Direct Connect, eDonkey, FastTrack, Kad, OpenNap, SoulSeek, HTTP/FTP	Multiplataforma	Sí
Morpheus	BitTorrent, Gnutella, Gnutella2	Windows	No
Shareaza	Gnutella, Gnutella2, eDonkey, BitTorrent, HTTP/FTP	Windows	No
Vagaa	BitTorrent, eDonkey, Kad	Windows	No
WinMX	WPNP, OpenNap	Windows	No
Zultrax	Gnutella, ZEPP	Windows	No

Figura 2: Aplicaciones multired.

1.5. CARACTERÍSTICAS DE LAS REDES P2P

A continuación se exponen algunas de las características más deseables de las redes P2P:

- ✓ **Escalabilidad**. Las redes P2P tienen un alcance mundial con cientos de millones de usuarios potenciales. En general, lo deseable es que cuantos más nodos estén conectados a una red P2P, mejor sea su funcionamiento. Así, cuando los nodos llegan y comparten sus propios recursos, los recursos totales del sistema aumentan. Esto es diferente en una arquitectura del modo servidor-cliente, con un sistema fijo de servidores, en los cuales la adición de clientes podría significar una transferencia de datos más lenta para todos los usuarios. Algunos autores advierten que, si proliferan mucho este tipo de redes cliente-servidor, podrían llegar a su fin, ya que a cada una de estas redes se conectarían muy pocos usuarios.
- ✓ **Robustez**. La naturaleza distribuida de las redes peer-to-peer también incrementa la robustez en caso de haber fallos en la réplica excesiva de los datos hacia múltiples destinos, y - en sistemas P2P puros - permitiendo a los peers encontrar la información sin hacer peticiones a ningún servidor centralizado de indexado. En el último caso, no hay ningún punto singular de falla en el sistema.
- ✓ **Descentralización**. Estas redes por definición son descentralizadas y todos los nodos son iguales. No existen nodos con funciones especiales, y por tanto ningún nodo es imprescindible para el funcionamiento de la red. En realidad, algunas redes comúnmente llamadas P2P no cumplen esta característica, como *Napster*, *eDonkey* o *BitTorrent*.
- ✓ **Distribución de costes entre los usuarios**. Se comparten o donan recursos a cambio de recursos. Según la aplicación de la red, los recursos pueden ser archivos, ancho de banda, ciclos de proceso o almacenamiento de disco.

- ✓ **Anonimato**. Es deseable que en estas redes quede anónimo el autor de un contenido, el editor, el lector, el servidor que lo alberga y la petición para encontrarlo, siempre que así lo necesiten los usuarios. Muchas veces el derecho al anonimato y los derechos de autor son incompatibles entre sí, y la industria propone mecanismos como el *DRM*[4] para limitar ambos.
- ✓ **Seguridad**. Es una de las características deseables de las redes P2P menos implementada. Los objetivos de un P2P seguro serían identificar y evitar los nodos maliciosos, evitar el contenido infectado, evitar el espionaje de las comunicaciones entre nodos, creación de grupos seguros de nodos dentro de la red, protección de los recursos de la red, etc. La mayor parte de las técnicas aún están bajo investigación, pero los mecanismos más prometedores son: cifrado multiclave, cajas de arena, gestión de derechos de autor (la industria define qué puede hacer el usuario, por ejemplo, la segunda vez que se oye la canción se apaga), reputación (permitir acceso sólo a los conocidos), comunicaciones seguras, comentarios sobre los ficheros, etc.

[4] Digital Rights Management - Gestión de Derechos Digitales

1.6. CLASIFICACIÓN DE LAS REDES P2P

Una posible clasificación de las redes P2P pudiera ser acorde a su grado de centralización:

- **Redes P2P centralizadas**

Este tipo de red P2P se basa en una arquitectura monolítica en la que todas las transacciones se hacen a través de un único servidor que sirve de punto de enlace entre dos nodos y que, a la vez, almacena y distribuye los nodos donde se almacenan los contenidos.

Poseen una administración muy dinámica y una disposición más permanente de contenido. Sin embargo, están muy limitadas en la privacidad de los usuarios y en la falta de escalabilidad de un sólo servidor, además de ofrecer problemas en puntos únicos de fallo, situaciones legales y enormes costos en el mantenimiento, así como el consumo de ancho de banda.

Una red de este tipo reúne las siguientes características:

- Se rige bajo un único servidor, que sirve como punto de enlace entre nodos y como servidor de acceso al contenido, el cual distribuye a petición de los nodos.
- Todas las comunicaciones (como las peticiones y encaminamientos entre nodos) dependen exclusivamente de la existencia del servidor.

Algunos ejemplos de este tipo de redes son *Napster* y *Audiogalaxy*.

- **Redes P2P híbridas, semi-centralizadas o mixtas**

En este tipo de red, se puede observar la interacción de un servidor central que sirve como *hub* y administra los recursos de banda ancha, enrutamientos y comunicación entre nodos pero sin saber la identidad de cada nodo y sin almacenar información alguna, por lo que el servidor no comparte archivos de ningún tipo a ningún nodo.

Tiene la peculiaridad de funcionar (en algunos casos como en *Torrent*) de ambas maneras, es decir, puede incorporar más de un servidor que gestione los recursos compartidos, pero también, en caso de que el servidor o los servidores que gestionan todo caigan, el grupo de nodos puede seguir en contacto a través de una conexión directa entre ellos mismos, con lo que es posible seguir compartiendo y descargando más información en ausencia de los servidores.

Este tipo de P2P presenta las siguientes características:

- Tiene un servidor central que guarda información en espera y responde a peticiones para esa información.
- Los nodos son responsables de hospedar la información (pues el servidor central no almacena la información) que permite al servidor central reconocer los recursos que se desean compartir, y poder descargar esos recursos compartidos a los usuarios que lo solicitan.
- Las terminales de enrutamiento son direcciones usadas por el servidor, que son administradas por un sistema de índices para obtener una dirección absoluta.

Algunos ejemplos de una red P2P híbrida son *KaZaA*, *Skype*, *BitTorrent*, *eDonkey* y *Direct Connect*.

- **Redes P2P "puras" o totalmente descentralizadas**

Las redes P2P de este tipo son las más comunes, siendo las más versátiles al no requerir de un gestionamiento central de ningún tipo, lo que permite una reducción de la necesidad de usar un servidor central, por lo que se opta por los mismos usuarios como nodos de esas conexiones y también como almacenadores de esa información. En otras palabras, todas las comunicaciones son directamente de usuario a usuario con ayuda de un nodo (que es otro usuario) quien permite enlazar esas comunicaciones.

Las redes de este tipo tienen las siguientes características:

- Los nodos actúan como cliente y como servidor.
- No existe un servidor central que maneje las conexiones de red.
- No hay un enrutador central que sirva como nodo y administre direcciones.

Algunos ejemplos de una red P2P "pura" son: *Kademlia*, *Ares Galaxy*, *Gnutella*, *Gnutella2* y *Freenet*.

La aplicación desarrollada queda enmarcada dentro de las redes totalmente descentralizadas, ya que cada usuario se conecta directamente con el resto de usuarios de su misma red sin la intervención de un servidor central que administre las direcciones o los recursos compartidos.

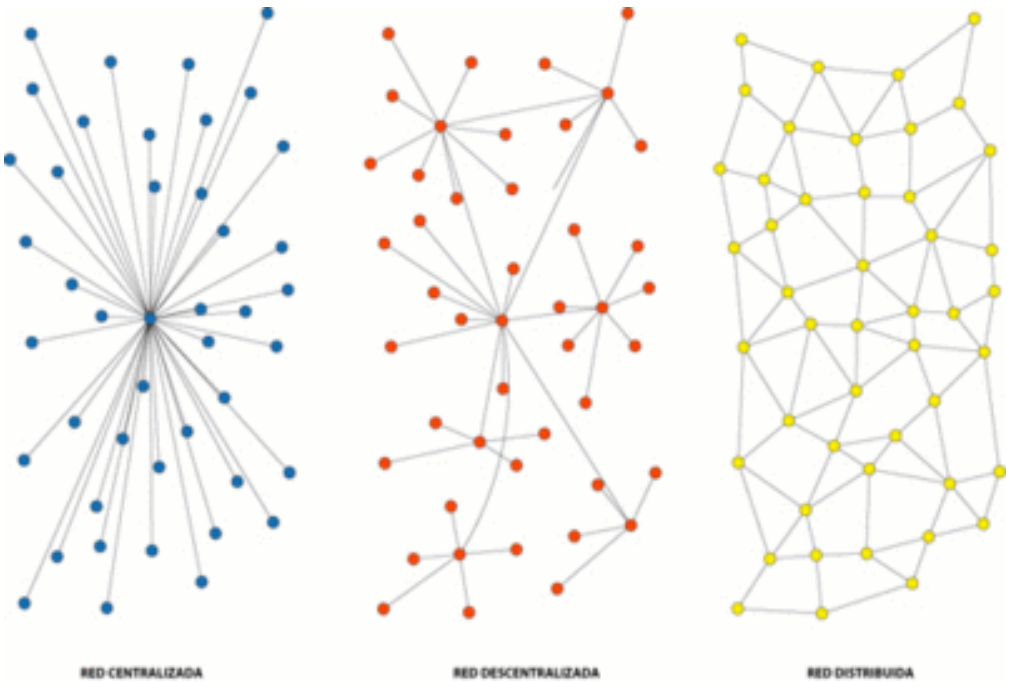


Figura 3. Las tres topologías de red según los famosos grafos de Paul Baran que se aplican también al diseño P2P.

Algunos prefieren clasificar las redes P2P según su estructuración:

- **Redes P2P sin estructura**

Se forma una red P2P no estructurada cuando los enlaces de la sobrecapa se establecen arbitrariamente. Tales redes pueden construirse muy fácilmente puesto que un usuario que desea unirse a la red puede copiar enlaces existentes de otro nodo y después formar sus propios enlaces en un plazo determinado.

En una red P2P no estructurada, si un usuario desea encontrar información específica en la red, la petición tiene que recorrer toda la red para encontrar tantos usuarios como sea posible, con el objetivo de conseguir a alguien que comparta los datos. La desventaja principal en estas redes es que las peticiones no siempre podrán resolverse. Es muy probable que un contenido popular esté disponible para varios usuarios, y es muy probable que cualquier usuario que lo busque lo encuentre; sin embargo, si un usuario está buscando datos no muy populares que comparten sólo unos cuantos usuarios, será altamente probable que la búsqueda no dé los resultados esperados. Dado que no hay correlación entre un usuario y el contenido compartido por él, entonces no hay garantía de que el usuario encuentre al que tiene la información deseada.

El *flooding* o inundación de mensajes, también incrementa el tráfico en la red y, por lo tanto, estas redes suelen tener una eficacia muy baja en los resultados de búsqueda. La mayor parte de las redes populares P2P (*Napster*, *Gnutella* y *KaZaA*) son redes P2P sin estructura.

- **Redes P2P estructuradas**

Las redes P2P estructuradas superan las limitaciones de las redes no estructuradas, mantienen una tabla de hash distribuida (DHT) y permiten que cada usuario sea responsable de una parte específica del contenido en la red.

Estas redes utilizan funciones de hash distribuido y asignan valores a cada contenido y a cada usuario en la red. Después siguen un protocolo global en la determinación de qué usuario es responsable de qué contenido.

De esta manera, siempre que un usuario desee buscar ciertos datos, utilizará el protocolo global para determinar al usuario o usuarios que lo tiene(n) y después dirigirá la búsqueda hacia éstos. Algunas redes P2P estructuradas son:

- Chord
- Pastry P2P Network
- Tapestry P2P Network
- Content Addressable Network
- Tulip Overlay
- CAN

La aplicación desarrollada queda enmarcada dentro de las redes sin estructura puesto que los enlaces entre los nodos se establecen arbitrariamente sin la intervención de una tabla hash distribuida y las peticiones de información entre nodos recorren toda la red, sin existir ningún protocolo que determine qué información comparte cada usuario.

También se podría clasificar las redes P2P según su generación:

- **Primera generación de P2P***:* Son literalmente las primeras redes P2P, que eran centralizadas.
- **Segunda generación de P2P***:* En esta generación se implementó por primera vez la característica de la descentralización, y esta característica es la más frecuente en los actuales P2P.
- **Tercera generación de P2P***:* Los más recientes, que implementan una comunicación no directa, cifrada y anónima.

Existe también la posibilidad de clasificar las redes P2P según sus características de anonimidad o exclusividad:

- Sin características de anonimidad
- Pseudónimo
- Red P2P Privada
- Friend-to-friend (de amigo-a-amigo)

Una reciente generación de sistemas peer-to-peer son los llamados metacomputing que se clasifican como middleware. Ejemplos de metacomputing son *Legion* y *Globus*, proyectos de investigación para el desarrollo de infraestructuras software de computación distribuida a escala mundial.

1.7. PROBLEMAS DE LAS REDES P2P

Algunos de los principales problemas de las redes entre iguales son:

- Diferentes redes P2P no pueden comunicarse entre sí (no es posible descargar un archivo desde un cliente *Kazaa* y utilizar en la misma red el *MSN Messenger*).
- No es posible la reconexión automática. Si una conexión cae, se debe recomenzar la descarga.
- La red está dividida: si un nodo cae, puede darse el caso de que una subred se desconecte.
- Los protocolos son muy específicos de cada aplicación.
- El proceso de búsqueda de los nodos es complejo. El problema de encontrar un par que se encuentre conectado a la red P2P es comúnmente solucionado realizando una conexión con un servidor, en el que la dirección IP es conocida. Con las P2P "puras" o totalmente descentralizadas, los clientes obtienen esta información sobre el resto de la red por Inundación de Pedidos (Multicast), y así pueden intercambiar información sin intervención de servidores.
- La conexión de pares sin dirección IP pública dificulta el proceso. La solución que se suele aplicar es la conexión de otro nodo que cumple la función de proxy, es decir, realiza una acción en representación de otro; en las redes P2P, el dispositivo proxy intercepta las conexiones de red que un cliente hace a un servidor de destino. El resto de los pares se conectan a este proxy, el cual envía la información que llega de uno al otro. Cualquier nodo que esté conectado a la red P2P y tenga una dirección IP pública puede ser elegido para que cumpla con el cometido de proxy.

Los principales problemas de la arquitectura P2P están ligados a aspectos no maduros de la tecnología. El avance del P2P dependerá del avance de los siguientes aspectos:

- ☑ Diseño y organización de los sistemas y mecanismos distribuidos de forma que sean eficientes para la obtención de datos.
- ☑ Mecanismos de localización y guiado distribuidos que permitan el acceso uniforme y disponible a los recursos correspondientes.
- ☑ Mecanismos de gestión distribuida que añadan los recursos a los múltiples pares y satisfagan el grado de calidad de servicio extremo a extremo, y peticiones de respuesta en tiempo real.
- ☑ Proveer mecanismos de seguridad y confidencialidad eficientes, como la autenticación, la encriptación, etc.

1.8. P4P, EL SIGUIENTE PASO EN EL DESARROLLO P2P

P4P es el acrónimo para *Proactive Network Provider Participation for P2P*, *Participación Activa del Proveedor de Red en P2P*. También es denominado *P2P Híbrido*.

Es el siguiente paso en el desarrollo de servicios P2P, ya que permite minimizar el número de saltos requeridos para las transferencias de archivos, eligiendo los pares más cercanos al usuario (pertenecientes al mismo proveedor) en primer lugar.

P4P no es un protocolo P2P, sino un medio para que los proveedores de Internet optimicen el tráfico de las redes P2P de sus usuarios.

Existe un grupo de trabajo que está definiendo exactamente este proceso: *P4P Working Group*.

Dos grandes proveedores de telecomunicaciones se han mostrado interesados en esta tecnología hasta el momento: *Telefónica* y *Verizon*

El P4P genera ciertos recelos en la comunidad de internautas ya que, gracias a esta tecnología, los *ISP*[5] pueden hacer negocio, no solo de ofrecer la conexión a internet a los usuarios, sino también de los contenidos que viajan por dicha red. Esta posibilidad puede suponer que las mismas ISPs discriminen a favor del tráfico que les es más rentable.

[5] Internet Service Provider - Proveedor de Servicios de Internet

2. MODELO DE NEGOCIO

El modelo de negocio, aunque no forma parte de la implementación como tal, aporta una amplia visión del contexto que rodea al producto; esta visión global es fundamental a la hora de integrar la aplicación dentro de un entorno de interacciones real.

Las aplicaciones P2P quedan enmarcadas dentro del contexto de las redes libres. A modo de esquema, se presentan los aspectos que rodean al concepto de red libre a tener en cuenta en el desarrollo de cualquier aplicación P2P.

2.1. MODELO DE COMUNICACIÓN EN INTERNET

El crecimiento actual de las redes y los masivos cambios en seguridad han creado nuevas vías que hacen difícil el funcionamiento de algunas aplicaciones. La arquitectura original de Internet, donde cada equipo se podía comunicar directamente con otro utilizando una IP única y global, ha sido remplazada por una nueva arquitectura de direcciones, consistente en una dirección global y muchas direcciones privadas interconectadas por NAT[6].

En esta nueva arquitectura, como se ve en la figura, sólo los equipos que tengan dirección pública pueden ser fácilmente conectados por otros en la red, ya que tienen una única, global y rutable dirección IP. Los equipos de las redes privadas pueden conectarse entre sí, o pueden establecer conexiones TCP o UDP con equipos que pertenezcan a la red pública.

Figura 4. *Esquema de conectividad en Internet*

Los dispositivos que realizan el NAT traducen la dirección y puertos de los paquetes que provienen de las redes privadas hacia las redes públicas, por lo tanto los dispositivos NAT generalmente permiten sólo conexiones salientes y bloquean cualquier tráfico entrante, a no ser que esté configurado específicamente de otra manera.

Esta arquitectura hace difícil la comunicación directa entre dos equipos que pertenezcan a distintas redes privadas. Con esto tenemos que muchas aplicaciones P2P que comparten archivos, realizan videoconferencia o juegos en línea, no funcionan a través de entornos con NAT.

[6] Network Address Translation - Traducción de Dirección de Red

2.2. BARRERAS DE INTERNET: FIREWALL, NAT Y PROXY

☒ Firewall

Uno de los problemas más comunes que el P2P encuentra en Internet es la existencia de barreras que impiden la comunicación directa entre varios pares. La barrera más común es el cortafuegos o firewall, cuya labor consiste en filtrar ciertos puertos de comunicación, limitar el acceso a determinados computadores y filtrar específicos patrones de mensajes. Prácticamente un cortafuegos sólo permite el paso de mensajes Http.

☒ NAT

Como ya se ha mencionado, otra barrera para la comunicación directa es NAT, estándar de Internet que le permite a una LAN usar un grupo de direcciones IP para el tráfico interno y otro grupo de direcciones IP para el tráfico externo. La NAT posee tres propósitos principales:

- ✓ Proveer un tipo de cortafuegos al ocultar las direcciones IP internas.
- ✓ Permitirle a una empresa usar más direcciones IP internas. Dado que son internas, no hay posibilidad de conflicto con IPs usadas por otras empresas u organizaciones.
- ✓ Permitirle a una empresa combinar múltiples conexiones ISDN[7] dentro de una misma conexión a Internet.

☒ Proxy

Por último, un servidor Proxy permite el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP pública. Provee servicios como el filtrado, cacheado y monitorización de tráfico.

[7] Integrated Services Digital Network - Red Digital de Servicios Integrados (RDSI)

Estas barreras han sido salvadas mediante el uso del protocolo Http. Mediante el uso de Http en una aplicación, conocido como *túnel Http* o *Http Tunneling*, se puede incrementar la posibilidad de atravesar las distintas barreras. La técnica de *tunneling* consiste en encapsular un protocolo de red sobre otro creando un túnel dentro de una red de comunicaciones.

Las aplicaciones de las técnicas de encapsulado por *Http Tunneling* son diversas, y varían desde una simple evasión para descargar un fichero desde *eMule*, hasta conectar *MSN Messenger*, realizar videoconferencias y todo lo que se tenga bloqueado encapsulando en el puerto 80.

El problema con la aplicación *túnel Http* es que la comunicación se ve desplazada. Una respuesta puede ser enlazada con su petición, pero el contexto entre cada ciclo de petición/respuesta se vería relacionado con aquellos posteriores o anteriores. Debido a ello, algunos tipos de identificador deben ser pasados desde el cliente al servidor de forma que se consiga mantener el contexto. Estos identificadores serían las cookies.

Para adecuar esta solución al P2P se debe hacer uso de pares relay (o superpares), que actúen de servidores Http. Si estos servidores disponen de una capacidad suficiente, y si existen grupos de relays a lo largo de la red estratégicamente colocados, el *tunneling* llega a adquirir una eficiencia mayor. Este uso de relays no parece un concepto muy P2P, pero realmente cualquier par podría ser uno de ellos. Se pueden considerar como simples intermediarios en una transacción.

2.3. PROTOCOLO DE COMUNICACIÓN EN REDES P2P

Una de las características principales de la tecnología P2P es que conforma una red de nodos para realizar sus tareas, ya sea compartir espacio, descargar archivos, distribuir tareas, etc. Por tanto, dichos nodos necesitan un protocolo de comunicación, tanto un protocolo de red para transmitir los mensajes, como un protocolo propio para que la información transmitida sea interpretada por los distintos nodos.

- ☑ **Nivel de Enlace.** Como las redes inalámbricas utilizan un medio compartido que transporta los datos que se emiten, es necesario que los nodos no interfieran en las comunicaciones que puedan estar manteniendo los demás, por eso el control de acceso al medio (MAC) es uno de los aspectos más importantes. Por lo general se usa un esquema de coordinación por contienda en el que no existe un control centralizado. En este esquema cualquier nodo puede transmitir cuando considere oportuno, sin embargo, es susceptible a colisiones, y cada nodo debe ser capaz de recuperase.
- ☑ **Nivel de Red.** En una red de este tipo cada nodo debe comportarse como un router, manteniendo individualmente las rutas a otros nodos. Esto implica que cuando crezca el número de nodos en la red, crecerán el número de rutas que ha de mantener cada nodo, y por consiguiente, las tablas con la información de enrutado hacia otros nodos. La escalabilidad, dependiendo del número de nodos y la movilidad de estos, se convierte en uno de los grandes problemas del mantenimiento de este tipo de redes.
- ☑ **Nivel de Transporte.** El protocolo de transporte UDP ofrece un servicio no confiable y no orientado a conexión de entrega de paquetes. Los mensajes de datos del nivel de aplicación son encapsulados en paquetes UDP y enviados confiando que alcanzarán correctamente su destino, aunque no se garantiza nada. Debido a su simpleza, su aplicación en este tipo de redes es directa y no presenta inconvenientes.
- ☑ **Nivel de Aplicación.** Para un uso comercial, se hace imprescindible que los protocolos usados en este nivel sean los mismos empleados para el mismo propósito en redes fijas.

2.4. WINDOWS Y EL FUNCIONAMIENTO DE LAS P2P

Tradicionalmente, los desafíos para el desarrollo de aplicaciones P2P han incluido la necesidad de desarrollar protocolos propietarios para el intercambio de mensajes, tener que buscar y conectar con las instancias de una aplicación que se ocultan detrás de una traducción de direcciones de red (NAT) o un firewall, y la necesidad inevitable de apoyar la infraestructura requerida para localizar aplicaciones en una red de área amplia (WAN).

Estas barreras se reducen drásticamente a partir de Windows Vista y el acompañante. NET Framework 3.0. La combinación de mejoras en Windows Vista como *PNRP*[8], *PNM*[9], y la introducción de *PeerChannel* en *Windows Communication Foundation* han hecho aplicaciones P2P mucho más accesible.

Para empezar, una aplicación P2P se conecta directamente con otras instancias de aplicaciones P2P. En el lenguaje de P2P, cada instancia de la aplicación se denomina nodo. Una agrupación de nodos conectados y con un nombre es llamada a menudo malla. Como resultado, las tecnologías que facilitan el desarrollo de aplicaciones P2P son referidas con frecuencia como tecnologías de malla.

PNRP, *PeerChannel* (en *Windows Communication Foundation*) y *PNM*, son ejemplos de tecnologías de malla de Windows Vista y posteriores. Todas ellas producen mallas que tienen aproximadamente la misma topología.

Una topología de malla es generalmente una abstracción del patrón de conexión entre los nodos de la malla.

[8] Peer Name Resolution Protocol - Protocolo de Resolución de Nombres de Pares

[9] People Near Me - Personas a mi alrededor

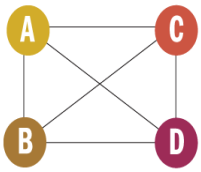


***Figura 5**. Malla totalmente conectada*

La Figura 5 muestra que cada uno de los cuatro nodos de la malla se conecta a todos los demás nodos. Una malla que cumple con este criterio se considera que es totalmente conectada. Mallas completamente conectadas rara vez son una buena idea.

Todos los sistemas operativos modernos aprovechan TCP/IP y UDP para la comunicación en red. Si TCP/IP es el transporte elegido por una malla completamente conectada, cada nodo de una malla de N nodos conectados debe crear o aceptar tomas de N-1 nodos. A medida que aumenta N, este modelo pasa a ser claramente inviable. Para resolver los problemas de escalabilidad y conectividad WAN, se debe buscar una malla de nodos parcialmente conectados, como la de la Figura 6.

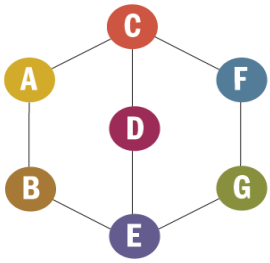


***Figura 6.** Malla parcialmente conectada.*

Como su nombre indica, los nodos de una malla parcialmente conectados están conectados sólo a unos pocos de los otros nodos de la malla. En el lenguaje P2P, estos nodos adyacentes se llaman vecinos. En general, una malla parcialmente conectada, con menos exigencias de recursos en cada nodo, incrementa drásticamente la escalabilidad de la malla. En teoría, una malla parcialmente conectada puede llegar a incluir todas las aplicaciones en todas las computadoras del mundo.

La forma en que un nodo se une a una malla depende de la tecnología de malla utilizada, pero en términos generales, un posible nodo debe utilizar el nombre de malla para resolver las direcciones físicas de uno o más nodos que ya estén en la malla. Después de conectarse, el nodo recién agregado debe entonces estar listo para responder a resoluciones posteriores de nombre de malla de otros nodos posibles.

La resolución del nombre de malla es un tema complejo. Gran parte de esta complejidad se debe al hecho de que, en muchos casos, la resolución de un nombre de malla depende de una o más mallas adicionales.

Una vez que un nodo está conectado a una malla, puede comunicarse con otros nodos a través de uno de los dos medios: las inundaciones de malla (también llamada mensajería multipartidista), o los mensajes de dirección.

La inundación de malla es un intento de enviar un mensaje a todos los nodos de la malla.

Por el contrario, la mensajería de dirección es un intento de dirigir un mensaje a un nodo en particular en la malla.

En general, un nodo en una red puede propagar un mensaje a todos los demás nodos vecinos. Al recibir este mensaje, el vecino del nodo emisor es responsable de reenviar el mensaje a sus vecinos, y así sucesivamente.

Las mallas son raramente estáticas. En la mayoría de las aplicaciones P2P, con frecuencia los nodos pueden unirse y dejar la malla, ya sea debido a cambios en la conectividad de red, o, en el caso de una aplicación de mensajería instantánea, debido a que un usuario inicia y detiene la aplicación. La mayoría de las tecnologías de malla deben tener algún mecanismo para mantenerse.

En general, el objetivo del mantenimiento de la malla es reparar o ajustar la malla para que opere de manera más eficiente y sea más robusta. Es importante señalar que cada tecnología de malla implementa el mantenimiento de malla de forma diferente.

PNRP está diseñado para resolver las direcciones físicas sobre la base de, entre otras cosas, un nombre de malla.

PNRP está disponible para Windows XP Service Pack 1 (SP1) con el Advanced Networking Pack, Windows XP SP2, Windows XP Professional x64 Edition, Windows Vista y Windows 7.

Al nivel más simple, *PNRP* es en sí mismo una aplicación P2P que toma la forma de un servicio de Windows, y la malla de nodos *PNRP* se utiliza exclusivamente para el descubrimiento de las direcciones físicas de los nodos que participan en otras mallas.

PNRP y *PNRP IPv6* está construido sobre el Protocolo de Internet versión 6 (IPv6). Dado que IPv6 es bastante nuevo, es importante mencionar al menos un aspecto importante antes de discutir la mecánica de *PNRP*. En IPv6, la dirección es un valor de 128 bits (que permite aproximadamente 3,4 x 1038 combinaciones de direcciones). Este tamaño del conjunto de direcciones IPv6 permite una de las características más importantes de IPv6: el direccionamiento de extremo a extremo, incluso cuando estas direcciones estén en varias subredes, y escondidas detrás de un NAT.

Una de las principales ventajas de *Windows Communication Foundation* es que ofrece un modelo de programación universal para muchos sabores diferentes de las aplicaciones distribuidas.

Una de las características menos conocidas de *Windows Communication Foundation* es su apoyo a la creación de aplicaciones P2P utilizando el mismo modelo de programación universal. Debido a su apoyo a las aplicaciones P2P, se puede mirar a *Windows Communication Foundation* como una tecnología de malla, pero en realidad, sólo el módulo *PeerChannel* está dedicado a la creación de aplicaciones P2P. Por esta razón, el término *PeerChannel* es comúnmente utilizado para referirse a las capacidades P2P de *Windows Communication Foundation*.

Independientemente de la forma en que se refieren a ellas, el *PeerChannel* en *Windows Communication Foundation* esconde casi toda la complejidad asociada tradicionalmente con el desarrollo de una aplicación P2P y es un cambio de juego en el campo de desarrollo de aplicaciones P2P.

La malla *PeerChannel* está diseñada para inundar mensajes. Sin embargo, también contiene mecanismos que pueden propagar un mensaje en parte de la malla, en lugar de a toda la malla. Por esta razón, es más exacto decir que una malla *PeerChannel* está diseñada para mensajería multipartidista.

Una aplicación P2P puede registrar muchos nodos con el servicio *PNRP*, y cada nodo de la malla *PNRP* se compone de un nombre P2P y un *ID PNRP*. En términos generales, un nombre de P2P representa un nombre de malla y es asignado por la aplicación P2P, mientras que el *ID PNRP* se asigna a un nodo registrado por la infraestructura de *PNRP*. La malla de *PNRP* está diseñada para enviar mensajes de dirección, y utiliza el *ID PNRP* para la estructura de la malla.

La estructura de la malla *PeerChannel* está impuesta por el número de vecinos que se conecta a cada nodo. Con este fin, una malla *PeerChannel* mantiene activamente la estructura de la malla. El efecto de este mantenimiento es una malla sólida, distribuida uniformemente. Para ser más específicos, un nodo de la malla se esfuerza por mantener entre dos y siete conexiones a los vecinos. Estos umbrales representan un acto de equilibrio entre la demanda de recursos en el nodo local y el mantenimiento de una malla sólida.

Es importante tener en cuenta que el código de aplicación no puede cambiar estos límites o ejercen ningún control sobre el mantenimiento de la malla. Estos detalles son atendidos en su totalidad por la infraestructura de base *PeerChannel* nodo por nodo.

PeerChannel ofrece *PNRP* y resoluciones personalizadas como un medio para descubrir las direcciones de los nodos que ya están en la malla. Independientemente del método elegido de resolución, la idea general es la misma: pasar el nombre de la malla para la resolución y recibir una lista de direcciones IP de los otros nodos de la malla.

Una vez que la resolución produce una lista de direcciones, el nodo se conecta a cada una de las direcciones posibles de forma concurrente. Cuando un nodo en una red *PeerChannel* recibe una de las solicitudes de conexión, podrá aceptar o rechazar la conexión.

Si se acepta la conexión, el nodo existente envía al nodo recién unido un mensaje de bienvenida que contiene, entre otras cosas, una lista de las direcciones de otros nodos de la malla. Si la conexión es rechazada, entonces el nodo existente envía al posible nodo un mensaje de rechazo que contiene un motivo de rechazo y una lista de las direcciones de otros nodos de la malla.

El punto importante aquí es que la resolución de la malla (mediante el *PNRP* o una resolución personalizada) no es la única forma de devolución de una lista de direcciones a un nodo en *PeerChannel*. Esta característica permite que los nodos que gozan de excelentes conexiones, devuelvan una lista de direcciones con mayor rapidez que si la *Resolución de Nombres de Malla* fuera el único medio por el cual un posible nodo pudiera obtener una dirección. Además, esta característica permite a los nodos de la malla ejercer control sobre cuántos vecinos tiene un nodo, que a su vez, afecta a la solidez de la malla.

La comunicación dentro de una malla *PeerChannel* se ajusta para mantener el paso de mensajes repetitivos a un mínimo. Cuando un nodo de la malla envía un mensaje a la malla, realmente está enviando un mensaje a sus vecinos. Tras la recepción de un mensaje, cada vecino inspecciona el mensaje y lo reenvía a sus vecinos. Si un nodo *PeerChannel* recibe un mensaje de un vecino, no envía el mensaje de vuelta a ese vecino. Además, si un nodo *PeerChannel* recibe con frecuencia un mensaje de un vecino que ya ha recibido y procesado, la conexión a ese vecino puede ser terminada en el siguiente ciclo de mantenimiento.

Estas características se implementan en una caché local en cada nodo. Internamente, cada nodo en una red *PeerChannel* almacena el valor del *ID WS-Addressing* (Identificador de Direccionamiento de Servicios Web) y un identificador del vecino que entregó el mensaje. Cada nodo comprueba la memoria caché para decidir a qué vecinos va a entregar el mensaje.

La combinación de estos rasgos da como resultado una malla que se ajusta para enviar mensajes a los nodos de la malla con un mínimo de repetición y de consumo de ancho de banda de red.

Como se ha mencionado anteriormente, un nodo *PeerChannel* también puede enviar un mensaje a un subconjunto de los nodos de la malla. Esto se logra mediante la asignación de un número de saltos a un mensaje, que es en realidad una forma de rastrear el número de nodos a través de los cuales ha sido transmitido el mensaje.

No hay que confundir este mecanismo con la mensajería de dirección, en la que se dirige un mensaje a un nodo en particular. Por el contrario, un número de saltos es un modo de trazar una frontera difusa alrededor de los nodos que van a recibir el mensaje.

Por ejemplo, si un nodo *PeerChannel* (Nodo A) tiene tres vecinos y envía un mensaje a la malla con un número de saltos de 1, entonces el mensaje será entregado a los tres nodos. Del mismo modo, si cada uno de los vecinos de un nodo tiene tres vecinos únicos, y un nodo envía un mensaje a la malla con un número de saltos de 2, entonces el mensaje será entregado a nueve nodos. Sin embargo, si cualquiera de los nodos vecinos de A tiene vecinos comunes, este número se reducirá proporcionalmente.

Físicamente, un número de saltos se expresa en un mensaje como un entero en un bloque de cabecera. Cuando un nodo recibe un mensaje que tiene un número de saltos, se inspecciona el valor del número de saltos. Si el valor es mayor que cero, el nodo disminuye el número de saltos de forma monótona, y reenvía el mensaje (con el valor del número de saltos) a los vecinos apropiados. Si recibe un mensaje que contiene un número de saltos de cero, entonces el mensaje no se envía.

PNM es otra tecnología de malla, integrada en Windows Vista y posteriores, que permite a los grupos de dispositivos cercanos descubrir, conectar, invitar, y colaborar con los demás.

Las capacidades ofrecidas por *PNM* son enormes, y es razonable suponer que una vez liberado, la comunidad de desarrolladores encontrarán formas nuevas e ingeniosas de aprovechar esta tecnología. Es importante señalar que *PNM* es completamente tecnología de malla y está desactiva por defecto.

La arquitectura *PNM* se compone de, entre otras cosas, una aplicación P2P llamada *p2phost.exe*. Cuando se ejecuta, este proceso crea una malla de conexión a instancias de *p2phost.exe* en otras máquinas. En general, el propósito de esta malla es la mensajería de dirección. Más específicamente, *PNM* está diseñado para resolver los nodos locales, y comunicarse con un subconjunto de estos nodos locales.

La API PNM está disponible como parte de la API de Windows, y en su mayor parte, se centra en la configuración del comportamiento de *p2phost.exe*.

El siguiente escenario ilustra cómo se puede utilizar *PNM* para proporcionar funciones de colaboración:

- *Usuario A* se instala la aplicación P2P, *Peer2PeerApp.exe*, en su ordenador portátil con Windows Vista o posterior. El instalador registra *Peer2PeerApp* con *PNM*.
- *Usuario A* lleva su ordenador portátil a un café e inicia sesión en *PNM*. Cuando está conectado, *p2phost.exe* comienza a descubrir otros nodos de la malla de *PNM*.
- *Usuario B*, que también se ha instalado *Peer2PeerApp*, está también en la cafetería con su ordenador portátil. Cuando *Usuario B* inicia sesión en *PNM*, *p2phost.exe* comienza a descubrir otros nodos de la malla de *PNM*.
- Desde *Peer2PeerApp*, integrado con *PNM*, *Usuario A* invita a *Usuario B* a utilizar *Peer2PeerApp*.

- ⇨ Tras la invitación enviada del usuario A al usuario B, *Usuario B* recibe una notificación en su barra de tareas de que *Usuario A* le ha invitado a comenzar *Peer2PeerApp*.
- ⇨ *Usuario B* acepta la invitación, y *PNM* inicia *Peer2PeerApp*. Ambos usuarios utilizan la aplicación *Peer2PeerApp*.
- ⇨ En algún momento del futuro, *Usuario A* y *Usuario B* están en diferentes ciudades.
- ⇨ *Usuario A* inicia sesión en *PNM* y abre *Peer2PeerApp*. De inmediato se da cuenta de que *Usuario B* también ha iniciado sesión en *PNM*.
- ⇨ *Usuario A* invita a *Usuario B*, a través de *Peer2PeerApp*, a utilizar la aplicación *Peer2PeerApp*.
- ⇨ *Usuario B* recibe y acepta la invitación de *Usuario A*. Tras la aceptación de la invitación, *PNM* lanza la aplicación *Peer2PeerApp*. Ambos usuarios utilizan la aplicación P2P desde ciudades diferentes.

La aplicación desarrollada se apoya totalmente en el funcionamiento del estándar *PeerChannel* de *Windows Communication Foundation*; utiliza *PNRP* para unirse a la malla y descubrir las direcciones de los nodos que ya están conectados, un esquema URI de tipo *net.p2p*, un binding o enlace *NetPeerTcpBinding* y la adición de la seguridad basada en contraseña.

En términos generales, puede aprovechar las principales características de la API de *PNM*, ofreciendo la posibilidad de registrar la aplicación con *PNM*, invitar a otros a unirse a una sesión de colaboración, crear un contacto persistente e invitar a un contacto que ya no es local.

2.5. SEGURIDAD

Como ha sucedido en otros contextos, las cuestiones de seguridad en tecnología P2P han sido sistemáticamente olvidadas o, en el mejor de los casos, tratadas de forma puntual y sin perspectiva global. Por otro lado, analizar la seguridad de esta clase de redes no resulta nada sencillo, dada la diversidad y heterogeneidad de las arquitecturas P2P existentes hasta la fecha.

Por eso, tras realizar un ejercicio de abstracción para identificar los elementos esenciales de todas ellas, que son: la comunidad de usuarios (nodos) que integra la red, la arquitectura de recubrimiento que define la estructura lógica de la red y los contenidos almacenados por los usuarios, se han podido identificar los problemas de seguridad en cada uno de los aspectos.

En primer lugar, la necesidad de garantizar la integridad de los datos distribuidos (para que ningún usuario los modifique sin autorización) presenta ciertas dificultades. Lo mismo ocurre cuando se requiere una comunicación confidencial entre nodos o se presenta la necesidad de un esquema de autorización que delimite los contenidos a los que puede acceder un usuario determinado.

Por otra parte, la alta transitoriedad de nodos (*churn*), o lo que es lo mismo, el dinamismo con el que los miembros entran y salen de la comunidad, supone un problema para casi todos los algoritmos y procedimientos del sistema. A todo esto habría que añadir otros peligros, como los ataques de denegación de servicio (saturar a un usuario honesto con solicitudes para colapsarle, por ejemplo), la utilización de identidades múltiples por parte de los usuarios (ataque *Sybil*) o el comportamiento egoísta exhibido por ciertos nodos que no comparten sus recursos (*free-riding*).

Existen algunas propuestas que tratan de resolver estos problemas y de encontrar soluciones contra estas conductas, pero lo cierto es que las redes de distribución usadas en la actualidad empiezan a necesitar arquitecturas más sofisticadas, capaces de soportar nodos móviles, una topología de red auto-organizativa y requisitos muy importantes de seguridad y eficiencia.

La principal limitación para la provisión de soluciones de seguridad en estas redes es, por lo general, la ausencia de autoridades centrales, y según algunos expertos la investigación debería centrarse en enfoques alternativos, como los protocolos de intercambio justo, de sociedades virtuales y de integridad y autenticación de contenidos, entre otros.

2.6. ANONIMATO

El nombre de peer-to-peer anónimo puede entenderse de forma equivocada. Esto es debido a su diseño; un nodo de la red debe tener un pseudónimo desde que tiene que tener una dirección para poder ser alcanzado por otro nodo para intercambiar datos. Sin embargo, normalmente esta dirección, especialmente en redes anónimas, no contiene ninguna información que pueda permitir la identificación. Por tanto, un usuario es casi pero no completamente anónimo.

Cuando se reciben datos en cualquier red, esta debe venir de algún sitio y los datos se deben haber pedido anteriormente por alguien. El anonimato viene de la idea en la que nadie sabe quién requiere la información ya que es difícil, pero no imposible, determinar si un usuario ha pedido los datos para él mismo o simplemente está pidiendo datos que le ha requerido otro usuario. El resultado final es que todo el mundo en una red anónima actúa como un emisor y un receptor universal para mantener el anonimato.

Un ideal común para las redes de iguales anónimas es hacer imposible que se pueda estorbar en la difusión de información. Esto es alcanzado típicamente a través del cifrado, haciendo un tipo de información indistinguible de otra.

Los promotores del P2P anónimo argumentan que esta tecnología es deseable y en algunos casos necesaria para asegurar la libertad de expresión y la libertad de movimiento de información. Reclaman que la verdadera libertad de expresión, especialmente en temas controvertidos, es difícil o imposible a menos que las personas puedan hablar bajo el anonimato. Argumentan que si el anonimato no es posible, uno podría sufrir amenazas o represalias por representar una vista no popular.

