UNIVERSIDAD DE EL SALVADOR

FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE

DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA

TRABAJO DE GRADUACIÓN

TEMA:

"DESARROLLO DE UNA APLICACIÓN .NET PARA APOYAR LA GESTIÓN ADMINISTRATIVA DEL DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA DE LA FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE"

PARA OPTAR AL GRADO DE:

INGENIERO DE SISTEMAS INFORMATICOS

PRESENTADO POR:

CERNA CALDERON, FREDY DARIO.

GROSS RUANO, MONICA MARIA AUXILIADORA.

DOCENTE DIRECTOR:

ING. JOSÉ ROLANDO CENTE MATAMOROS.

26 de ABRIL , 2010

[SANTA ANA EL SALVADOR CENTRO AMERICA]

UNIVERSIDAD DE EL SALVADOR

RECTOR

ING. Y MSC. RUFINO QUEZADA SANCHEZ

VICE-RECTOR ACADEMICO

ARQ. Y MASTER ANGEL PEREZ RAMOS

VICE-RECTOR ADMINISTRATIVO

LICDO. Y MASTER OSCAR NOE NAVARRETE

SECRETARIO GENERAL

LICDO. DOUGLAS VLADIMIR ALFARO CHAVEZ

FISCAL GENERAL

DR. RENE MADECADEL PERLA JIMENEZ

FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE

DECANO

LICDO. JORGE MAURICIO RIVERA

VICE-DECANO

LICDO. Y MASTER ELADIO EFRAÍN ZACARÍAS ORTEZ

SECRETARIO DE LA FACULTAD

LICDO. VICTOR HUGO MERINO QUEZADA

JEFE DE DEPARTAMENTO

ING. RAÚL ERNESTO MARTINEZ BERMUDEZ

DEDICATORIA

Antes que nada quiero agradecer a Dios todo poderoso, por permitirme culminar mis estudios y finalizar este ambicioso trabajo de grado. Por darme la fortaleza para superar todos los obstáculos y desafíos en el camino.

A toda mi familia a mi madre Gloria Elizabeth Calderón de Cerna por estar a mi lado siempre, por regalarme todo su amor y cariño sin el cual hubiese sido imposible lograr mis metas, y de forma muy especial a mi padre Fredy Darío Cerna Cortez, que el recuerdo de su memoria siempre me ha dado fortaleza para seguir adelante. A mis hermanas Martha Elizabeth Cerna Calderón y Alma Raquel Cerna Calderón por brindarme su cariño.

A mis queridas sobrinas Alicia Elizabeth Ovando Cerna y Emily Nicole Gonzales Cerna quienes me dan un incentivo para seguir adelante.

A mi docente asesor Ing. José Rolando Cente Matamoros, por regalarme su conocimiento de forma desinteresada, con paciencia y por ser un verdadero maestro y amigo y enseñarme a ver la vida desde otra perspectiva.

De forma muy especial a la familia Barillas Perdomo: Luis Alonso Barillas Cornejo, Cristina de los Angeles Perdomo de Barillas, Gabriela Leonor, Luisa Iliana y Astrid Elena Barillas Perdomo por brindarme su apoyo, amistad desinteresada, sincero afecto en todo momento. Gracias.

A mis queridos amigos Luis Mario Echeverria Moreno, Edwin Arturo Figueroa Molina, Cecybeth María Flores Lucero, Michael Alexander Posada Figueroa por brindarme su valiosa e invaluable amistad, su apoyo y confianza, Gracias.

A mi compañera de Tesis Mónica María Auxiliadora Gross Ruano, por acompañarme en esta tarea tan grande e importante para mi vida.

Fredy Darío Cerna Calderón.

DEDICATORIA

Gracias primeramente **a TI DIOSITO** por ser tan bueno y misericordioso conmigo, gracias por darme la Vida, salud y sobre todo la Fuerza y Gracia de culminar una meta más en la vida, ya que la he logrado solo con tú ayuda**,** para TI sea LA GLORIA.

A mis amados Padres Antonio Salvador Gross Mendoza y Teodolinda Ruano de Gross, por el ejemplo de lucha constante, apoyo, amor, consejos, oraciones y todas las buenas características, que sin duda alguna puedo decir que han sido, son y serán los mejores papis que Dios me pudo haber dado.

A mis Hermanos, Antonio Salvador Gross Ruano, Fátima Estefani Gross Ruano por estar conmigo en las malas y en las buenas, los quiero montonnnn.

A mi Abuelo, Por el cariño, por ser tan grande ejemplo de sacrificio, bondad, Honradez, Acción, y sobre todo la enseñanza a no desfallecer, sea cual sea el obstáculo que se esté pasando.

A mis Tíos, Ing. Mauricio Ernesto García Eguizábal, por su apoyo, sus constantes consejos a lo largo de toda la carrera. Y en general a todos ellos por incentivarme a culminar con este proyecto.

A mi asesor de Tesis, Ing. Rolando Cente Matamoros por el tiempo tan valioso que nos dedico impartiéndonos de su conocimiento tanto intelectual como de la vida diaria.

A mi compañero de Tesis Fredy Darío Cerna Calderón por su amistad y apoyo incondicional durante todo el proceso del desarrollo de la presente tesis. **A su Familia,** por apoyarnos y sus finas atenciones todo el tiempo.

A mí siempre amigo, por incentivarme a ser mejor persona y enseñarme a ver la vida desde un punto de vista real, su cariño y su respaldo en todo momento.TQM!

Ing. Mónica María A. Gross Ruano.

INDICE

Capítulo VI - Plan de Implementación de la Aplicación

Capítulo VII - Conclusiones y Recomendaciones

Anexos

Introducción

En este documento se presenta el proyecto de tesis denominado "DESARROLLO DE UNA APLICACIÓN .NET PARA APOYAR LA GESTIÓN ADMINISTRATIVA DEL DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA DE LA FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE."

En el que inicialmente se exponen una serie de etapas que han sido necesarias antes de la elaboración del aplicativo. Primero se muestran los objetivos en los que se define cual es la finalidad del proyecto. Luego, una investigación mediante la cual se identifico la importancia que tiene para el Departamento de Ingeniería y Arquitectura la puesta en marcha del aplicativo propuesto, también se delimitan los alcances en los que se basa este desarrollo, y finalmente la metodología de trabajo a utilizar.

Una vez terminada la investigación previa, se describe mediante los capítulos que se presenta a continuación, la evolución del proyecto.