El movimiento de pornografía es común en las redes P2P anónimas, y algunos creen que estas redes ayudan al terrorismo. Hay varias respuestas a estas críticas.

La primera, es que la información es neutral y que hay gente actuando sobre la información, lo que es bueno o malvado. Una segunda, es que estas cuestiones son ejemplos de pánicos morales, y que si las redes de iguales anónimas hubieran estado activas durante los años 1950 o 1960, seguramente habrían sido atacadas por portar información sobre los derechos civiles o el anarquismo.

2.7. SITUACIÓN LEGAL DE LAS REDES P2P

La *RIAA*[10], *SGAE*[11], *MPA*[12] y otros, no sólo han optado por llevar a juicio a los creadores de los programas de intercambio de archivos, sino que también han realizado algunas acciones en contra de sus usuarios.

En septiembre de 2003, la *RIAA* demandó a 261 internautas por copiar música de manera ilegal. Resultó particularmente polémico que, entre esos 261 internautas, se encontrara una niña de 12 años que acabó condenada a pagar 2000 dólares por compartir cerca de mil canciones.

También ha habido acciones legales en contra de sitios web que almacenan y permiten buscar elinks, Torrents, archivos NZB (abreviatura de *Newzbin*, utilizados en grupos de news para buscar partes de archivos que, cuando se combinan juntos, forman un archivo de gran tamaño) y otros links de P2P.

También se ha implementado el uso de servidores *fakes*. Un servidor *fake* se caracteriza porque en realidad no está dedicado verdaderamente a actuar como servidor de la red *eDonkey 2000*, sino que intenta obtener información de los clientes que a él se conectan (para posibles pleitos posteriores) o contaminar la red con elinks falsos, corruptos, o simplemente llenos de basura. Como ejemplo cabe destacar los servidores *Razorback 2.2, 2.3,..., 2.6*, que aparecieron "un buen día" al calor del éxito de *Razorback 2.0 y 2.1*.

En el 2006, los servidores verdaderos *Razorback 2.0 y 2.1* fueron cerrados por acciones legales de la *Motion Picture Association* (*MPA*). *Razorback 2* era el servidor más importante de la red P2P de *eDonkey 2000*, albergaba 3 millones de usuarios diarios, y manejaba más de 1,3 millones de conexiones simultáneamente, indexando más de 170 millones de archivos; actualmente los responsables de *Razorback 2* se disponen a reabrir el servidor.

---

[10] Recording Industry Association of America - Asociación Estadounidense de Discográficas

[11] Sociedad General de Autores y Editores

[12] Music Publishers Association - Asociación de Editores de Música

Sin embargo, a pesar del cierre de estos servidores, sólo se logró que los usuarios se cambiaran a otros servidores de la red *eDonkey 2000*, y que se diera a conocer mucho más la red *Kademlia*.

En septiembre de 2006, *MetaMachine*, la compañía propietaria de *eDonkey*, se comprometió con la *RIAA* a pagar una multa de $30 millones en un acuerdo extrajudicial, para evitar posibles demandas de la industria discográfica; debido a esto, en el sitio web de *eDonkey* se colocó un aviso que informa sobre la ilegalidad de compartir música y vídeos que tengan copyright. Asimismo, el programa cliente *eDonkey 2000* dejó de funcionar, desplegando este mismo mensaje e iniciando su desinstalación automáticamente. No obstante, la red *eDonkey 2000* no pudo ser cerrada, y sigue funcionando hasta la actualidad debido a otros programas como *eMule*, *MLDonkey*, etc.

Respecto al protocolo *BitTorrent*, también ha habido algunas acciones en contra. En 2006, a finales de marzo, *TorrentPluribrain*, un buscador de Torrents para el escritorio, tuvo que cesar la actividad de sus servidores debido a una denuncia interpuesta en el nombre de la *Société Civile des Producteurs Phonographiques* de París ya que, pese a que su desarrollador es español, los servidores estaban localizados en Francia.

Desde febrero de 2010 hasta febrero de 2011 se produjeron cerca de 100.000 demandas contra los usuarios de redes P2P por violación de derechos de autor en Estados Unidos.

Los principales demandados han sido los usuarios del programa *BitTorrent*, a los que se informaba a través de una carta de que habían sido acusados de facilitadores de piratería y de distribuir creaciones sin autorización de la red. Según ha informado *TorrentFreak*, los demandantes, normalmente responsables de la industria del porno, a través de estos juicios en masa tratan de obtener los datos personales de la mayoría de los usuarios que presuntamente han compartido material en línea. Después, una vez que se han entregado sus datos, les ofrecen resolver el problema a cambio de unos cientos o miles de dólares y que el caso no vaya a más, por lo que los usuarios prefieren pagar y evitar así un juicio con sanciones más duras.

Al parecer, el sistema judicial de los Estados Unidos se está sobrecargando con demandas de este tipo, según el blog de tecnología. Asimismo, organizaciones de consumidores como la *Electronic Frontier Foundation* (*EFF*), ya se han hecho eco de estas demandas y han explicado que "los propietarios de derechos de autor tienen el derecho a proteger sus obras, pero no pueden utilizar tácticas de mala calidad para hacerlo", según la directora de Propiedad Intelectual del *EFF*, Corynne McSherry.

La información sobre el número de casos que van surgiendo sobre las demandas contra usuarios de P2P, se puede ir viendo a través de una hoja de cálculo de Excel en Google Docs que se irá actualizando de forma periódica (https://docs.google.com/leaf?id=0B8oEqWOAwM3SY2MxNjlhYzUtZDE4Mi00MDcwLWFhZDEtZjdjMmFlODQwYjBk&hl=en).

2.8. SITUACIÓN LEGAL EN ESPAÑA

En marzo del 2010 el Ejecutivo aprobó en el Consejo de Ministros la *Ley de Economía Sostenible* (*LES*), en cuya disposición final primera se da cobertura legal al cierre de las webs que enlazan a contenidos protegidos por derechos de autor. Junto a la *LES* también se tramitó la denominada *ley Sinde*, por el nombre de la titular de Cultura, Ángeles González-Sinde.

El anuncio de la aprobación de la *LES*, seguido al minuto desde las redes sociales, desató un aluvión de protestas en Internet. La más impactante, el apagón de webs de enlaces tan populares como *Cinetube* o *Series Yonkis*, en cuya portada, sobre fondo negro, se leía "Por la libertad en la Red. No al cierre de webs".

En julio de 2011 el ministerio de Cultura, mediante un comunicado, anunció que, una vez vencido el plazo de tres meses, ni la Comisión Europea ni ninguno de los países miembros de la UE habían presentado observaciones o dictámenes (opiniones) sobre el Real Decreto que regula el funcionamiento de la *Comisión de Propiedad Intelectual*, es decir, el núcleo duro de la llamada *ley Sinde*, controvertida norma aprobada en febrero del presente año y que busca frenar las descargas no autorizadas de contenidos protegidos en Internet.

El reglamento regula el funcionamiento de las dos secciones que conforman la *Comisión de Propiedad Intelectual*. La primera se centra en cuestiones de arbitraje extrajudicial de litigios y la segunda, y más discutida por influyentes internautas, juristas y sectores de la industria de contenidos, establece los mecanismos para bloquear (previa autorización judicial) las páginas web que ofrezcan, sin autorización, descargas de contenidos protegidos por derechos de propiedad intelectual.

También en julio de 2011 la *OSCE* (*Organización para la Seguridad y la Cooperación en Europa*), que agrupa más de cincuenta países, publicó un informe sobre la libertad de expresión en Internet donde expuso severas críticas a las políticas que suponen el corte del acceso a Internet a un ciudadano y el bloqueo de sitios web.

El texto, que documenta las prácticas existentes en diferentes países (cita la *ley Hadopi*, en Francia, y la *ley Sinde*, en España), considera que se está imponiendo la medida de bloquear el acceso a un sitio cuando las autoridades no pueden exigir cambios en sus contenidos ya que el mismo se alberga en un tercer país. El informe sostiene que el derecho a la libertad de expresión tiene sus limitaciones pero éstas deben ser definidas de forma clara y restrictiva por parte de las leyes de cada Estado. Añade que, por este motivo, el bloqueo de sitios debería contemplarse como una medida extrema por su impacto en la libertad de expresión y no debería contemplarse como una solución o sanción permanente y, subraya que, en cualquier caso, debe ser producto de una resolución judicial.

Una sentencia de la Audiencia de Barcelona contra *Indice-web.com*, resuelta también en estas fechas, dictaminó que las páginas web que enlazan a contenidos protegidos por los derechos de autor no ponen nada a disposición del público y, en consecuencia, no realizan un acto de comunicación pública. El fallo comprometió a la llamada *ley Sinde*, que pronosticó el cierre rápido de las webs de enlaces por una comisión administrativa, ya que en teoría vulneran, precisamente, el derecho exclusivo de comunicación pública de los creadores, recogido en la *Ley de Propiedad Intelectual* (*LPI*).

No obstante el pasado marzo, la misma sala de la Audiencia Provincial de Barcelona condenó, mediante sentencia recurrida, al titular de la web *Elrincondejesus*, Jesús Guerra Calderón, a indemnizar a la Sociedad General de Autores y Editores (*SGAE*) con 2.041 euros en compensación por las cantidades dejadas de percibir por "facilitar la descarga directa de obras musicales de su repertorio" con enlaces a servidores como *Megaupload*. Asimismo, instaba a la web a que retire los enlaces, imponiéndole el pago adicional a *SGAE* de 1.546 euros por los gastos de investigación.

Y así, a lo largo de la historia, las demandas solo han llevado a los usuarios habituales a cambiar a programas P2P con tecnologías más difíciles de detener, y el número de usuarios, lejos de disminuir, cada vez es mucho mayor.

3. OBJETIVOS DEL PROYECTO

El proyecto va a consistir en la implementación de una aplicación de ocio y entretenimiento mediante P2P, basándose en el entorno de desarrollo .NET y en el módulo *PeerChannel* de *Windows Communication Foundation*.

Para dicho desarrollo, la primera parte del proyecto consiste en la familiarización con un nuevo lenguaje de desarrollo integrado en la plataforma Visual Studio: C#.NET. Se ha optado por este lenguaje ya que existe gran documentación sobre él, y es capaz de acceder a todas las librerías que proporciona .NET.

Este entorno de desarrollo tiene una "orientación a objetos" de la que hasta este momento solo se había obtenido conocimientos teóricos en una o dos asignaturas de la carrera cursada (Ingeniería Técnica en Informática de Gestión), lo cual supone un considerable esfuerzo de aprendizaje, así como un paso más en la aplicación de los conocimientos adquiridos en la misma.

En segundo lugar, una vez adquiridos los conocimientos necesarios para el desarrollo de aplicaciones en dicha plataforma, se implementará una aplicación de colaboración totalmente descentralizada mediante tecnología P2P.

Este software de ocio y entretenimiento, permitirá a los usuarios conectados a la misma red, chatear, intercambiar archivos, realizar descargas, visualizar fotos o cualquier otro archivo de imagen y reproducir audio y video de forma comunitaria.

Cabe decir que los principales objetivos que a posteriori puedo definir como alcanzados son los siguientes:

- Lograr una gran satisfacción por la calidad de la aplicación, en cuanto a funcionalidad, eficiencia, corrección y estética.
- Aprender nuevos lenguajes y herramientas, así como explorar un tipo de arquitecturas y protocolos, para mí, totalmente desconocidos.
- Aplicar e integrar todos los conocimientos, teóricos y técnicos, así como la capacidad de creatividad y originalidad.

4. TECNOLOGÍAS EMPLEADAS

4.1. PROGRAMACIÓN

4.1.1. WINDOWS PRESENTATION FOUNDATION (WPF)

Windows Presentation Foundation (*WPF*) es una tecnología de Microsoft presentada como parte de Windows Vista.

Permite el desarrollo de interfaces de interacción en Windows tomando las mejores características de las aplicaciones Windows y de las aplicaciones web.

WPF ofrece una amplia infraestructura y potencia gráfica con la que es posible desarrollar aplicaciones visualmente atractivas, con facilidades de interacción, que incluyen animación, vídeo, audio, documentos, navegación o gráficos 3D.

Separa, con el lenguaje declarativo XAML y los lenguajes de programación de .NET, la interfaz de interacción de la lógica del negocio, propiciando una arquitectura Modelo Vista Controlador para el desarrollo de las aplicaciones.

WPF se incluye con Windows 7, Windows Vista y Windows Server 2008 y también está disponible para Windows XP Service Pack 2 o posterior y Windows Server 2003.

Los componentes principales de *WPF* se muestran en la ilustración siguiente.

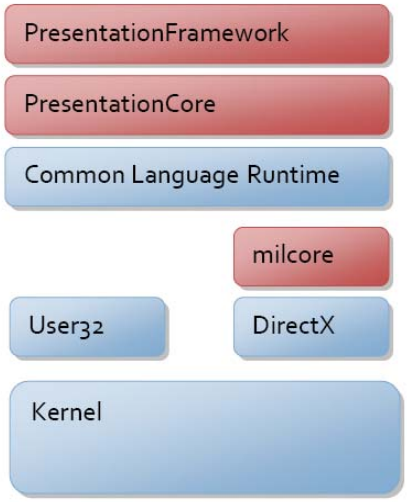


Figura 7. *Componentes principales de WPF*

Las secciones rojas del diagrama (*PresentationFramework*, *PresentationCore* y *milcore*) son los componentes principales del código de *WPF*. De éstos, sólo uno es un componente no administrado, *milcore*. *Milcore* se ha escrito en código no administrado para conseguir una estrecha integración con DirectX (colección de APIs desarrolladas para facilitar las complejas tareas relacionadas con multimedia en la plataforma Microsoft Windows). La visualización en *WPF* se realiza a través del motor de *DirectX*, lo que permite una presentación eficaz mediante hardware y software. El motor de composición de *milcore* es sumamente sensible al rendimiento, y requirió renunciar a muchas de las ventajas de CLR para obtener mejoras en el rendimiento.

WPF está diseñado para permitir crear sistemas de presentación dinámicos y controlados por datos. Cada parte del sistema está diseñada para crear objetos mediante conjuntos de propiedades que controlan el comportamiento. El enlace de datos es una parte fundamental del sistema y está integrado en cada capa. Las aplicaciones tradicionales crean una presentación y, a continuación, enlazan a algunos datos. En *WPF*, todo lo concerniente al control, cada aspecto de la presentación, está generado por algún tipo de enlace de datos. *WPF* ofrece la posibilidad de crear aplicaciones mucho más ricas y controladas fundamentalmente por los datos.

4.1.2. WINDOWS COMMUNICATION FOUNDATION (WCF)

El desarrollo de aplicaciones distribuidas y orientadas a servicios, requiere la interacción de una amplia variedad de escenarios. Históricamente, la plataforma Windows ha tenido una variedad de tecnologías para dar soporte a este tipo de aplicaciones (*DCOM*, *MSMQ*, *COM+*, *ASMX*, *WSE* entre otros). Pero a medida que han surgido nuevos desafíos siempre ha sido complicado para el desarrollador poder seleccionar que tecnología es más adecuada para cada problema.

A partir del lanzamiento de .NET 3.0 con *Windows Communication Foundation* (*WCF*) la integración y facilidad de programación de esta variedad de tecnologías y el soporte a nuevos estándares se ha hecho muy sencillo para los programadores.

WCF es un modelo de programación unificado para la creación y puesta en marcha de sistemas distribuidos orientados a una arquitectura de servicios.

Integra todas las tecnologías para el desarrollo de aplicaciones distribuidas existentes hasta la fecha: *MSMQ*, *Web Services*, *.Net Remoting*, *Enterprise Services* y *Web Services Enhacements* (*WSE*).

Brinda:

- ⇨ *Web Services* seguros
- ⇨ Aplicaciones distribuidas transaccionables y confiables
- ⇨ Interoperabilidad basada en *Web Services* (*WS-**)
- ⇨ Variedad de transportes
- ⇨ Varias posibilidades para alojar servicios (en IIS, en una Aplicación de WinForms, etc.)
- ⇨ Conjunto de clases accesibles a través del namespace *System.ServiceModel* para desarrollar servicios

Abarca:

- ⇨ Transporte
- ⇨ Sistemas de seguridad
- ⇨ Patrones de mensajería
- ⇨ Sistemas de representación de datos
- ⇨ Modelos de alojamiento

Está disponible en Windows 7 / Windows Vista / XP / Server 2003 / Windows "Longhorn" Server y forma parte de lo que se llama .NET 3.0.

Se podría decir que una aplicación distribuida orientada a objetos es un conjunto de componentes con fuerte acoplamiento ya que se construye a partir de bibliotecas de clases, las cuales tienen dependencias e interactúan llamando a objetos distribuidos. Para esto deben compartir un sistema de tipos de datos común, por tanto no sería la mejor manera de construir una aplicación distribuida.

La orientación a servicios es un complemento de la orientación a objetos donde las comunicaciones son el centro de la arquitectura. Un sistema distribuido orientado a servicios es un conjunto de componentes débilmente acoplados que brindan servicios comunicándose por medio de mensajes.

Los pilares de la Orientación a Servicios (OS) son:

- ☑ **Los límites son explícitos:** los sistemas distribuidos orientados a servicios interactúan enviando mensajes a través de límites formales y explícitos, no se debe considerar lo que hay detrás (la implementación) ya que esto es justamente lo que preserva la flexibilidad. Cruzar éstos límites tiene un costo en tiempo de procesamiento y en las comunicaciones. Por esto las llamadas a servicios deben ser simples y eficientes. Cada interacción implica un cruce de límites.
- ☑ **Los servicios son autónomos:** los servicios se despliegan, administran y versionan en forma independiente. Los servicios no deberían fallar si cualquier servicio dependiente no está disponible. No existe una entidad controladora de servicios. Deben tener contemplado el manejo de los errores, ya sean internos propios del servicio, o errores generados por los clientes por mal formación de los mensajes, etc.
- ☑ **Los servicios comparten Esquemas y Contratos, no clases y tipos:** los servicios utilizan contratos para pasar datos y especificar comportamientos, no pasan clases o tipos. Los servicios no tienen ningún conocimiento del entorno de ejecución del cliente u otro servicio, ya que la información que se intercambia es independiente de la plataforma.
- ☑ **Compatibilidad basada en políticas:** cada servicio brinda una política, es decir, una descripción de sí mismo y de sus capacidades y requerimientos. Esto permite que dos sistemas encuentren un área común para poder interactuar. Las políticas separan las interacciones que puede tener, de las restricciones.

Algunos de los beneficios más importantes de los servicios son:

- ✓ **Aislamiento:** la modificación del comportamiento de un servicio no implica modificar o recompilar otros servicios o clientes.
- ✓ **Independencia de ubicación:** es transparente si un servicio se ejecuta localmente o en un equipo remoto, la forma de utilización es la misma.
- ✓ **Flexibilidad de transporte:** en cuanto a protocolo y formato de representación, ya que existen varias alternativas o incluso pueden crearse en forma personalizada.
- ✓ **Independencia de la plataforma e implementación:** no necesitamos conocer el contexto de ejecución del servicio para poder interactuar con él.
- ✓ **Independencia del tiempo:** un servicio no necesita estar en línea para poder interactuar con él. Por ejemplo si utilizamos *MSMQ*.

Las capas que componen una aplicación con *WCF* y algunos de los puntos más sobresalientes de cada una son:

1. **Capa de Contratos:** define los puntos en común que tienen que tener los servicios y clientes para poder interactuar.
   - **Data Contract:** define los objetos y estructuras complejas comunes.
   - **Message Contract:** define cuál información viaja en la cabecera y cuál en el cuerpo de los mensajes.
   - **Service Contract:** define el comportamiento que el servicio expondrá al entorno.
   - **Policy y Binding:** define las características del canal de comunicaciones y requisitos que hay que cumplir para poder interactuar con el servicio.
2. **Capa de ejecución del servicio:** esta capa describe el comportamiento del servicio cuando está operando. Opera principalmente sobre el cuerpo de un mensaje recibido.
   - **Throlling behavior:** define la cantidad de mensajes que podrá procesar el servicio.
   - **Error behavior:** define que hacer en caso de error y como se lo informa al cliente.
   - **Metadata behavior:** define como el servicio publicará una descripción de sí mismo.
   - **Instante behavior:** define la forma en que se instanciará el servicio propiamente dicho, es decir la clase que implementa un contrato.
   - **Transaction behavior:** define las operaciones que podrán formar parte de transacciones y realizar un rollback en caso de error.
   - **Dispatch:** controla como WCF procesará un mensaje.
   - **Message inspection:** permite inspeccionar las distintas partes de un mensaje.
   - **Parameter filtering:** permite que ocurra una acción predefinida según el contenido de la cabecera de los mensajes.

3. **Capa de Mensajería:** esta capa está formada por canales de comunicaciones y opera sobre la cabecera y el cuerpo del mensaje. Permite formar un stack con características de un canal de comunicaciones acorde a los requerimientos de nuestro sistema.
   - **WS-Security Channel:** implementación de la especificación de WS que permite seguridad a nivel de mensaje.
   - **WS-Reliable Messaging Channel:** implementación de la especificación de WS que permite asegurar la entrega de mensajes.
   - **Encoders:** define el conjunto de formatos de representación de datos que podemos utilizar.
   - **Http Channel:** define que el transporte de mensajes se realizará a través del protocolo Http.
   - **TCP Channel:** define que el transporte de mensajes se realizará utilizando el protocolo Tcp.
   - **Transaction Flow Channel:** define el patrón de los mensajes para posibilitar transacciones.
   - **NamedPipe Channel:** posibilita la comunicación entre procesos a través de tuberías.
   - **MSMQ Channel:** posibilita la interacción con un servicio a través de una cola de mensajes.
4. **Hosting y Activación:** esta capa define las posibles formas de ejecutar y activar un servicio
   - **Self-Hosted:** a través de un archivo ejecutable (.exe).
   - **Administrado:** a través de un agente externo (Internet Información Service o Windows Activation Service).
   - Servicio de Windows.
   - Los componentes COM+ pueden ser alojados como un servicio WCF.

Algunos conceptos de WCF son:

- **Servicio:** es un programa que expone sus capacidades a través de una colección de Endpoints y su descripción.
- **Descripciones del servicio:** se publican utilizando WSDL, XSD, WS-Policy y WS-MetaDataExchange.
- **Cliente:** es un programa que intercambia mensajes con uno o más Endpoints y también puede exponer sus propios Endpoints. Un cliente debe conocer la Dirección, el Contrato y el Binding de un servicio para poder comunicarse con él.
- **Mensaje:** representa la unidad de intercambio de datos entre un Cliente y un Servicio.

  Podemos encontrar los siguientes patrones:

  - **Simplex:** mensajes de una sola vía donde no se espera respuesta alguna.
  - **Duplex:** mensajes de dos vías de modo asíncrono.
  - **Request-Reply:** mensajes de dos vías de modo síncrono, es decir, se envía el mensaje y se aguarda por la respuesta.

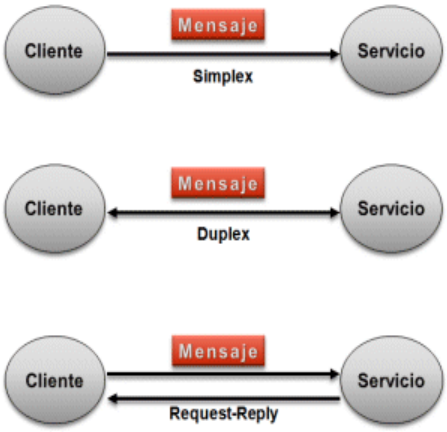


Figura 8. *Patrones de mensajes*

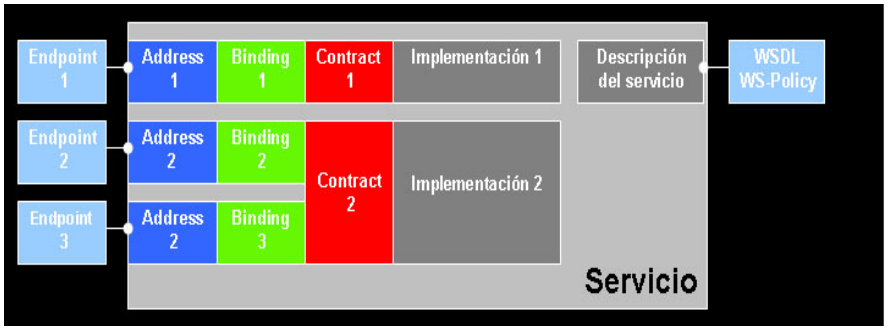


Figura 9. Estructura de un servicio

Un servicio se accede a través de un Endpoint (Punto de Conexión) que es el medio a través del cual interactúa con su entorno. Está formado por una dirección, un enlace y un contrato, lo que se llama habitualmente el ABC (Address, Binding, Contract) del servicio.

- **Address:** es la dirección de red del Endpoint, o URI (Uniform Resource Identifier). Es una forma de especificar una dirección de un recurso, está formada por un protocolo (Http, net.TCP, net.Pipe, net.p2p, net.msmq) seguido del nombre del host, puerto y path:

  Protoclo://Host [:Puerto] [/path/[.../]]

- **Binding:** especifica como un servicio se comunica con el entorno. Permite crear un stack de características para el canal de comunicaciones acorde a los requerimientos del servicio. El canal no se abre hasta que el servicio reciba el primer mensaje. Permite definir:
  - El medio de transporte (HTTP, TCP, MSMQ, etc).
  - La codificación de los datos (texto, MTOM, binario).
  - La seguridad requerida (HTTP, SOAP, Windows NTLM/Kerberos, user/pass, etc).
  - La confiabilidad de las comunicaciones (sesión).
  - El patrón de mensajes (Simplex, Duplex o Request-Reply).
  - La interoperabilidad deseada (Basic WS 1° generación, WS-*, .Net, MSMQ).

Binding	Intero-perabilidad	Seguridad (Default)	Sesión	Transacción (Default)	Duplex	Codificación
BasicHttpBinding	Basic Profile 1.1	(Ninguno), Transporte, Mensaje, Mixto	No	No	n/a	Texto, (MTOM)
WSHttpBinding	WS	Transporte, (Mensaje), Mixto	Si	(No) Si	n/a	Texto, (MTOM)
WSDualHttpBinding	WS	(Mensaje), Ninguno	Si	(No) Si	Si	Texto, (MTOM)
WSFederationHttpBinding	WS - Federation	(Mensaje), Mixto, Ninguno		(No) Si	No	Texto, (MTOM)
NetTcpBinding	.Net	(Transporte), Mensaje, Ninguno,Mixto	SI	(No) Si	Si	Binaria
NetNamedPipeBinding	.Net	(Transporte), Mensaje, Ninguno,Ambos	Si	(No) Si	Si	Binaria
NetMsmqBinding	.Net	Ninguno,(Transporte)	No	(No) Si	No	-
NetPeerTcpBinding	Peer	Mensaje, (Transporte), Ambos	No	(No)	Si	-
MsmqIntegrationBinding	MSMQ	(Transporte)	No	(No) Si	n/a	-

Figura 10. *Tipos de bindings disponibles*

- **Contract:** es una colección de operaciones; especifica lo que el Endpoint puede comunicar al entorno. Los tipos de contratos que podemos especificar son:
  - **Service Contract:** es una interfaz que implementa un servicio. Se publica utilizando WSDL. El Operation Contract son los métodos que forman parte de la interfaz, y que podrán ser accedidos por los clientes del servicio.
  - **Data Contract:** define las clases y estructuras complejas que puede retornar o recibir un servicio. Es implícito si el tipo de dato es serializable, es decir el tipo de dato se puede pasar directamente sin declararlo en el contrato. Es explícito en caso contrario y se debe especificar en un data contract (Serializacion/Deserializacion). Se publica utilizando un XML Schema. El Member Contract son los atributos o propiedades del objeto que forma parte de un data contract y podrán ser serializados.
  - **Message Contract:** especifica la estructura, cabecera y cuerpo, del mensaje SOAP (Message Header y MessageBody). Nos permite definir si un atributo de una clase viaja en la cabecera o en el cuerpo de un mensaje.

El **Service Behavior** representa las distintas características de ejecución del servicio en ejecución y se especifica como parámetros del contrato, en archivos de configuración o mediante la codificación. Permite manejo de errores, diferentes formas de instanciación y manejo de concurrencia entre otras particularidades.

- **Instanciación:** determina cómo se instancian las clases que implementan un contrato.
  - **Singleton:** una instancia atiende a todos los clientes.
  - **Por llamada:** se crea una instancia de clase por cada mensaje de petición y luego se destruye.
  - **Sesión privada:** se crea una instancia de clase por cada solicitud de un cliente.
  - **Sesión compartida:** igual que la anterior pero los clientes pueden compartir una instancia.
- **Concurrencia:** determina cómo se relacionan las instancias con los Threads.
  - **Simple:** especifica que el código del objeto servicio no puede ser ejecutado por más de un thread al mismo tiempo.
  - **Múltiple:** especifica que el código del objeto servicio puede ser ejecutado por múltiples threads al mismo tiempo por lo que se deben establecer mecanismos de exclusión mutua.
  - **Reentrante:** especifica que el código del objeto servicio no puede ser ejecutado por más de un thread al mismo tiempo.
- **Throlling:** define condiciones de performance del servicio.
- **Manejo de Errores:** los errores pueden ser procesados por el programador o por el framework. También se puede reportar errores al cliente.
- **Metadata:** establece la publicación de los metadatos que brindan la descripción del servicio.

- **Tiempo de vida:** un servicio puede controlar el tiempo de vida de las sesiones de un cliente, especificando las operaciones de inicio y fin de sesión.
- **Seguridad:** establece la confidencialidad e integridad de los mensajes, autenticación, autorización, auditoría y detección de reintentos.
- **Transacciones:** el programador especifica si un servicio aceptará transacciones y el cliente controla el alcance y el tiempo de vida de una transacción.

4.1.3. C#

El nombre *C Sharp* fue inspirado por la notación musical, donde ♯ (sostenido, en inglés *sharp*) indica que la nota (*C* es la nota *do* en inglés), es un semitono más alta, sugiriendo que C# es superior a C/C++. Además, el signo '#' viene de cuatro '+' pegados.

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270).

Es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común (CLI [13]), que es una especificación estandarizada que describe un entorno virtual para la ejecución de aplicaciones, cuya principal característica es la de permitir que aplicaciones escritas en distintos lenguajes de alto nivel puedan luego ejecutarse en múltiples plataformas, tanto de hardware como de software, sin necesidad de reescribir o recompilar su código fuente.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco *Mono - DotGNU*, el cual genera programas para distintas plataformas como Windows, Unix y GNU/Linux.

[13] Common Language Infrastructure – Infraestructura de Lenguaje Común

Entre las características de C# destacan:

- ✓ Provee el beneficio de un ambiente elegante y unificado.
- ✓ No soporta herencia múltiple, solamente el runtime .NET permite la herencia múltiple en la forma de interfaces, las cuales no pueden contener implementación.
- ✓ No maneja apuntadores, para emular la función de los apuntadores se utilizan *delegates* que proveen las bases para el *.NET event model*.
- ✓ Por defecto trabaja con código administrado.
- ✓ La Plataforma .NET provee un colector de basura que es responsable de administrar la memoria en los programas C#.
- ✓ El manejo de errores está basado en excepciones.
- ✓ Soporta conceptos como encapsulación, herencia y polimorfismo de la programación orientada a objetos.
- ✓ El modelo completo de clases está construido en la cima del .NET *Virtual Object System* (*VOS*). El modelo de objetos es parte de la infraestructura y ya no forma parte del lenguaje de programación.
- ✓ No existen funciones globales, variables o constantes. Todo deber ser encapsulado dentro de la clase, como un miembro de la instancia (accesible vía instancia de clase) o un miembro estático (accesible vía tipo).
- ✓ Los métodos que se definen en las clases son por defecto no virtuales (no pueden ser sobre escritos al derivar clases)
- ✓ Soporta los modificadores de acceso private, protected, public y agrega un cuarto modificador, internal.
- ✓ Solamente se permite una base clase, si se requiere herencia múltiple es posible implementar intefaces.
- ✓ No es posible utilizar variables no inicializadas.
- ✓ No es posible hacer el cast de un entero a un tipo de referencia (objeto).
- ✓ Los parámetros que son pasados son *type-safe (*tipo seguro).

- ✓ El soporte de versiones lo provee el *CLR*[14].
- ✓ Permite acceder a diferentes APIs a través del .NET *Common Language Specification*, el cual define el estándar de interoperabilidad entre lenguajes que se adhieran a este estándar.
- ✓ La Plataforma .NET provee un acceso transparente a *COM*[15].
- ✓ Soporta *OLE Automation* (mecanismo formal de comunicación entre procesos basado en *COM*)
- ✓ Permite la interoperabilidad con APIs al estilo *C* y *DLLs*[16]; esta característica para acceder a APIs nativas es llamada *Platform Invocation Services* (*PInvoke*)
- ✓ Por defecto el código es *safe-mode*, pero es posible declarar clases o sólo métodos *unsafe*, esta declaración permite utilizar apuntadores, estructuras y almacenamiento de arreglos estáticos.
- ✓ C# depende del *runtime* que provee la Plataforma .NET, el *runtime* administra la ejecución de código.

[14] Common Language Runtime - Lenguaje Común en Tiempo de Ejecución

[15] Component Object Model - Modelo de Objetos de Componentes

[16] Dynamic Link Library - Biblioteca de Enlace Dinámico

4.1.4. PLATAFORMA .NET

.NET es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permite un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de *Oracle Corporation* y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones - o como la misma plataforma las denomina, soluciones - permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo. Esta solución es el producto principal en la oferta de Microsoft, y pretende ser utilizada por la mayoría de las aplicaciones creadas para la plataforma Windows.

.NET Framework se incluye en Windows Server 2008, Windows Vista y Windows 7. De igual manera, la versión actual de dicho componente puede ser instalada en Windows XP, y en la familia de sistemas operativos Windows Server 2003. Una versión reducida de .NET Framework está disponible para la plataforma Windows Mobile, incluyendo teléfonos inteligentes.

Los principales componentes del marco de trabajo son:

- ☑ **El conjunto de lenguajes de programación.** Debido a la publicación de la norma para la i*nfraestructura común de lenguajes* (*CLI*), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta ya más de 20 lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje.

Algunos de los lenguajes desarrollados para el marco de trabajo .NET son: C#, Visual Basic .NET, Delphi (Object Pascal), C++, F#, J#, Perl, Python, Fortran, Prolog (existen al menos dos implementaciones, el P# y el Prolog.NET), Cobol y PowerBuilder.

- ☑ **La Biblioteca de Clases Base o BCL.** La *Biblioteca de Clases Base* (*BCL*[17]) maneja la mayoría de las operaciones básicas que se encuentran involucradas en el desarrollo de aplicaciones. Incluyen soporte para cualquiera de los archivos de entrada/salida y base de datos de entrada/salida para XML y SOAP.

  La *Biblioteca de Clases Base* se clasifica en cuatro grupos clave:

  - ✓ ASP.NET y Servicios Web XML
  - ✓ Windows Forms
  - ✓ ADO.NET
  - ✓ .NET

  La funcionalidad del *BCL* incluye:

  - ✓ Clases colección, tales como consultas, arreglos, pilas y tablas *hash*.
  - ✓ Clases de acceso a bases de datos
  - ✓ Clases IO (*input-output*)
  - ✓ Clases WinForms, para crear interfaces de usuario
  - ✓ Clases Network

- ☑ **El Lenguaje Común en Tiempo de Ejecución o CLR, fundamento de la Plataforma .NET.** El *Lenguaje Común en Tiempo de Ejecución* .NET (*CLR*) es un ambiente basado en componentes y C# está diseñado para facilitar la creación de componentes.

  Todos los objetos son escritos como componentes y los componentes son el centro de acción, por ello que reciba el nombre de lenguaje céntrico de componentes (*component-centric*).

[17] Base Class Libraries - Biblioteca de Clases Base

Los componentes creados en C# son totalmente auto-describibles y pueden ser utilizados sin un proceso de registro. C# ayuda en la creación de componentes mediante el *runtime* y *framework* .NET, los cuales proveen un sistema de tipo unificado en el cual todo puede ser tratado como un objeto.

El *runtime* es una agente que administra el código en tiempo de ejecución al proveer de servicios principales como la administración de memoria, administración de hilos, implementación de tipos estrictos de seguridad y otras formas de verificación de código que aseguran la seguridad y robustez.

Es, por tanto, un entorno administrado o controlado, lo cual significa que administra la memoria por el programador a través del recolector de basura y un entorno verificado, ya que en tiempo de ejecución el entorno verifica que la ejecución del código sea de tipo segura (*type-safe*). En el *runtime* .NET todos los errores son reportados vía excepciones.

La plataforma .NET puede almacenar componentes no administrados que son cargados por el *CLR* en sus procesos e inicializados por la ejecución de código administrado, de esta manera se crea un ambiente de software que puede explotar tanto las características de código administrado como las de código no administrado.

4.2. PLATAFORMAS DE DESARROLLO

4.2.1. GANTTPROJECT

El *diagrama de GANTT* es una herramienta que le permite al usuario modelar la planificación de las tareas necesarias para la realización de un proyecto. Esta herramienta fue inventada por Henry L. Gantt en 1917.

Debido a la relativa facilidad de lectura de los *diagramas de GANTT*, esta herramienta es utilizada por casi todos los directores de proyecto en todos los sectores. El *diagrama de GANTT* es una herramienta para el director del proyecto que le permite realizar una representación gráfica del progreso del proyecto, pero también es un buen medio de comunicación entre las diversas personas involucradas en el proyecto.

Este tipo de modelo es particularmente fácil de implementar con una simple hoja de cálculo, pero también existen herramientas especializadas como *GanttProject*.

GanttProject permite planificar proyectos mediante *diagramas de GANTT*. Cada proyecto puede dividirse en subtareas, con su propia fecha de comienzo, duración, dependencias, progreso y anotaciones. Almacena la información en formato XML y puede exportar imágenes PNG/JPG, páginas web o documentos PDF. Es un programa Java, por lo que se necesita tener instalado *Java Runtime Environment 1.4.1* (o posterior).

4.2.2. VISUAL PARADIGM PARA UML

El *Modelado Visual* es el proceso que permite representar gráficamente el sistema software, permitiendo resaltar los detalles más importantes.

Un buen modelo:

- ✓ Identifica requisitos y comunica información.
- ✓ Se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos.
- ✓ Permite ver las relaciones entre los componentes del diseño.
- ✓ Mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico.

Visual Paradigm es una herramienta *CASE*[18] que da soporte al modelado visual con UML[19] 2.1

Ofrece:

- ☑ Entorno de creación de diagramas para UML 2.1.
- ☑ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ☑ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ☑ Capacidades de ingeniería directa e inversa.
- ☑ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ☑ Disponibilidad de múltiples versiones, para cada necesidad.
- ☑ Disponibilidad de integrarse en los principales IDEs[20].
- ☑ Disponibilidad en múltiples plataformas.

---

[18] Computer Aided Software Engineering - Ingeniería de Software Asistida por Computadora

[19] Unified Modeling Language - Lenguaje Unificado de Modelado

[20] Integrated Development Environment - Entorno de Desarrollo Integrado

4.2.3. VISUAL STUDIO 2010

Microsoft Visual Studio es un entorno de desarrollo integrado (*IDE*) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Visual Studio 2010 es la versión más reciente de esta herramienta, acompañada por *.NET Framework 4.0.* La fecha de lanzamiento de la versión final fue el 12 de abril de 2010.

Hasta ahora, uno de los mayores logros de la versión 2010 de *Visual Studio* ha sido el de incluir las herramientas para el desarrollo de aplicaciones para Windows 7 - tales como herramientas para el desarrollo de las características de Windows 7 (*System.Windows.Shell*) - y la *Ribbon Preview* para *WPF*.

Entre sus más destacables características, se encuentran la capacidad para utilizar múltiples monitores, así como la posibilidad de desacoplar las ventanas de su sitio original y acoplarlas en otros sitios de la interfaz de trabajo. Además de esto, aparece una edición que compila las características de todas las ediciones comunes de *Visual Studio* (*Professional*, *Team Studio* y *Test)*, conocida como *Visual Studio Ultimate*.

5. PROCESO DE DESARROLLO

5.1. METODOLOGÍAS ÁGILES DE DESARROLLO E INGENIERÍA DEL SOFTWARE (*MADIS*)

Los modelos del proceso, ciclos de vida o paradigmas de ingeniería de software más reconocidos y aplicados son el modelo de proceso en cascada, el desarrollo evolutivo, el desarrollo formal y los modelos iterativos.

Los modelos iterativos son modelos híbridos entre el modelo en cascada y el modelo evolutivo. Entre los modelos iterativos destacan el desarrollo incremental, el desarrollo en espiral y los procesos ágiles de desarrollo**.**

Las *MADIS* (*Metodologías Ágiles de Desarrollo e Ingeniería de Software*) están construidas sobre la definición del desarrollo iterativo e incremental. Las iteraciones son pequeños proyectos compuestos de varias actividades cuyo objetivo es entregar una parte de un sistema parcialmente completo, probado, integrado y estable. Todo el software es integrado en cada entrega de cada iteración hasta obtener el producto software completo en la última iteración.

Las metodologías ágiles más reconocidas por los especialistas en el tema de desarrollo ágil son *XP*[21] (*eXtreme Programming*), *Scrum*, la familia de metodologías *Crystal* y *FDD*[22] (Feature Driven Development).

La *programación extrema* o *eXtreme Programming* (*XP*) es un enfoque de la ingeniería de software formulado por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change* (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

[21] eXtreme Programming - Programación Extrema

[22] Feature Driven Development - Desarrollo Basado en Funcionalidades

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarla de manera dinámica durante el ciclo de vida del software.

Los principios originales de la programación extrema son: simplicidad, comunicación, retroalimentación (*feedback*) y coraje. Un quinto principio, respeto, fue añadido en la segunda edición de *Extreme Programming Explained*. Los cinco principios se detallan a continuación:

- **Simplicidad.** La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hace que la complejidad aumente exponencialmente. Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece. También se aplica la simplicidad en la documentación, de esta manera el código debe comentarse en su justa medida, intentando, eso sí, que el código esté autodocumentado. Para ello se deben elegir adecuadamente los nombres de las variables, métodos y clases. Los nombres largos no decrementan la eficiencia del código ni el tiempo de desarrollo gracias a las herramientas de autocompletado y refactorización que existen actualmente. Aplicando la simplicidad junto con la autoría colectiva del código y la programación por parejas se asegura que cuanto más grande se haga el proyecto, todo el equipo conocerá más y mejor el sistema completo.

- **Comunicación.** La comunicación se realiza de diferentes formas. Para los programadores el código comunica mejor cuanto más simple sea. Si el código es complejo hay que esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que éstos últimos pronto quedan desfasados con el código a medida que es modificado. Debe comentarse sólo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método. Las pruebas unitarias son otra forma de comunicación ya que describen el diseño de las clases y los métodos al mostrar ejemplos concretos de como utilizar su funcionalidad. La comunicación con el cliente es fluida ya que el cliente forma parte del equipo de desarrollo. El cliente decide que características tienen prioridad y siempre debe estar disponible para solucionar dudas.

- **Retroalimentación (*feedback*).** Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante. Considérense los problemas que derivan de tener ciclos muy largos. Meses de trabajo pueden tirarse por la borda debido a cambios en los criterios del cliente o malentendidos por parte del equipo de desarrollo. El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las pruebas unitarias informan sobre el estado de salud del código. Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código.

- **Coraje o valentía.** Para los gerentes la programación en parejas puede ser difícil de aceptar, porque les parece como si la productividad se fuese a reducir a la mitad ya que solo la mitad de los programadores está escribiendo código. Hay que ser valiente para confiar en que la programación por parejas beneficia la calidad del código sin repercutir negativamente en la productividad.

- **Respeto.** El respeto se manifiesta de varias formas. Los miembros del equipo se respetan los unos a otros, porque los programadores no pueden realizar cambios que hagan que las pruebas existentes fallen o que demore el trabajo de sus compañeros. Los miembros se respetan su trabajo porque siempre están luchando por la alta calidad en el producto y buscando el diseño óptimo o más eficiente para la solución a través de la refactorización del código.

Las características fundamentales del método son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera - el código es revisado y discutido mientras se escribe - es más importante que la posible pérdida de productividad inmediata.
- Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

  La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Cuanto más simple es el sistema, menos tendrá que comunicar sobre éste, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

5.2. JUSTIFICACIÓN DEL PROCESO DE DESARROLLO ELEGIDO

La realidad del proyecto a realizar manifiesta los siguientes factores:

- ⇨ Se basa en una plataforma de desarrollo nueva, y el desarrollo es complejo, con lo que el aprendizaje en el transcurso de la realización del proyecto es continuo, y frecuentemente se descubren nuevos mecanismos para optimizar ciertas partes del desarrollo ya realizado.
- ⇨ Es un desarrollo complejo, en el que las pruebas son constantes, y la retroalimentación con la información recibida de anteriores ciclos es clave para el desarrollo. Esta retroalimentación obliga a veces a redefinir partes del producto.
- ⇨ Aunque a menudo infravalorado, las necesidades que pueden ir surgiendo sobre la marcha pueden redefinir ciertas características del producto, y por tanto de los requisitos, que necesitan ser refinados en sucesivas fases. En un ciclo de vida con etapas cerradas, esto puede suponer un problema.

Por todos estos factores he elegido un desarrollo iterativo e incremental basado en programación extrema y que ofrece las siguientes ventajas:

- ✓ Se divide el desarrollo en pequeños ciclos, que afinan y redefinen posteriores ciclos del desarrollo, optimizando el análisis de los mismos, y produciendo un desarrollo más rápido y con menos posibilidad de errores.
- ✓ La división del trabajo en pequeños ciclos hace más sencilla su finalización al tener objetivos más pequeños y definidos.
- ✓ Si esto lo unimos a la retroalimentación con la información proporcionada por anteriores ciclos, y la disminución de errores que esto provoca, obtenemos también una reducción del tiempo y coste de ejecución del proyecto.

Si bien es cierto que, aunque no se ha seguido la programación extrema en todas sus características fundamentales (principalmente porque soy una única persona y no un equipo), sí se han seguido todos sus principios y características en la medida de lo posible: desarrollo iterativo e incremental, pruebas unitarias continuas, corrección de todos los errores antes de añadir nueva funcionalidad, simplicidad en el código y refactorización para aumentar su legibilidad y mantenibilidad.

6. METODOLOGÍA DE DESARROLLO

6.1. PLANIFICACIÓN INICIAL

6.1.1. DIAGRAMA DE GANTT INICIAL

Nombre	Duración	Fecha de inicio	Fecha de fin
Proyecto Final de Carrera	181	1/10/10	18/06/11
Comienzo	8	1/10/10	14/10/10
Fijar objetivos	1	1/10/10	2/10/10
Búsqueda de bibliografía	7	4/10/10	14/10/10
Aprendizaje	25	14/10/10	19/11/10
Investigación de nuevos conceptos	13	14/10/10	3/11/10
Familiarización con herramietas y software	12	3/11/10	19/11/10
Establecer requisitos	14	19/11/10	11/12/10
Captura de requisitos	7	19/11/10	30/11/10
Especificación de requisitos	7	30/11/10	11/12/10
Fase de Análisis	21	13/12/10	12/01/11
Análisis de requisitos	11	13/12/10	28/12/10
Elaboración de diagramas de análisis	10	28/12/10	12/01/11
Fase de Diseño	28	12/01/11	19/02/11
Diseño de arquitectura	5	12/01/11	19/01/11
Diseño de componentes	12	19/01/11	4/02/11
Elaboración de diagramas de diseño	11	4/02/11	19/02/11
Fase de Implementación	40	21/02/11	16/04/11
Fase de Pruebas	21	18/04/11	17/05/11
Fase de Documentación	6	17/05/11	25/05/11
Manual de instalación	3	17/05/11	20/05/11
Manual de usuario	3	20/05/11	25/05/11
Finalización	18	25/05/11	18/06/11
Escribir memoria definitiva	12	25/05/11	10/06/11
Preparar presentación oral	6	10/06/11	18/06/11

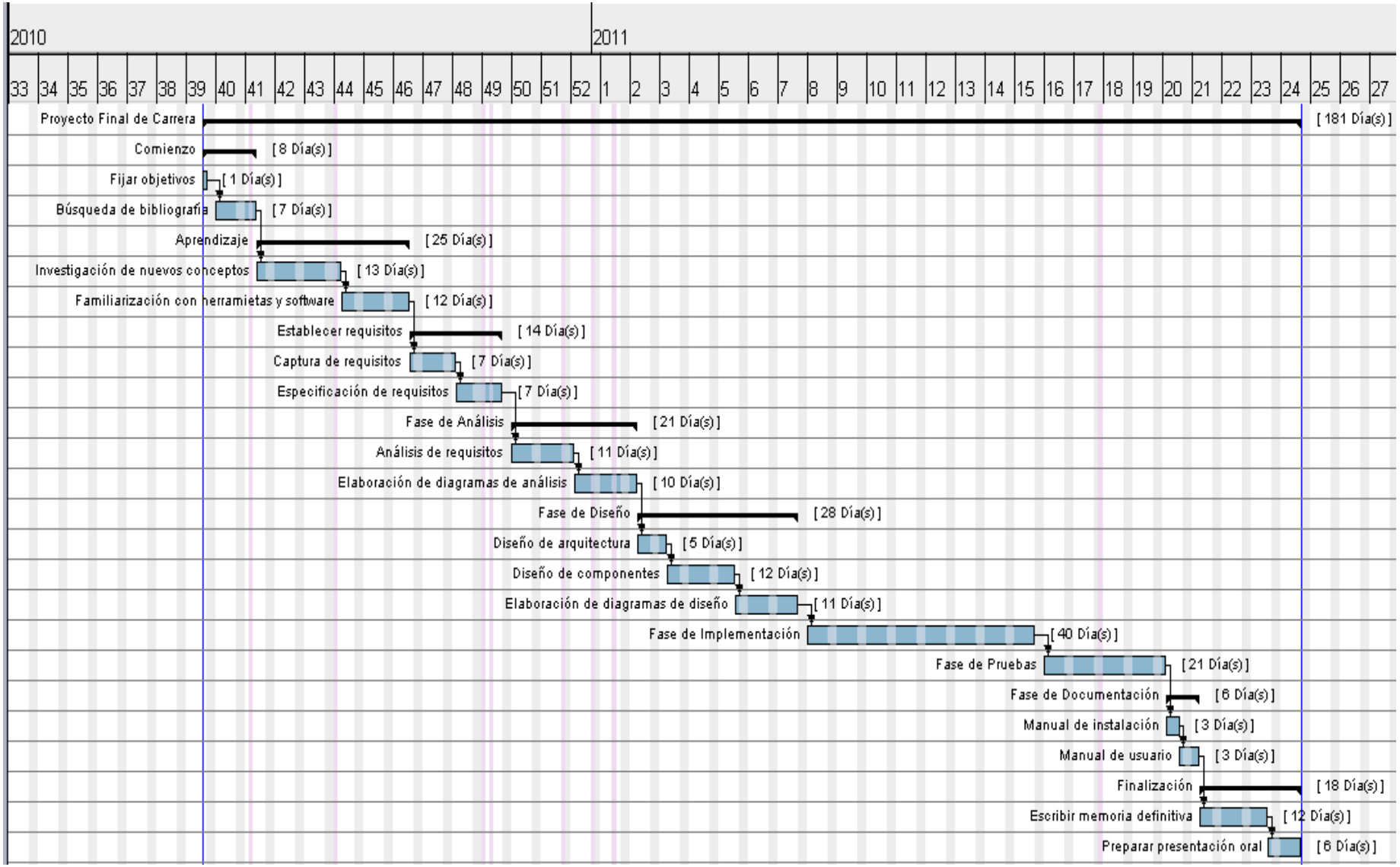


Diagrama de Gantt Inicial

6.2. CAPTURA DE REQUISITOS

Extraer los requisitos y requerimientos de un producto de software es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere de habilidad y experiencia en la ingeniería de software para reconocer requerimientos incompletos, ambiguos o contradictorios.

La captura, análisis y especificación de requerimientos (incluso pruebas de ellos), es una parte crucial; de esta etapa depende en gran medida el logro de los objetivos finales. Se han ideado modelos y diversos procesos de trabajo para estos fines. Aunque aún no está formalizada, ya se habla de la *Ingeniería de requerimientos*, por ejemplo en dos capítulos del libro de Sommerville "Ingeniería del Software" titulados "Requerimientos del software" y "Procesos de la Ingeniería de Requerimientos".

El propósito de la ingeniería de requisitos es hacer que los mismos alcancen un estado óptimo antes de alcanzar la fase de diseño en el proyecto. Los buenos requisitos deben ser medibles, comprobables y sin ambigüedades o contradicciones.

El resultado del análisis de requerimientos con el cliente se plasma en el documento *ERS*, *Especificación de Requerimientos del Sistema*, cuya estructura puede venir definida por varios estándares, tales como *CMMI*[23], modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

Una especificación de requisitos del software es una descripción completa del comportamiento del sistema a desarrollar. Incluye un conjunto de casos de uso que describen todas las interacciones que se prevén que los usuarios tendrán con el software. También contiene requisitos no funcionales (o suplementarios).

[23] Capability Maturity Model Integration - Integración de Modelos de Madurez de Capacidades

Las estrategias recomendadas para la especificación de los requisitos del software están descritas por el estándar IEEE 830-1998. Este estándar describe las estructuras posibles, contenido deseable y calidades de una especificación de requisitos del software.

6.2.1. REQUISITOS FUNCIONALES

Los requisitos funcionales determinan la funcionalidad de la aplicación. Son los comportamiento que el usuario necesita que efectúe el software.

Serán descritos en tablas basadas en la normativa IEEE 830.

RF-000	(Nombre del requisito funcional)
Descripción	(breve descripción del requisito)

Tabla de Requisitos Funcionales

RF – 001	Gestión de la Red
Descripción	Para usar la aplicación el usuario debe autentificarse con un nombre de usuario, una contraseña y un nombre de red. El nombre de usuario y el nombre de red serán obligatorios, mientras que la contraseña se podrá omitir, en cuyo caso el sistema asignará una contraseña por defecto. Tras la validación, esta quedará deshabilitada impidiendo al usuario modificar estos datos. El sistema se conecta a la red (o malla de conexión), identificada por el nombre de red y la contraseña especificados por el usuario en la validación. El sistema habilita las funcionalidades de la aplicación: descargas, chat, intercambio de archivos, visor de imágenes y reproductor multimedia (audio/vídeo). El sistema se desconecta de la red. El sistema deshabilita las funcionalidades de la aplicación: descargas, chat, intercambio de archivos, visor de imágenes y reproductor multimedia (audio/vídeo). El sistema habilita la validación de usuario, haciendo posible una nueva conexión.

Requisito Funcional 1

RF – 002	Gestión de la funcionalidad Chat
Descripción	Permite a los nodos conectados a la misma malla de conexión, enviar y recibir texto entre ellos.

Requisito Funcional 2

RF – 003	Gestión de la funcionalidad Intercambiar archivo
Descripción	Permite a los nodos conectados a la misma malla de conexión, enviar archivos a dicha malla que serán aceptados o rechazados por el resto de usuarios participantes.

Requisito Funcional 3

RF – 004	Gestión de la funcionalidad Visor de imágenes
Descripción	Permite a los nodos conectados a la misma malla de conexión, mostrar archivos de imagen, como por ejemplo fotos, que serán visualizados por el resto de usuarios participantes.

Requisito Funcional 4

RF – 005	Gestión de la funcionalidad Reproductor multimedia de Audio
Descripción	Permite a los nodos conectados a la misma malla de conexión, enviar archivos de audio a dicha malla que serán aceptados o rechazados por el resto de usuarios participantes. El archivo de audio se escuchará en el nodo emisor y en todos los nodos participantes que hayan aceptado el archivo.

Requisito Funcional 5

RF – 006	Gestión de la funcionalidad Reproductor multimedia de Vídeo
Descripción	Permite a los nodos conectados a la misma malla de conexión, enviar archivos de vídeo a dicha malla que serán aceptados o rechazados por el resto de usuarios participantes. El archivo de vídeo se visualizará en el nodo emisor y en todos los nodos participantes que hayan aceptado el archivo.

Requisito Funcional 6

RF – 007	Gestión de la funcionalidad Descargas
Descripción	Permite a los nodos conectados a la misma malla de conexión, buscar y descargar archivos que el resto de usuarios participantes tengan almacenados en sus respectivas carpetas compartidas, creadas, de forma automática, por la aplicación.

Requisito Funcional 7

6.2.2. REQUISITOS NO FUNCIONALES

Los requisitos no funcionales imponen restricciones al diseño o funcionamiento del sistema (tal como requisitos de funcionamiento, estándares de calidad, o requisitos del diseño). Son los "recursos" para que trabaje el sistema de información (redes, tecnología).

Serán descritos en tablas basadas en la normativa IEEE 830.

RNF-000	**(Nombre del requisito no funcional)**
Descripción	(breve descripción del requisito)

Tabla de Requisitos No Funcionales

RNF - 001	**Acceso a la funcionalidad a través de Internet**
Descripción	Es necesaria la conexión a Internet para tener acceso a las funcionalidades de la aplicación.

Requisito No Funcional 1

RNF - 002	**Fiabilidad de la conexión**
Descripción	Es responsabilidad del sistema crear, por cada usuario que se conecte, un enlace seguro punto a punto bajo el protocolo TCP, proporcionando a cada participante una dirección de red única con la que cada nodo podrá buscar a otros nodos, comunicarse con los servicios expuestos por estos y exponer sus servicios para que otros nodos puedan comunicarse con él.

Requisito No Funcional 2

RNF – 003	Protocolo de comunicación TCP
Descripción	La aplicación debe utilizar el protocolo TCP (Protocolo de Control de Transmisión). En la pila de protocolos TCP/IP, TCP es la capa intermedia entre el protocolo de internet (IP) y la aplicación. La aplicación necesita que la comunicación sea fiable y, dado que la capa IP aporta un servicio de datagramas no fiable (sin confirmación), TCP añade las funciones necesarias para prestar un servicio que permita que la comunicación entre dos sistemas se efectúe libre de errores, sin pérdidas y con seguridad. Los servicios provistos por TCP corren en el anfitrión (host) de cualquiera de los extremos de una conexión, no en la red. Por lo tanto, TCP es un protocolo para manejar conexiones de extremo a extremo.

Requisito No Funcional 3

RNF – 004	Acceso simultáneo para un alto número de usuarios
Descripción	La aplicación debe estar optimizada para al acceso simultáneo de un número de usuarios alto.

Requisito No Funcional 4

RNF – 005	Interfaz amigable
Descripción	El diseño de la interfaz ha de ser la adecuada para usuarios que pueden no estar familiarizados con la informática. El uso de las funcionalidades de la aplicación debe ser asequible de realizar para cualquier usuario.

Requisito No Funcional 5

6.3. PLANIFICACIÓN FINAL

6.3.1. DIAGRAMA DE GANTT FINAL

Nombre	Duración	Fecha de inicio	Fecha de fin
Proyecto Final de Carrera	188	3/01/11	24/09/11
Comienzo	8	3/01/11	14/01/11
Fijar objetivos	1	3/01/11	4/01/11
Búsqueda de bibliografía	7	4/01/11	14/01/11
Aprendizaje	32	14/01/11	1/03/11
Investigación de nuevos conceptos	17	14/01/11	8/02/11
Familiarización con herramietas y software	15	8/02/11	1/03/11
Establecer requisitos	14	1/03/11	19/03/11
Captura de requisitos	7	1/03/11	10/03/11
Especificación de requisitos	7	10/03/11	19/03/11
Fase de Análisis	21	21/03/11	19/04/11
Análisis de requisitos	11	21/03/11	5/04/11
Elaboración de diagramas de análisis	10	5/04/11	19/04/11
Fase de Diseño	28	19/04/11	27/05/11
Diseño de arquitectura	5	19/04/11	26/04/11
Diseño de componentes	12	26/04/11	12/05/11
Elaboración de diagramas de diseño	11	12/05/11	27/05/11
Fase de Implementación	40	27/05/11	22/07/11
Fase de Pruebas	21	22/07/11	23/08/11
Fase de Documentación	6	23/08/11	31/08/11
Manual de instalación	3	23/08/11	26/08/11
Manual de usuario	3	26/08/11	31/08/11
Finalización	18	31/08/11	24/09/11
Escribir memoria definitiva	12	31/08/11	16/09/11
Preparar presentación oral	6	16/09/11	24/09/11

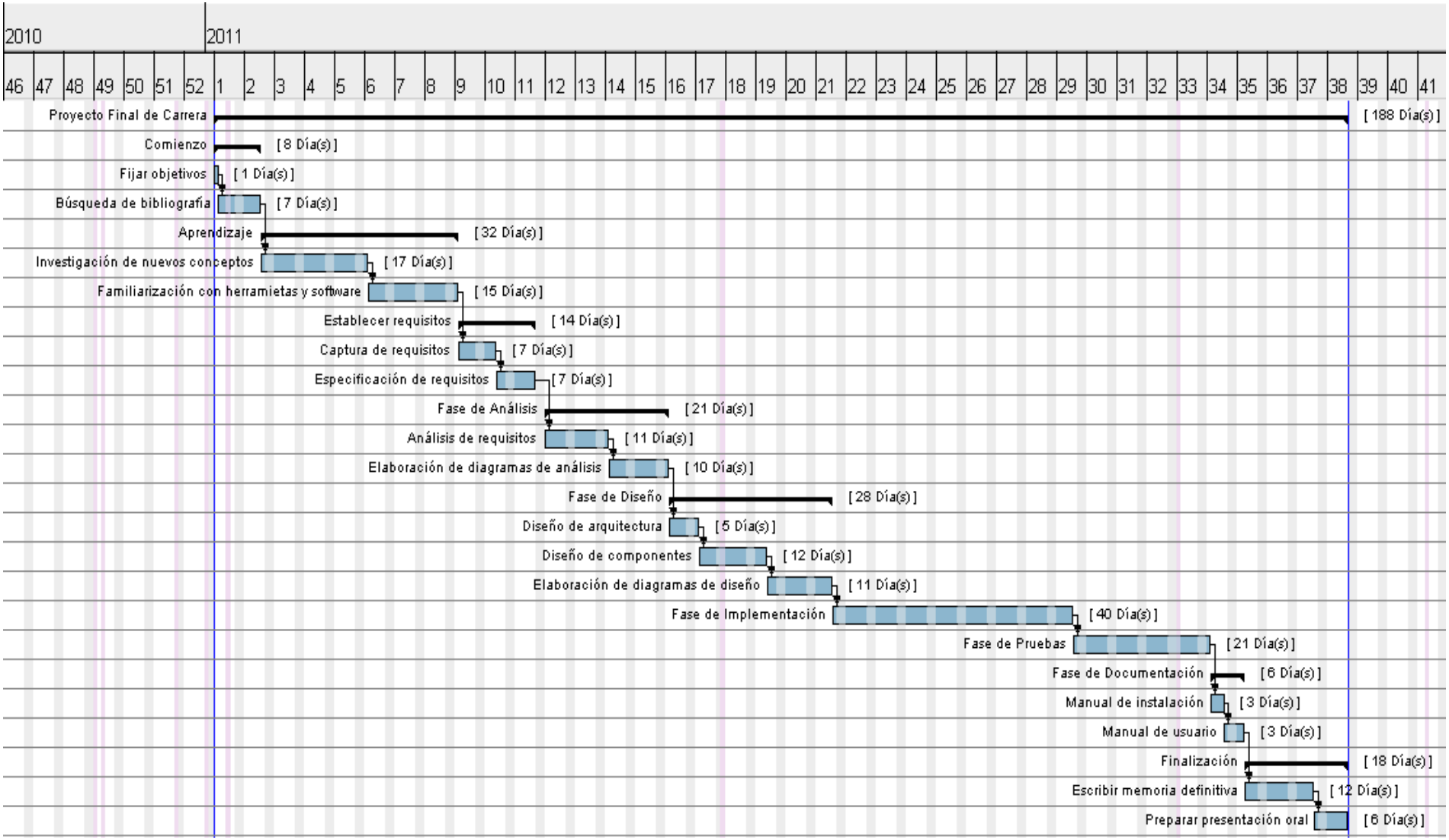


Diagrama de Gantt Final

6.4. ANÁLISIS

Para que el desarrollo de un proyecto de software concluya con éxito, es de suma importancia que antes de empezar a codificar los programas que constituirán la aplicación de software, se tenga una completa y plena comprensión de los requisitos del software.

Roger S. Pressman, internacionalmente reconocido en la mejora de procesos de software y en tecnologías de Ingeniería de software, establece que la tarea del análisis de requisitos es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refina en detalle el ámbito del software, y se crean modelos de los requisitos de datos, flujo de información y control, y del comportamiento operativo. Se analizan soluciones alternativas y se asignan a diferentes elementos del software. El análisis de requisitos permite al desarrollador especificar la función y el rendimiento del software, indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software.

6.4.1. ANÁLISIS DE REQUISITOS

Analizar los requisitos significa detectar y corregir las falencias comunicativas, transformando los requisitos obtenidos en condiciones apropiadas para ser tratados por el diseño.

El análisis o especificación de requisitos describe el comportamiento esperado en el software una vez desarrollado. Gran parte del éxito de un proyecto de software radicará en la identificación de las necesidades del negocio así como la interacción con los usuarios funcionales para la recolección, clasificación, identificación, priorización y especificación de los requisitos.

La IEEE Std. 830-1998 normaliza la creación de las *Especificaciones de Requerimientos de Software* (*Software Requirements Specification*).

Para llevar a cabo el análisis de requisitos se van a elaborar Diagramas de Paquetes, Diagramas de Actores y Diagramas de Casos de Uso.

6.4.2. DIAGRAMA DE PAQUETES

En el *Lenguaje Unificado de Modelado* (*UML*), un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

Los paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes. Con estas líneas maestras sobre la mesa, los paquetes son buenos elementos de gestión. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.

A continuación se muestra el diagrama de paquetes general de la aplicación con el fin de obtener una visión más clara de la lógica del sistema.

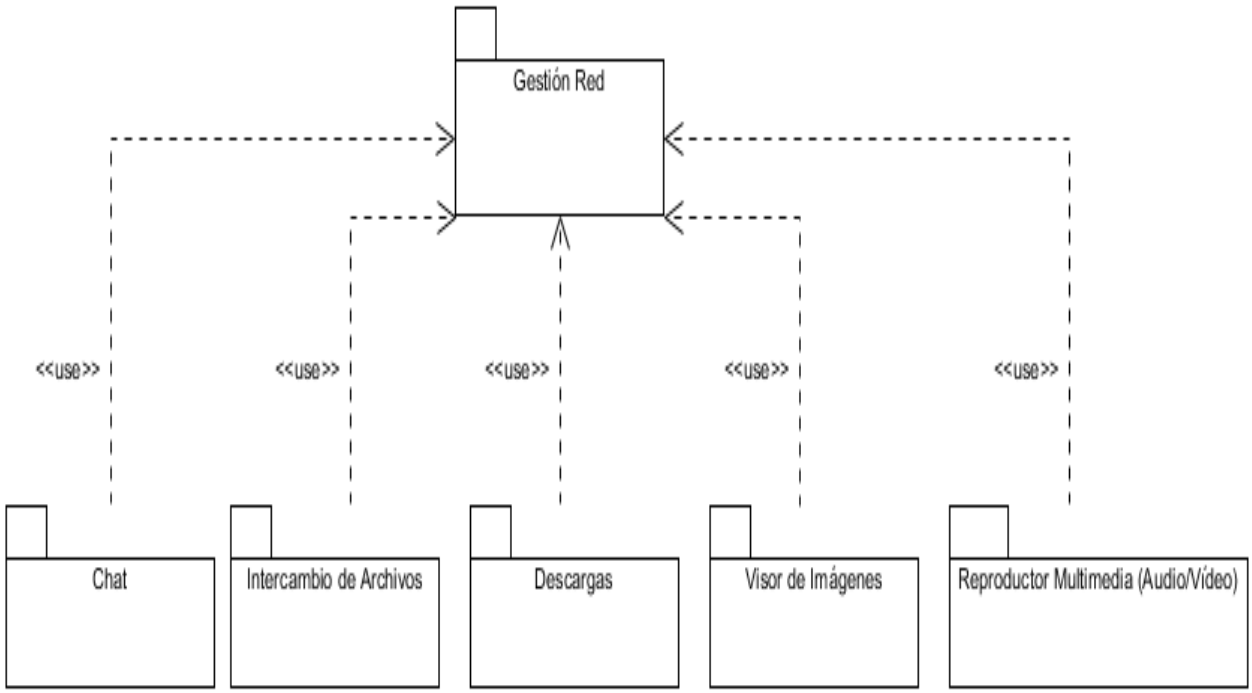


Diagrama de Paquetes GENERAL

6.4.3. DIAGRAMA DE ACTORES

En el *Lenguaje Unificado de Modelado* (*UML*), un actor especifica un rol jugado por un usuario o cualquier otro sistema que interactúa con el sujeto.

Se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Esto incluye a los operadores humanos pero también incluye a todos los sistemas externos, además de entidades abstractas, como el tiempo.

Los actores pueden representar roles jugados por usuarios humanos, hardware externo, u otros sujetos. Un actor no necesariamente representa una entidad física específica, sino simplemente una faceta particular (es decir, un "rol") de alguna actividad que es relevante a la especificación de sus casos de uso asociados. Así, una única instancia física puede jugar el rol de muchos actores diferentes y, asimismo, un actor dado puede ser interpretado por múltiples instancias diferentes.

El sistema estará compuesto por los siguientes actores:

- **Peer:** es un nodo o usuario que se conecta de forma directa a otros nodos o usuarios, formando entre todos una red o malla de conexión friend-to-friend totalmente descentralizada y sin estructura. Los nodos o peers que participan en la misma malla podrán chatear, intercambiar archivos, visualizar fotos, reproducir vídeos, escuchar música o buscar y descargar recursos compartidos por otros nodos de la misma malla.
- **PeerPseudoServidorHttp:** añade a cada peer la funcionalidad de servidor HTTP[24], protocolo usado en cada transacción de la World Wide Web. Así cada nodo podrá actuar tanto de cliente, realizando peticiones HTTP, como de servidor, dando respuesta a solicitudes HTTP.

  Con el “roll” PeerPseudoServidorHttp se pretende dar soporte a las funcionalidades de reproducir vídeo y escuchar audio, puesto que se requiere que el emisor del archivo multimedia actúe como servidor del mismo y los nodos receptores, actúen como clientes.

[24] Hypertext Transfer Protocol - Protocolo de Transferencia de Hipertexto

A continuación se muestra el diagrama de actores del sistema.

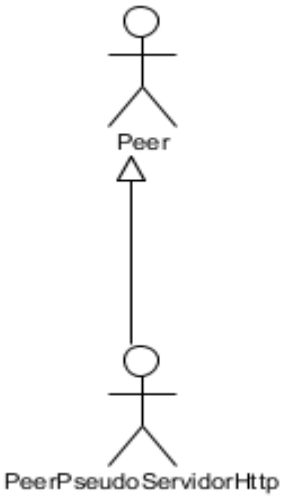


Diagrama de Actores

6.4.4. DIAGRAMAS DE CASOS DE USO

En el *Lenguaje de Modelado Unificado* (*UML*), un Diagrama de Casos de Uso es una especie de diagrama de comportamiento puesto que sirve para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, es un diagrama que muestra la relación entre los actores y los casos de uso en un sistema.

Un Caso de Uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

Los diagramas de casos de uso son a menudo confundidos con los casos de uso. Mientras los dos conceptos están relacionados, los casos de uso son mucho más detallados que los diagramas de casos de uso.

Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona este a eventos que se producen en su ámbito o en él mismo.

La técnica de caso de uso tiene éxito en sistemas interactivos, ya que expresa la intención que tiene el actor (su usuario) al hacer uso del sistema.

Como técnica de extracción de requerimiento permite que el analista se centre en las necesidades del usuario, qué espera éste lograr al utilizar el sistema, evitando que la gente especializada en informática dirija la funcionalidad del nuevo sistema basándose solamente en criterios tecnológicos.

A su vez, durante la extracción de requisitos el analista se concentra en las tareas centrales del usuario describiendo por lo tanto los casos de uso que mayor valor aportan al negocio. Esto facilita luego la priorización del requerimiento.

A continuación se muestran los diagramas de casos de uso del sistema. Estos diagramas son importantes para visualizar, especificar y documentar el comportamiento de cada elemento facilitando que los sistemas, subsistemas y clases sean abordables y comprensibles al presentar una vista externa de cómo pueden utilizarse estos elementos en un contexto dado.

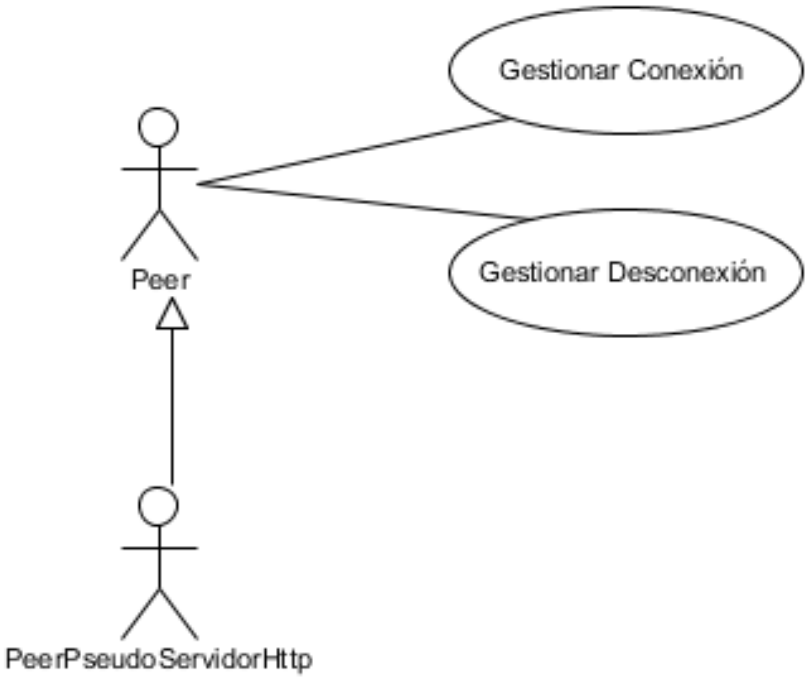


Diagrama de Casos de Uso GESTIÓN RED

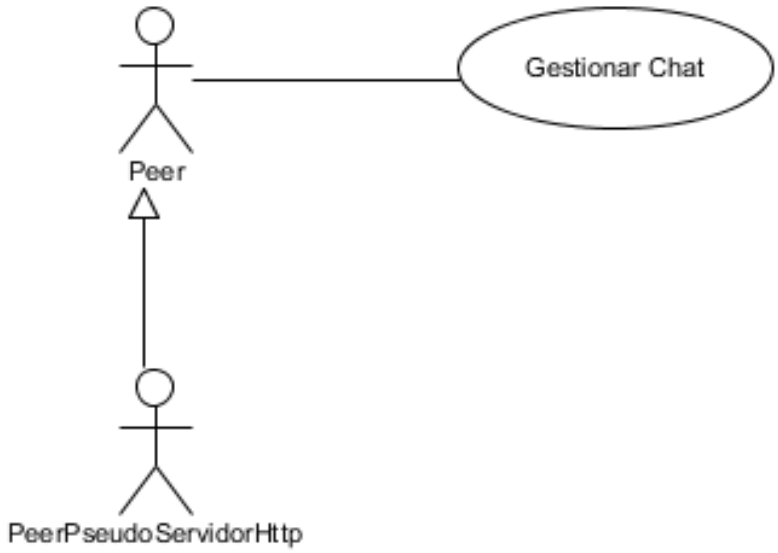


Diagrama de Casos de Uso CHAT

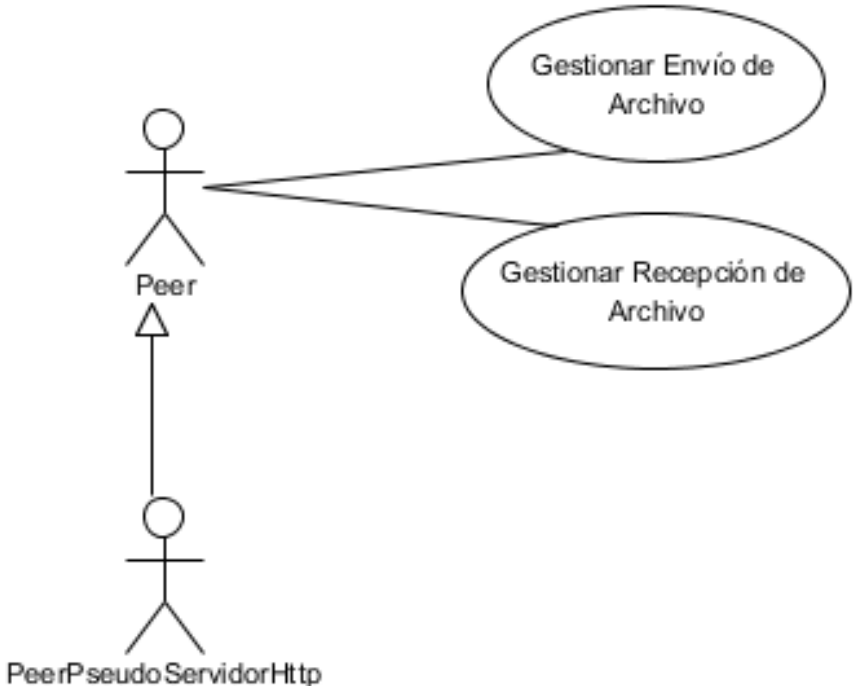


Diagrama de Casos de Uso INTERCAMBIO DE ARCHIVOS

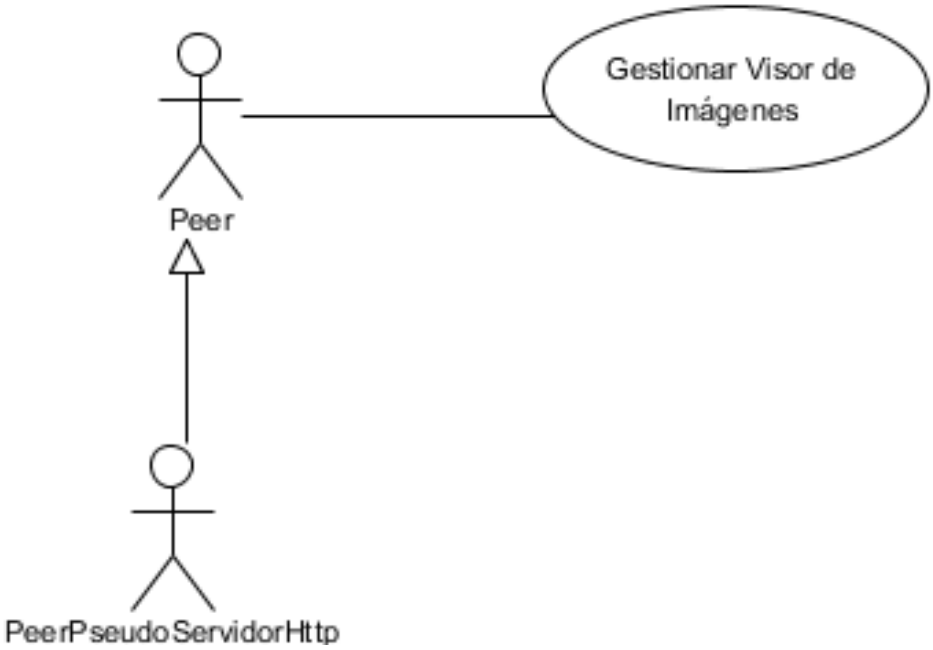


Diagrama de Casos de Uso VISOR DE IMÁGENES

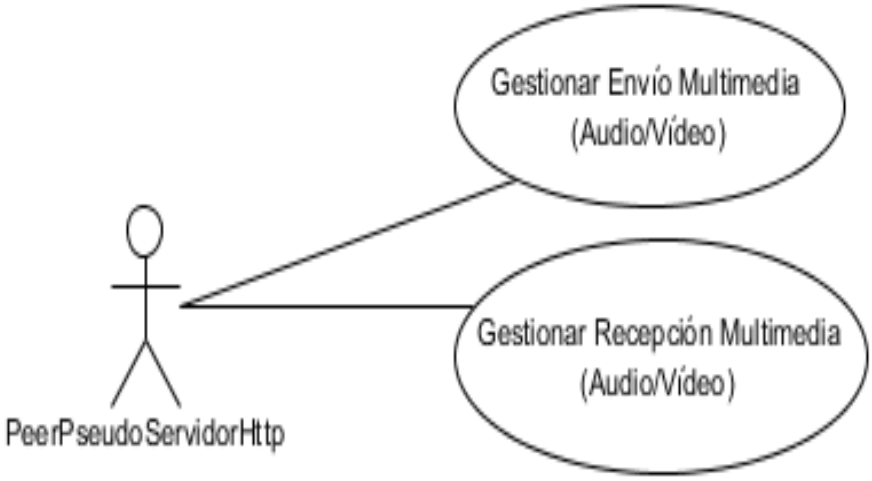


Diagrama de Casos de Uso REPRODUCTOR MULTIMEDIA (AUDIO/VÍDEO)

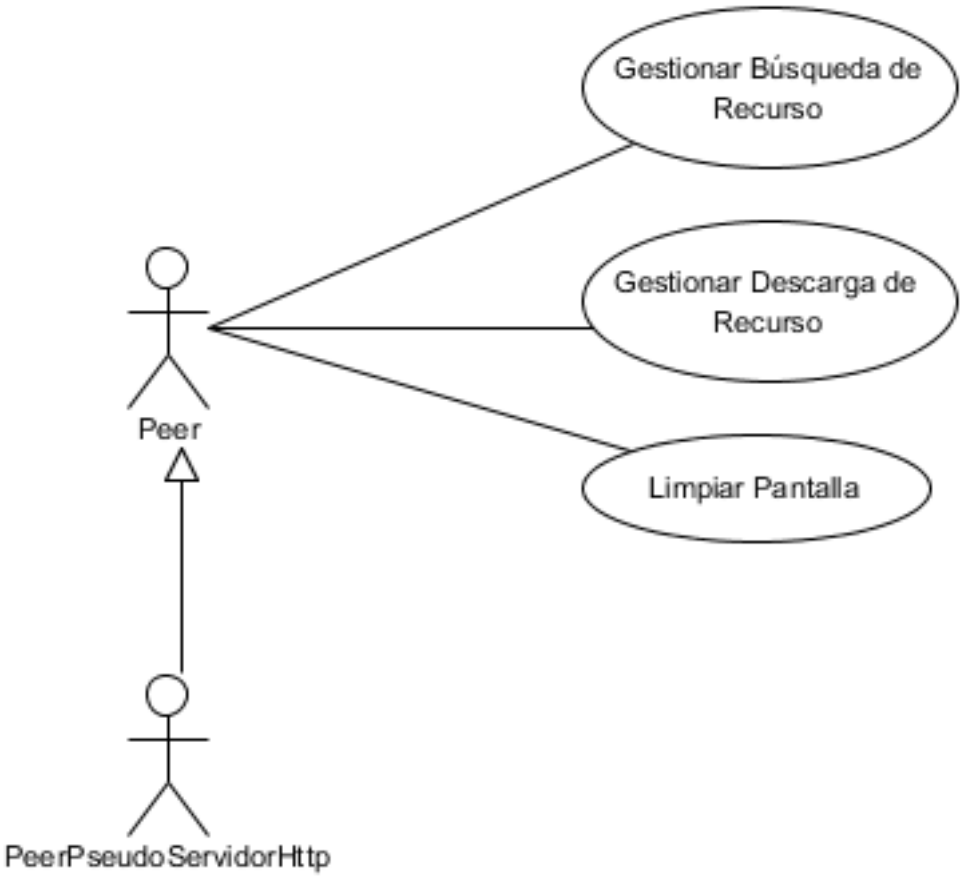


Diagrama de Casos de Uso DESCARGAS

6.4.5. ESPECIFICACIÓN DE CASOS DE USO

El comportamiento de un caso de uso se puede especificar describiendo un flujo de eventos de forma textual, lo suficientemente claro para que alguien ajeno al sistema lo entienda fácilmente. Cuando se escribe este flujo de eventos se debe incluir cómo y cuándo empieza y acaba el caso de uso, cuándo interactúa con los actores y qué objetos se intercambian, el flujo básico y los flujos alternativos del comportamiento.