En el capítulo uno se mencionan generalidades como la evolución de la Facultad Multidisciplinaria de Occidente junto con su Misión y Visión así mismo la evolución del Departamento de Ingeniería y Arquitectura y el organigrama de cómo está formado en lo que respecta al personal que ahí labora. También se presentan los diferentes servicios que este presta a la población estudiantil.

En el capitulo dos se detalla el análisis Preliminar, el cual consta de la explicación de la situación Actual de Departamento de Ingeniería y Arquitectura para poder identificar la problemática que ahí se tiene y con esto poder establecer la solución a este.

En el capitulo tres se describen los requerimientos de información, de la cual se valió para el desarrollo del aplicativo, así como también requerimientos operativos y técnicos que el usuario que desea hacer uso de este debe considerar para el uso efectivo, y finalmente se describen el análisis de Factibilidad del proyecto.

En el capitulo cuatro se habla del diseño que se realizo para la elaboración del aplicativo Sistema de Comunicación Integral Punto a Punto (SCIP2P) , en este se explica claramente cómo funciona cada una de las partes que conforma el aplicativo.

En el capitulo cinco se muestra como haciendo uso del diseño, se desarrollo el aplicativo y así también las pruebas que se le realizaron.

Finalmente, en el capitulo Seis se presenta el plan de implementación del Sistema de Comunicación Integral Punto a Punto.

Objetivos

Objetivo General.

Desarrollar un sistema informático que dé respuesta a las necesidades de comunicación y apoyo a las tareas de gestión administrativa, actuales del departamento de Ingeniería y Arquitectura de la Facultad Multidisciplinaria de Occidente, integrando todas las entidades que lo constituyen.

Objetivos Específicos

- Indagar las necesidades del personal y entes que se ven involucrados en las actividades del Departamento, por medio de herramientas de investigación.
- Hacer posible una comunicación eficiente entre todas las entidades relacionadas con las actividades del Departamento de Ingeniería y Arquitectura, para lograr una mayor productividad de las labores que se presentan diariamente.
- Facilitar la administración sistemática del activo que pertenece al Departamento de Ingeniería y Arquitectura, para lograr mantener una base de datos de todo lo que este posee de una manera ordenada.
- Proporcionar a los alumnos mecanismos que mejoren el acceso a la información y su participación en las diferentes actividades académicas.
- Brindar a los alumnos herramientas que simplifiquen las tareas que realizan a diario.

Antecedentes

Durante la vida del Departamento de Ingeniería y Arquitectura se han realizado múltiples esfuerzos por mejorar el desempeño de las labores, entre algunos de los más significativos o relacionados con el proyecto que se pretende implementar tenemos el desarrollo de los siguientes trabajos de grado:

- Diseño e Implantación de una Intranet con Servicios de Chat, Fórum de Discusión, Correo Electrónico, Servidor de Archivos y a Nivel de Propuesta los Servicios de Voz y Datos sobre Protocolo de Internet.

  Año: 2002.

- Diseño e Instalación de una Red Informática ARPA en el Edificio de Usos Múltiples de la Facultad Multidisciplinaria de Occidente y su Interconexión con Red Interna.

  Año: 2002

- Diseño de una Red Inalámbrica en la Facultad Multidisciplinaria de Occidente de la Universidad de El Salvador.

  Año: 2005

- Diseño, Desarrollo e Implementación de una Enciclopedia Web para el Departamento de Ingeniería FMO UES.

  Año: 2005

- Propuesta para la Implementación del Laboratorio de Hardware y Elaboración de las Guías Prácticas de las Asignaturas de Especialidad de la Carrera de Ingeniería de Sistemas Informáticos.

  Año: 2006

- Desarrollo de una Aplicación Web, Configuración y Logística para Transmitir Capacitaciones Electrónicas en la Universidad de El Salvador.

  Año: 2006

- Diseño del Sitio Web de Contenidos Universitarios y Sistemas de Control en Línea de Egresados y Graduados del Departamento de Ingeniería y Arquitectura de la F.M.O

  Año: 2007

- Diseño, Desarrollo e Implementación de una Herramienta Colaborativa en Línea para Apoyo a Catedráticos en la Enseñanza Universitaria.

  Año: 2007

- Análisis, Desarrollo e Implementación de un Software Multimedia Interactivo de Apoyo al Aprendizaje de Matemática de las Carreras de Ingeniería de la F.M.O.

  Año: 2007

- Propuesta de Creación e Implantación de una Sección de Servicios de Ingeniería y Arquitectura de la Facultad Multidisciplinaria de Occidente UES.

  Año: 2007

- Diseño, Desarrollo e Implementación de un Sistema para el Registro y Control del Activo Fijo de la F.M.O

  Año: 2008

- Estructuración Documental para el Sistema de Gestión de Calidad de la Escuela de Ingeniería y Arquitectura de la F.M.O Basada en la Norma ISO – 9001 – 2000.

  Año: 2008

- Propuesta para la Creación e Implantación de la Unidad de Recursos Multimedia del Departamento de Ingeniería y Arquitectura de la Facultad Multidisciplinaria de Occidente.

  Año: 2008

Otros Esfuerzos Informáticos Realizados en el Departamento de Ingeniería y Arquitectura.

1. **Infraestructura de Red del Departamento de Ingeniería y Arquitectura.**

   - Año 2004 - Primera fase de instalación de la red.
   - Año 2005 - Segunda Fase de Instalación de la red, adición de puntos de red e instalación de un nuevo router.
   - Finales de 2006 - inicios de 2007 - Actualización final de la red .
   - Inicios de 2008 implementación de Redes Privadas Virtuales (VPN) sobre la arquitectura lógica de la red.

2. Instalación e implementación de herramientas colaborativas como apoyo académico docente – alumno.

- Año 2004 – Prueba piloto con la herramienta colaborativa *Basic Support for Cooperative Working* (BSCW).
- Año 2005 – Prueba piloto con la herramienta Aulita Virtual diseñada por estudiantes de la carrera de Ingeniería en Sistemas Informáticos.
- Año 2007 – Instalación e implementación de la herramienta Moodle. Desactivada a finales de 2007 debido a problemas de DNS.
- Inicios de 2008 – Actualización de Skin del Moodle.

Justificación

Debido a que los tiempos han cambiado, es necesario que las organizaciones vayan a la vanguardia de la tecnología, y siendo los desarrolladores de este proyecto de tesis parte del Departamento de ingeniería y Arquitectura de FMO, específicamente de la carrera de Ingeniería en Sistemas Informáticos tienen la responsables de hacer que esto se cumpla.