CU-000	**Nombre del Caso de Uso**
Descripción	Funcionalidad del Caso de Uso
Actores	Usuarios que intervienen en el Caso de Uso
Precondiciones	Condiciones que se deben cumplir para realizar el Caso de Uso
Flujo Normal	Pasos que componen el Caso de Uso
Flujo Alternativo	Pasos que se realizan en caso de producirse una excepción

Tabla de muestra de Caso de uso

CU-001	Gestionar Conexión
Descripción	El usuario se autentifica en la aplicación proporcionando, además de un nombre de usuario, una contraseña y un nombre o identificador de red, que tendrán la finalidad de interconectar en la misma malla a todos los usuarios o nodos asociados a esa contraseña y a ese nombre de red. Una vez que el usuario ha sido validado correctamente el sistema habilita las funcionalidades de la aplicación: descargas, chat, intercambio de archivos, visor de imágenes y reproductor multimedia (audio/vídeo) y deshabilita la validación de usuario, evitando así la posibilidad de modificar estos valores.
Actores	Peer, PeerPseudoServidorHttp.
Precondiciones	Cada usuario, peer o nodo participante debe tener conexión a Internet.
Flujo Normal	1. El usuario introduce un nombre de usuario, una contraseña y un nombre de red en los campos correspondientes en la pantalla de conexión a la red. Tanto el nombre de usuario como el nombre de red serán campos obligatorios, mientras que la contraseña se podrá omitir, en cuyo caso se asignará una contraseña por defecto. Para formar parte de la misma malla de conexión se requiere que el nombre de red y la contraseña sean idénticos en cada nodo participante. Para el correcto funcionamiento de la aplicación, el sistema no permite que los nombres de usuario dentro de una misma malla de conexión sean iguales entre sí. 2. El sistema crea, por cada usuario que se conecta, un enlace seguro punto a punto bajo el protocolo TCP, proporcionando a cada usuario una dirección de red única con la que cada nodo podrá buscar a otros nodos conectados a la misma red, comunicarse con los servicios expuestos por estos y exponer sus servicios para que otros nodos puedan comunicarse con él. 3. El sistema habilita las funcionalidades de la aplicación: descargas, chat, intercambio de archivos, visor de imágenes y reproductor multimedia (audio/vídeo) y deshabilita la validación de usuario, evitando así la posibilidad de modificar estos valores.

Flujo Alternativo	2. 'En caso de que el nombre de usuario ya exista en esa malla de conexión, el sistema no se conectará. Se le informa de esto al usuario y se le sugiere que intente conectarse de nuevo con un nombre de usuario distinto. 3. 'Si el sistema no logra conectarse, las funcionalidades de la aplicación estarán deshabilitadas y la validación de usuario se habilitará permitiendo al usuario conectarse de nuevo. 'En caso de que se produzca algún error durante la conexión y el sistema sea incapaz de recuperarse, la aplicación se cerrará.

Caso de Uso 1

CU-002	**Gestionar Desconexión**
Descripción	El usuario se desconecta de la malla de conexiones.
Actores	Peer, PeerPseudoServidorHttp.
Precondiciones	El usuario debe de estar conectado.
Flujo Normal	1. El usuario se desconecta de la malla de conexiones. 2. El sistema cierra la conexión punto a punto del usuario dejando de exponer sus servicios y haciendo imposible la comunicación con este nodo. 3. El sistema deshabilita las funcionalidades de la aplicación: descargas, chat, intercambio de archivos, visor de imágenes y reproductor multimedia (audio/vídeo) y habilita la validación de usuario permitiendo al usuario conectarse de nuevo.
Flujo Alternativo	2. 'En caso de que se produzca algún error durante la desconexión y el sistema sea incapaz de recuperarse, la aplicación se cerrará.

Caso de Uso 2

CU-003	Gestionar Chat
Descripción	El usuario *chatea* con todos los nodos conectados a su misma red. Las conversaciones se muestran en la pantalla del chat.
Actores	Peer, PeerPseudoServidorHttp.
Precondiciones	El usuario debe estar conectado para que la funcionalidad de chat esté habilitada.
Flujo Normal	1. El usuario chatea con todos los nodos conectados a su misma red, esto es, envía y recibe mensajes de texto. 2. El sistema muestra por pantalla las conversaciones entre los usuarios. 3. El sistema muestra por pantalla el nombre de los usuarios que se unen a la malla de conexiones y el nombre de los que la abandonan.
Flujo Alternativo	2'. 'En caso de que alguno de los nodos pierda la conexión, la comunicación con esos nodos será imposible.

Caso de Uso 3

CU-004	Gestionar Envío de Archivo
Descripción	El usuario selecciona de cualquier directorio de su ordenador el archivo que desea enviar al resto de nodos conectados a su misma malla.
Actores	Peer, PeerPseudoServidorHttp.
Precondiciones	El usuario debe estar conectado para que la funcionalidad de intercambiar archivo esté habilitada.
Flujo Normal	1. El sistema muestra al usuario el árbol de directorios del equipo para que seleccione el archivo de desea enviar. 2. El usuario envía el archivo seleccionado a la malla de conexiones. 3. El sistema muestra información del envío en la pantalla del nodo emisor.
Flujo Alternativo	2'. 'En caso de que el nodo emisor pierda la conexión, el archivo no se enviará a ningún nodo.

Caso de Uso 4

CU-005	Gestionar Recepción de Archivo
Descripción	Cada vez que se produce el envío de un archivo a la malla, el sistema informa al resto de participantes de esa malla sobre el emisor del envío y el nombre del archivo enviado, ofreciendo a estos la posibilidad de aceptar o rechazar el envío.
Actores	Peer, PeerPseudoServidorHttp.
Precondiciones	Los usuarios deben estar conectados para que la funcionalidad de intercambiar archivo esté habilitada y otro nodo participante debe haber enviado un archivo a la malla de conexiones.
Flujo Normal	1. Por cada archivo enviado, el sistema informa, al resto de participantes de la misma malla, sobre el emisor del envío y el nombre del archivo enviado y ofrece a estos la posibilidad de aceptar o rechazar el envío. 2. El sistema muestra, a cada usuario que acepta el archivo, el árbol de directorios del equipo, permitiendo seleccionar la ubicación donde se guardará el archivo aceptado. 3. El sistema almacena el archivo aceptado en el directorio especificado. 4. El sistema muestra información del envío en cada nodo que haya aceptado el archivo.
Flujo Alternativo	3'. 'En caso de que alguno de los nodos participantes pierda la conexión, el archivo no llegará a esos nodos.

Caso de Uso 5

CU-006	Gestionar Visor de Imágenes
Descripción	El usuario selecciona, de cualquier directorio de su ordenador, el archivo de imagen, como por ejemplo una foto, que desea que el resto de nodos conectados a la misma malla visualice. La imagen se mostrará en la pantalla de cada nodo participante.
Actores	Peer, PeerPseudoServidorHttp.
Precondiciones	El usuario debe estar conectado para que la funcionalidad de visor de imágenes esté habilitada.
Flujo Normal	1. El sistema muestra al usuario el árbol de directorios del equipo para que seleccione el archivo de imagen que desea mostrar al resto de nodos conectados a la misma malla. 2. El sistema muestra en la pantalla del nodo emisor la imagen seleccionada. 3. El sistema muestra en las pantallas del resto de nodos conectados a la malla la imagen seleccionada por el emisor.
Flujo Alternativo	2'. 'En caso de que el nodo emisor pierda la conexión, la imagen seleccionada no se visualizará en ningún nodo. 3'. 'En caso de que caiga la conexión de alguno de los otros nodos conectados la malla, la imagen no se visualizará en esos nodos.

Caso de Uso 6

CU-007	Gestionar Envío Multimedia (Audio/Vídeo)
Descripción	El usuario selecciona, de cualquier directorio de su ordenador, el archivo de audio, si se encuentra en la pantalla de funcionalidad de Audio, o de vídeo, si se encuentra en la pantalla de funcionalidad de Video, que desea escuchar o visualizar con el resto de participantes.
Actores	PeerPseudoServidorHttp.
Precondiciones	El usuario debe estar conectado para que las funcionalidades multimedia estén habilitadas.
Flujo Normal	1. El sistema muestra al usuario el árbol de directorios del equipo para que seleccione el archivo multimedia de desea escuchar, pantalla de Audio, o visualizar, pantalla de Video, con el resto de participantes conectados. 2. El usuario envía el archivo multimedia seleccionado a la malla de conexiones. 3. El sistema convierte al nodo emisor en pseudoservidor del archivo multimedia y arranca el servicio de audio o vídeo en este nodo. 4. El sistema muestra información del servicio multimedia en la pantalla del nodo emisor.
Flujo Alternativo	2'. 'En caso de que el nodo emisor pierda la conexión, el servicio multimedia no funcionará.

Caso de Uso 7

CU-008	Gestionar Recepción Multimedia (Audio/Vídeo)
Descripción	Cada vez que se produce el envío de un archivo multimedia a la malla, el sistema informa al resto de participantes de esa malla sobre el emisor del envío y el nombre del archivo enviado, ofreciendo a estos la posibilidad de aceptar o rechazar el archivo multimedia.
Actores	PeerPseudoServidorHttp.
Precondiciones	Los usuarios deben estar conectados para que las funcionalidades multimedia estén habilitadas y otro nodo participante debe haber enviado un archivo multimedia a la malla de conexiones.
Flujo Normal	1. Por cada archivo multimedia enviado, el sistema informa, al resto de participantes de la misma malla, sobre el emisor del envío y el nombre del archivo enviado y ofrece a estos la posibilidad de aceptar o rechazar el envío. 2. El sistema convierte a cada nodo que acepta el archivo multimedia en cliente del servicio y arranca en estos nodos la audición de música, si se está utilizando la funcionalidad de Audio, o la visualización de vídeo, si se está utilizando la funcionalidad de Video. 3. El sistema muestra información del servicio multimedia en la pantalla de cada nodo que haya aceptado el archivo.
Flujo Alternativo	2'. 'Si cae la conexión de alguno de los nodos invitados, el servicio no funcionará en esos nodos.

Caso de uso 8

CU-009	Gestionar Búsqueda de Recurso
Descripción	El usuario envía a todos los nodos conectados a su misma malla una petición de búsqueda con el nombre del recurso que desea descargar.
Actores	Peer, PeerPseudoServidorHttp.
Precondiciones	El usuario debe estar conectado para que la funcionalidad de descargas esté habilitada.
Flujo Normal	1. El usuario introduce el nombre del recurso que desea descargar y envía una petición de búsqueda a todos los nodos de su misma malla. 2. El sistema examina, en cada nodo que recibe la petición de búsqueda, su correspondiente carpeta compartida (creada de forma automática) y en caso de encontrar algún archivo cuyo nombre coincida totalmente o al menos en parte, con el nombre del recurso buscado, envía, al nodo emisor de la petición de búsqueda, una respuesta por cada archivo encontrado. 3. El sistema muestra por pantalla un listado con información de cada recurso encontrado: nombre del archivo, tamaño y nodo poseedor del recurso. 4. El sistema muestra, en la pantalla del nodo emisor, información del estado del proceso.
Flujo Alternativo	2'. 'Si los nodos que reciben las peticiones de búsqueda no poseen ningún archivo cuyo nombre tenga alguna coincidencia con el recurso buscado, no envían ninguna respuesta. 3'. 'Si cae la conexión del nodo emisor de la petición de búsqueda, la operación no se podrá llevar a cabo.

Caso de uso 9

CU-010	Gestionar Descarga de Recurso
Descripción	El usuario selecciona el archivo que desea descargar del listado de archivos encontrados y envía una petición de descarga al nodo poseedor del recurso.
Actores	Peer, PeerPseudoServidorHttp.
Precondiciones	El usuario debe haber realizado una búsqueda del recurso y debe haber recibido como mínimo una respuesta de archivo encontrado.
Flujo Normal	1. El usuario selecciona del listado de archivos encontrados, el archivo que desea descargar y envía una petición de descarga al nodo poseedor del recurso. El archivo seleccionado desaparece del listado de recursos encontrados y se muestra en el listado de descargas activas. 2. Cuando el nodo poseedor del recurso recibe la petición de descarga, el sistema localiza el archivo en su carpeta compartida (creada de forma automática) y se lo envía al nodo solicitante como respuesta a su petición de descarga. 3. El sistema guarda el archivo recibido en el directorio compartido del nodo destinatario. 4. El sistema muestra, en la pantalla del nodo destinatario, información del estado del proceso.
Flujo Alternativo	2'. 'Si cae la conexión del nodo poseedor del recurso, la operación de descarga no se podrá llevar a cabo. 3'. 'Si cae la conexión del nodo destinatario del recurso, la operación de descarga no se podrá llevar a cabo.

Caso de uso 10

CU-011	Limpiar Pantalla
Descripción	El usuario borra del listado de archivos encontrados y del listado de descargas activas los recursos que desea que no se muestren en pantalla.
Actores	Peer, PeerPseudoServidorHttp.
Precondiciones	El usuario debe haber realizado una búsqueda con respuesta de al menos un archivo encontrado y/o una descarga de al menos un archivo, es decir, los listados de archivos encontrados y/o descargas activas deben tener información.
Flujo Normal	1. El usuario selecciona del listado de archivos encontrados los archivos que desea eliminar de la pantalla. 2. El usuario selecciona del listado de descargas activas los archivos que desea eliminar de la pantalla. 3. El sistema borra los archivos seleccionados en ambos listados. Estos archivos, en el caso de que se hayan descargado, no se eliminarán del ordenador del usuario, permanecerán en su carpeta compartida, sólo desaparecerán de la pantalla informativa de la aplicación.
Flujo Alternativo	3'. 'Si la conexión del usuario cae, estos listados no se mantendrán, es decir, al reiniciar la aplicación los listados aparecerán vacíos.

Caso de uso 11

6.4.6. DIAGRAMA DE CLASES EN LA FASE DE ANÁLISIS

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

A continuación se muestra el diagrama de clases obtenido tras llevar a cabo la especificación de los casos de uso. Es un diagrama a grandes rasgos ya que aún no se han identificado todos los elementos que se identificarán en la fase de diseño.

Diagrama de Clases GENERAL - Fase de Análisis

6.4.7. DIAGRAMAS DE SECUENCIA EN LA FASE ANÁLISIS

Los diagramas de secuencia forman parte de los llamados diagramas de interacción y se utilizan para modelar los aspectos dinámicos de los sistemas.

Un diagrama de interacción muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos.

Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes. Gráficamente, un diagrama de secuencia es una tabla que representa objetos, dispuestos a lo largo del eje X, y mensajes, ordenados según se suceden en el tiempo, a lo largo del eje Y.

A continuación se muestran los diagramas de secuencia obtenidos tras llevar a cabo la especificación de los casos de uso. A igual que con el diagrama de clases, en la fase de análisis aún no se han identificado todos los elementos que se identificarán en la fase de diseño, por tanto son diagramas a grandes rasgos.

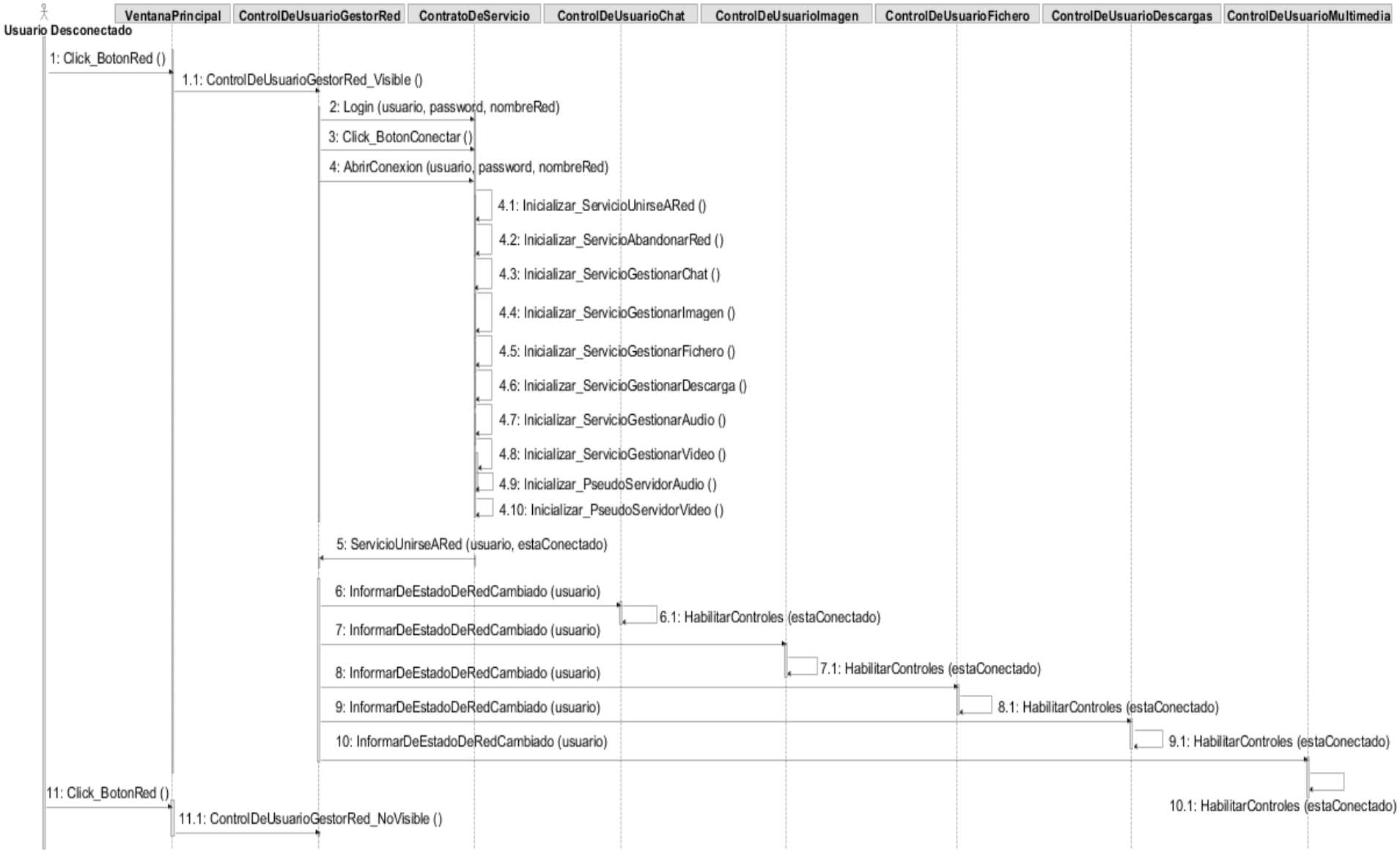


Diagrama de Secuencia CONECTAR - Fase de Análisis

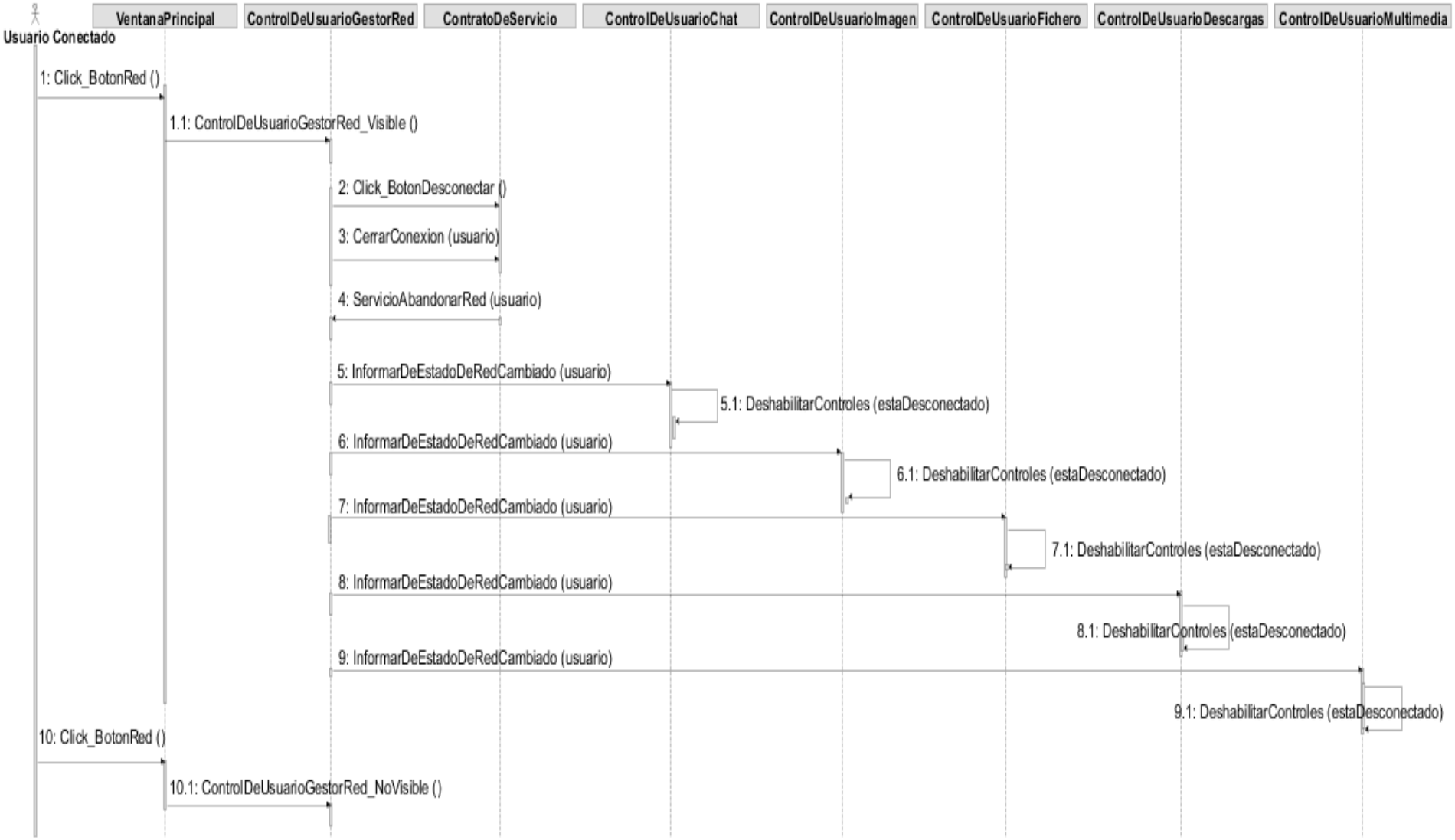


Diagrama de Secuencia DESCONECTAR - Fase de Análisis

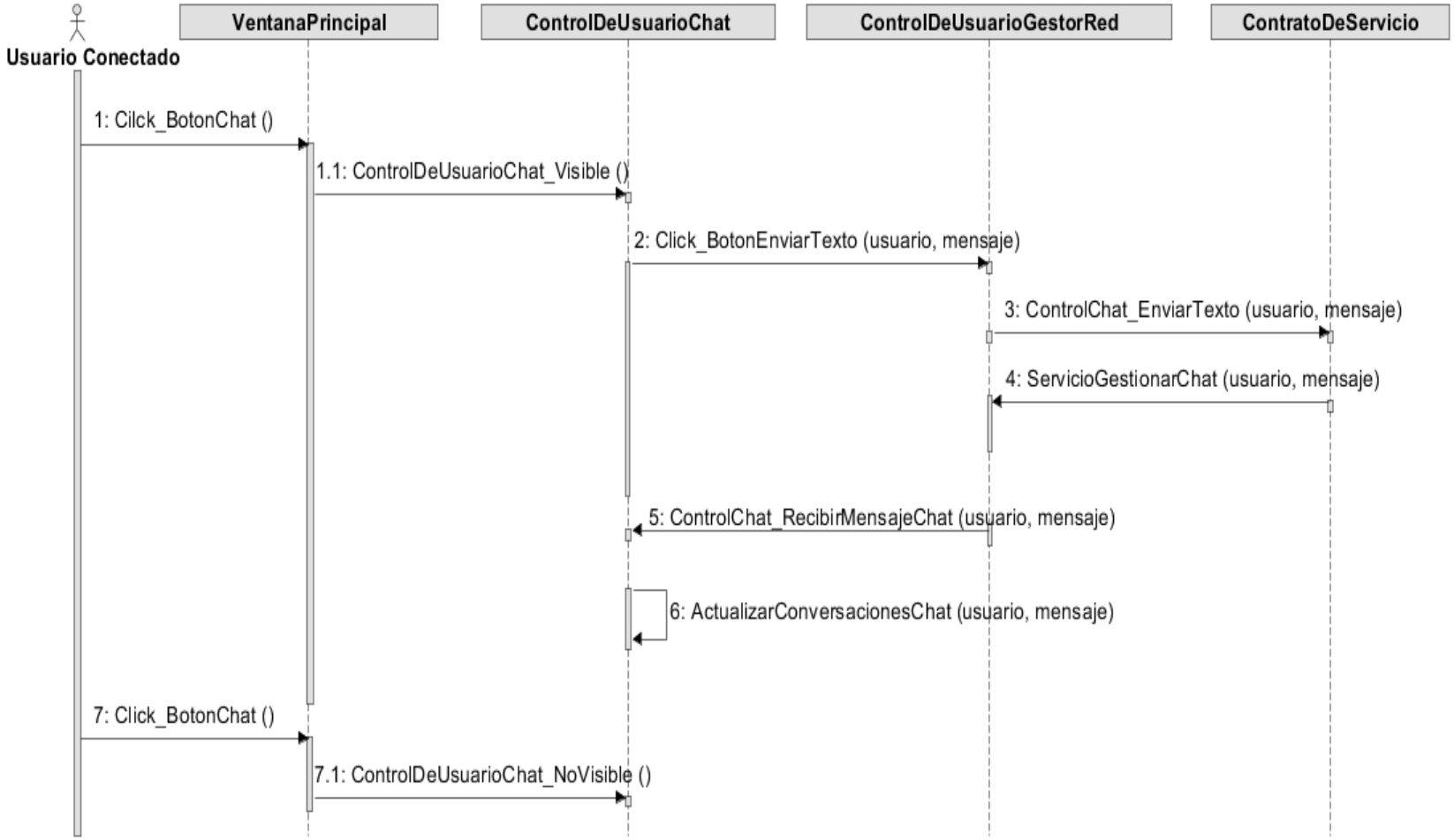


Diagrama de Secuencia CHAT - Fase de Análisis

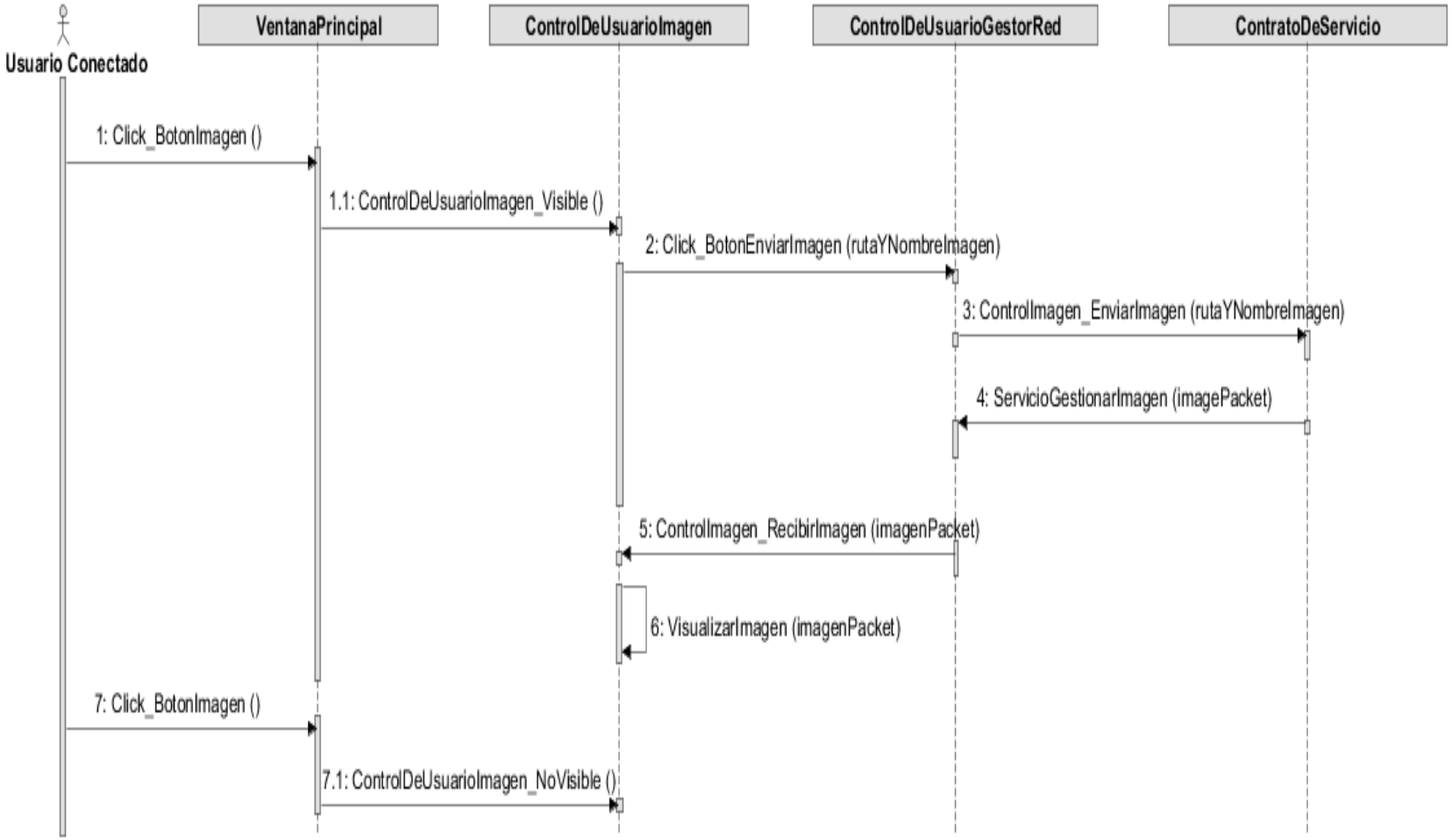


Diagrama de Secuencia VISOR DE IMÁGENES - Fase de Análisis

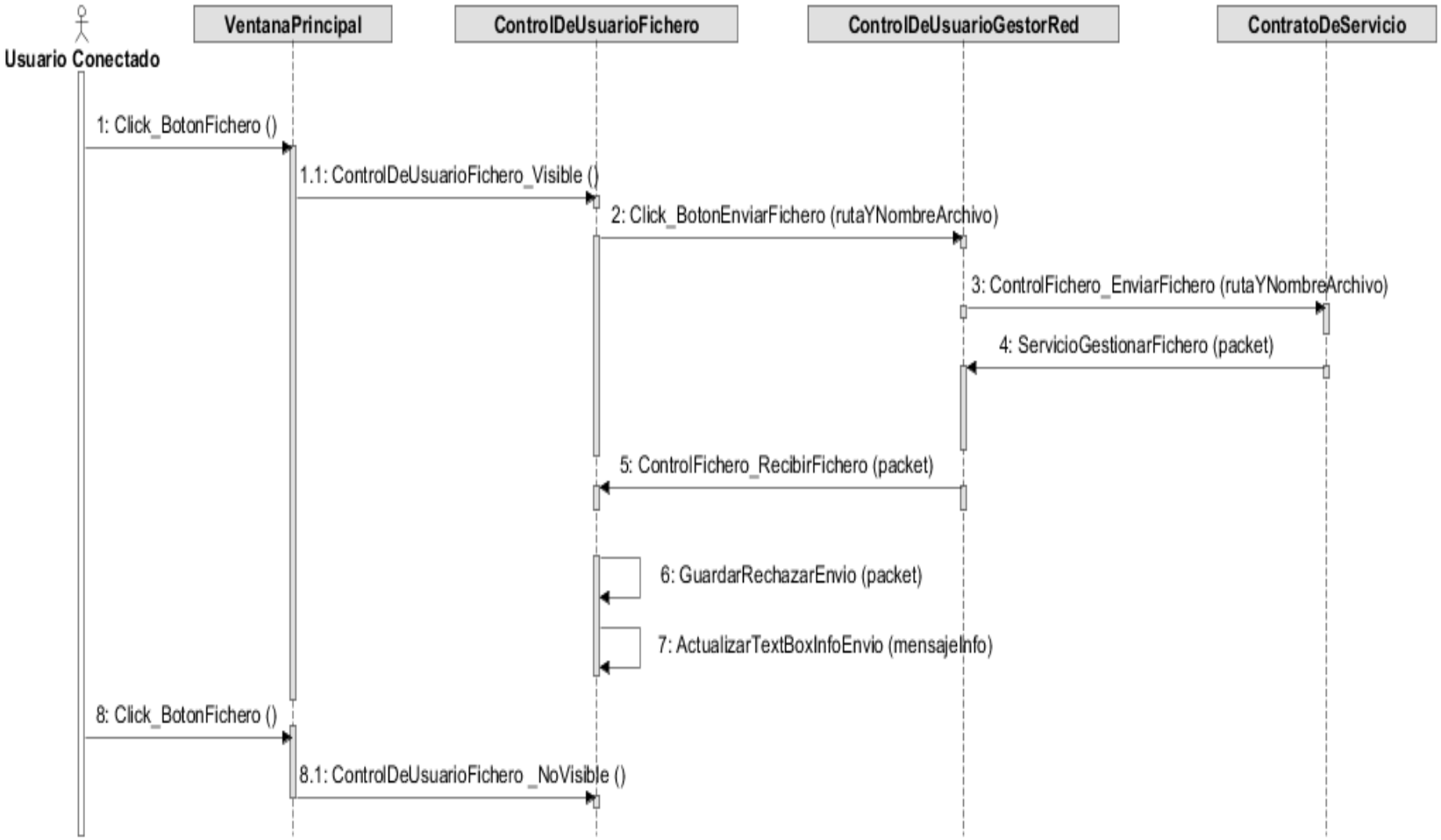


Diagrama de Secuencia INTERCAMBIO DE ARCHIVOS - Fase de Análisis

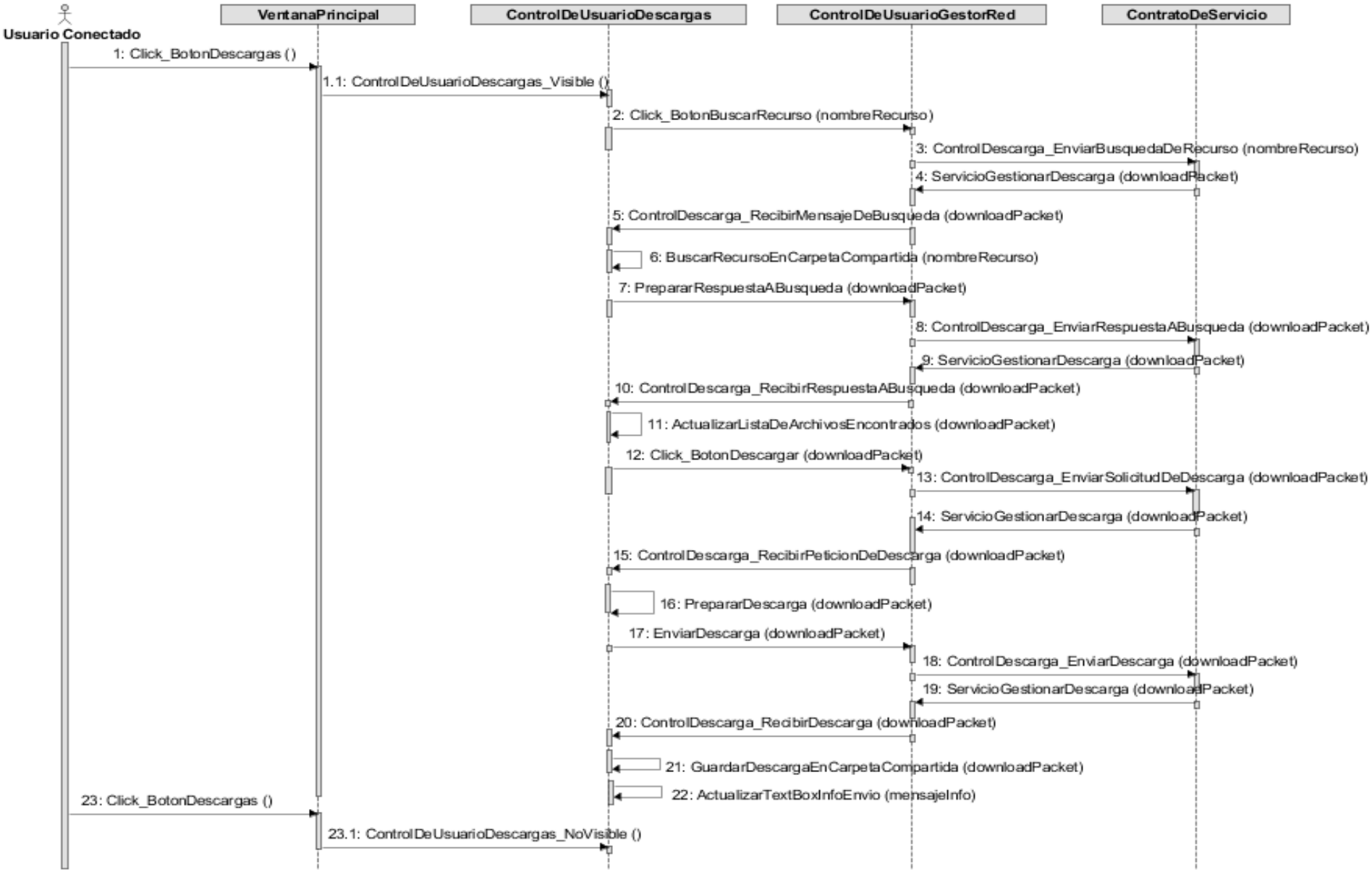


Diagrama de Secuencia DESCARGAS - Fase de Análisis

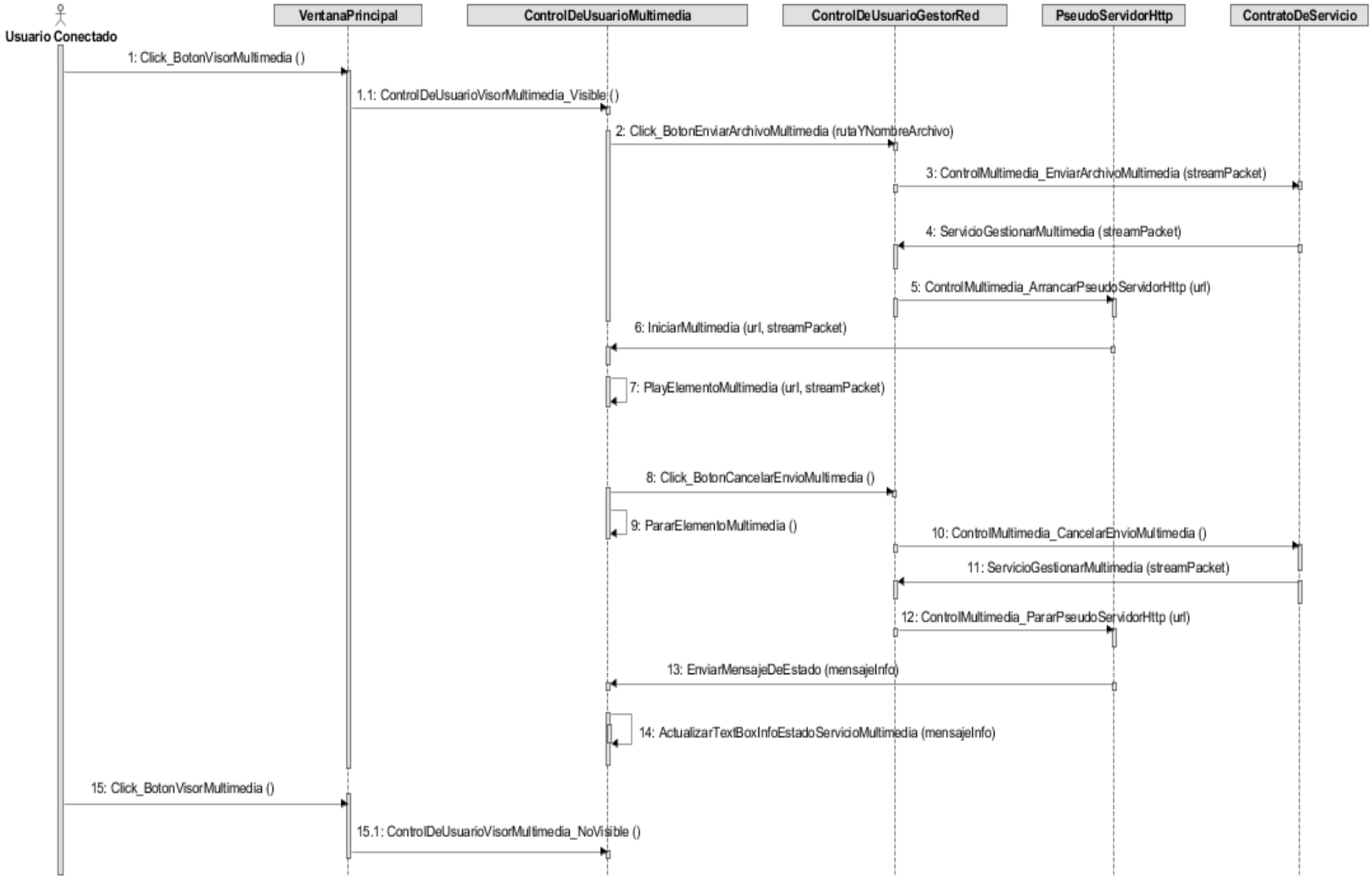


Diagrama de Secuencia REPRODUCTOR MULTIMEDIA (AUDIO/VÍDEO) - Fase de Análisis

6.5. DISEÑO

6.5.1. DISEÑO DE ARQUITECTURA BASADA EN EVENTOS

La familia de aplicaiones peer to peer, también llamada de componentes independientes, enfatiza la modificabilidad por medio de la separación de las diversas partes que intervienen en la computación. Consiste, por lo general, en procesos independientes o entidades que se comunican a través de mensajes. Cada entidad puede enviar mensajes a otras entidades, pero no controlarlas directamente. Los mensajes pueden ser enviados a componentes nominados o propagados mediante broadcast.

En términos de patrones de diseño, el patrón que corresponde más estrechamente a este estilo es el que se conoce como *Observer (*subconjunto del modelo *Publish/Subscribe, Publicación/Suscripción)*, un término que se hizo popular en *Smalltalk* a principios de los ochenta.

Las arquitecturas basadas en eventos se vinculan históricamente con sistemas basados en actores, daemons y redes de conmutación de paquetes (publicación-suscripción). Los conectores de estos sistemas incluyen procedimientos de llamada tradicionales y vínculos entre anuncios de eventos e invocación de procedimientos. La idea dominante en la invocación implícita es que, en lugar de invocar un procedimiento en forma directa (como se haría en un estilo orientado a objetos) un componente puede anunciar mediante difusión uno o más eventos. Un componente de un sistema puede anunciar su interés en un evento determinado asociando un procedimiento con la manifestación de dicho evento.