Por lo cual, se realizo una investigación de campo en la que se observa que hay una serie de actividades que pueden ser simplificadas y en las cuales los mayores beneficiados son las diferentes entidades que forman parte del Departamento de Ingeniería y Arquitectura.

Mediante el Aplicativo, los usuarios podrán tener una comunicación eficaz y eficiente, mediante la cual se reducirá el tiempo de realización de varias actividades que ahí se realizan.

Alcances

- Diseñar, Desarrollar e implementar un aplicativo que de soporte a las diferentes entidades del Departamento de Ingeniería y Arquitectura.

- Crear un ambiente agradable en el aplicativo para que el usuario no lo perciban engorroso en cuanto a la manipulación, si no que quede invitado a volver a usarlo.

- Mejorar la comunicación entre las diferentes entidades que conforman el Departamento de Ingeniería y Arquitectura, creando grupos por materia para una fácil localización tanto del docente encargado como de compañeros de materia.

- Facilitar el acceso a los documentos y material de apoyo, por parte de todos los usuarios del Aplicativo.

- Actualizar frecuentemente la calendarización de las diferentes actividades en las que participaran: Jefatura, docentes y estudiantes, de una forma ágil.

- Controlar el manejo de activos como equipo audio visual y de biblioteca del Departamento.

- Facilitar un mecanismo que permita la utilización, creación y distribución de los diferentes tipos de documentos utilizados en el departamento, haciendo uso de plantillas u otros formatos pre elaborado.

- Establecer un plan de implementación que permita asegurar un adecuado funcionamiento y uso del sistema.

- Se sugerirán los recursos tecnológicos necesarios para el mejor aprovechamiento del aplicativo.

- Se elaborara el manual de usuario de la herramienta desarrollada.

Metodología de la investigación

Entre las técnicas a utilizar para la recolección de datos, están las siguientes:

- Cuestionarios

Son listas que exponen preguntas y se distribuyen entre las personas que eran parte de la investigación. Para el caso de este estudio estos han sido elaborados con el fin de indagar ampliamente en qué consiste actualmente todos los procesos que cada una de las entidades involucradas realiza.

- Entrevista

Que son el medio de recopilación de información de una forma directa. Se prepararan guías de entrevistas: para las personas encargadas de los siguientes puestos: Jefe del Departamento, Encargados de carrera, docentes, Directores de Trabajos de Grado, Jefe de servicio social, asesores de servicio social y secretaria.

- Revisión Documental

Este consiste en consultas a libros, tesis y cualquier otro material que aporte información útil para el tema a desarrollar.

- Consultas Web

Esta manera de investigación tiene que ver con documentarse por medio de internet en algunas dudas que durante el desarrollo de este trabajo se presenten.

CAPITULO I

GENERALIDADES DEL PROYECTOS

CAPITULO I – GENERALIDADES DEL PROYECTO

1.1 Marco Histórico de la Facultad multidisciplinaria de Occidente y del Departamento de Ingeniería y Arquitectura

1.1.1 Facultad Multidisciplinaria de Occidente

En la ciudad de Santa Ana, la Universidad de El Salvador abrió sus puertas a la población de occidente el 15 de octubre de 1874 y por falta de apoyo desapareció en 1880; transcurrió el tiempo hasta que en 1965 se logro mediante el acuerdo Nº 46 del Honorable Consejo Superior Universitario, la fundación del Centro Universitario de Occidente (16 de julio de 1965).

Para esa época el centro ofrecía los estudios que integran las distintas carreras profesionales de la Universidad de El Salvador; así poseía los mismos departamentos de servicios generales existentes en la Unidad Central de San Salvador; como son: Física y Matemáticas, Ciencias Biológicas y Química; además el Centro contaba con un departamento de Ciencias Sociales, Filosofía y Letras.

En la década de los noventa, específicamente para la fecha del 4 de junio de 1992, en sesión ordinaria del Consejo Superior Universitario, se llegó al acuerdo Nº 39-91-95-IX denominado “Proyecto de Acuerdo del Consejo Superior Universitario, sobre la Creación de las Facultades Regionales Multidisciplinarias”, en el que se establece la Facultad Multidisciplinaria Occidental con sede en el Departamento de Santa Ana, se organizará y funcionará en el terreno e instalaciones que ocupa el Centro Universitario de Occidente.

1.1.2 Departamento de Ingeniería y Arquitectura

En lo que respecta al Departamento de Ingeniería, inicialmente nace atendiendo las especialidades de Física, Matemática y los dos primeros años de las carreras propias de ingeniería, teniendo los alumnos en esa época que trasladarse a San Salvador para culminar sus estudios en la Facultad de Ingeniería y Arquitectura de la misma.

Posteriormente, a finales de la época de los 80, el departamento de Ingeniería se separa de las especialidades de Física y Matemática y comienza atender las asignaturas propias de sus carreras. Ya en la década de los 90 se empieza a valorar la posibilidad de completar algunas de las carreras de Ingeniería, que hasta esa fecha solo atendían los dos primeros años de todas las especialidades.

Cuando se estaba finalizando la carrera de ingeniería civil, y luego de evaluar el éxito en la gestión económica que generaba la recién fundada sociedad de padres, se siguió con la carrera de ingeniería industrial; posteriormente se hizo la gestión para iniciar la carrera de Ingeniería de Sistemas Informáticos.

Ya para el mes de octubre del año 2004 el Consejo Superior Universitario aprobó la creación de la carrera de Arquitectura en este departamento, además se estuvo trabajando para presentar antes de finalizar el año 2006, el proyecto del Diplomado en Informática Educativa. De esta manera el nombre del departamento de ingeniería fue actualizado, por lo que ahora es llamado Departamento de Ingeniería y Arquitectura, teniendo como oferta académica las carreras completas de Ingeniería Civil, Industrial y de Sistemas Informáticos, Arquitectura y los dos primeros años de las carreras de Ingeniería Eléctrica, Mecánica y Química.