Desde el punto de vista arquitectónico, los componentes de un estilo de invocación implícita son módulos cuyas interfaces proporcionan tanto una colección de procedimientos (igual que en el estilo de tipos de datos abstractos) como un conjunto de eventos. Los procedimientos se pueden invocar a la manera usual en modelos orientados a objeto, o mediante el sistema de suscripción que se ha descrito.

En la estrategia arquitectónica de Microsoft, este estilo (llamado más bien un “patrón recurrente de diseño”) juega un papel de cierta importancia.

En este estilo y en ese contexto, los eventos se disparan bajo condiciones de negocios particulares, debiéndose escribir código para responder a esos eventos. Este patrón puede utilizarse cuando se desea que sucedan varias actividades, tal que todas ellas reciben los mismos datos de iniciación y no pueden comunicarse entre sí. Diferentes implementaciones del evento pueden o no ejecutarse, dependiendo de información de filtro específica. Si las implementaciones se han configurado para que corran secuencialmente, el orden no puede garantizarse.

Las prescripciones de Microsoft para el uso del modelo de eventos señalan que este estilo puede utilizarse cuando:

- ✓ Se desea manejar independientemente y de forma aislada diversas implementaciones de una “función” específica.
- ✓ Las respuestas de una implementación no afectan la forma en que trabajan otras implementaciones.
- ✓ Todas las implementaciones son de escritura solamente o de “dispararse y olvidar”, tal que la salida del proceso de negocios no está definida por ninguna implementación, o es definida sólo por una implementación de negocios específica.

Entre las ventajas enumeradas en relación con el modelo se señalan:

- ☑ Se optimiza el mantenimiento haciendo que procesos de negocios que no están relacionados sean independientes.
- ☑ Se alienta el desarrollo en paralelo, lo que puede resultar en mejoras de performance.
- ☑ Es fácil de empaquetar en una transacción atómica.
- ☑ Es agnóstica en lo que respecta a si las implementaciones corren sincrónica o asincrónicamente porque no se espera una respuesta.
- ☑ Se puede agregar un componente registrándolo para los eventos del sistema; se pueden reemplazar componentes.
- ☑ El estilo no permite construir respuestas complejas a funciones de negocios.
- ☑ Un componente no puede utilizar los datos o el estado de otro componente para efectuar su tarea.
- ☑ Cuando un componente anuncia un evento, no tiene idea sobre qué otros componentes están interesados en él, ni el orden en que serán invocados, ni el momento en que finalizan lo que tienen que hacer. Pueden surgir problemas de performance global y de manejo de recursos cuando se comparte un repositorio común para coordinar la interacción.

6.5.2. DIAGRAMAS DE CLASES EN LA FASE DE DISEÑO

Como ya se ha dicho, los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones.

Estos diagramas en la fase de diseño son mucho más precisos ya que se han podido identificar todos los elementos que formarán el sistema.

Los diferentes elementos que se van a mostrar en estos diagramas son:

Este símbolo representa las clases frontera, que son las clases encargadas de la interfaz del sistema.

Este otro símbolo representa las clases controladores, encargadas de redirigir las distintas peticiones y realizar operaciones de la lógica de negocio.

Este símbolo representa las clases entidad y representan algo acerca de lo cual el sistema tiene que almacenar alguna información.

A continuación se muestran los diagramas de clases del sistema

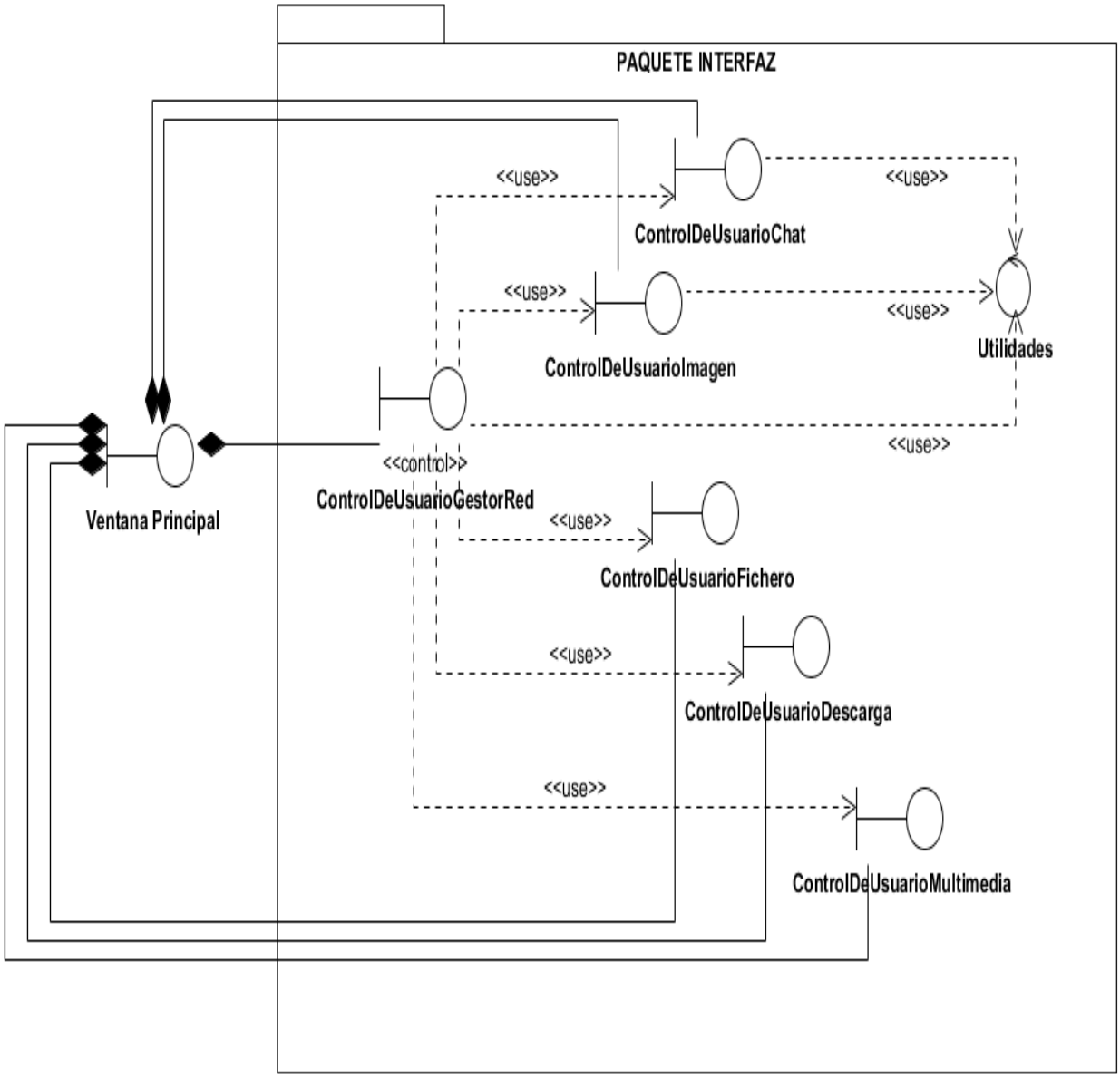


Diagrama de Clases PAQUETE INTERFAZ - Fase de Diseño

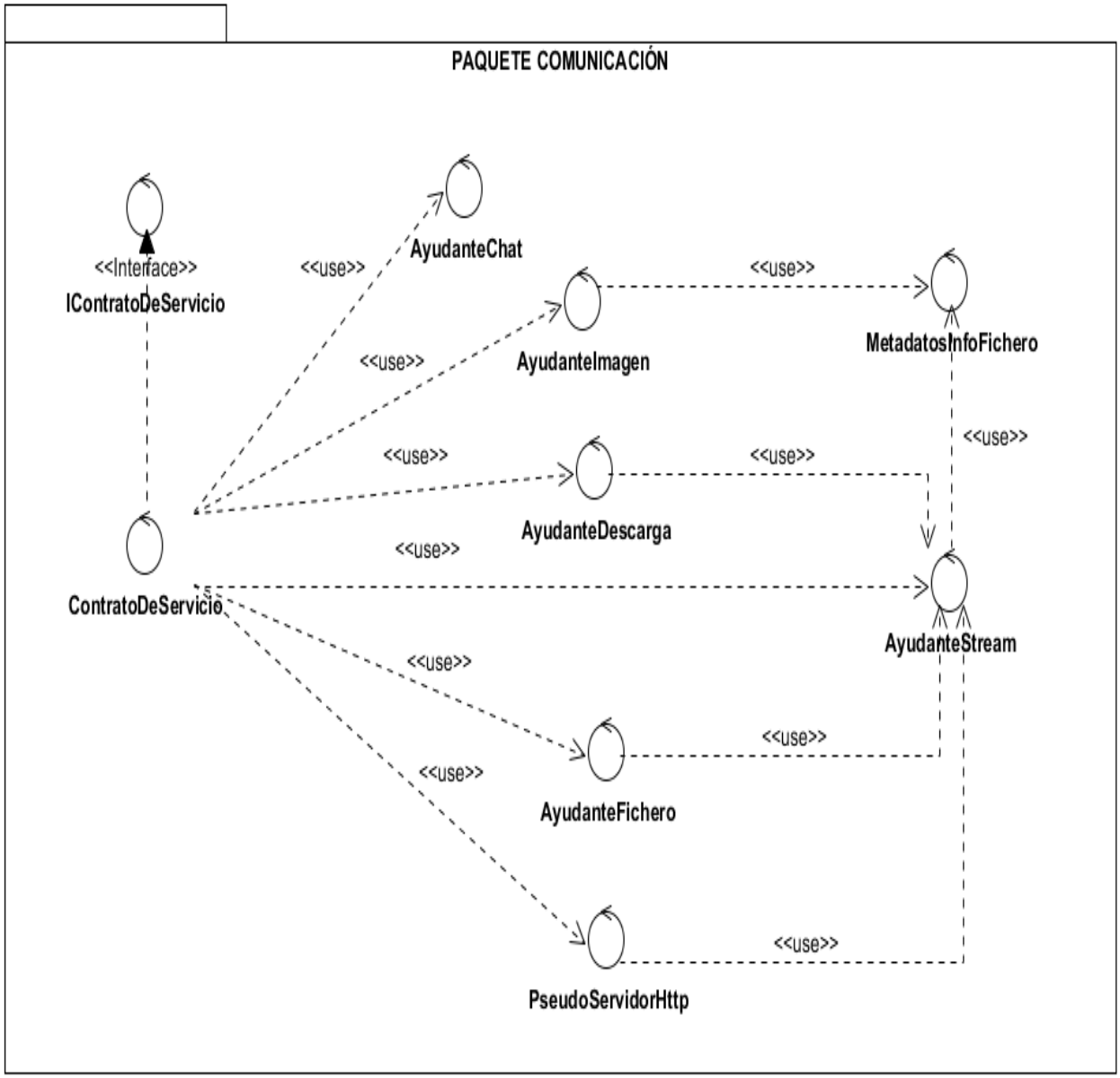


Diagrama de Clases PAQUETE COMUNICACIÓN - Fase de Diseño

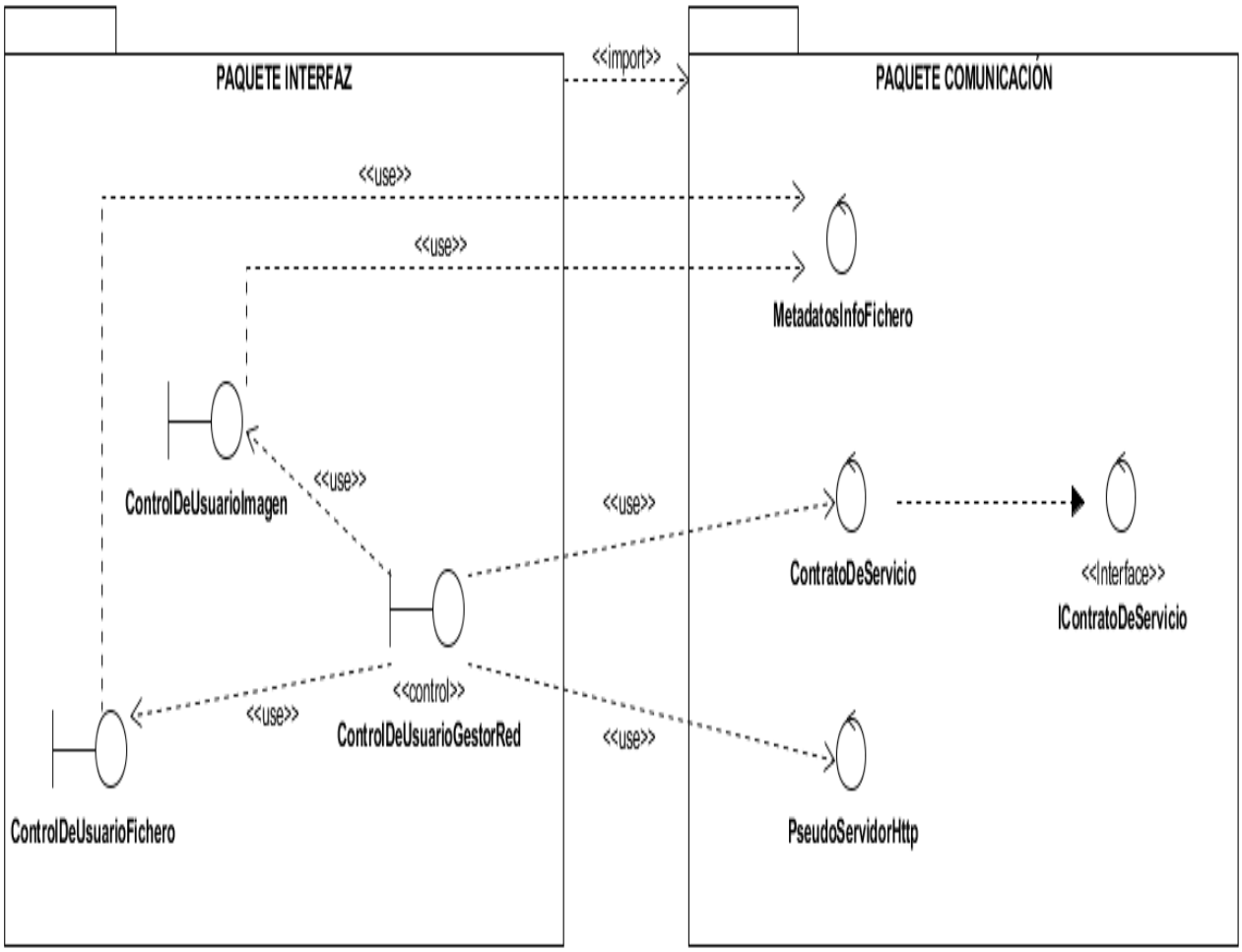


Diagrama de Clases PAQUETE INTERFAZ & PAQUETE COMUNICACIÓN - Fase de Diseño

6.5.3. DIAGRAMAS DE SECUENCIA EN LA FASE DE DISEÑO

Como ya se ha dicho, los diagramas de secuencia son un tipo de diagramas usados para modelar la interacción entre objetos en un sistema.

Los diagramas de secuencia se utilizan con frecuencia para validar los casos de uso. Documentan el diseño desde el punto de vista de los casos de uso. Observando qué mensajes se envían a los objetos, componentes o casos de uso y viendo, a grosso modo, cuánto tiempo consume el método invocado, los diagramas de secuencia nos ayudan a comprender los cuellos de botella potenciales, para así poder eliminarlos.

A continuación se muestran los diagramas de secuencia de los casos de uso más importantes que, como ocurre con los diagramas de clases, son mucho más precisos que los mismos diagramas en la fase de análisis.

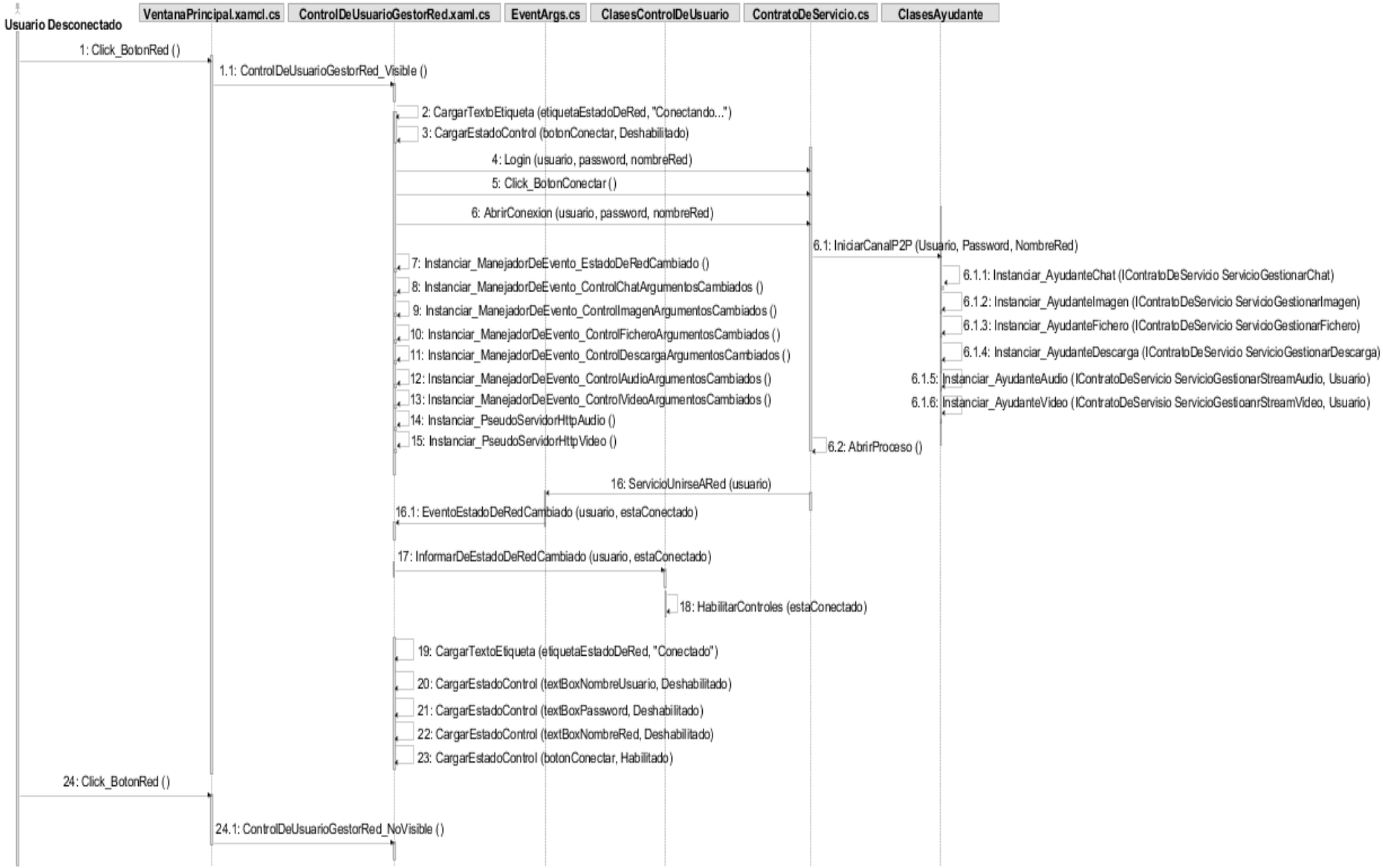


Diagrama de Secuencia CONECTAR - Fase de Diseño

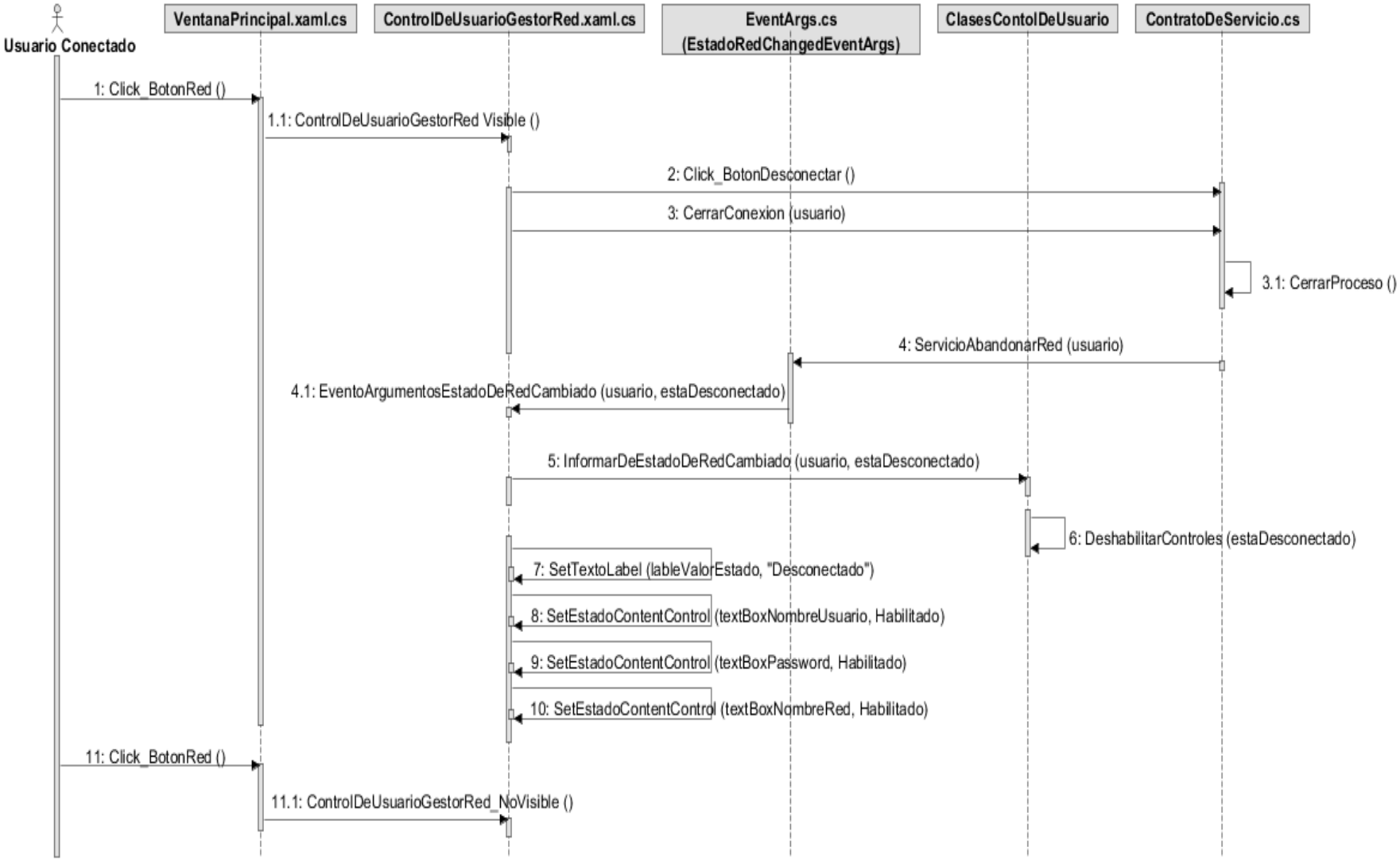


Diagrama de Secuencia DESCONECTAR - Fase de Diseño

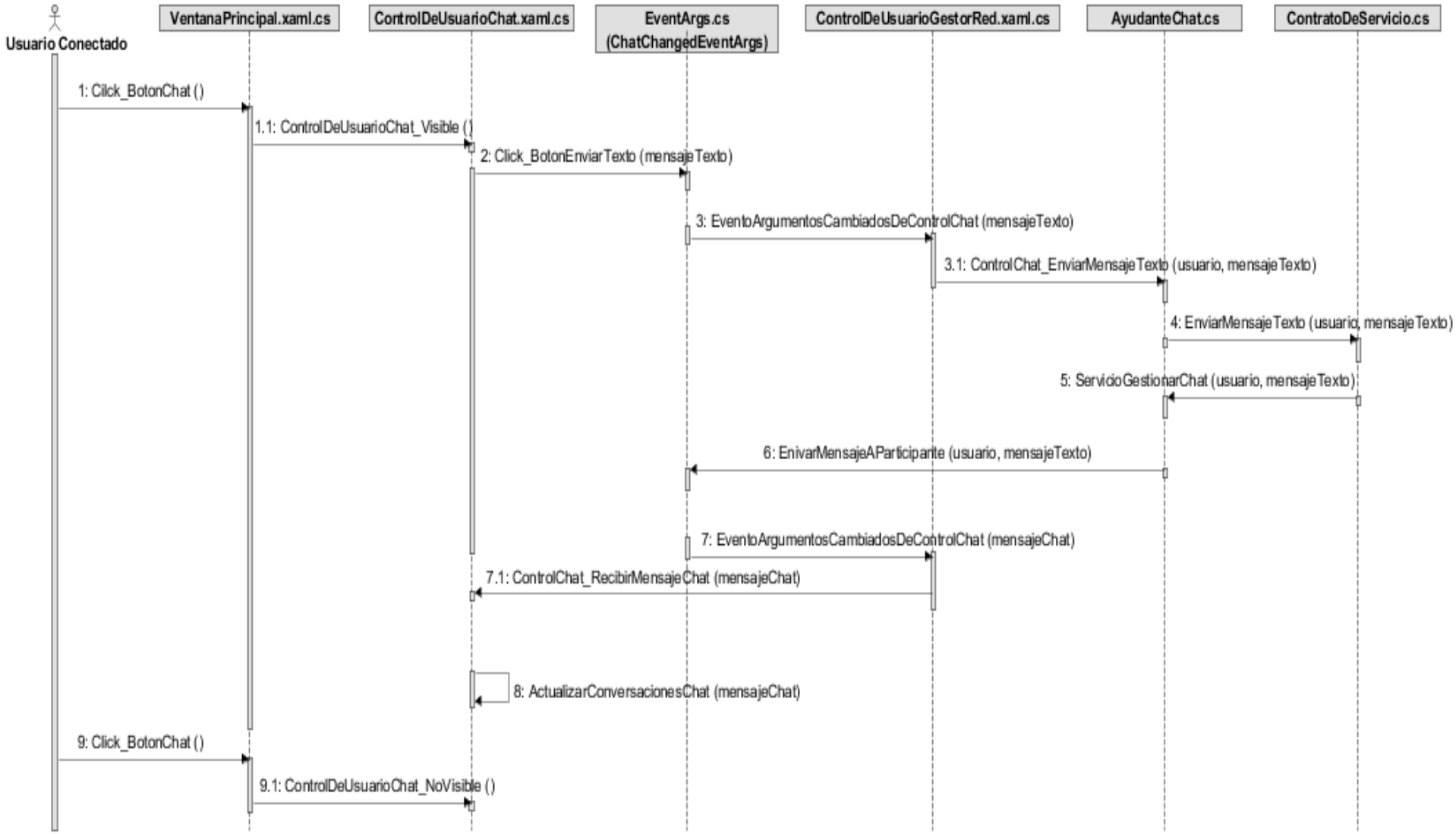


Diagrama de Secuencia CHAT - Fase de Diseño

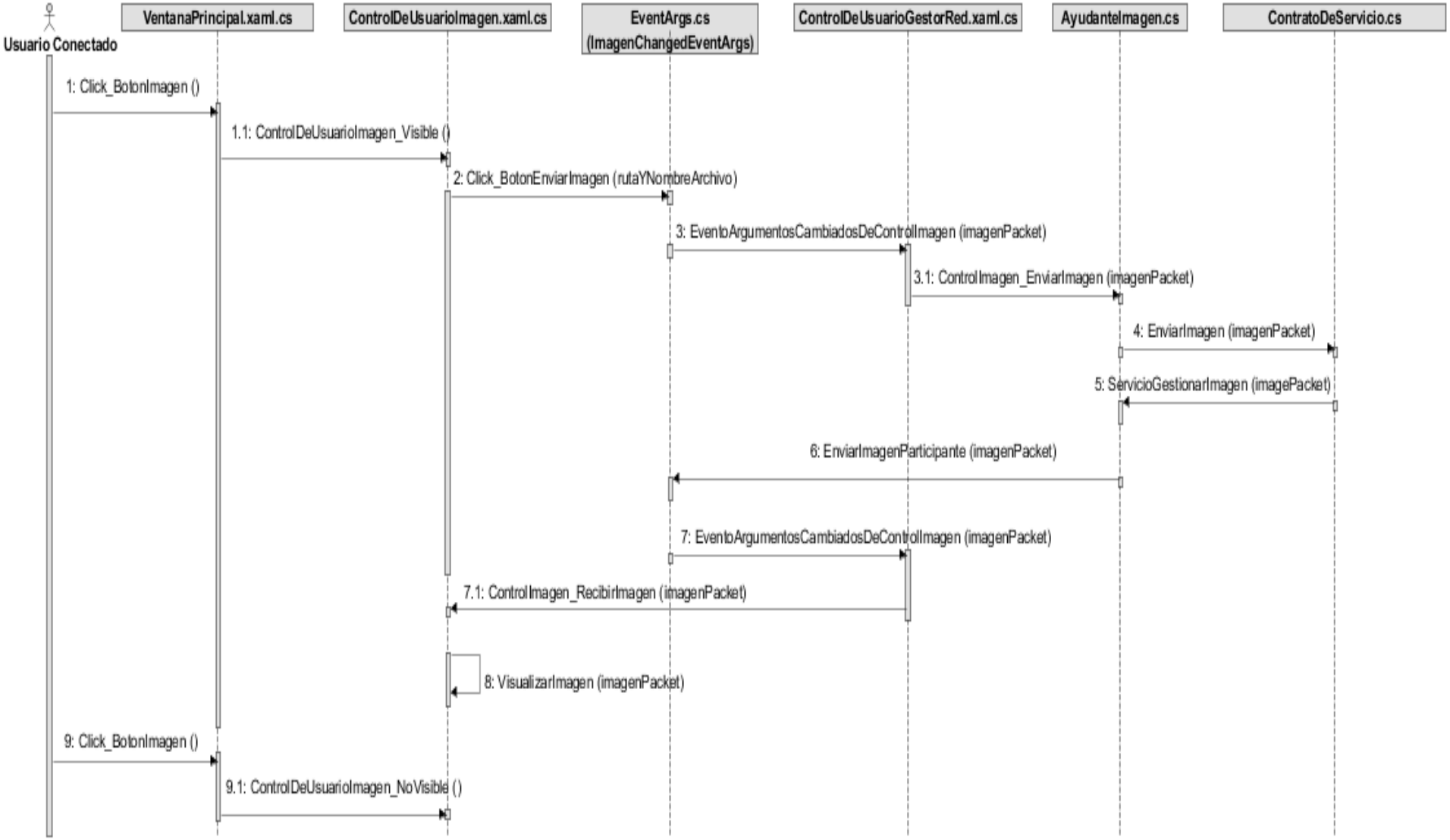


Diagrama de Secuencia VISOR DE IMÁGENES - Fase de Diseño

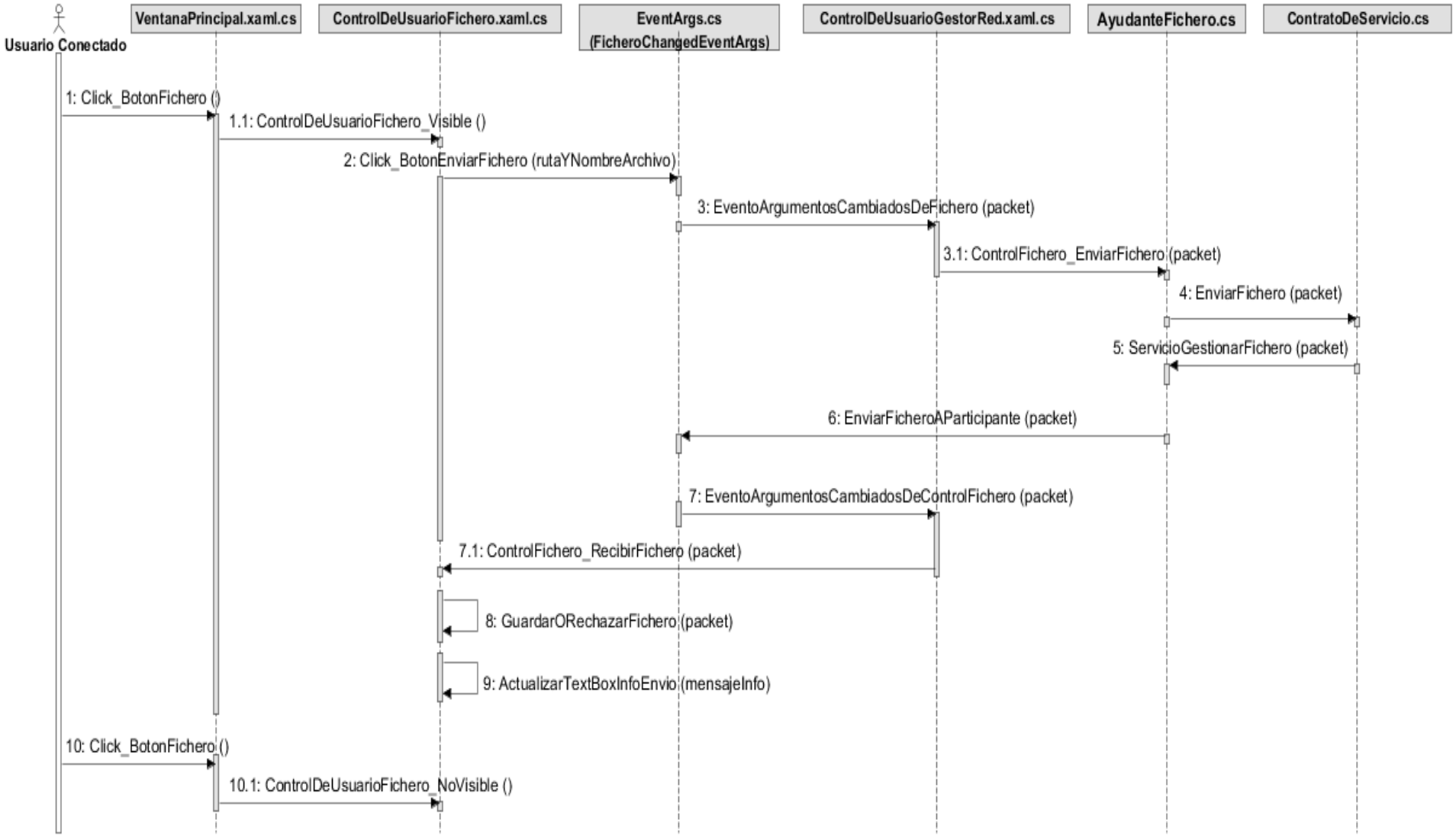


Diagrama de Secuencia INTERCAMBIO DE ARCHIVOS - Fase de Diseño

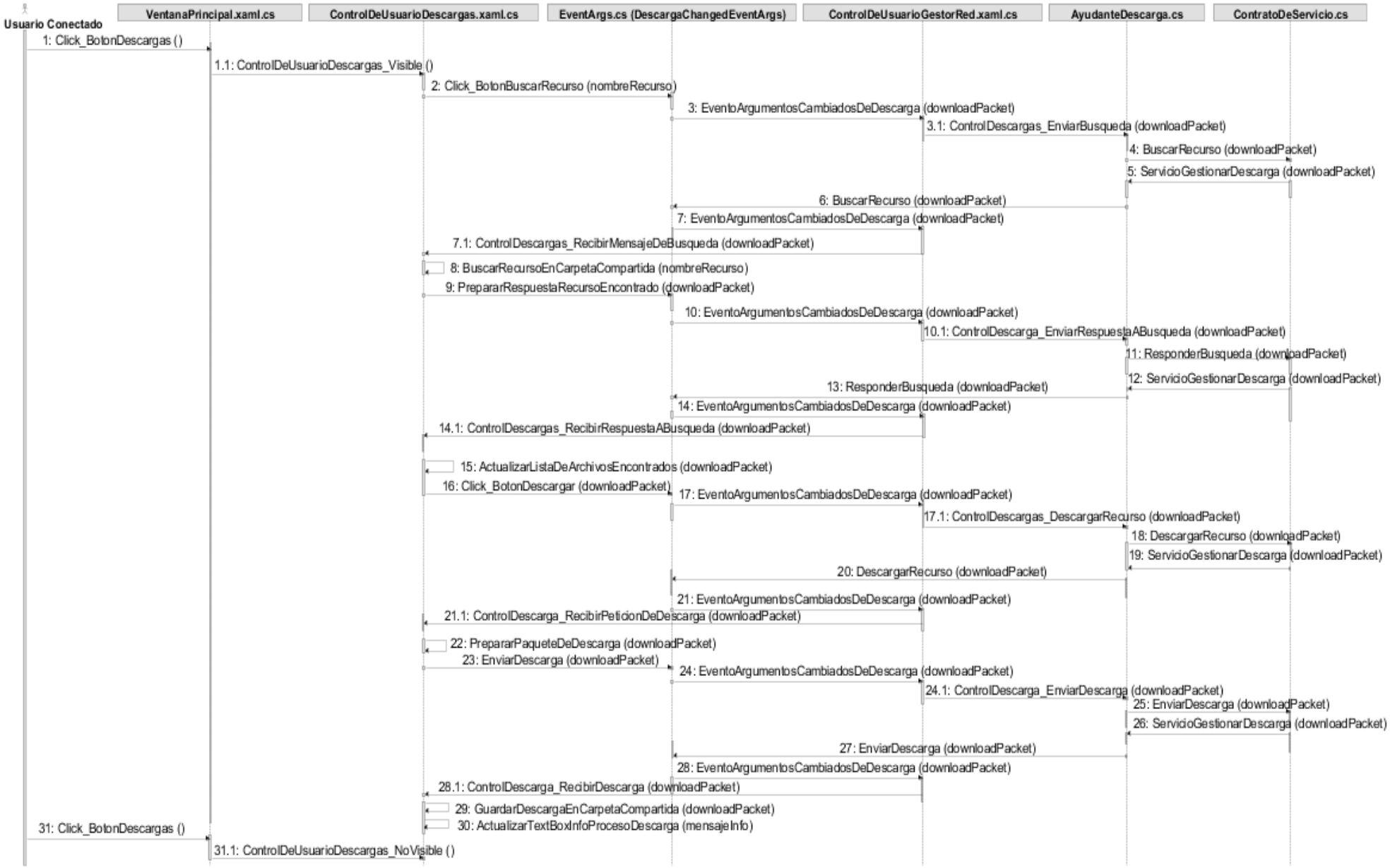


Diagrama de Secuencia DESCARGAS - Fase de Diseño

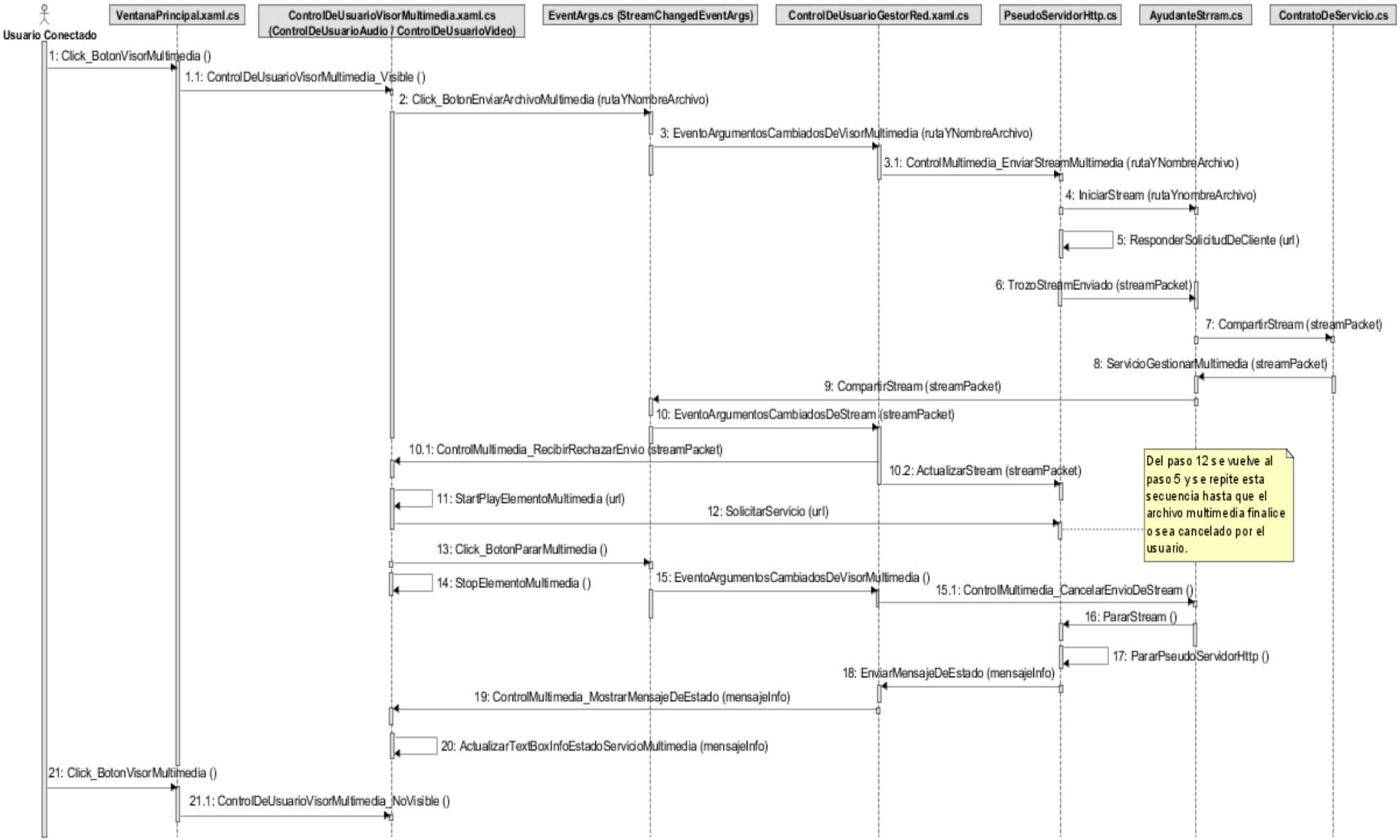


Diagrama de Secuencia REPRODUCTOR MULTIMEDIA (AUDIO/VÍDEO) - Fase de Diseño

6.5.4. DISEÑO DE INTERFAZ

Las IU son importantes porque afectan a las sensaciones, las emociones y el humor de tus usuarios. Si la IU está mal y el usuario siente que no puede controlar la aplicación, literalmente no serán felices y culparán a tu programa de ello. Si la IU está bien y las cosas funcionan de la manera que los usuarios esperan, estarán contentos cuando consigan completar pequeñas tareas.