El Departamento de Ingeniería y Arquitectura, actualmente imparte lo que son siete carreras académicas, de las cuales solamente tres no se desarrollan completamente, ya que solo existen los primeros dos años, teniendo que continuar hasta su finalización en la sede central de San Salvador (Tabla 1)

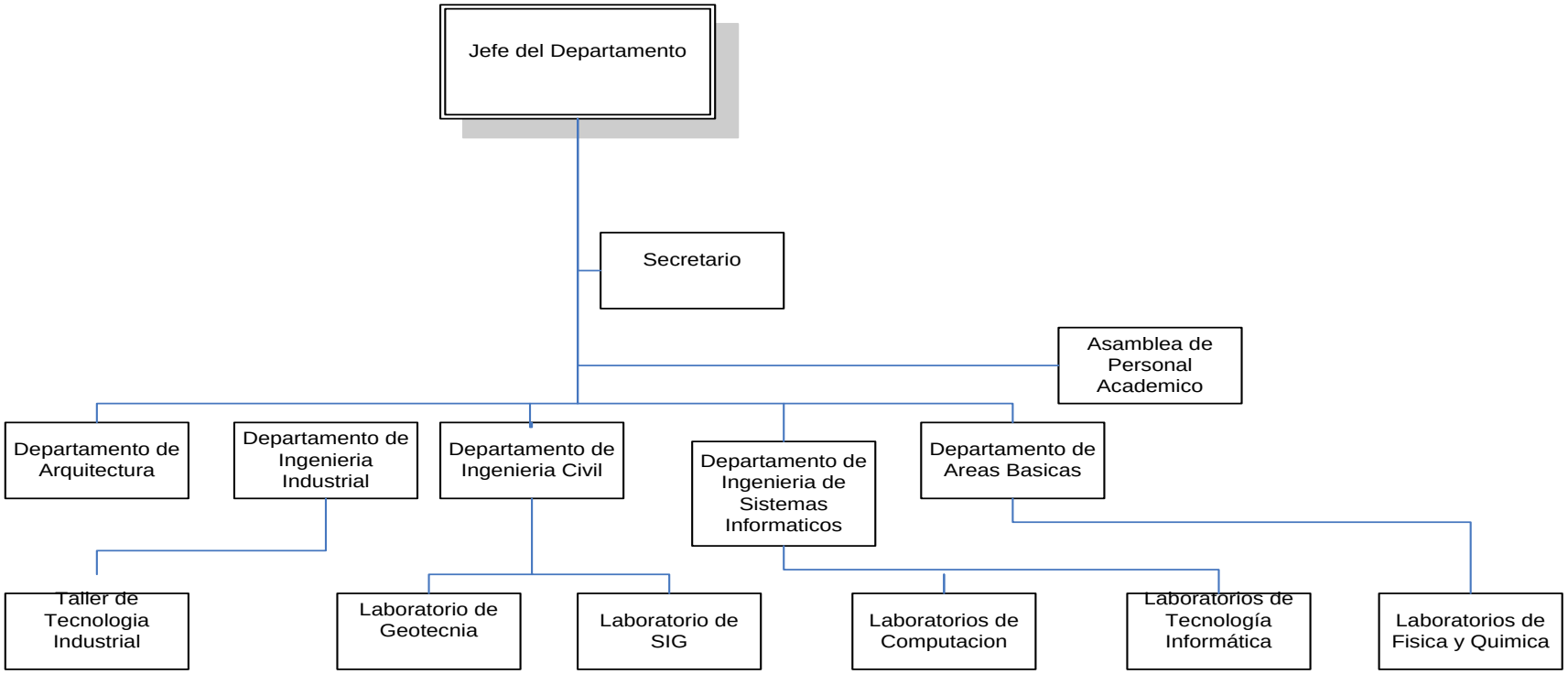
CARRERA	CODIGO	ESTADO
Ingeniería Civil	I30501	Completa
Ingeniería Industrial	I30502	Completa
Ingeniería de Sistemas Informáticos	I30515	Completa
Arquitectura	A30507	Completa
Ingeniería Mecánica	I30504	Incompleta
Ingeniería Eléctrica	I30504	Incompleta
Ingeniería Química	I30506	Incompleta

Tabla 1

1.1.3 Infraestructura del Departamento de Ingeniería y Arquitectura.

En el año 2002 gracias a la celebración de los Juegos Centroamericanos y del Caribe, la Universidad de El Salvador se ve beneficiada, y especialmente la Facultad Multidisciplinaria de Occidente, con la construcción de dos nuevos edificios denominados: Edificio de Usos Múltiples y de ciencias de la Salud Medicina. Actualmente, el Departamento de Ingeniería se encuentra ubicado en el segundo nivel del edificio de usos múltiples, al costado norte del departamento de Idiomas.

1.2 Estructura del Departamento de Ingeniería y Arquitectura



Digrama 1

1.3 Descripción de los servicios que presta el Departamento de Ingeniería y Arquitectura.

Mostrar esta descripción ha sido posible gracias a entrevistas realizadas a las diferentes entidades que lo conforman, las cuales son:

1.3.1 Jefatura del Departamento.

Entre algunos de los principales procesos Administrativos a cargo de la jefatura se pueden mencionar:

- Gestionar con el Decanato la compra de recursos didácticos y equipo para el uso de este.
- Gestionar ante el Decanato las capacitaciones a nivel de docentes.
- Programación y Aprobación de eventos de origen académico.
- Autorización de Permisos e incapacidades.
- Coordinar con Administración Académica:
  - ✓ Establecimiento de Horarios.
  - ✓ Asignación de Aulas.
  - ✓ Validación de Materias.
  - ✓ Tramites como Equivalencias y Cambios de Asignatura.
- Programación y asignación de la Carga Académica.

1.3.2 Secretaría del Departamento.

Las actividades que la Secretaria realiza están orientadas principalmente, a brindar apoyo en la gestión administrativa de la jefatura. Sin embargo, algunas veces también brinda servicios a las demás entidades ya que sirve como nexo de comunicación entre estas y la Jefatura, por ejemplo es ella quien se encarga de notificar la programación de sesiones y reuniones.

Entre algunas de sus actividades se pueden mencionar:

1.3.2.1 Apoyo en la Elaboración de Documentos importantes**.**

- Elaboración, edición y entrega de cartas y memorándums.

- Elaboración y edición de constancias y solicitudes, entre las que podemos mencionar:
  - ✓ Elaboración de constancias de conducta para alumnos.
  - ✓ Elaboración de constancia de tercera matrícula.
  - ✓ Elaboración de solicitud de sexta matrícula.
  - ✓ Elaboración de constancias de retiro de asignaturas.
  - ✓ Elaboración de solicitud para realización de visitas técnicas.
  - ✓ Elaboración de constancia de trabajo de grado inscrito.
  - ✓ Elaboración de constancia de entrega de trabajo de grado en formato electrónico.
- Elaboración de solicitud para mantenimiento de equipo o infraestructura del departamento. Etc.