Para hacer feliz a la gente, tienes que permitirles sentirse al mando de su entorno. Para hacer esto, necesitas interpretar correctamente sus acciones. La interfaz necesita comportarse de la manera que ellos esperan que se comporte.

Así, el axioma fundamental de todo diseño de interfaz de usuario es:

Una interfaz de usuario está bien diseñada cuando el programa se comporta exactamente como el usuario piensa que lo haría.

Como Hillel[25] dijo, todo lo demás son comentarios. Todas las demás reglas de buen diseño de IU son solo corolarios.

Joel Spolsky, fundador de Fog Creek Software, en Nueva York; ha trabajado como programador y gerente en Microsoft, Viacom y Juno.

[25] Hilel o Hillel, llamado *el Viejo* (*Ha-Zaquen*) o *el Sabio* (c. 70 a. C.-10 d. C.) fue un rabino y maestro judío, el primer erudito que sistematizó la interpretación de la ley escrita.
Sus ideas sobre la piedad personal y la preocupación por los demás, fueron precursoras de las nociones cristianas equivalentes, hasta tal punto que si en aquellos tiempos hubiese existido la SGAE, los nietos de Hillel podrían haber considerado seriamente una sustanciosa demanda por violación de derechos de autor.

No obstante existen múltiples principios para el diseño e implementación de las IU que también se han tenido en cuenta durante el desarrollo del sistema:

- ☑ **Anticipación.** Las aplicaciones deberían intentar anticiparse a las necesidades del usuario.
- ☑ **Autonomía.** Los programas son usados por usuarios con distintos niveles de conocimientos, desde principiantes hasta expertos.

  Se debe dar al usuario el ambiente flexible para que pueda aprender rápidamente a usar la aplicación.

  Es importante utilizar mecanismos indicadores de estado del sistema que mantengan a los usuarios alertas e informados. Además, se debe mantener información del estado del sistema en ubicaciones fáciles de visualizar.
- ☑ **Eficiencia del usuario.** Se debe considerar la productividad del usuario antes que la productividad de la máquina. Si el usuario debe esperar la respuesta del sistema por un período prolongado, estas pérdidas de tiempo se pueden convertir en pérdidas económicas para la organización. Los mensajes de ayuda deben ser sencillos y proveer respuestas a los problemas. Los menús y etiquetas de botones deben tener las palabras claves del proceso.
- ☑ **Ley de Fitts.** La *Ley de Fitts* es un modelo del movimiento humano que predice el tiempo necesario para moverse rápidamente desde una posición inicial hasta una zona destino final como una función de la distancia hasta el objetivo y el tamaño de éste. La *Ley de Fitts* se usa para modelar el acto de apuntar, tanto en el mundo real como en los ordenadores, por ejemplo con un ratón.

  Es por ello que es conveniente usar objetos grandes para las funciones importantes.

- ☑ **Interfaces explorables.** El usuario debe sentirse seguro de poder salir del sistema cuando lo desee. Es por ello que la IU debe tener un objeto fácil de accionar con el cual poder finalizar la aplicación.

  Para aquellos usuarios que sean noveles en el uso de la aplicación, se deberá proveer de guías para realizar tareas que no sean habituales.

  Es conveniente que el usuario pueda incorporar elementos visuales estables que permitan, no solamente un desplazamiento rápido a ciertos puntos del trabajo que esté realizando, sino también un sentido de "casa" o punto de partida.

- ☑ **Uso de metáforas.** Las buenas metáforas crean figuras mentales fáciles de recordar. La IU puede contener objetos asociados al modelo conceptual en forma visual, con sonido u otra característica perceptible por el usuario que ayude a simplificar el uso del sistema.

- ☑ **Curva de aprendizaje.** El aprendizaje de un producto y su usabilidad no son mutuamente excluyentes. El ideal es que la curva de aprendizaje sea nula, y que el usuario principiante pueda alcanzar el dominio total de la aplicación sin esfuerzo.

- ☑ **Reducción de latencia.** Siempre que sea posible, el uso de tramas (multi-threading) permite colocar la latencia en segundo plano (background). Las técnicas de trabajo multitarea posibilitan el trabajo ininterrumpido del usuario, realizando las tareas de transmisión y computación de datos en segundo plano.

- ☑ **Legibilidad.** Para que la IU favorezca la usabilidad del sistema de software, la información que se exhiba en ella debe ser fácil de ubicar y leer. Para lograr obtener este resultado se deben tener en cuenta algunas cosas como: el texto que aparezca en la IU deberá tener un alto contraste, se debe utilizar combinaciones de colores como el texto en negro sobre fondo blanco o amarillo suave; el tamaño de las fuentes tiene que ser lo suficientemente grande como para poder ser leído en monitores estándar; es importante hacer clara la presentación visual (colocación/agrupación de objetos, evitar la presentación de excesiva información, etc).

6.6. PRUEBAS

La fase de pruebas en el desarrollo de software es necesaria para determinar la calidad del software producido además de representar una revisión final de las especificaciones, del diseño y de la codificación. Deben realizarse pruebas de todas las entidades generadas durante el desarrollo del producto: especificaciones de requisitos, casos de uso, diagramas desarrollados, código fuente y bases de datos en caso de que las haya. La práctica óptima consiste en aplicar distintas técnicas de prueba para cada tipo de producto software.

Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Por esta razón, se debe definir en el proceso de la ingeniería del software una plantilla para las pruebas del software: un conjunto de pasos en los que podamos situar los métodos específicos de diseño de casos de prueba.

Las pruebas comienzan a nivel de módulo y trabajan “hacia fuera”, hacia la integración de todo el sistema basado en computadora. Según el momento, son apropiadas diferentes técnicas de prueba.

La prueba la lleva a cabo el responsable del desarrollo del software y posteriormente un grupo de usuarios del mismo. La prueba y la depuración son actividades diferentes, pero la depuración se debe incluir en cualquier estrategia de prueba.

La prueba del software es un elemento de un tema más amplio que, a menudo, es conocido como verificación y validación.

La verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica. La validación se refiere al conjunto de actividades que aseguran que el software construido se ajusta a los requisitos del cliente.

Las pruebas constituyen el último recurso desde el que se puede evaluar la calidad y, de forma más pragmática, descubrir los errores pero las pruebas no deben ser vistas como una red de seguridad.

6.6.1. PRUEBAS UNITARIAS

Son una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las pruebas de integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

La idea es realizar casos de prueba para cada función no trivial o método en el módulo de forma que cada caso sea independiente del resto.

Cada clase compilada ha sido sometida a numerosos casos de prueba buscando el comportamiento deseado, estudiando valores límite y comprobando repeticiones de bucles.

A continuación se adjuntan las tablas de Casos de Prueba unitarias.

CP-000	**Nombre del Caso de Prueba**
Propósito	Descripción del aspecto del sistema que está siendo probado.
Prerrequisitos	Suposiciones que deben cumplirse antes de ejecutar el Caso de Prueba.
Datos de Prueba	Lista de variables y sus posibles valores usados en el Caso de Prueba. El caso de prueba deberá ser ejecutado una vez por cada combinación de valores.
Descripción	Explicación descriptiva de la prueba.
Resultado Esperado	Resultado del Caso de Prueba esperado.
Resultado Obtenido	Resultado del Caso de Prueba obtenido.

Tabla de muestra de Caso de Prueba

CP-001	Login_1: Inicio normal de sesión de usuario
Propósito	Probar que los usuarios pueden iniciar sesión con nombre de usuario, contraseña y nombre de red apropiado.
Prerrequisitos	El usuario no ha iniciado sesión todavía.
Datos de Prueba	Nombre de Usuario = {testuser, bogususer, válido, inválido, vacío} Contraseña = {válida, inválida, vacío} Nombre de Red = {testnet, bogusnet, inválido, vacío}
Descripción	Varios usuarios inician sesión en la misma red, es decir, se logan con el mismo Nombre de Red y la misma Contraseña.
Resultado Esperado	Si el inicio de sesión es correcto, el usuario se conecta con el resto de participantes que hayan iniciado sesión con ese mismo Nombre de Red y esa misma Contraseña. Las cajas de texto de inicio de sesión se deshabilitan y los botones de funcionalidad del resto de ventanas se habilitan. Si el inicio de sesión es incorrecto, debido por ejemplo a que se introducen espacios en alguna caja de texto, o a que el Nombre de Usuario está ya en uso, el sistema muestra el mensaje de error "Login inválido", donde se explica la causa del error y el usuario no se conecta. Si ocurre esto, las cajas de texto de inicio de sesión se habilitan y los botones de funcionalidad del resto de ventanas se deshabilitan. La ventana "Red" se muestra y se oculta sin interrumpir la funcionalidad.
Resultado Obtenido	☑ Correcto

Caso de Prueba 1

CP-002	**Login_2: Inicio de sesión con un Nombre de Usuario que ya existe en la red.**
Propósito	Probar que un usuario no puede iniciar sesión con un nombre de usuario que ya existe en la red a la que se quiere conectar.
Prerrequisitos	El usuario no ha iniciado sesión todavía.
Datos de Prueba	Nombre de Usuario = {testuser, bogususer, inválido} Contraseña = {válida, vacío} Nombre de Red = {testnet, bogusnet, válido}
Descripción	Un usuario intenta iniciar sesión con un Nombre de Usuario que está siendo utilizado en el mismo momento y en la misma red por otro usuario.
Resultado Esperado	El sistema no permite que se conecten a la misma red dos usuarios con el mismo Nombre de Usuario. El sistema muestra el mensaje de error “Nombre de Usuario actualmente en uso” donde se le sugiere al usuario que elija un Nombre de Usuario distinto. El campo correspondiente a “Nombre de Usuario” muestra por unos segundos el texto “_USUARIO_FALLIDO_”. El usuario no se conecta y los campos correspondientes a “Nombre de Usuario”, “Contraseña” y “Nombre de Red” se habilitan.
Resultado Obtenido	☑ Correcto

Caso de Prueba 2

CP-003	**Login_3: Inicio de sesión con conexión a Internet a través de Proxy.**
Propósito	Probar que un usuario no puede usar la aplicación si la conexión a Internet es a través de proxy.
Prerrequisitos	El usuario no ha iniciado sesión todavía.
Datos de Prueba	Nombre de Usuario = {testuser, bogususer, válido} Contraseña = {válida, vacío} Nombre de Red = {testnet, bogusnet, válido}
Descripción	Un usuario con conexión a Internet a través de Proxy, intenta iniciar sesión.
Resultado Esperado	El sistema no es capaz de funcionar si la conexión a Internet es a través de Proxy puesto que las conexiones con proxy no son accesibles para los sockets. El sistema muestra un mensaje de “Error crítico” donde se le informa al usuario que la red no es accesible debido a la existencia de un proxy. A continuación se cierra la aplicación.
Resultado Obtenido	☑ Correcto

Caso de Prueba 3

CP-004	Chat: Envío y recepción de mensajes por el Chat.
Propósito	Probar que los usuarios conectados a la misma red pueden chatear entre ellos.
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Mensaje Usuario_1 = {mensaje de texto} Mensaje Usuario_2 = {mensaje de texto} Mensaje Usuario_n = {mensaje de texto}
Descripción	Varios usuarios conectados chatean entre sí.
Resultado Esperado	Todos los usuarios conectados a la misma red envían y reciben mensajes del resto de participantes. Cada mensaje de texto es recibido por todos los usuario conectados. El sistema muestra las conversaciones en la ventana “Chat”. Cada vez que un nuevo usuario se une a la red, el sistema informa al resto de participantes de la red sobre la incorporación del nuevo usuario y muestra esta información en el área correspondiente de la ventana “Chat”. Cada vez que un usuario abandona la red, el sistema informa al resto de participantes de la red sobre la retirada de este usuario y muestra la información en el área correspondiente de la ventana “Chat”. La ventana “Chat” se muestra y se oculta sin interrumpir la funcionalidad.
Resultado Obtenido	☑ Correcto

Caso de Prueba 4

CP-005	VisorDeImagen: Visualización de archivos de imagen.
Propósito	Probar que los usuarios conectados a la misma red pueden visualizar imágenes entre ellos.
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Foto Usuario_1 = {archivo de imagen} Foto Usuario_2 = {archivo de imagen} Foto Usuario_n = {archivo de imagen}
Descripción	Varios usuarios conectados se muestran sus fotos o cualquier otro archivo de imagen.
Resultado Esperado	Todos los usuarios conectados a la misma red pueden visualizar sus fotos o cualquier archivo de tipo imagen (*.gif, *.jpg, *.png) con el resto de participantes conectados. El sistema muestra en el área correspondiente de la ventana “Imagen” de todos los participantes conectados, las imágenes con sus respectivos títulos. La ventana “Imagen” se muestra y se oculta sin interrumpir la funcionalidad.
Resultado Obtenido	☑ Correcto

Caso de Prueba 5

CP-006	Intercambio de Archivos: Aceptar/Rechazar envíos de archivos.
Propósito	Probar que los usuarios conectados a la misma red pueden intercambiar archivos entre ellos.
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Archivo Usuario_1 = {archivo de imagen} Archivo Usuario_2 = {archivo de música} Archivo Usuario_3 = {archivo de audio} Archivo Usuario_n = {archivo de texto}
Descripción	Varios usuarios conectados se envían archivos de distinto tipo.
Resultado Esperado	Todos los usuarios conectados a la misma red pueden enviar archivos de cualquier tipo con el resto de participantes conectados. El sistema ofrece a los receptores la posibilidad de aceptar o rechazar el archivo. El sistema muestra en el área correspondiente de la ventana "Archivo" del emisor, información del envío: ruta y nombre del archivo que se envía y éxito o fracaso en el proceso de envío. El sistema muestra en el área correspondiente de la ventana "Archivo" de todos los receptores que hayan aceptado el envío, información del archivo recibo: nombre de archivo y éxito o fracaso en el proceso de envío. El sistema muestra en las pantallas de todos los nodos receptores que hayan rechazado el envío, el nombre del archivo rechazado. La ventana "Archivo" se muestra y se oculta sin interrumpir la funcionalidad.
Resultado Obtenido	☑ Correcto

Caso de Prueba 6

CP-007	Reproductor de Audio: Escuchar música.
Propósito	Probar que los usuarios conectados a la misma red pueden reproducir y escuchar música entre ellos.
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Archivo Usuario_1 = {archivo de música } Archivo Usuario_2 = {archivo de música} Archivo Usuario_n = {archivo de música }
Descripción	Varios usuarios conectados, reproducen archivos de audio y envían invitaciones al resto de participantes para que también ellos puedan escuchar la canción seleccionada.
Resultado Esperado	Todos los usuarios conectados a la misma red pueden escuchar música simultáneamente. El sistema ofrece a los receptores la posibilidad de aceptar o rechazar la invitación de audio. El sistema arranca el servicio en el nodo emisor y en de todos los nodos receptores que hayan aceptado la invitación, y muestra en el área correspondiente de la ventana “Audio” información del archivo de audio: Título, Artista y Álbum. El sistema muestra a emisores y receptores información del proceso. Si el emisor interrumpe el audio, este dejará de escucharse en todos los nodos receptores. Si cualquiera de los receptores interrumpe el audio, este solo dejará de escucharse en su equipo. La ventana “Audio” se muestra y se oculta sin interrumpir la funcionalidad.
Resultado Obtenido	☑ Correcto

Caso de Prueba 7

CP-008	Reproductor de Vídeo: Reproducir y visualizar vídeos.
Propósito	Probar que los usuarios conectados a la misma red pueden reproducir y visualizar archivos de vídeo entre ellos.
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Archivo Usuario_1 = {archivo de vídeo} Archivo Usuario_2 = {archivo de vídeo} Archivo Usuario_n = {archivo de vídeo}
Descripción	Varios usuarios conectados, reproducen archivos de vídeo y envían invitaciones al resto de participantes para que también ellos puedan visualizar el vídeo seleccionado.
Resultado Esperado	Todos los usuarios conectados a la misma red pueden reproducir archivos de vídeo e invitar a los otros participantes a visualizar al vídeo elegido de forma simultánea, esto es, al mismo tiempo que está siendo reproducido. El sistema ofrece a los receptores la posibilidad de aceptar o rechazar la invitación de vídeo. El sistema arranca el servicio en el nodo emisor y en de todos los nodos receptores que hayan aceptado la invitación, y muestra en el área correspondiente de la ventana “Video” información del archivo de vídeo: Título y Artista. El sistema muestra a emisores y receptores información del proceso multimedia. Si el emisor interrumpe el vídeo, este dejará de visualizarse en todos los nodos receptores. Si cualquiera de los receptores interrumpe el vídeo, este solo dejará de reproducirse en su equipo. La ventana “Video” se muestra y se oculta sin interrumpir la funcionalidad.
Resultado Obtenido	☑ Correcto

Caso de Prueba 8

CP-009	Descargas: Buscar y descargas recursos compartidos.
Propósito	Probar que los usuarios conectados a la misma red pueden buscar y descargar recursos entre ellos.
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Archivo Usuario_1 = {archivo de texto } Archivo Usuario_2 = {archivo de imagen} Archivo Usuario_3 = {archivo de música} Archivo Usuario_n = {archivo de vídeo }
Descripción	El usuario conectado introduce texto en la caja de búsquedas para que el sistema proceda a la búsqueda de algún archivo con algún tipo de similitud en el nombre. El sistema busca el archivo en las carpetas compartidas de los otros participantes. Si la búsqueda tiene éxito, el usuario puede proceder a su descarga.
Resultado Esperado	El usuario recibe respuesta a sus búsquedas cuando alguno de los otros participantes posee, en su carpeta compartida por la aplicación, algún archivo cuyo nombre coincida, en al menos una palabra, con el recurso buscado. El sistema muestra un listado de los archivos encontrados. El usuario puede descargar cualquiera de estos archivos. El sistema procede a la descarga y almacena el archivo en la carpeta compartida del usuario. El sistema muestra en los nodos que solicitan las búsquedas y/o las descargas información del proceso. La ventana “Descargas” se muestra y se oculta sin interrumpir la funcionalidad.
Resultado Obtenido	☑ Correcto

Caso de Prueba 9

6.6.2. PRUEBAS DE INTEGRACIÓN

Se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez. Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos.

Las pruebas de integración y las pruebas del sistema completo que se han realizado se reducen al uso de la aplicación entre, a lo sumo, cuatro usuarios y comprobar que el funcionamiento es el esperado.

A continuación se adjuntan las tablas de Casos de Prueba de integración.

CP-000	**Nombre del Caso de Prueba**
Propósito	Descripción del aspecto del sistema que está siendo probado.
Prerrequisitos	Suposiciones que deben cumplirse antes de ejecutar el Caso de Prueba.
Datos de Prueba	Lista de variables y sus posibles valores usados en el Caso de Prueba. El caso de prueba deberá ser ejecutado una vez por cada combinación de valores.
Descripción	Explicación descriptiva de la prueba.
Resultado Esperado	Resultado del Caso de Prueba esperado.
Resultado Obtenido	Resultado del Caso de Prueba obtenido.

Tabla de muestra de Caso de Prueba

CP-001	**Chat_VisorDeImagen: Funcionamiento conjunto de Chat y Visor de Imagen.**
Propósito	Probar que los usuarios pueden usar simultáneamente las funcionalidades de “Chat” y “Visor de Imagen”.
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Usuario_1 = {Chat, Visor de Imagen} Usuario _2 = {Chat, Visor de Imagen} Usuario_n = {Chat, Visor de Imagen}
Descripción	Varios usuarios conectados chatean y visualizan archivos de imagen.
Resultado Esperado	El sistema permite que varios usuarios conectados a la misma red chateen entre sí, a la vez que visualizan archivos de imagen. Es posible ocultar y mostrar ambas ventanas, a la vez o por separado, sin que la funcionalidad se interrumpa.
Resultado Obtenido	☑ Correcto

Caso de Prueba 1

CP-002	**Chat_VisorDeImagen_IntercambioDeArchivo: Funcionamiento conjunto de Chat, Visor de Imagen e Intercambio de Archivo.**
Propósito	Probar que los usuarios pueden usar simultáneamente las funcionalidades de “Chat”, “Visor de Imagen” e “Intercambio de Archivo”
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Usuario_1 = {Chat, Visor de Imagen, Intercambio de Archivo} Usuario _2 = {Chat, Visor de Imagen, Intercambio de Archivo } Usuario_n = {Chat, Visor de Imagen, Intercambio de Archivo }
Descripción	Varios usuarios conectados chatean, visualizan fotos e intercambian archivos.
Resultado Esperado	El sistema permite que varios usuarios conectados a la misma red chateen entre sí, visualicen archivos de imagen y se envíen archivos. El sistema permite que algunos usuarios acepten los envíos de archivos y que otros los rechacen. Es posible ocultar y mostrar las ventanas, a la vez o por separado, sin que la funcionalidad se interrumpa.
Resultado Obtenido	☑ Correcto

Caso de Prueba 2

CP-003	**Chat_VisorDeImagen_IntercambioDeArchivo_ReproductorDeAudio: Funcionamiento conjunto de Chat, Visor de Imagen, Intercambio de Archivo y Reproductor de Audio.**
Propósito	Probar que los usuarios pueden usar simultáneamente las funcionalidades de “Chat”, “Visor de Imagen” , “Intercambio de Archivo” y “Reproductor de Audio”
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Usuario_1 = {Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio} Usuario _2 = {Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio} Usuario_n = {Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio}
Descripción	Varios usuarios conectados chatean, visualizan fotos, intercambian archivos y reproducen archivos de audio.
Resultado Esperado	El sistema permite que varios usuarios conectados a la misma red chateen entre sí, visualicen archivos de imagen, intercambien archivos y reproduzcan música de forma simultánea. El sistema permite que algunos usuarios acepten los envíos de archivos y que otros los rechacen. El sistema permite que algunos usuarios acepten las invitaciones audio y que otros las rechacen. Si el emisor de audio interrumpe la reproducción, o bien porque para la canción, o bien porque se desconecta, el resto de nodos receptores no escuchan el audio. Si cualquiera de los receptores de audio interrumpe la reproducción, o bien porque para la canción, o bien porque se desconecta, el audio deja de sonar solo en su equipo. Es posible ocultar y mostrar las ventanas, a la vez o por separado, sin que la funcionalidad se interrumpa.
Resultado Obtenido	☑ Correcto

Caso de Prueba 3

CP-004	**Chat_VisorDeImagen_IntercambioDeArchivo_ReproductorDeAudio_ReproductorDeVídeo: Funcionamiento conjunto de Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio y Reproductor de Vídeo.**
Propósito	Probar que los usuarios pueden usar simultáneamente las funcionalidades de “Chat”, “Visor de Imagen” , “Intercambio de Archivo”, “Reproductor de Audio” y “Reproductor de Video”
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Usuario_1 = {Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio, Reproductor de Video} Usuario _2 = {Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio, Reproductor de Video} Usuario_n = {Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio, Reproductor de Video}
Descripción	Varios usuarios conectados chatean, visualizan fotos, intercambian archivos y reproducen archivos de audio y vídeo.
Resultado Esperado	El sistema permite que varios usuarios conectados a la misma red chateen entre sí, visualicen archivos de imagen, intercambien archivos y reproduzcan música y vídeos de forma simultánea. El sistema permite que algunos usuarios acepten los envíos de archivos y que otros los rechacen. El sistema permite que algunos usuarios acepten las invitaciones audio y que otros las rechacen. El sistema permite que algunos usuarios acepten las invitaciones vídeo y que otros las rechacen. Al igual que con archivos de audio, si el emisor de vídeo interrumpe la reproducción, o bien porque para el vídeo, o bien porque se desconecta, el resto de nodos receptores no visualizan el vídeo. Al igual que con archivos de audio, si cualquiera de los receptores de vídeo interrumpe la reproducción, o bien porque para el vídeo, o bien porque se desconecta, el vídeo deja de visualizarse solo en su equipo. Es posible ocultar y mostrar las ventanas, a la vez o por separado, sin que la funcionalidad se interrumpa.
Resultado Obtenido	☑ Correcto

Caso de Prueba 4

CP-005	Prueba completa: Todos los componentes funcionando a la vez.
Propósito	Probar que los usuarios pueden usar simultáneamente todas las funcionalidades: “Chat”, “Visor de Imagen” , “Intercambio de Archivo”, “Reproductor de Audio”, “Reproductor de Video” y “Descargas”
Prerrequisitos	Varios usuarios han iniciado sesión y están conectados a la misma red (mismo Nombre de Red y misma Contraseña).
Datos de Prueba	Usuario_1 = {Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio, Reproductor de Video, Descargas} Usuario _2 = {Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio, Reproductor de Video, Descargas} Usuario_n = {Chat, Visor de Imagen, Intercambio de Archivo, Reproductor de Audio, Reproductor de Video, Descargas}
Descripción	Varios usuarios conectados chatean, visualizan fotos, intercambian archivos y reproducen audio y vídeo, a la vez que realizan descargas.
Resultado Esperado	El sistema permite que varios usuarios conectados a la misma red chateen entre sí, visualicen imágenes, intercambien archivos, reproduzcan música y vídeos y descarguen archivos compartidos. El sistema permite que algunos usuarios acepten los envíos de archivos y que otros los rechacen. El sistema permite que algunos usuarios acepten las invitaciones audio y que otros las rechacen. El sistema permite que algunos usuarios acepten las invitaciones vídeo y que otros las rechacen. Al igual que con archivos de audio, si el emisor de vídeo interrumpe la reproducción, o bien porque para el vídeo, o bien porque se desconecta, el resto de nodos receptores no visualizan el vídeo. Al igual que con archivos de audio, si cualquiera de los receptores de vídeo interrumpe la reproducción, o bien porque para el vídeo, o bien porque se desconecta, el vídeo deja de visualizarse solo en su equipo. Los usuarios pueden buscar y descargar recursos localizados en las respectivas carpetas compartidos del resto de participantes conectados. Es posible ocultar y mostrar las ventanas, a la vez o por separado, sin que la funcionalidad se interrumpa.
Resultado Obtenido	☑ Correcto

Caso de Prueba 5

7. MANUALES

7.1. MANUAL DE INSTALACIÓN

Este manual se ha escrito para ayudar a los usuarios de la aplicación a instalar el sistema y configurar el software requerido en ambientes Windows para su correcto funcionamiento.

REQUERIMIENTOS DEL SISTEMA

Para el correcto funcionamiento de la aplicación, el entorno Windows debe tener al menos las siguientes características:

☞ El SO del equipo debe ser uno de los siguientes:

- Windows XP Service Pack 1 (SP1) con el Advanced Networking Pack
- Windows XP SP2
- Windows XP Professional x64 Edition
- Windows Vista
- Windows 7

☞ Servicios que obligatoriamente deben estar iniciados:

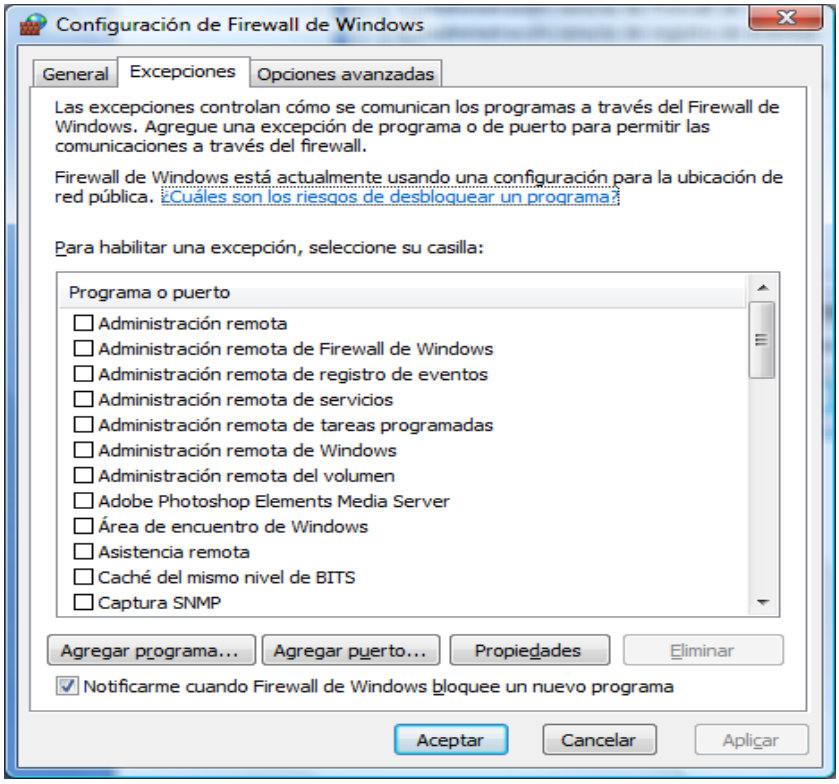
- p2pimsvc: Administración de identidad de redes de mismo nivel.
- PNRPsvc: Protocolo de resolución de nombres de miso nivel.
- WinHttpAutoProxySvc: Servicio de detección automática de proxy web WinHTTP.
- PNRPAutoReg: Servicio de publicación de nombres de equipo PNRP.
- Audiosrv: Audio de Windows.

☞ Servicios opcionales relacionados con el rendimiento:

- MMCSS: Programador de aplicaciones multimedia.
- WPFFontCache_v0400: Windows Presentation Foundation Font Cache 4.0.0.0.

☞ Configuración de Firewall de Windows. En caso de tener activado el Firewall de Windows deben estar habilitadas las siguiente excepciones:

- Servicio de registro de nombres de equipo de Windows Collaboration.
- Windows Peer to Peer Collaboration Foundation.
- Windows Communication Foundation.
- También en la pestaña *Excepciones* de la ventana *Configuración de Firewall de Windows* debe chequearse la casilla "*Notificarme cuando Firewall de Windows bloquee un nuevo programa*".



☞ Configuración de Firewall del antivirus, en caso de tener instalado un antivirus:

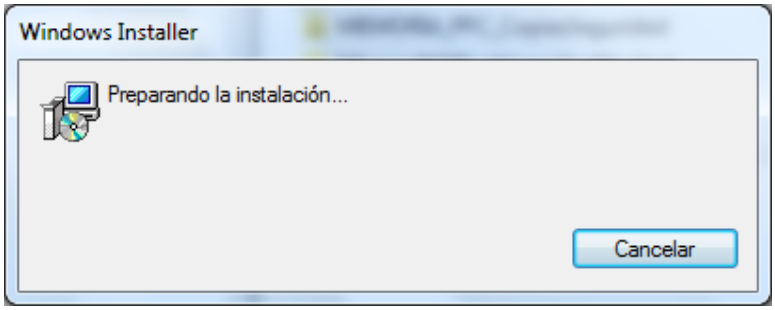
- Los niveles de seguridad deben ser como mínimo de tipo "*Acceso supervisado*" o similar, que permita a los programas conocidos conectarse automáticamente a Internet y notifique cuando los programas no se reconozcan.
- Los permisos para la aplicación, *Peer2PeerApp*, de acceso entrante y saliente a Internet deben ser de acceso completo.

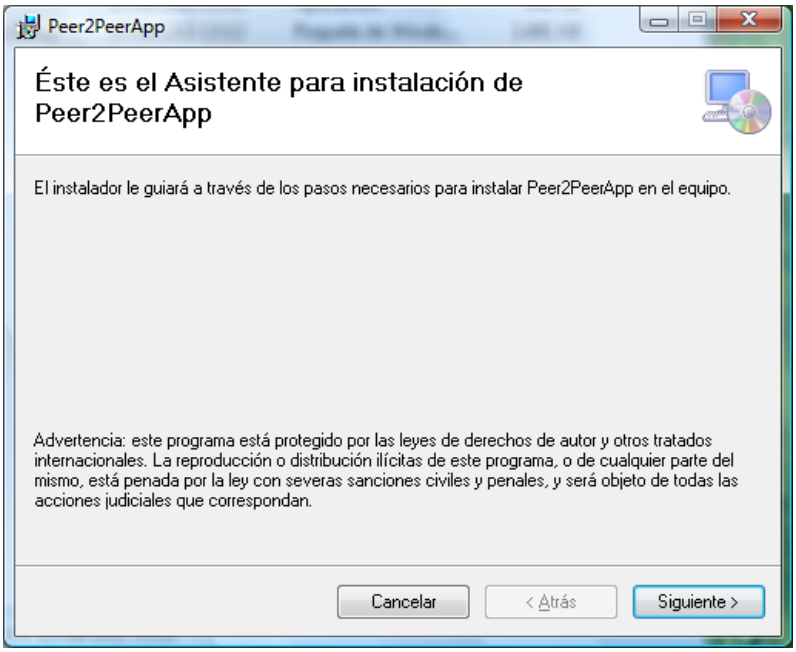
PROCESO DE INSTALACIÓN

La subcarpeta “Instalador de la Aplicación”, dentro de la carpeta “Documentación” (Ver CD-ROM de “Ejemplar Informatizado de PFC”) contiene dos archivos:

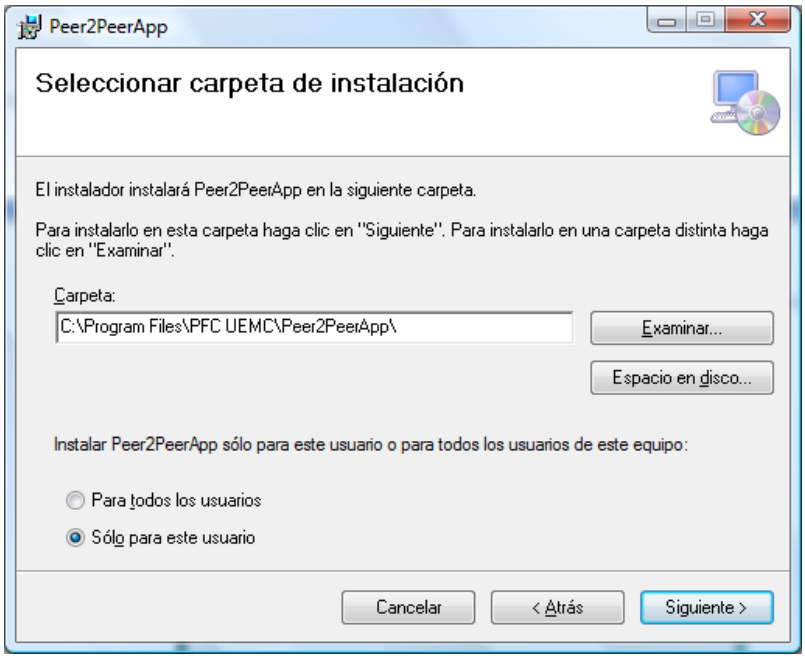
- setup: Archivo que determina si el instalador de Windows se encuentra en el equipo de destino e instala el instalador de Windows si es necesario.
- Peer2PeerApp Setup: Archivo de instalación de paquetes que contienen toda la información necesaria para la instalación de la aplicación.

1. Hacer doble click en el icono “Peer2PeerApp Setup”, situado en la subcarpeta “Instalador de la Aplicación”.





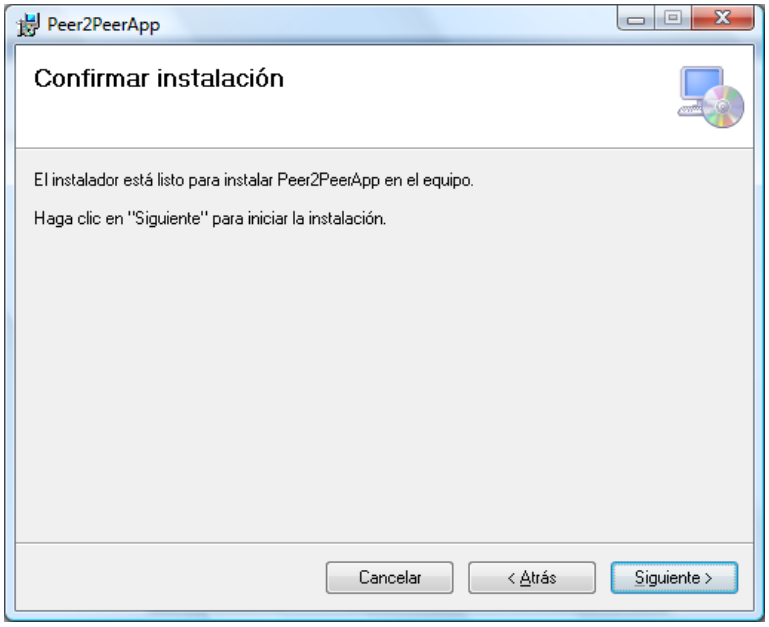
2. Hacer click en Siguiente.



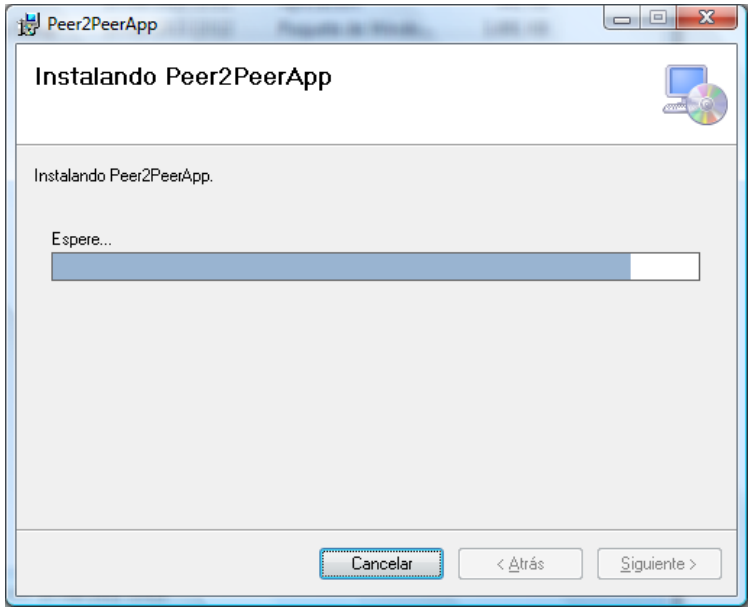
Ver: Se puede cambiar la ubicación donde se instalará la aplicación.

Ver: Se puede instalar la aplicación para todos los usuarios o sólo para el usuario actual.

3. Hacer click en Siguiente.

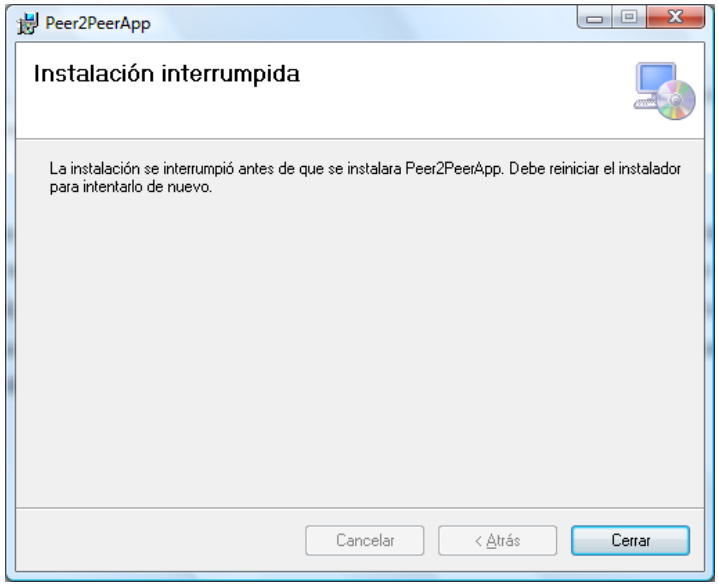


4. Hacer click en Siguiente.

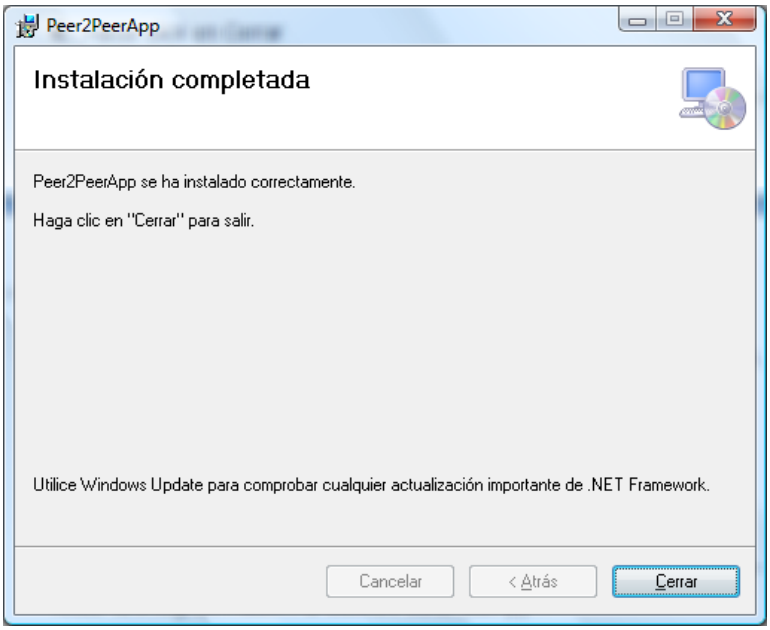


- Si se desea cancelar la instalación, hacer click en Cancelar.

  Ver: En caso de que la instalación se cancele, aparece la ventana "Instalación interrumpida".



Ver: En caso de que la instalación no se cancele, aparece la ventana "Instalación completada".



- Hacer click en Cerrar

Ver: En el escritorio se muestra el icono de acceso rápido a la aplicación, "Peer2PeerApp".

Ver: También se puede navegar hasta el ejecutable en

Inicio>>Programas>>PFC UEMC – Peer2PeerApp>>Peer2PeerApp

7. Navegar hasta la carpeta en la que se ha instalado la aplicación. Por defecto "C:\Program Files\PFC UEMC\Peer2PeerApp".

Ver: En la carpeta "CODIGO FUENTE" creada por el instalador, se encuentra el código fuente de la aplicación.

7.2. MANUAL DE USUARIO

Este manual ha sido elaborado con la intención de ofrecer la información necesaria para aprender a utilizar todas las funcionalidades básicas de la aplicación.

Este manual sigue una estructura concreta; está dividido en secciones independientes, empezando por la de conexión, y centrándose en los detalles de cada funcionalidad en las secciones consecutivas.

Con el fin de facilitar la comprensión del manual, se incluyen gráficos explicativos.

FUNCIÓN "CONEXIÓN"

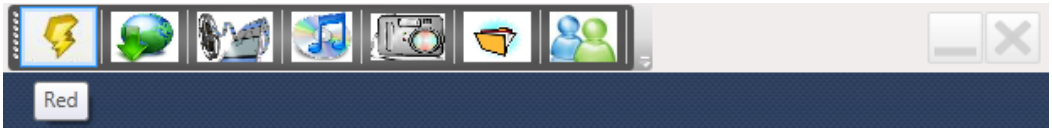
1. Hacer doble click en el icono de acceso directo de la aplicación situado en el escritorio, o navegar hasta el ejecutable:

   Inicio>>Programas>>PFC UEMC – Peer2PeerApp>>Peer2PeerApp

Peer2PeerApp

2. Se muestran los botones de la aplicación.

3. Hacer click en el botón "Red".

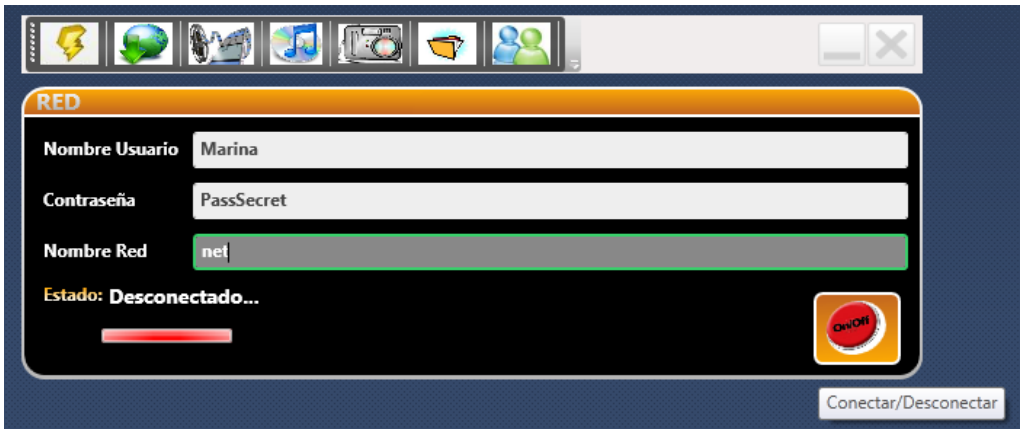


   Ver: Ventana "Red" abierta.

   Ver: La etiqueta "Estado" en la ventana de Red muestra "Desconectado…"

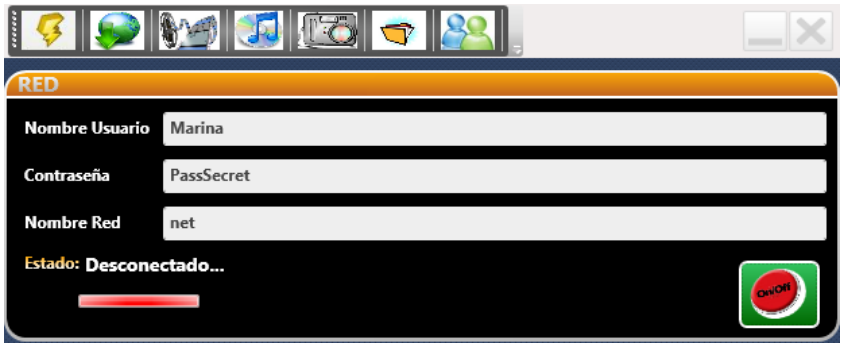
4. Teclear Nombre de Usuario, Contraseña y Nombre de Red en las cajas de texto correspondientes.

5. Hacer click en el botón "Conectar/Desconectar".



- Ver: En el momento de hacer click en el botón “Conectar/Desconectar”, este muestra un color verde, indicando que la operación solicitada acaba de iniciarse.

  Esto mismo ocurre al hacer click en cualquiera de los botones de la aplicación.

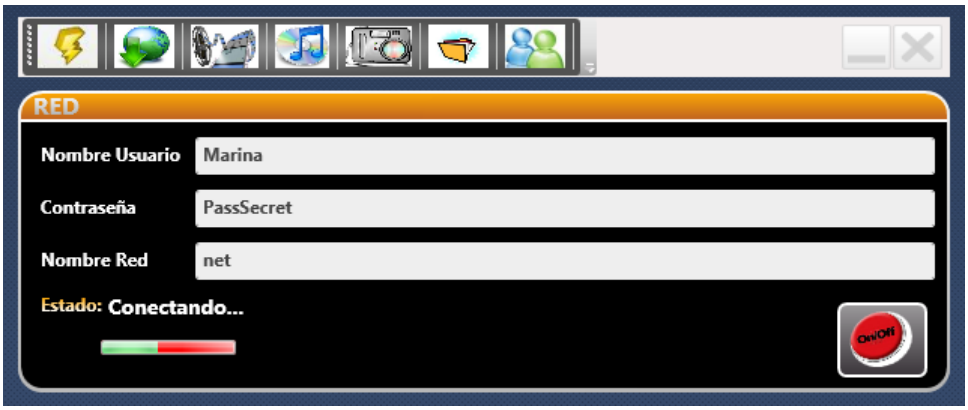


Ver: La etiqueta de “Estado” de la ventana de “Red” cambia a “Conectando…”

Ver: La barra de progreso empieza a tomar color verde, indicando que el sistema está en proceso de conexión.

Ver: Las cajas de texto correspondientes a “Nombre de Usuario”, “Contraseña” y “Nombre de Red” están deshabilitados.

Ver: El botón “Conectar” muestra un color gris que indica que está deshabilitado.



7. Ver: La etiqueta de “Estado” de la ventana “Red” cambia a “Conectado…”

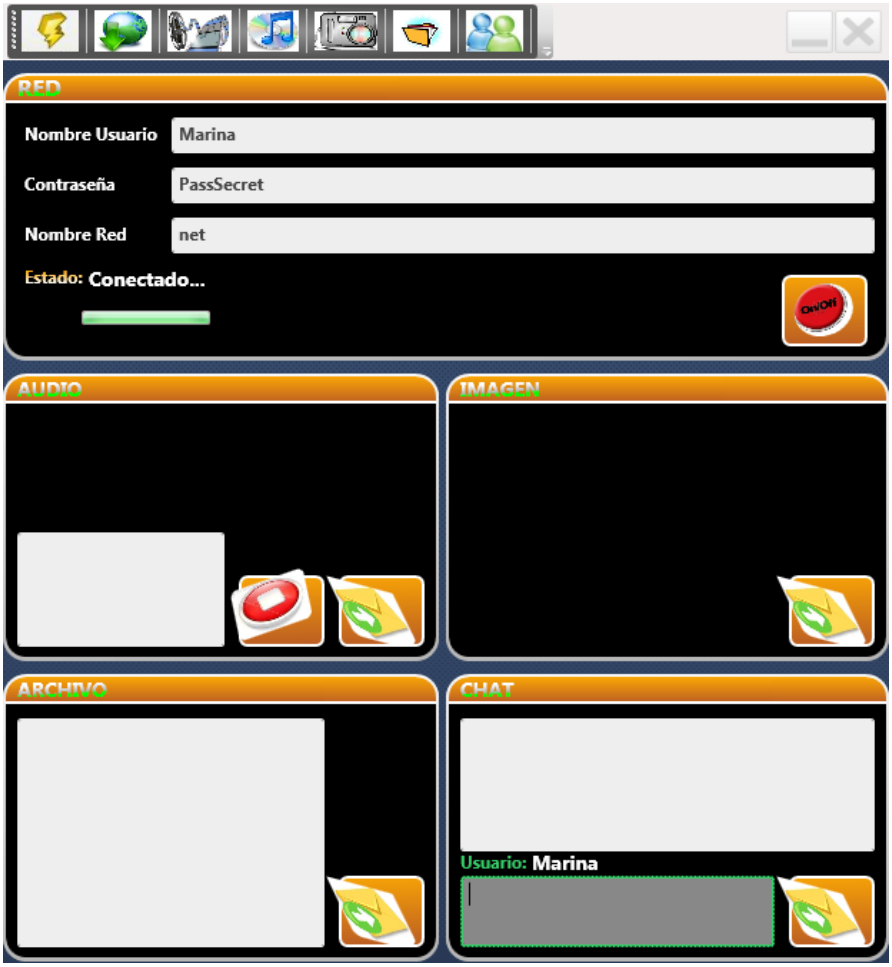
   Ver: La barra de progreso se muestra de color verde.

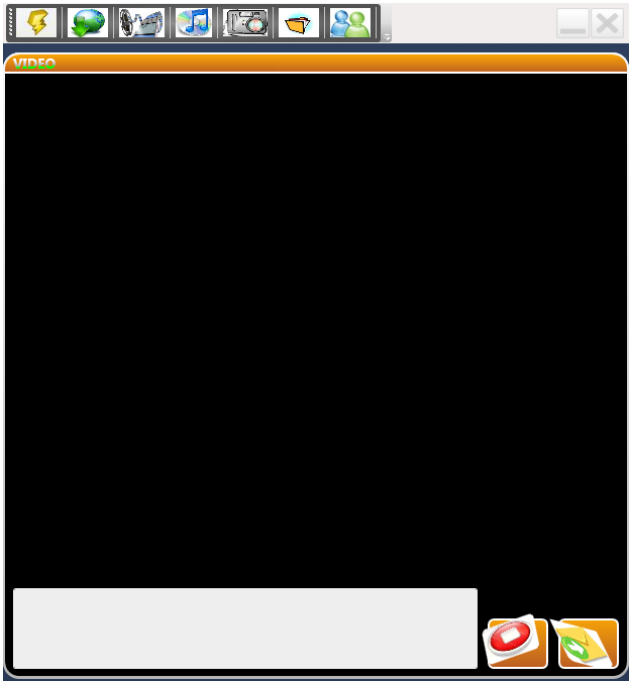
   Ver: Las cajas de texto correspondientes a “Nombre de Usuario”, “Contraseña” y “Nombre de Red” están deshabilitados.

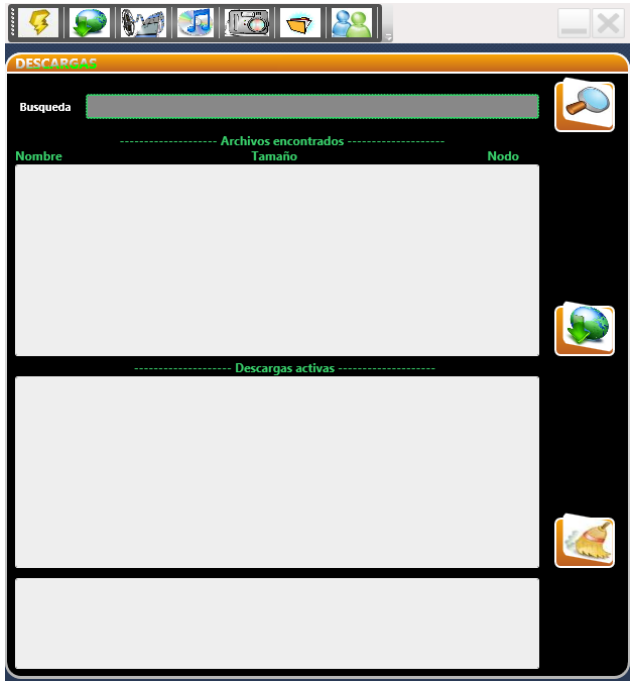
   Ver: El botón “Conectar” muestra un color naranja que indica que está habilitado.

   Ver: Los botones del resto de ventanas (“Chat”, “Imagen”, “Archivo”, “Audio”, “Video” y “Descargas”) se muestran en color naranja, por tanto habilitados.

   Ver: Los títulos de las ventanas muestran el texto en un tono verde, que indica que el usuario está conectado.







8. Hacer click de nuevo en el botón “Red”.

   Ver: Ventana “Red” oculta.

NOTA: Si el inicio de sesión es correcto, el usuario se conecta con el resto de usuarios que hayan iniciado sesión con el mismo Nombre de Red y la misma Contraseña.

- Si el usuario no está conectado, los botones de funcionalidad de las demás ventanas mostrarán un color gris, indicando que están deshabilitados.

- Si el inicio de sesión es incorrecto, el sistema muestra el mensaje de error “Login inválido”, donde se explica el motivo del error, y el usuario no se conecta.

  El sistema sólo admite como datos válidos de inicio de sesión, caracteres alfanuméricos y sin espacios en sus cajas de texto (Nombre de Usuario, Nombre de Red y Contraseña).

- El sistema no permite que se conecten a la misma red dos usuarios con el mismo Nombre de Usuario.

  Si esto ocurre, el sistema muestra el mensaje de error “Nombre de Usuario actualmente en uso”, donde se le sugiere al usuario que elija un Nombre de Usuario distinto:

  Ver: La caja de texto correspondiente a “Nombre de Usuario” muestra por unos segundos el texto “_USUARIO_FALLIDO_”.

  Ver: Las cajas de texto correspondientes a “Nombre de Usuario”, “Contraseña” y “Nombre de Red” están habilitadas.

  Ver: La caja de texto correspondiente a “Nombre de Usuario” se muestra sin texto, las cajas de texto correspondientes a “Nombre de Red” y “Contraseña” mantienen los datos introducidos.

  Ver: El botón “Conectar” se muestra color naranja, por tanto habilitado.

- El sistema es incapaz de funcionar si la conexión a Internet es a través de Proxy puesto que las redes con proxy no son accesibles para los sockets.

  Ver: El sistema muestra un mensaje de “Error crítico” donde se le informa al usuario de que la red no es accesible debido a la existencia de un proxy. A continuación se cierra la aplicación.

- Si el usuario no está conectado a Internet, el sistema será incapaz de comunicarse con el exterior, y la aplicación no funcionará.

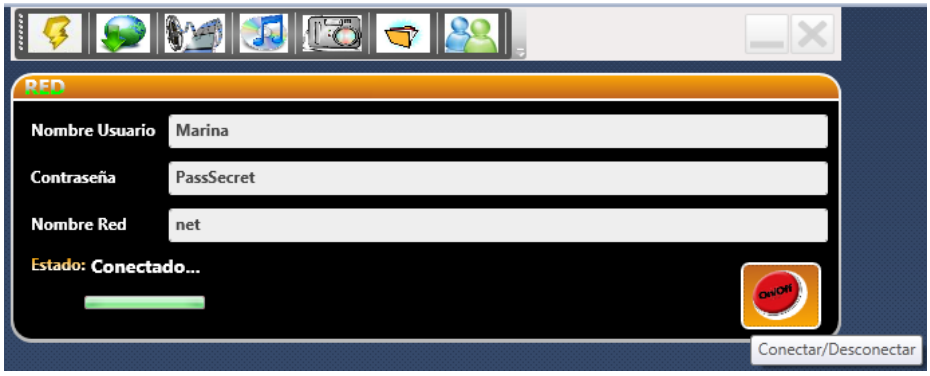
- El usuario puede ocultar y mostrar la ventana “Red” sin que la conexión se interrumpa.

FUNCIÓN "DESCONEXIÓN"

1. Hacer click en el botón "Red".

   Ver: Ventana "Red" abierta.

2. Hacer click en el botón "Conectar/Desconectar".



3. El sistema desconecta al usuario.

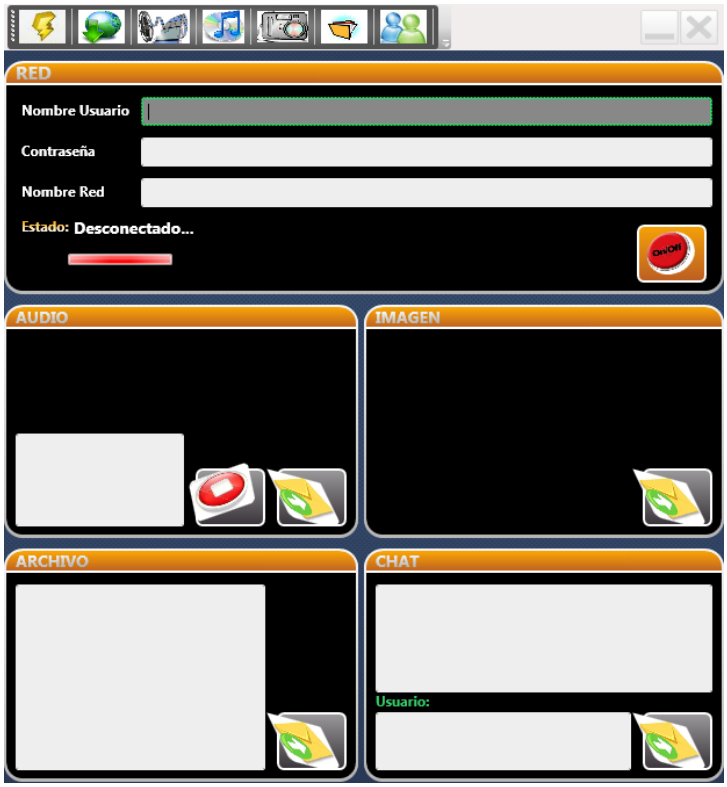
   Ver: Tras la desconexión, el botón "Conectar/Desconectar" se muestra en color naranja, por tanto habilitado.

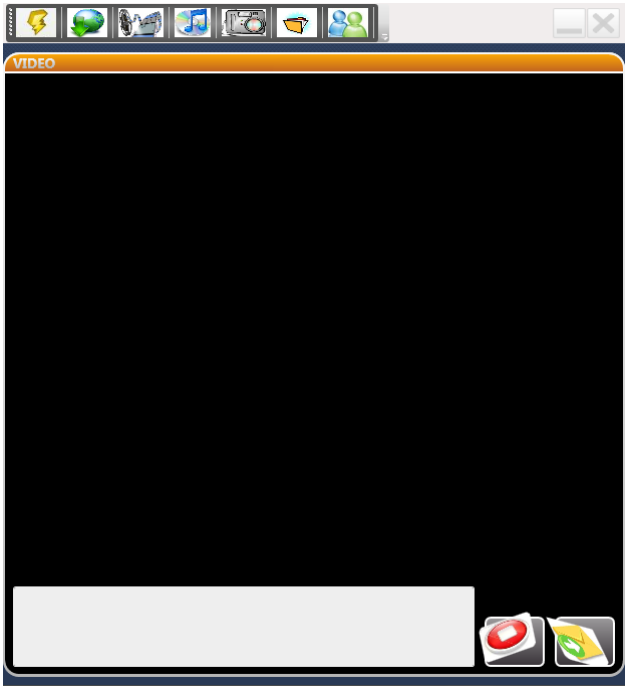
   Ver: Los botones de funcionalidad de las demás ventanas se muestran en color gris, por tanto deshabilitados.

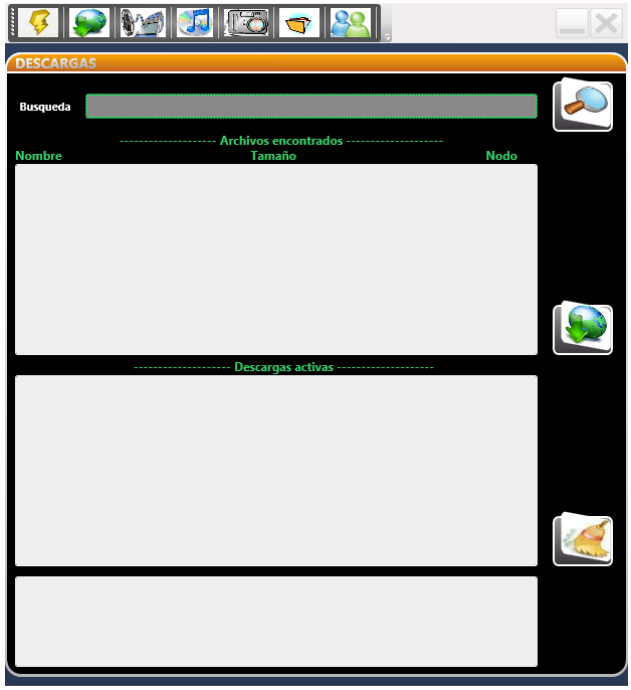
   Ver: La etiqueta de "Estado" de la ventana "Red" cambia a "Desconectado…"

   Ver: La barra de progreso se muestra de color rojo, que indica que el usuario está desconectado.

   Ver: Los títulos de las ventanas muestran el texto en blanco mate, sin tono verde, indicando que el usuario está desconectado.


RED
Nombre Usuario
Contraseña
Nombre Red
Estado: Desconectado...
On/Off
AUDIO
IMAGEN
ARCHIVO
CHAT
Usuario:


VIDEO


DESCARGAS
Busqueda
---------- Archivos encontrados ----------
Nombre
Tamaño
Nodo
---------- Descargas activas ----------

FUNCIÓN "CHAT"

1. Hacer click en el botón "Chat".

Ver: Ventana "Chat" abierta.

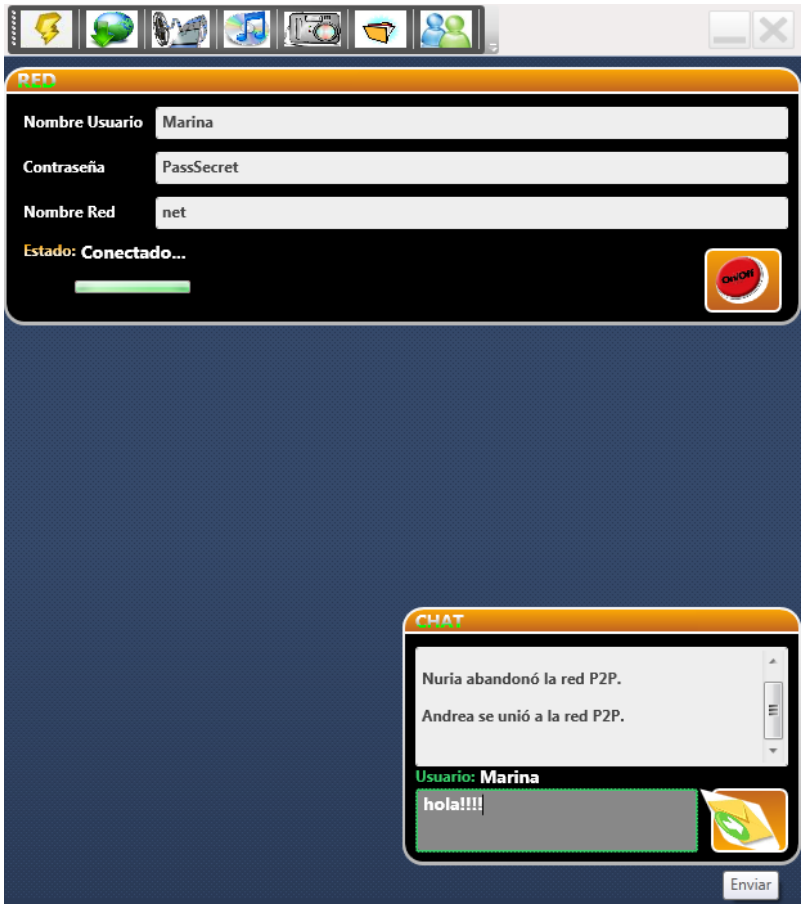
2. Teclear un mensaje en el área de texto correspondiente.

3. Hacer click en el botón "Enviar", o pulsar la tecla Intro.

Ver: El mensaje de texto se muestra en la ventana correspondiente de "Chat" en todos los usuarios conectados a la misma red.

Ver: Cada vez que un nuevo usuario se une a la red, el sistema informa al resto de usuarios de la red sobre la incorporación del nuevo usuario y muestra el nombre de este nuevo usuario en el área correspondiente de la ventana "Chat".

Ver: Cada vez que un usuario abandona la red, el sistema informa al resto de usuarios de la red sobre la retirada de este usuario y muestra su nombre en el área correspondiente de la ventana "Chat".



4. Hacer click de nuevo en el botón “Chat”.

5. Ver: Ventana “Chat” se oculta.

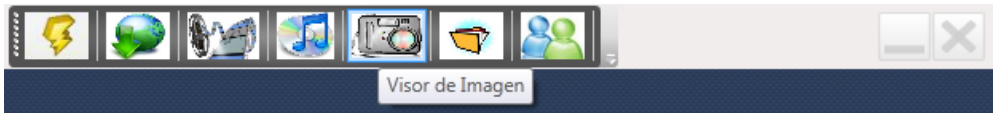
   Ver: Al cabo de un rato, el usuario puede hacer click de nuevo en el botón “Chat”; el sistema muestra en el área correspondiente de la ventana “Chat”, las conversaciones entre participantes hasta ese momento.

NOTA: Todos los usuarios conectados a la misma red envían y reciben mensajes entre sí. El sistema muestra las conversaciones en el área correspondiente de las respectivas ventanas “Chat”.

- El usuario puede ocultar y mostrar la ventana de “Chat” sin que la funcionalidad se interrumpa.

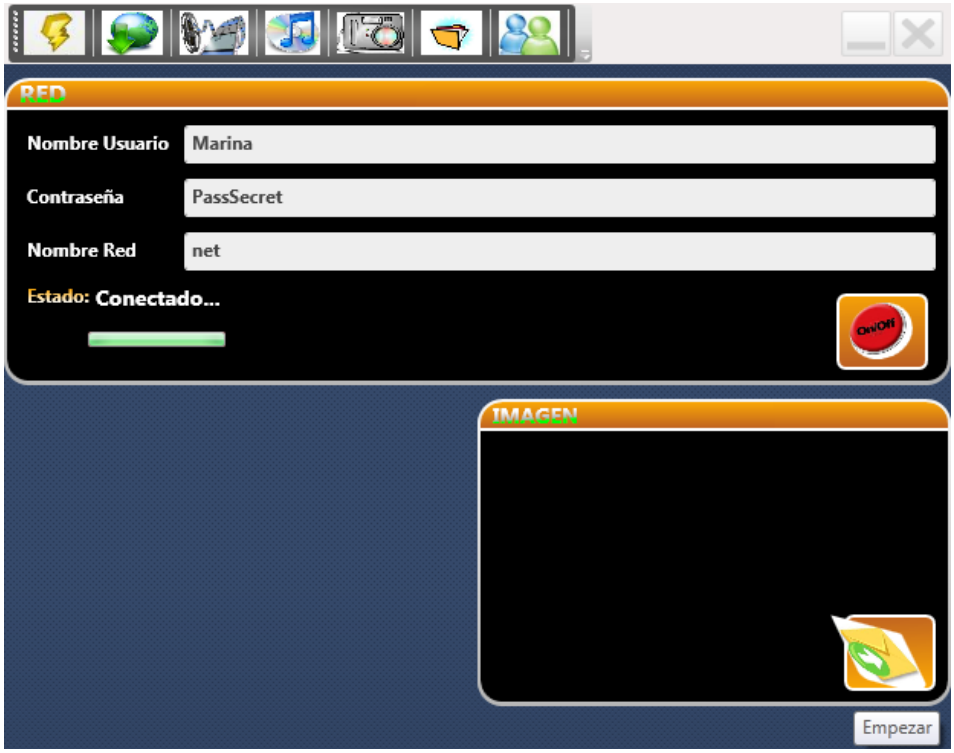
FUNCIÓN "VISOR DE IMAGEN"

1. Hacer click en el botón "Visor de Imagen".



Ver: Ventana "Imagen" abierta.

2. Hacer click en el botón "Empezar".

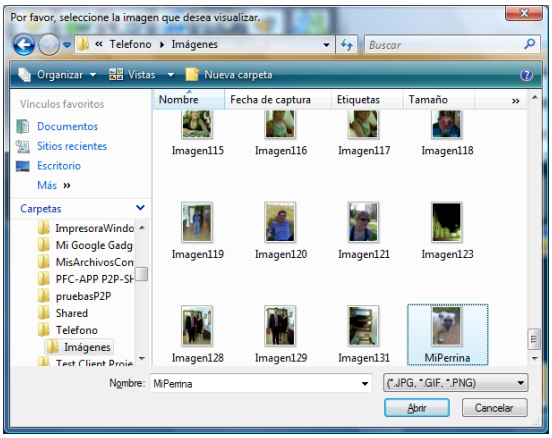


Ver: El sistema muestra el árbol de directorios de los archivos con formato de imagen (.jpg, .gif y .png) del equipo.

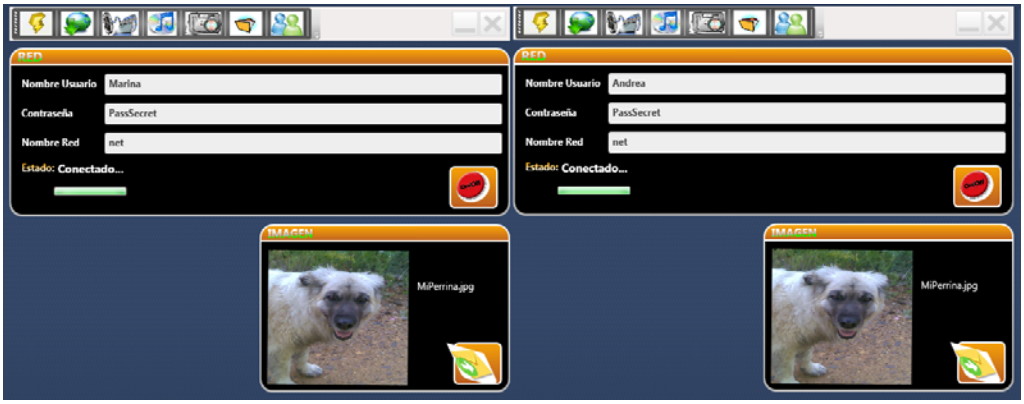
3. Seleccionar del árbol de directorios, el archivo de que se quiere visualizar con el resto de usuarios conectados.

Ver: El nombre del archivo seleccionado aparece en el campo correspondiente en el árbol de directorios.

4. Hacer click en el botón “Abrir” en el diálogo del árbol de directorios.



5. Ver: El sistema muestra en el área correspondiente de la ventana “Imagen” de todos los usuarios, la imagen seleccionada y su título.



6. Hacer click de nuevo en el botón “Visor de Imagen”.

7. Ver: Ventana “Imagen” oculta.

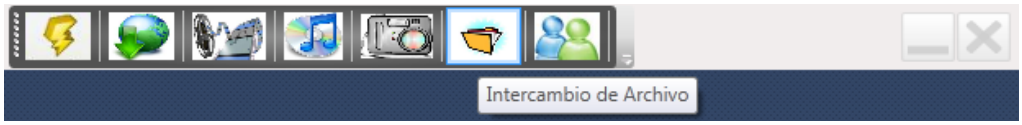
   Ver: Al cabo de un rato, el usuario puede hacer click de nuevo en el botón “Visor de Imagen”; el sistema mantiene en el área correspondiente de la ventana “Imagen”, el último archivo que se ha compartido en la red por cualquiera de los participantes.

NOTA: Todos los usuarios conectados a la misma red visualizan los archivos de imagen (*.jpg, *.gif, *.png) de todos los participantes conectados.

- El usuario puede ocultar y mostrar la ventana “Imagen” sin que la funcionalidad se interrumpa.

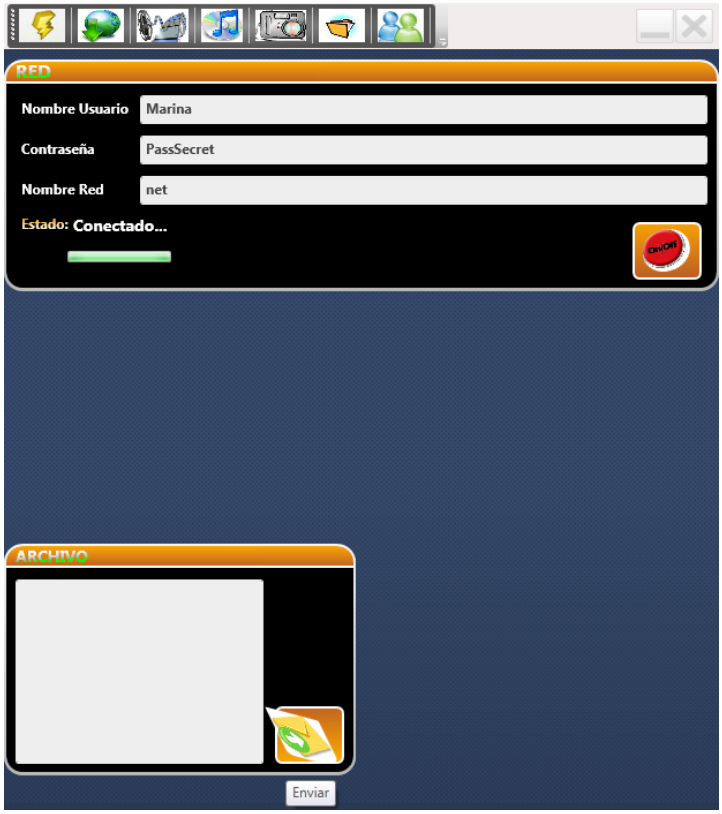
FUNCIÓN "INTERCAMBIO DE ARCHIVO"

1. Hacer click en el botón de "Intercambio de Archivo".



Ver: Ventana "Archivo" abierta.

2. Hacer click en el botón "Enviar".

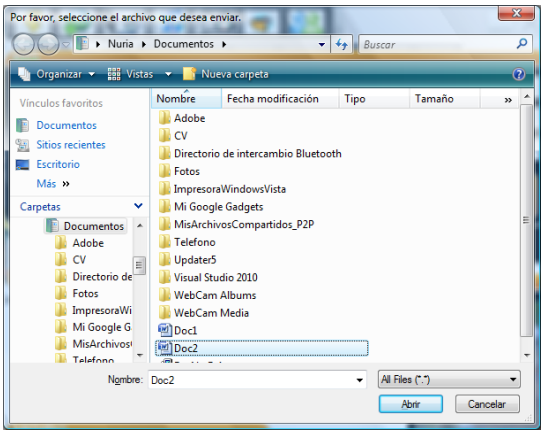


Ver: El sistema muestra el árbol de directorios del equipo.

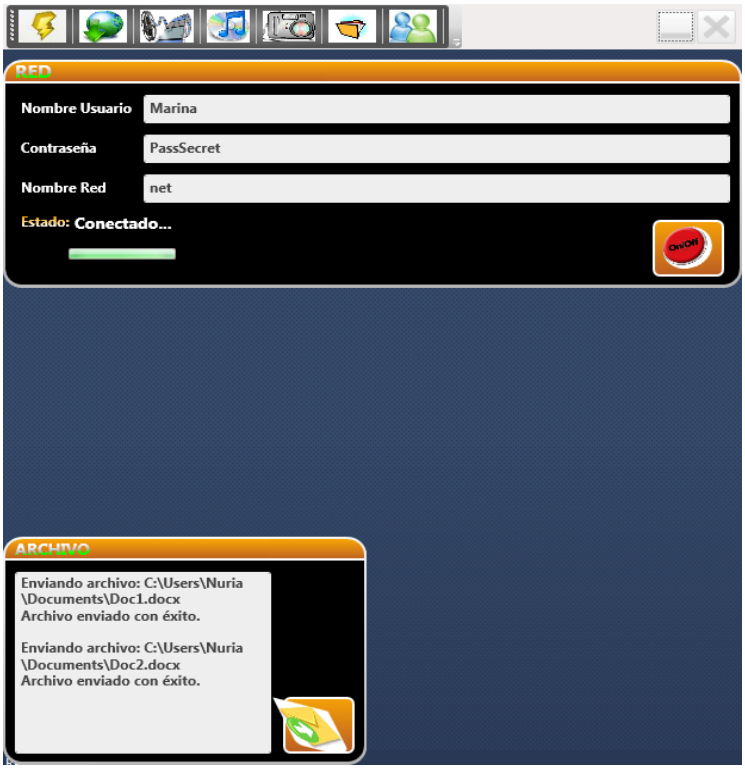
3. Seleccionar del árbol de directorios el archivo, con formato de cualquier tipo, que se quiera enviar al resto de participantes conectados.

Ver: El nombre del archivo seleccionado aparece en el campo correspondiente en el árbol de directorios.

4. Hacer click en el botón “Abrir” en el diálogo del árbol de directorios.



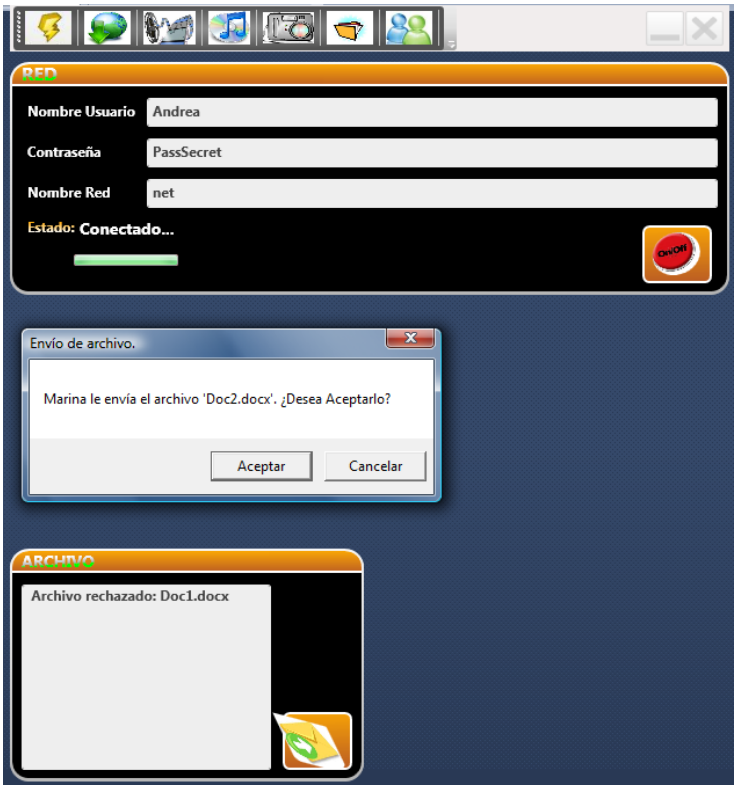
Ver: El sistema muestra en el área correspondiente de la ventana “Archivo” del emisor, información del envío: ruta y nombre del archivo que se envía y éxito o fracaso del proceso de envío.



Ver: El sistema muestra en las pantallas del resto de participantes conectados, el mensaje de “Envío de archivo”, donde se informa del nombre del emisor y el nombre del archivo y se ofrece la posibilidad de Aceptar o Rechazar el envío.

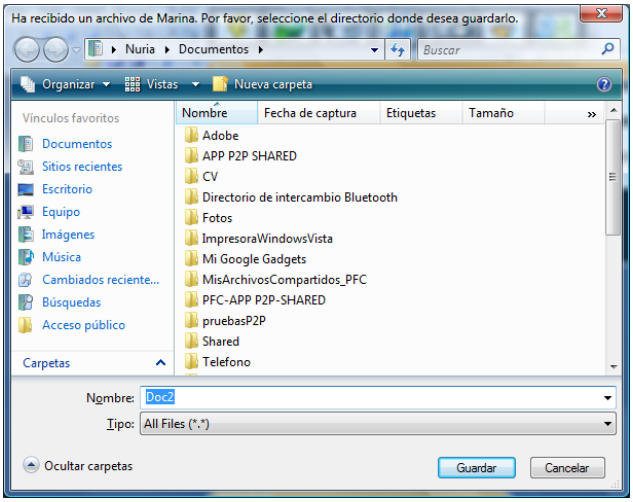
5. Si se quiere aceptar el envío, hacer click en “Aceptar” en el mensaje de “Envío de archivo”.

   Si se quiere rechazar el envío, hacer click en “Cancelar” en el mensaje de “Envío de archivo”.

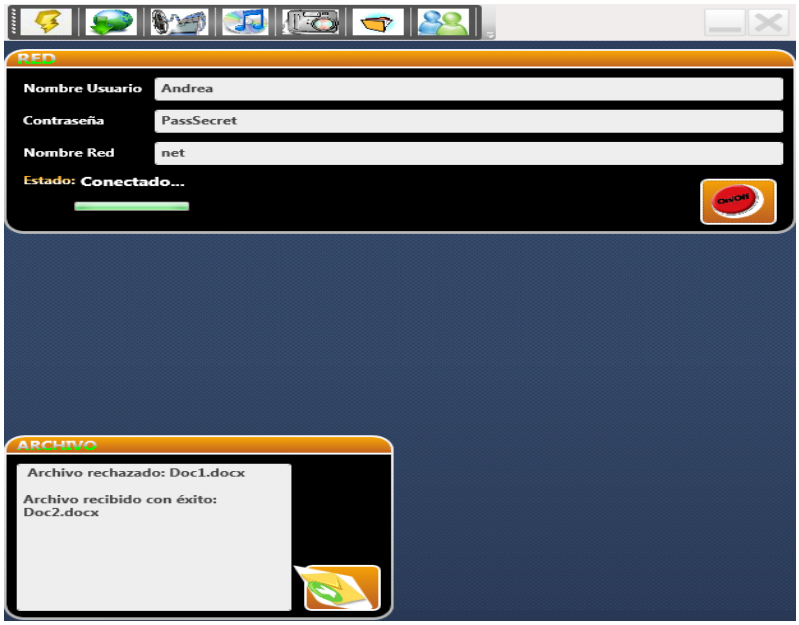


   Ver: El sistema muestra en las pantallas de todos los usuarios receptores que hayan aceptado el envío, sus respectivos árboles de directorios y se les pide que seleccionen un directorio para guardar el archivo recibido.

6. Hacer click en “Guardar” en el diálogo del árbol de directorios.



Ver: El sistema muestra en el área correspondiente de la ventana "Archivo" de los receptores, información de los archivos recibos (nombre de archivo y éxito o fracaso del proceso de envío), e información de los archivos rechazados (nombre del archivo rechazado).



7. Hacer click de nuevo en el botón "Intercambio de Archivo".

8. Ver: Ventana "Archivo" oculta.

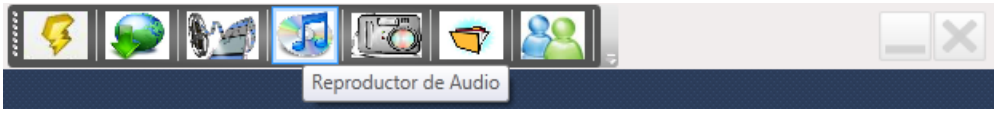
Ver: Al cabo de un rato, el usuario puede hacer click de nuevo en el botón de "Intercambio de Archivo"; el sistema mantiene en el área correspondiente de la ventana "Archivo", información de los archivos enviados, aceptados y rechazados.