1.3.2.2 Apoyo en Actividades Diversas.

- Recepción y gestión de solicitudes para préstamo de equipo audiovisual.
- Recepción y registro de inscripciones de trabajos de grado.
- Recepción de ejemplares de trabajos de grado.
- Además de apoyar en las diversas actividades realizadas por la Jefatura del Departamento.

1.3.3 Cuerpo Docente

Entre las principales tareas que desarrollan y gestionan los docentes del Departamento de Ingeniería y Arquitectura durante el desarrollo de su carga académica podemos mencionar:

- Planificación de clases.
- Planificación de Metodologías de Aprendizaje.
- Impartir Clases.
- Brindar consulta a alumnos.
- Preparación y revisión de evaluaciones y trabajos ex aula.
- Elaboración de lineamientos para los instructores en servicio social.

- Registrar notas en el sistema ADACAD.

Además la carga académica de un docente puede incluir otras funciones como:

- Asesoría en proyectos de diseño y tareas varias.
- Apoyo en actividades académicas del Departamento.
- Participación en Actividades Académicas Institucionales.
- Participación en Actividades Académicas en coordinación con alumnos.

1.3.3.1 Relaciones del Cuerpo Docente con la Jefatura del Departamento.

En el desempeño de sus actividades, los docentes gestionan algunas tareas con la Jefatura de departamento, alguna de ellas son:

- Gestionar la Asignación de la Carga Académica.
  - ✓ Gestión de Horarios.
  - ✓ Asignación de Horarios.
  - ✓ Gestionar Cambios de Horario.
- Gestión de aulas.
  - ✓ Asignación de Aulas para la oferta de asignaturas.
  - ✓ Problemas relacionados a aulas.
- Gestión de Permisos.
- Reuniones de Trabajo.
- Capacitaciones de Personal.
- Gestión de asignación de transporte para visitas técnicas o actividades académicas.
- Otras.

1.3.4 Población Estudiantil.

La población estudiantil como es natural en toda organización educativa, se relaciona de forma directa y constante con los docentes que imparten las asignaturas.

Es así, como además de recibir clases, asistencia a evaluaciones parciales, exposiciones, etc. Existen muchas otras actividades que requieren una comunicación efectiva entre docentes y alumnos. Entre algunas de estas actividades podemos mencionar:

- Distribución de documentación y material de apoyo concerniente a la asignatura impartida.
- Comunicación de avisos importantes referentes a programación de actividades, programación de evaluaciones, notas de evaluaciones, visitas técnicas, etc.
- Notificación de cambios en la programación de actividades académicas.
- Dar respuesta a dudas e interrogantes de los estudiantes.
- Formación y coordinación de grupos de trabajo.
- Recepción de tareas y trabajos ex aula.
- Brindar asesoría en la elaboración de proyectos.

1.3.5 Docente Director de Trabajo de Graduación.

Gracias a encuestas a Docentes Directores de Trabajos de Grado del Departamento de Ingeniería y Arquitectura podemos identificar de forma general:

- El proceso que deben seguir los estudiantes en trabajo de grado:

  - ✓ Presentación del Perfil de Trabajo de Grado, esperando la debida revisión y aprobación.
  - ✓ Elaboración del Anteproyecto.
  - ✓ Celebración de reuniones de coordinación y presentación de avances al Docente Director de Trabajos de Grado durante todo el proceso.
  - ✓ Desarrollar el trabajo de grado.
  - ✓ Definir pautas para dar continuidad al trabajo.
  - ✓ Rendir Evaluaciones: 2 privadas y 1 pública. Distribuidas durante todo el proceso de desarrollo del trabajo de grado.

Por lo cual sus actividades se diversifican, ya que no solamente estará a cargo de uno de estos proyectos, si no también le serán asignadas materias.

En lo referente a Trabajos de grado las funciones del docente son las siguientes:

1.3.5.1 Brindar seguimiento al desarrollo de cada uno de los trabajos de grado

- Registrando a través de Bitácoras:
- Hora y Día de Reunión.
- Observaciones y Sugerencias.
- Tareas Pendientes.
- Compromisos.
- Acuerdos, etc.

A través de la elaboración de una calendarización que incluya:

- Reuniones Periódicas.
- Definición de Objetivos.

1.3.5.2 Evaluación del trabajo de grado

El desarrollo del trabajo de grado es evaluado a través de tres presentaciones: dos privadas y una pública, realizadas ante un jurado evaluador, quien escucha y valora estas presentaciones, brinda una revisión del material o documentación presentada, realiza preguntas y observaciones, con el fin de poder brindar la evaluación pertinente.

Apoyándose en formatos que definen criterios de evaluación y su respectiva ponderación, el jurado evaluador puede brindar su valoración final a la presentación del trabajo de grado.

Todos estos trabajos de grado se encuentran disponibles posteriormente en la biblioteca a través de documentos impresos o en formato electrónico en disco compacto.

1.3.6 Docente Coordinador de Proyección Social.

Esta entidad es la encargada de gestionar entre el departamento y Proyección Social, todos los trámites necesarios para que un estudiante obtenga la carta de comprobación de horas sociales. Pero para ello, hay requisitos que tiene que cumplir los cuales son:

- Haber cumplido con el 80% de su carrera.
- Poseer un CUM de 7.00

Luego de comprobar que cumple los requisitos antes mencionados, el estudiante se tiene que someter al siguiente procedimiento:

1.3.6.1 Inscripción del Proyecto de Servicio Social.

Los estudiantes aspirantes a realizar alguno de estos proyectos de servicio social siguen una serie de pasos, para poder inscribir el proyecto. Entre estas tareas se pueden mencionar:

- Verificar la disponibilidad de proyectos o instructoras para realizar el Servicio Social.
- Presentar la propuesta de elaboración del proyecto (anteproyecto) al docente Coordinar de Proyección Social, para su aprobación.
- Llenar la ficha de Inscripción de Servicio Social, adjuntando un record de notas firmado por administración académica.

1.3.6.2 Proceso de Seguimiento y Evaluación del Servicio Social

Durante el desarrollo del proyecto de servicio social, cada alumno debe entregar mensualmente un reporte que permite dar seguimiento a la labor realizada por éste. Este documento refleja la descripción de todas las actividades orientadas a cumplir con el desarrollo del proyecto. El registro de cada una de estas actividades debe contar con la aprobación del Docente Director del Proyecto. Este documento se comienzan a entregar 30 días después de haber inscrito el proyecto de servicio social.

1.4 Mecanismos de Comunicación entre Entidades

Actualmente el Departamento de Ingeniería y Arquitectura dispone de mecanismos de comunicación sumamente diversos, no unificados ni estandarizados. Se utiliza comunicación directa, llamadas telefónicas, mensajes de texto, e-mail, publicaciones Web, memorándums etc

1.4.1 Comunicación Jefatura – Cuerpo Docente.

La jefatura del departamento, mandos intermedios, cuerpo docente en general, incluidos los diferentes cargos y comisiones, son responsables del enlace y correcta transmisión de mensajes en ambos sentidos.

Existen múltiples mecanismos de comunicación utilizados por estas entidades entre las que podemos mencionar:

- Boletines.
- Memorándums.
- Pizarrón Informativo.
- Juntas.
- Reuniones.
- Etc.

Es importantes destacar que la secretaria del departamento actualmente juega un papel de suma importancia en la comunicación interna del departamento, como medio de comunicación entre entidades y se encarga de comunicar avisos y disposiciones importantes de la Jefatura, notifica la programación de algunas sesiones y reuniones, entrega memorándums, registra acuses de recibido, etc.

1.4.2 Comunicación Docente – Estudiante

Actualmente existe un esquema de comunicación principalmente unidireccional entre los docentes y sus alumnos, de tal forma que los docentes son emisores de avisos, disposiciones y mensajes, los alumnos se convierten en receptores de estos avisos a través de diversos medios no unificados de comunicación, entre algunos de estos mecanismos podemos mencionar:

- Boletines.
- Pizarrón Informativo.
- Correos Electrónicos.
- Páginas Web.

Sin embargo en la mayor parte de los casos cumple un papel muy importante el alumno representante de asignatura, quien mantiene una comunicación directa con el docente a través de:

- Comunicación frente a frente.
- Correos electrónicos.
- Mensajes de Texto.
- Llamadas telefónicas.

De tal forma que el representante de asignatura es el encargado de notificar al alumnado en general avisos importantes, notificaciones y disposiciones, cambios en programaciones recientes. Sin embargo actualmente no existen mecanismos establecidos y unificados de comunicación entre estudiantes incluso de la misma asignatura.

1.5 Misión y Visión de la Facultad Multidisciplinaria de Occidente.

1.5.1 Misión

Institución en nuestro país eminentemente académica, rectora de la educación superior, formadora de profesionales con valores éticos firmes, garante del desarrollo, de la ciencia, el arte, la cultura y el deporte. Crítica de la

realidad, con capacidad de proponer soluciones a los problemas nacionales a través de la investigación filosófica, científica artística y tecnológica; de carácter universal.

1.5.2 Visión

Ser una universidad transformadora de la educación superior y desempeñar un papel protagónico relevante, en la transformación de la conciencia crítica y prepositiva de la sociedad salvadoreña, con liderazgo en la innovación educativa y excelencia académica, a través de la integración de las funciones básicas de la universidad: la docencia la investigación y la proyección social.

CAPITULO II

ANALISIS PRELIMINAR

CAPITULO II - ANALISIS PRELIMINAR

2.1 Situación Actual de Departamento de Ingeniería y Arquitectura y Planteamiento del Problema.

2.1.1 Hardware

En lo que respecta al hardware como base de la plataforma tecnológica, se ha realizado un estudio para determinar el tipo y las características que este tiene. Gracias a las respectivas visitas que se realizaron en el Departamento se encontró el siguiente equipo distribuido en un área específica.

- Servicio de Internet de la Facultad Multidisciplinaria de Occidente.

Está constituida por un solo router el cual direcciona el tráfico proveniente y saliente hacia internet, encaminando sus paquetes a la red central ubicada en San Salvador por medio de un receptor de fibra óptica con un enlace dedicado ATM de 4 Mbps y cuyo proveedor es TELECOM, la cual se encarga de toda la comunicación entre facultades y red externa.

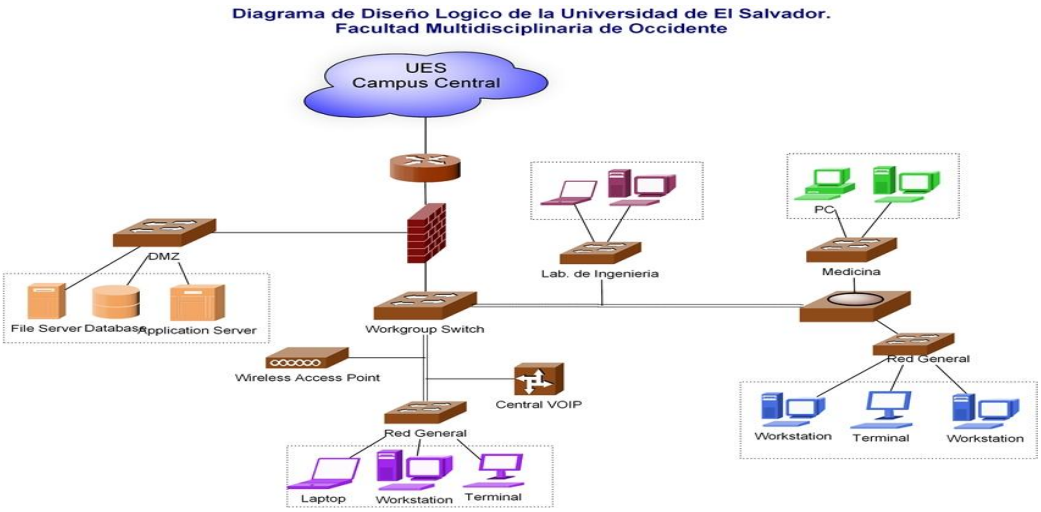


Diagrama 2

- ✓ Además posee un firewall/PIX de marca CISCO bidireccional propiedad de TELECOM, el cual se ocupa de la seguridad de la red local, evitando así cualquier contenido peligroso a través de ciertos protocolos a través de reglas.

- ✓ Existe un CONTENTFILTER que trabaja de tal manera que la información sale a través de una aplicación dentro del servidor. Bloqueando así todas la URL y está sujeto a un PROXY.