NOTA: Todos los usuarios conectados a la misma red intercambian archivos de cualquier tipo con el resto de usuarios conectados. El sistema ofrece a los usuarios receptores la posibilidad de aceptar o rechazar el archivo. El sistema muestra en el área correspondiente de las respectivas pantallas de emisores y receptores, información del proceso de envío y de los archivos aceptados o rechazados.

✍ El usuario puede ocultar y mostrar la ventana "Archivo" sin que la funcionalidad se interrumpa.

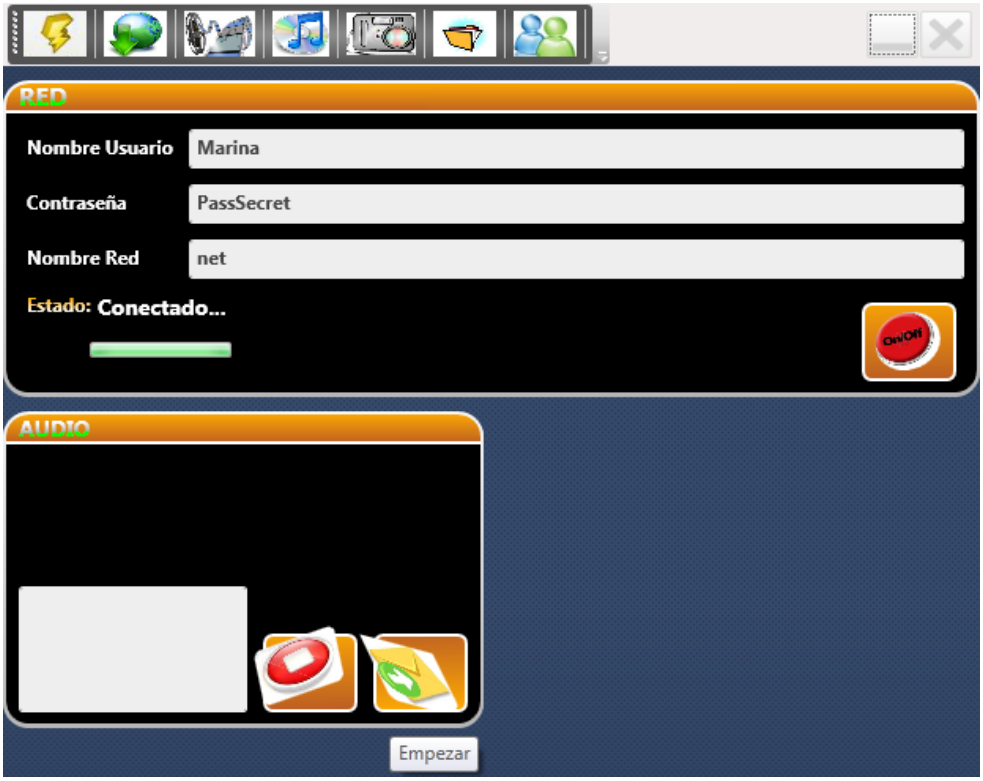
FUNCIÓN "REPRODUCTOR DE AUDIO"

1. Hacer click en el botón "Reproductor de Audio".



Ver: Ventana "Audio" abierta.

2. Hacer click en el botón "Empezar".

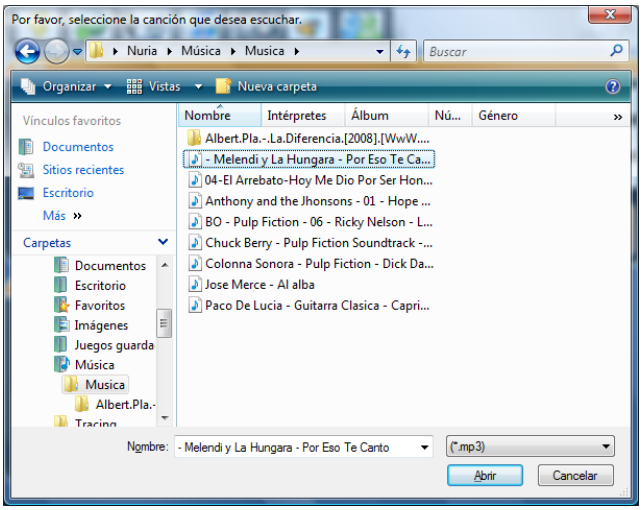


Ver: El sistema muestra el árbol de directorios los archivos con formato de audio (*.mp3, *. wma) del equipo.

3. Seleccionar del árbol de directorios el archivo que se quiere escuchar con el resto de participantes conectados.

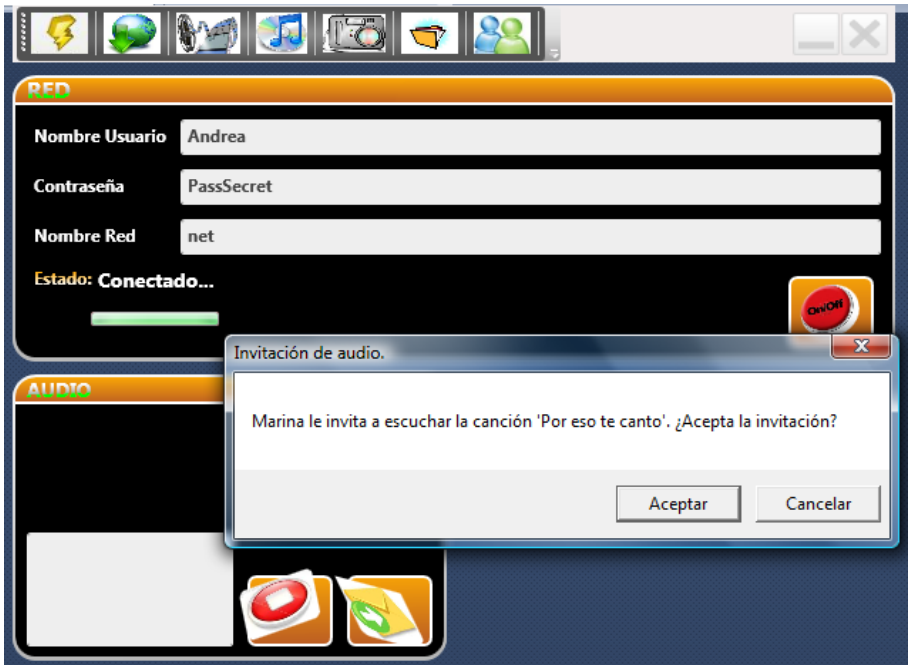
Ver: El nombre del archivo seleccionado aparece en el campo correspondiente en el árbol de directorios.

4. Hacer click en el botón “Abrir” en el diálogo del árbol de directorios.

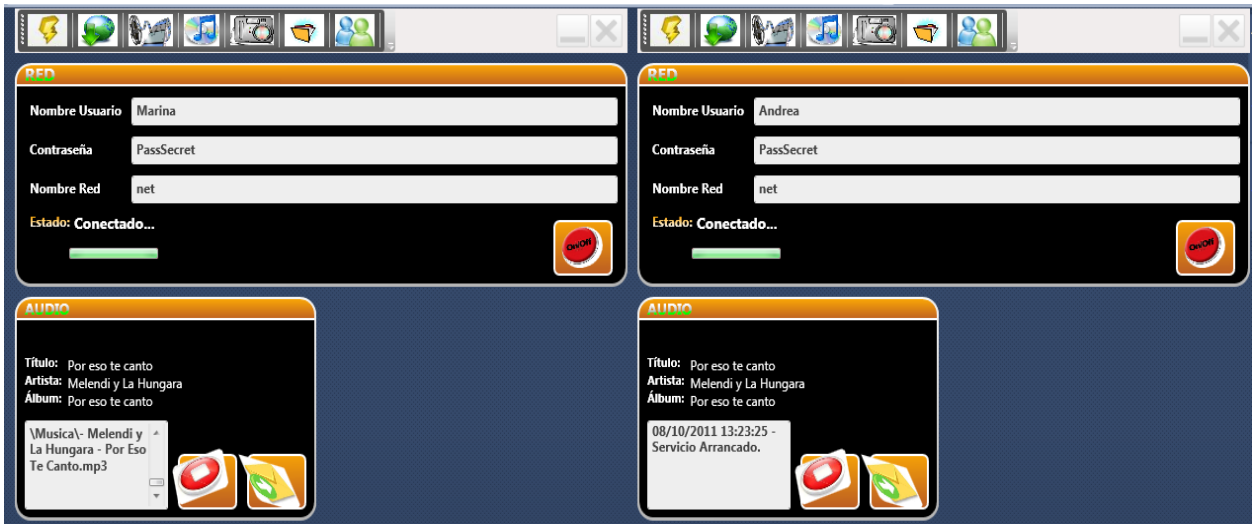


Ver: El sistema muestra en las pantallas del resto de usuarios conectados, el mensaje de “Invitación de audio”, donde informa del nombre del emisor y el nombre del archivo de audio y ofrece la posibilidad de Aceptar o Rechazar la invitación.

5. Si se desea escuchar la canción, hacer click en “Aceptar” en el mensaje de “Invitación de audio”.



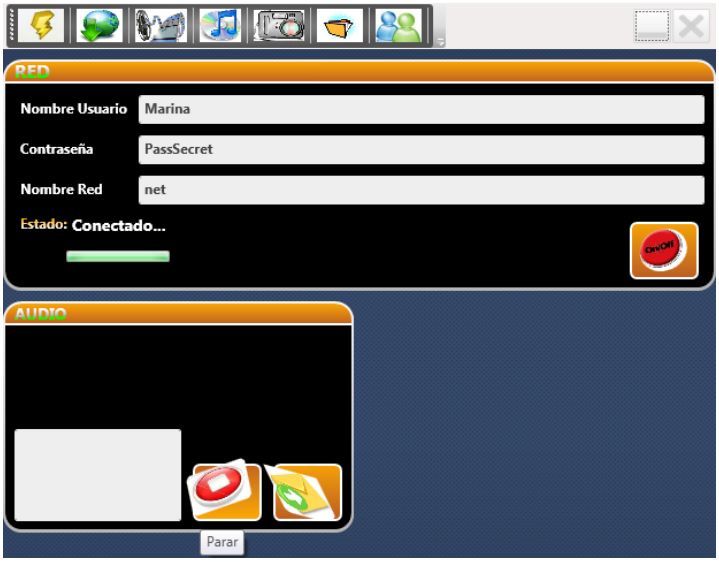
Ver: El sistema arranca el servicio de audio en el usuario emisor y en de todos los usuarios receptores que hayan aceptado la invitación, y muestra en el área correspondiente información del servicio e información del archivo de audio: Título, Artista y Álbum.



- Si se desea dejar de escuchar la canción, hacer click en el botón “Parar”.

Ver: Si el usuario que para el audio es un emisor, es decir, ha enviado la invitación de audio, la canción deja de escucharse en los equipos de todos los participantes de esa red.

Ver: Si el usuario que para el audio es un receptor, es decir, ha aceptado una invitación de audio, la canción solo deja de escucharse en su equipo.



7. Hacer click de nuevo en el botón “Reproductor de Audio”.

8. Ver: Ventana “Audio” oculta.

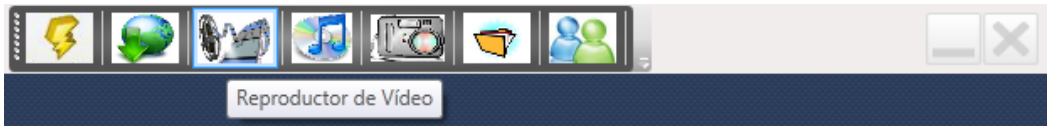
   Ver: Al cabo de un rato, el usuario puede hacer click de nuevo en el botón “Reproductor de Audio”; el sistema mantiene en el área correspondiente de la ventana “Audio”, la información del servicio de audio recopilada hasta ese momento.

NOTA: Todos los usuarios conectados a la misma red escuchan música simultáneamente. El sistema ofrece a los receptores la posibilidad de aceptar o rechazar las invitaciones de audio. El sistema muestra a emisores y receptores información del proceso multimedia y del archivo de audio.

- El usuario puede ocultar y mostrar la ventana “Audio” sin que la funcionalidad se interrumpa.

FUNCIÓN "REPRODUCTOR DE VÍDEO"

1. Hacer click en el botón "Reproductor de Vídeo".



Ver: Ventana "Video" abierta.

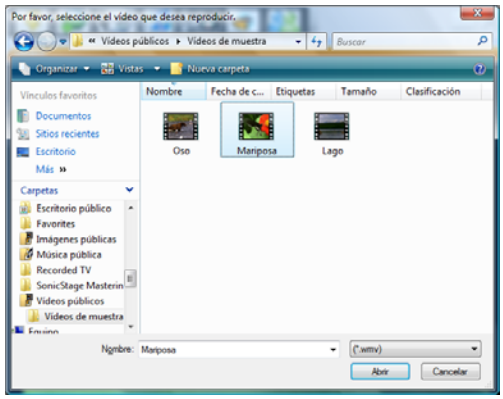
2. Hacer click en el botón "Empezar".

Ver: El sistema muestra el árbol de directorios los archivos con formato de vídeo (*. wmv) del equipo.

3. Seleccionar del árbol de directorios el archivo que se quiere visualizar con el resto de usuarios conectados.

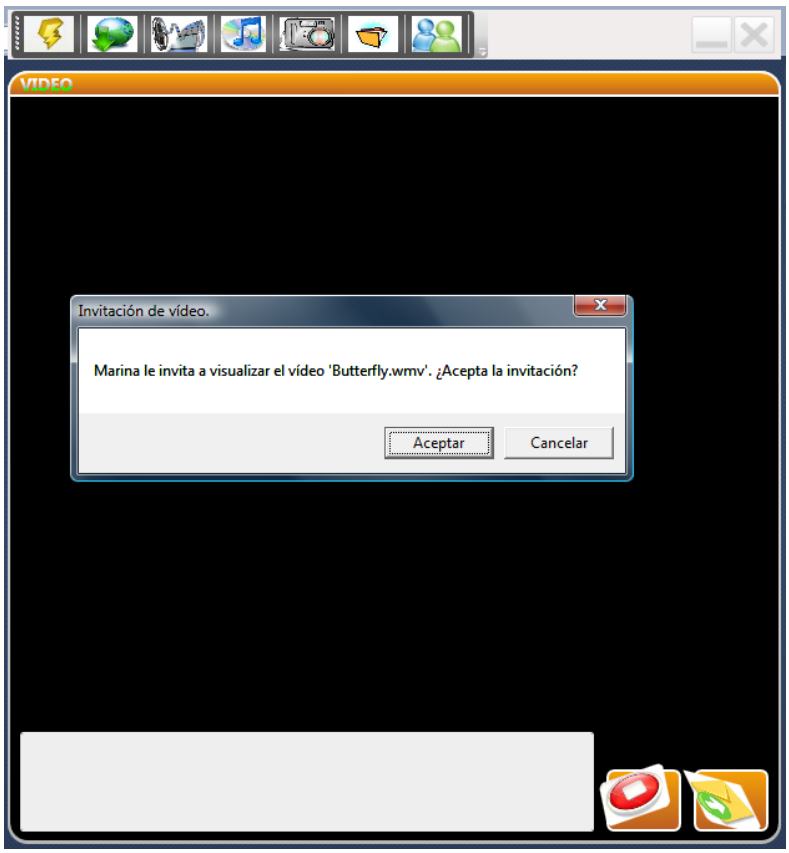
Ver: El nombre del archivo seleccionado aparece en el campo correspondiente en el árbol de directorios.

4. Hacer click en el botón “Abrir” en el diálogo del árbol de directorios.

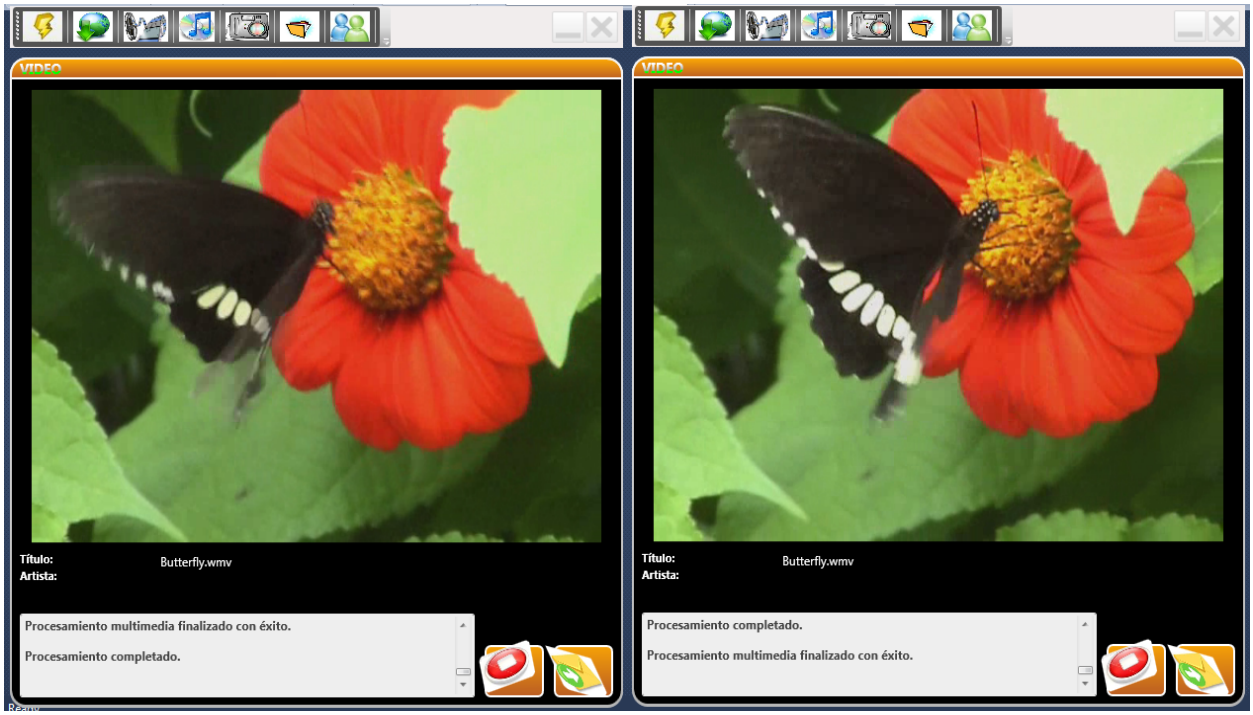


Ver: El sistema muestra en las pantallas del resto de participantes conectados, el mensaje de “Invitación de vídeo”, donde informa del nombre del emisor y el nombre del archivo de vídeo y ofrece la posibilidad de Aceptar o Rechazar la invitación.

5. Si se desea visualizar el vídeo, hacer click en “Aceptar” en el mensaje de “Invitación de vídeo”.



Ver: El sistema arranca el servicio de vídeo en el usuario emisor y en de todos los usuarios receptores que hayan aceptado la invitación, y muestra en el área correspondiente información del servicio e información del archivo de vídeo: Título y Artista.



- Si se desea dejar de visualizar el vídeo, hacer click en el botón “Parar”.

Ver: Si el usuario que para el vídeo es un nodo emisor, es decir, ha enviado la invitación de vídeo, el vídeo deja de visualizarse en los equipos de todos los participantes de esa red.

Ver: Si el usuario que para el vídeo es un nodo receptor, es decir, ha aceptado una invitación de vídeo, el vídeo solo deja de visualizarse en su equipo.

7. Hacer click de nuevo en el botón “Reproductor de Vídeo”.

8. Ver: Ventana “Video” oculta.

   Ver: Al cabo de un rato, el usuario puede hacer click de nuevo en el botón “Reproductor de Video”; el sistema mantiene en el área correspondiente de la ventana “Video”, información del servicio de vídeo recopilada hasta ese momento.

NOTA: Todos los usuarios conectados a la misma red visualizan vídeos simultáneamente. El sistema ofrece a los receptores la posibilidad de aceptar o rechazar las invitaciones de vídeo. El sistema muestra a emisores y receptores información del proceso multimedia y del archivo de vídeo.

- El usuario puede ocultar y mostrar la ventana “Video” sin que la funcionalidad se interrumpa.

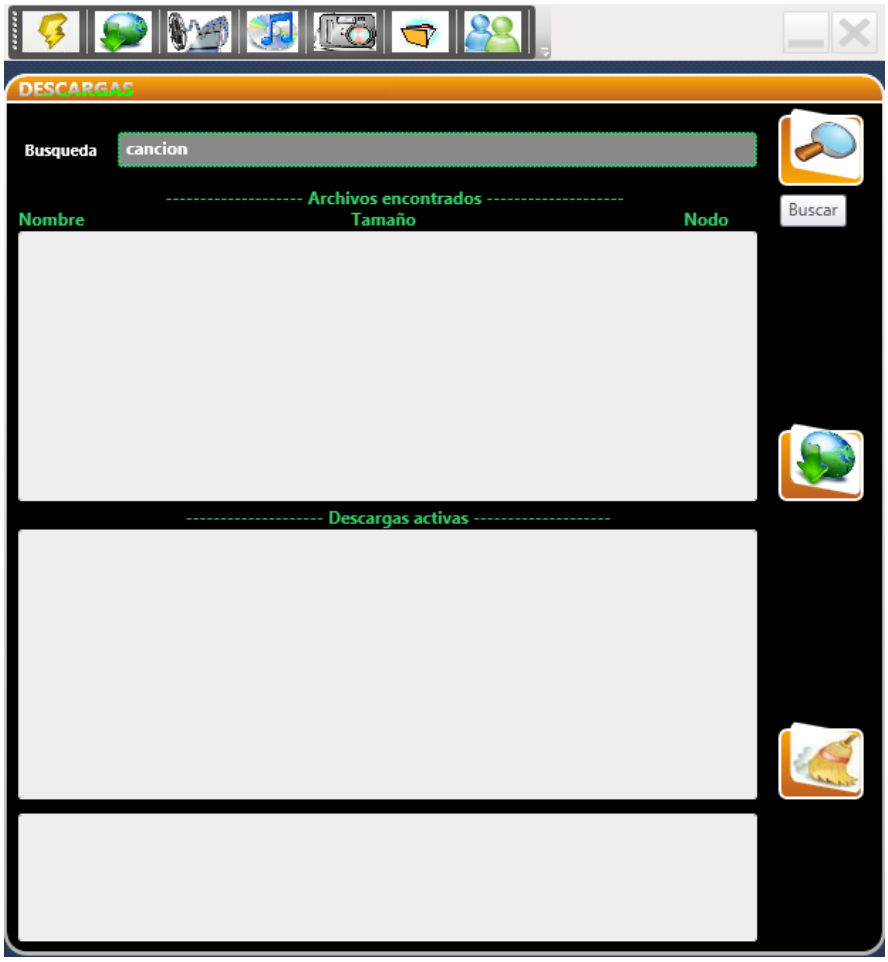
FUNCIÓN "DESCARGAS"

1. Hacer click en el botón "Descargas".

Ver: Ventana "Descargas" abierta.

2. Teclear un criterio de búsqueda (por ejemplo el título de una canción o el autor) en la caja de búsquedas y hacer click en el botón "Buscar".

   Ver: Si la caja de búsqueda está en blanco y el usuario hace click en "Buscar", el sistema muestra el mensaje de información "Acción requerida" donde solicita al usuario que introduzca algún criterio de búsqueda.

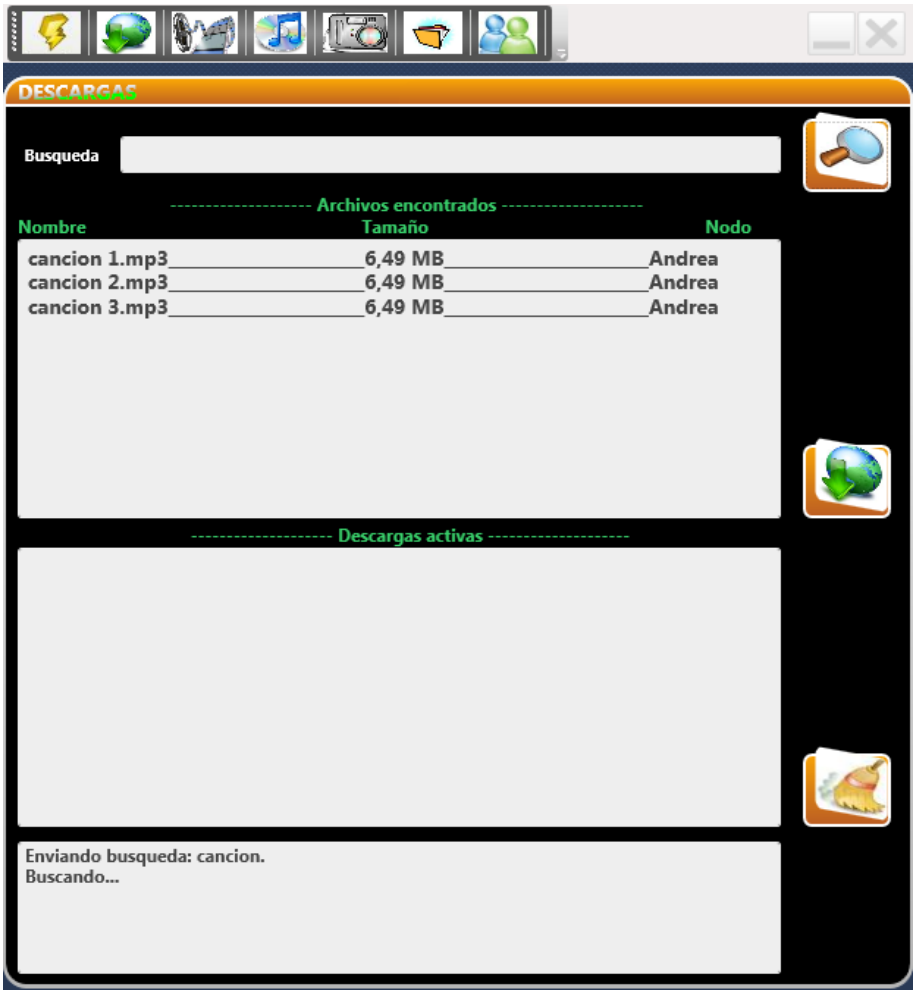


3. El sistema envía una solicitud de búsqueda a todos los usuarios conectados a la misma red.

   Ver: El sistema muestra información sobre el proceso de búsqueda, en el área correspondiente de la ventana "Descargas" del usuario que ha realizado la búsqueda.

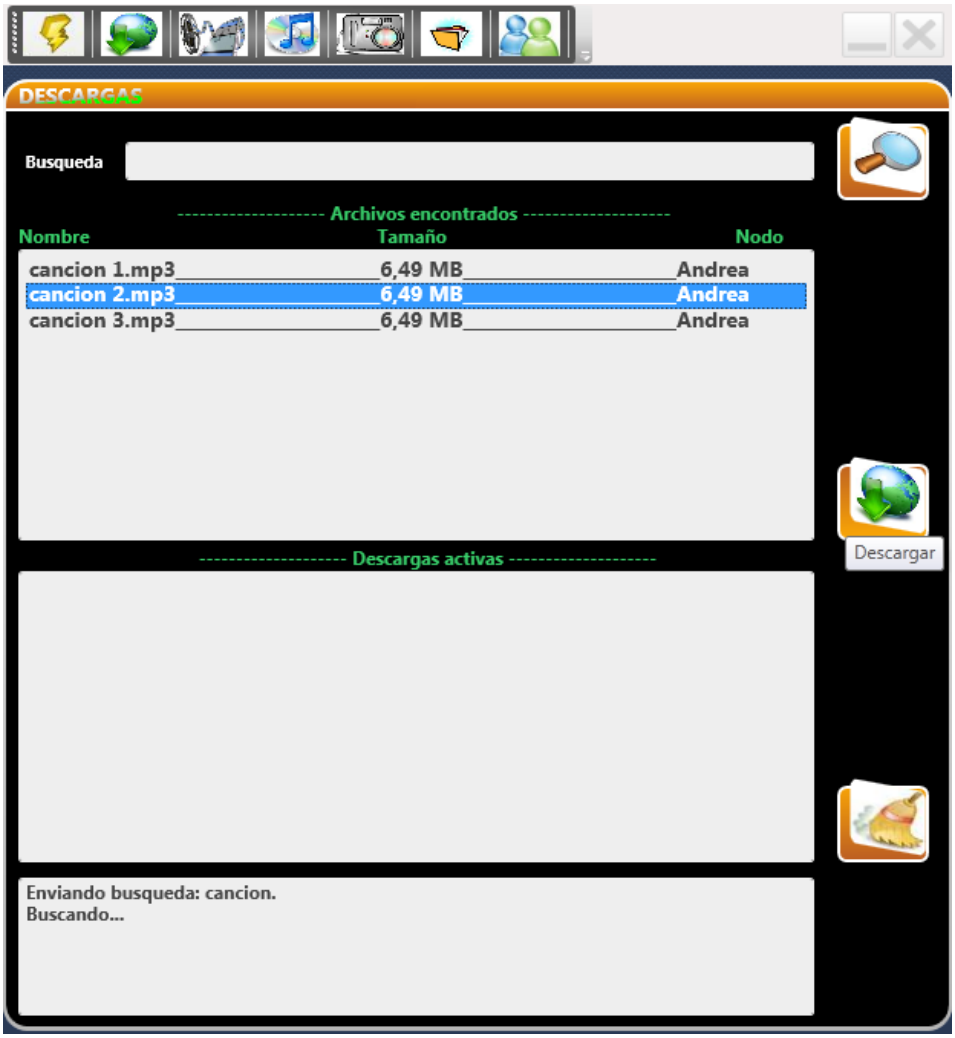
   El sistema busca en la carpeta compartida, "MisArchivosCompartidos_P2P" (creada de forma automática por la aplicación), de todos los participantes de la misma red, algún archivo cuyo nombre coincida al menos en una palabra con el recurso buscado. El usuario que ha realizado la búsqueda recibe una respuesta por cada archivo encontrado.

   Ver: El sistema muestra un listado de los archivos encontrados en el área "Archivo encontrados".



4. Seleccionar un archivo de la lista “Archivos encontrados” y hacer click en el botón “Descargar”.

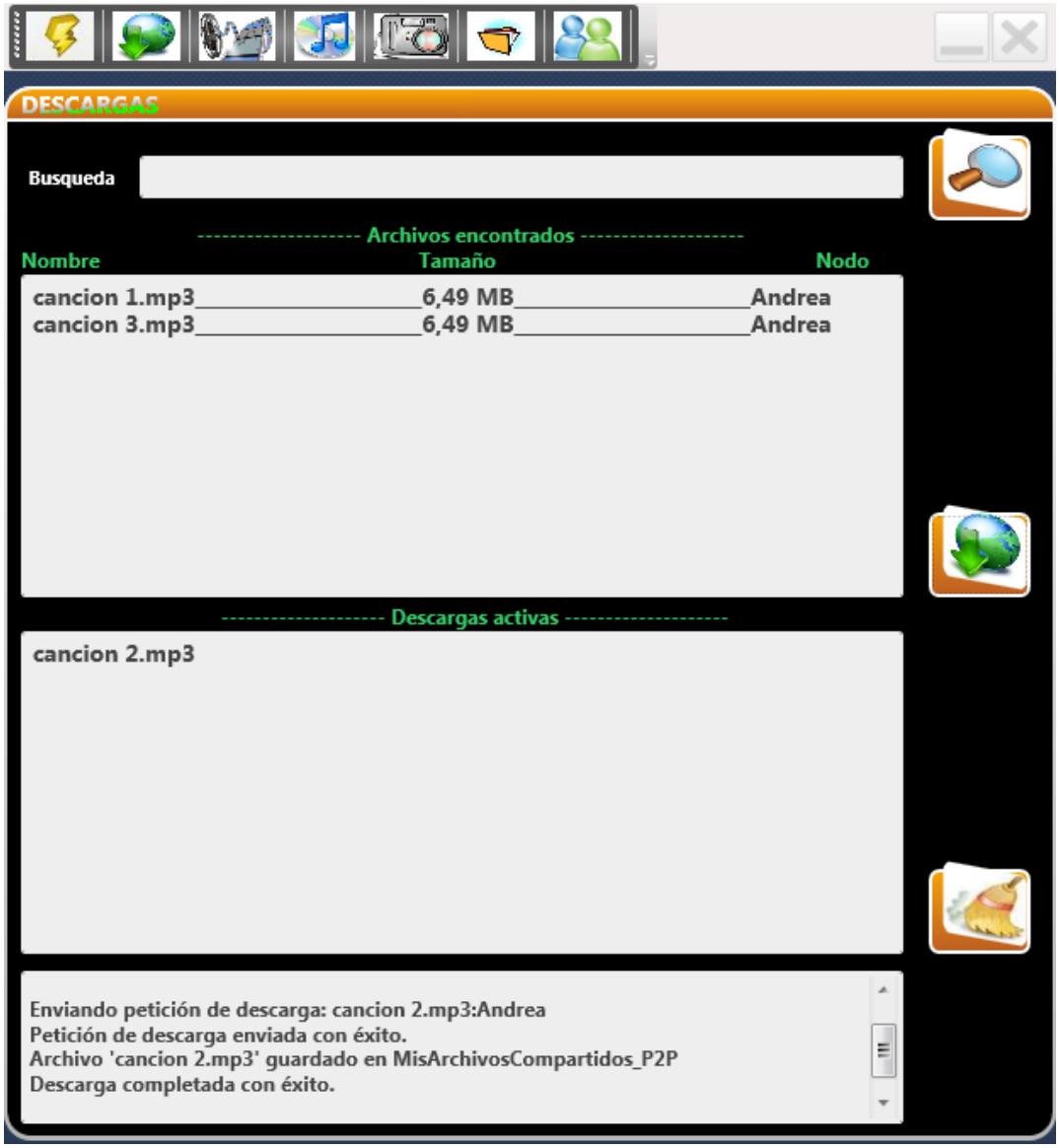
   Ver: Si el usuario no selecciona ningún archivo y hace click en “Descargar”, el sistema muestra el mensaje de información “Acción requerida”, donde solicita al usuario que seleccione un archivo de la lista.



5. El sistema elimina el archivo seleccionado de la lista de “Archivos encontrados” y lo muestra en la lista de “Descargas activas”.

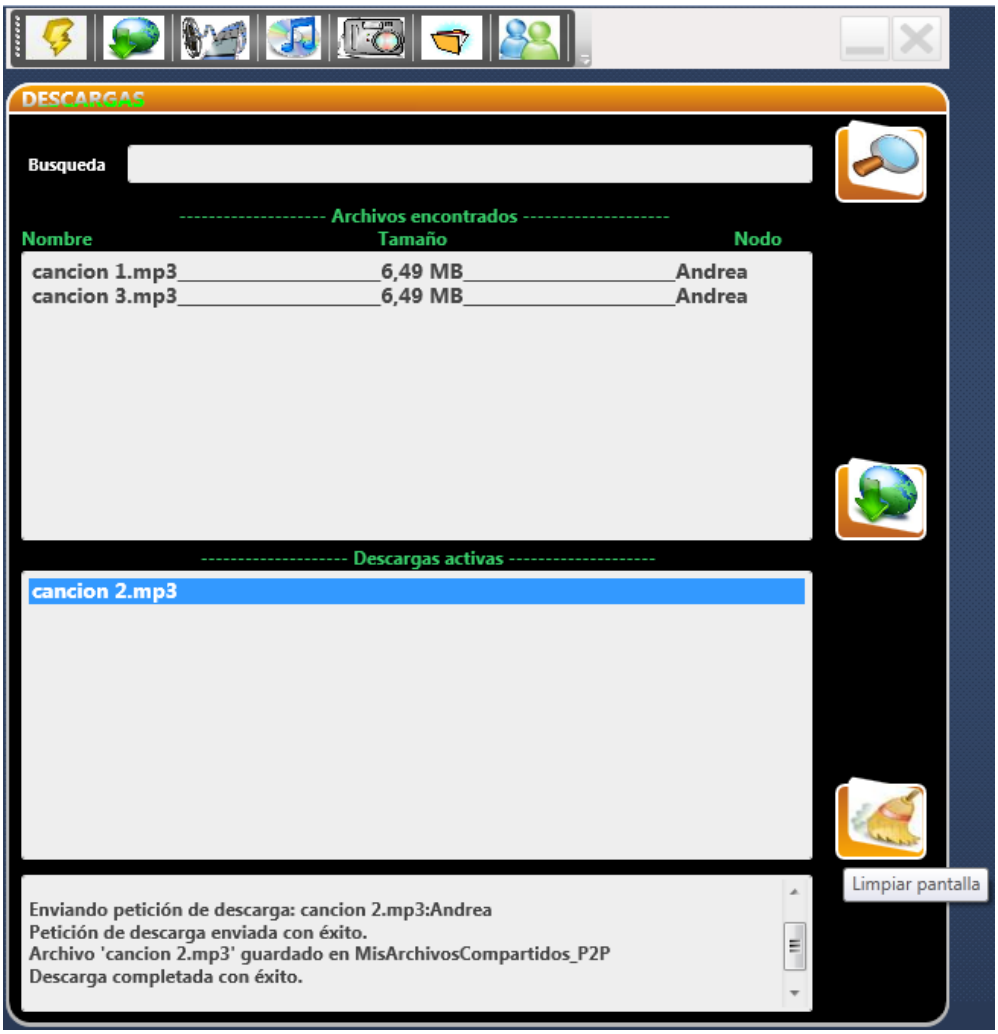
   El sistema procede a la descarga y guarda el archivo descargado en la carpeta compartida del usuario (MisArchivosCompartidos_P2P).

   Ver: El sistema muestra información sobre el proceso de descarga, en el área correspondiente de la ventana “Descargas” del usuario que ha descargado el archivo.



- Seleccionar un archivo de la lista de “Archivos encontrados” y/o de la lista de “Descargas activas” y hacer click en el botón “Limpiar pantalla”.

  El sistema borra de ambas listas los archivos seleccionados.



NOTA: Todos los usuarios conectados a la misma red buscan y descargan archivos entre sí. El sistema muestra información de los procesos de búsqueda y descarga en el área correspondiente de la ventana “Descargas” de cada usuario que realiza una búsqueda y/o una descarga.

- El usuario puede ocultar y mostrar la ventana “Descargas” sin que la funcionalidad se interrumpa.

8. CONCLUSIONES

8.1. CONCLUSIONES

No voy a negarlo, he encontrado muchas dificultades a lo largo del desarrollo. Más satisfactorio aún ha sido irlas resolviendo todas aunque sin duda la más complicada ha sido la lucha contra los plazos. En ocasiones al programar o al encontrar un error e intentar solventarlo, llegas a un atolladero sin salida en el que la paciencia y que no cunda el pánico son lo más importante.

Han sido muchas, muchísimas horas dándole vueltas a la cabeza, buscando caminos alternativos mejores, investigando métodos, estudiando códigos ajenos, buscando soluciones…

Sin duda he aprendido mucho acerca del proceso de ingeniería del software. A lo largo de la carrera puedes estudiar partes del mismo por separado, pero no es hasta que lo llevas a cabo, cuando te haces una idea general de lo que supone.

En cuanto a las tecnologías utilizadas tal vez me ha resultado más difícil de lo que esperaba. Desde el lenguaje de programación C#, hasta el desarrollo de aplicaciones P2P, pasando por la plataforma .NET, WPF o WCF, han sido áreas totalmente nuevas y desconocidas para mí, lo cual también ha resultado ser un reto importante.

Me hubiera gustado saber como resultaría el mismo proyecto si lo comenzara en este instante... pero eso ya será otro proyecto.

8.2. LÍNEAS FUTURAS

En este apartado se comentan las futuras mejoras y ampliaciones que se pueden desarrollar sobre la aplicación.

Una de las mejoras sería añadir una base de datos dinámica que actualice información de los usuarios, como nombres de los archivos que comparten o su identificador de usuario, cada vez que estos se conectan a la red.

También sería interesante que la red fuera semi-descentralizada, gestionada por super-peers elegidos aleatoriamente y encargados de proporcionar información sobre la red, como por ejemplo, qué usuarios tienen qué archivos o qué inicios de sesión no son válidos por existir ya un usuario con ese mismo nombre.

Otra mejora importante podría ser que las descargas de archivos se llevaran a cabo de manera más distribuida, permitiendo descargar trozos distintos del mismo archivo en varias máquinas.

Se podría mejorar también la funcionalidad de los recursos multimedia, implementando cámaras web, micrófonos, etc

También sería muy interesante que la aplicación pudiera comunicarse con otras redes y otros protocolos, como *Kademlia*, *Gnutella* o *BitTorrent*.

9. REFERENCIAS

9.1. BIBLIOGRÁFICAS

- Ingeniería del Software, Un enfoque práctico – Roger S. Pressman. McGraw Hill.
- Ingeniería de Software Orientado a Objetos – Bernd Bruege, Allen H. Dutoit. Prentice Hall
- Construcción de Software Orientado a Objetos – Bertrand Meyer. Prentice Hall
- Patrones de Diseño – Erich Gamma. Addison Wesley
- Enciclopedia de Microsoft C# – Fco. Javier Ceballos. RaMa
- Head First C# - Andrew Stellman & Jennifer Greene. O'Reilly
- Programming WCF Services – Juval Löwy. O'Reilly
- Peer to Peer: Harnessing the Power of Disruptive Technologies – Andy Oram. O'Reilly
- Professional C# 4 and .NET 4 – Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson, Morgan Skinner. John Wiley & Sons

9.2. ELECTRÓNICAS

- http://msdn.microsoft.com/en-us/magazine/cc188685.aspx
- http://msdn.microsoft.com/es-es/library/ms123401.aspx
- http://www.elpais.com/articulo/sociedad/Gobierno/deja/intacta/ley/antidescargas/elpepusoc/20100319elpepusoc_3/Tes
- http://www.elpais.com/articulo/cultura/ley/Sinde/pasa/examen/europeo/elpepucul/20110722elpepucul_16/Tes
- http://www.elpais.com/articulo/tecnologia/OSCE/reivindica/acceso/Internet/derecho/elpeputec/20110719elpeputec_6/Tes
- http://www.elpais.com/articulo/tecnologia/sentencia/compromete/filosofia/ley/Sinde/elpeputec/20110713elpeputec_4/Tes
- http://www.noticiasdot.com/wp2/2011/02/02/100-000-usuarios-de-redes-p2p-han-sido-demandados-en-eeuu-durante-el-ultimo-ano/
- http://www.willydev.net/descargas/WillyDev_InterfazdeUsuarioparaDesarrolladores1.pdf
- http://www.ebooksdownloadfree.com/Cisco-Networking/Peer-To-Peer-ApplicationDevelopment-Cracking-The-Code-BI796.html