- ✓ Luego se tiene la DMZ (Zona Desmilitarizada) dentro de la cual se encuentran todos los servidores, cuya cantidad es de 15 divididos en 13 servidores Linux y 2 servidores Windows los cuales proveen servicios a Ingeniería y a Biblioteca.

- ✓ Además provee red inalámbrica la antena omnidireccional se encuentra ubicada en el edificio de usos múltiples, dicha disposición fue elegida debido a un mayor alcance. Físicamente está formada por 2 Access Point y están conectados como clientes LAN recibiendo una IP Dinámica.

- ✓ VOIP es otro servicio que es suministrado a la red a través de un enlace dedicado E1 que posee una capacidad de ancho de banda de 2.048 Mbps y provee de acceso telefónico a toda la facultad.

- ✓ La red se comunica por medio de enlaces de fibra óptica hacia el edificio de medicina, académica y el edifico de agua.

- ✓ Por último la distribución de la red se hace por medio de una serie de swichtes los cuales conmutan los paquetes hacia los host destino y permiten la comunicación en la red local.

Por otra parte, el diseño Físico está constituido de la siguiente manera:

- ✓ La Distribución física de la red de la Facultad Multidisciplinaria de Occidente de la Universidad de El Salvador, está distribuida desde el edificio de usos múltiples conectando por medio de fibra óptica a los edificios de medicina, de agua y académica.

- ✓ Desde Académica se interconectan por medio de cable de par trenzado UTP categoría 5 los Departamento de Biología, el Centro de Cómputo de Economía, Colecturía, Proyección Social, Unidad Socioeconómica y Unidad de Proyección Literaria (Letras); además de interconectar por medio de cable coaxial al Departamento de Economía y Unidad de Maestrías.

- ✓ Como punto de origen, la unidad de maestrías se distribuye por medio de cable de par trenzado UTP categoría 5 a la unidad Planes Especiales (PAES) y al Departamento de Física.

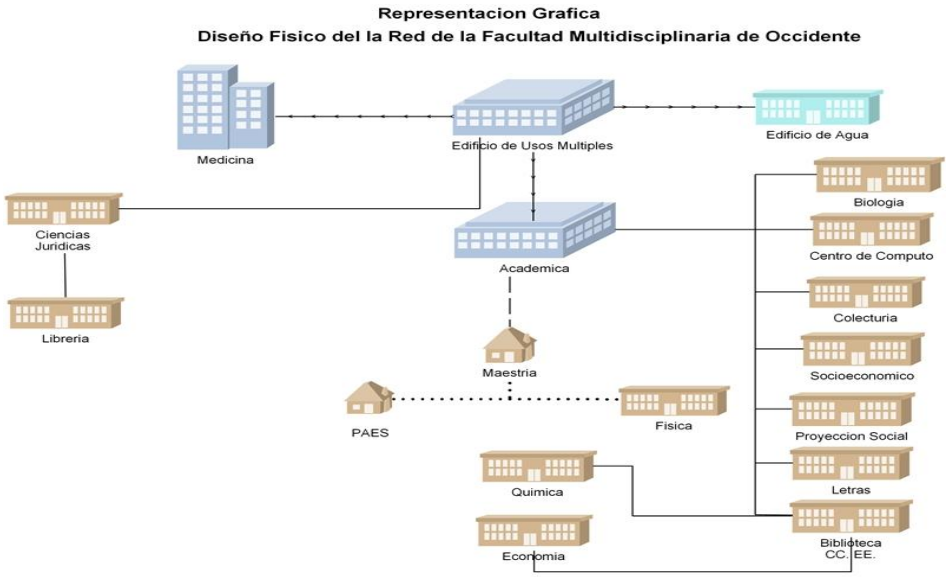


Diagrama 3

- ♦ Infraestructura de Red del Departamento de Ingeniería y Arquitectura.

El departamento actualmente cuenta con un switch principal, ver diagrama 4, éste da salida a la red tanto de la universidad como a la de internet. Así mismo, se encuentran otros dos switches con menor capacidad pero que fueron necesarios debido aumento de PCs en la red de este lugar. Para los usuarios que se conectan mediante el medio inalámbrico, está configurado un Access point propio del área. Toda esta infraestructura y Dispositivos de red son de los se harán uso para la implementación de la aplicación para gestión administrativa.

Diagrama de Red del Departamento de Ingeniería y Arquitectura.

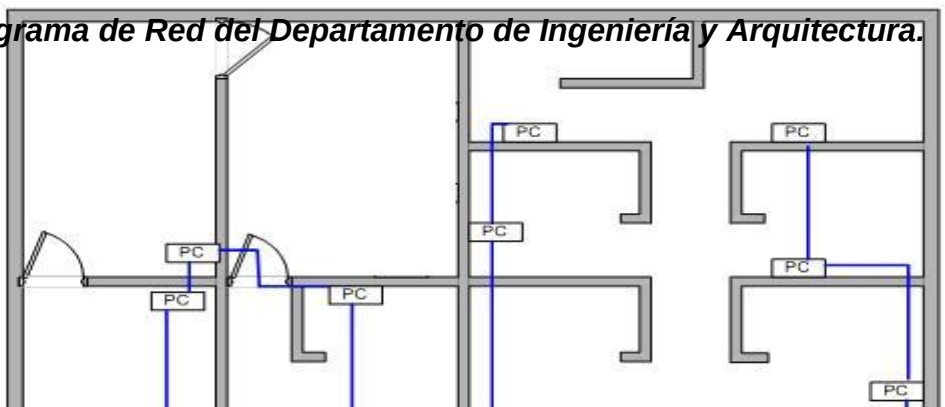


Diagrama 4

2.1.2 Software

Descripción con Respecto a Hardware	Sistema Operativo	Posee punto de Red
Motherboard: Intel Pentium 4 **Procesador : 2.8 GHz** **RAM : 512 Mb** **Disco Duro: 80 Gb**	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	**Si**

Tarjeta/Red: Si (una)		
Motherboard: Intel Pentium 4 **Procesador : 1.8 GHz** **RAM : 512 Mb** **Disco Duro: 80 Gb** **Tarjeta/Red: Si (una)**	**S. O. : XP Profesional Service Pack 2** **Navegador: Explorer**	**Si**

Descripción de Equipo Informático del Departamento de Ingeniería y Arquitectura.

Motherboard: Intel Celeron Procesador : 1.8 GHz RAM : 256Mb Disco Duro: 40 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	Si
Motherboard: Intel Celeron Procesador : 1.8 GHz RAM : 256Mb Disco Duro: 40 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	Si
Motherboard: Intel Pentium 4 Procesador : 2.4 GHz RAM : 256Mb Disco Duro: 10 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	Si

Motherboard: Intel Pentium 4 Procesador : 1.8 GHz RAM : 128Mb Disco Duro: 60 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	No Switch propio
Motherboard: Intel Procesador : 200 MHz RAM : 64 Mb Disco Duro: 1 Gb Tarjeta/Red: Si (una)	S. O. : Windows 98 Navegador: Explorer y Mozilla	Si
Motherboard: Intel Pentium 4 Procesador : 2.4 GHz RAM : 256Mb Disco Duro: 40 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	Si
Motherboard: Intel Procesador : Celeron 2.4 GHz RAM : 256Mb Disco Duro: 40 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	Si
Motherboard: Intel Procesador : Celeron 1.8 GHz RAM : 256Mb Disco Duro: 40 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	Si
Motherboard: Intel Pentium 3 Procesador : 500 MHz RAM : 128Mb Disco Duro: 40 Gb Tarjeta/Red: Si (una)	S. O. : Windows 98 Navegador: Explorer	Si

Motherboard: Intel Pentium 3 Procesador : 500 MHz RAM : 128Mb Disco Duro: 40 Gb Tarjeta/Red: Si (una)	S. O. : Windows 98 Navegador: Explorer	Si
Motherboard: Pentium 4 Intel Procesador : 2.6 GHz RAM : 256Mb Disco Duro: 40 TB Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	No Switch propio
Motherboard: Intel Pentium 4 Procesador : 1.8 MHz RAM : 128Mb Disco Duro: 20 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Antivirus: Norton Navegador: Explorer	Si
Motherboard: Intel Pentium 3 Procesador : 800 MHz RAM : 128Mb Disco Duro: 40 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	Si
Motherboard: Intel Procesador : Pentium 3 750 MHz RAM : 128Mb Disco Duro: 20 Gb Tarjeta/Red: Si (una)	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	Si
Motherboard:AMD athlon xp 2100 Procesador : 2.1 GHz RAM : 128 Mb	S. O. : XP Profesional Service Pack 2 Navegador: Explorer	Si

Disco Duro: 20 TB Tarjeta/Red: Si (una)		

Tabla 2

2.2 Planteamiento del Problema.

Gracias a cuestionarios e investigaciones realizadas sobre los diferentes procesos que realizan cada una de las entidades del Departamento de Ingeniería y Arquitectura, se pueden identificar varios aspectos que no favorecen y muchas veces, dificultan el trabajo que realiza el personal.

2.2.1 Falta de Mecanismos Eficientes de Comunicación.

Quizás uno de los puntos primordiales identificado, es la clara falta de mecanismos adecuados de comunicación dentro del departamento, que permitan una comunicación eficiente y efectiva entre todas y cada una de las entidades del mismo.

Esto suele limitar el flujo necesario de información, lo que conlleva a muchas otras problemáticas y dificultades para el desarrollo de las actividades cotidianas. Impidiendo la adecuada integración y consenso de actividades y dejando casi inexistente la posibilidad de un trabajo colaborativo y de apoyo.

A pesar de las múltiples soluciones informáticas existentes en la actualidad, estas no satisfacen las necesidades de comunicación propias y particulares del Departamento. Ya que estas no ofrecen facilidades para el trabajo colaborativo, apoyo en actividades, fácil traslado de cualquier tipo de información. Lo cual trae efectos negativos como son:

- Recepción de información incompleta o retraso en la recepción de la misma.
- Asignación de funciones poco definidas.
- Malos entendidos.
- Desconocimiento.
- Etc.

2.2.2 Falta de una Estandarización de Procesos Académicos y de Gestión.

A pesar de que existen estudios realizados que buscan identificar, unificar y estandarizar las tareas realizadas por cada uno de las entidades, tanto de aspecto académicas como de gestión. Aun existen muchas actividades en las cuales no hay un consenso del proceso detallado que se debe seguir para su cumplimiento.

Esta falta de uniformidad de los procesos impide que herramientas convencionales puedan servir de apoyo a las funciones académicas y de gestión que realizan las entidades del Departamento de Ingeniería y Arquitectura. Se necesitan herramientas que permitan cierto grado de flexibilidad para brindar un verdadero apoyo en estas tareas, sin perder de vista los objetivos bien definidos que posee cada entidad en el rol que desempeña.

2.2.3 Dificultad y Problemáticas que presentan Entidades del Departamento en el desarrollo de sus labores.

Además de estas dificultades y problemáticas que atañen a las entidades del departamento en general, existen otras que son propias de las entidades estudiadas, que están directamente relacionadas a la labor que desempeñan en sus diversas funciones, las cuales pudieron ser identificadas gracias al estudio realizado. Entre algunas de las más importantes se pueden mencionar:

1. Jefatura del Departamento.

   - Inexistencia de mecanismos que faciliten la programación y calendarización de actividades y eventos. Así como su fácil distribución.
   - Dificultad para notificar convocatorias, disposiciones, avisos importantes de manera rápida y eficiente, garantizando su correcta recepción.

2. Secretaría del Departamento.

   - Debido a la inexistencia de mecanismos de comunicación eficientes entre entidades del departamento principalmente entre la jefatura del departamento y las demás entidades. La secretaria del departamento debe

encargarse de comunicar avisos importantes, entregar memorándums, convocatoria a juntas, etc. solicitando acuses de recibido.

- Necesidad de elaborar, de forma repetitiva, documentos como avisos, memorándums, constancias, solicitudes sin la facilidad para reutilizar formatos de manera eficiente que le permitan reducir esfuerzo y tiempo en su elaboración.
- Falta de herramientas adecuadas que le ayuden en su labor de apoyo a las actividades de gestión administrativa del departamento, como por ejemplo: En la gestión del préstamo de equipo audio visual y equipo para prácticas.

3. Población Docente.

- Falta de herramientas adecuadas que faciliten la planificación de la carga académica y ficha didáctica del docente. Elaboración y registro de temarios, programación y calendarización de clases, evaluaciones, trabajos, prácticas, etc. así como la distribución adecuada de esta información.
- Dificultad para distribuir material de apoyo, folletos, guías de laboratorio, diapositivas, etc. a los estudiantes inscritos en la asignatura a su cargo.
- Carencia de mecanismos que faciliten la comunicación práctica, ágil y eficiente con los alumnos para la notificación de avisos importantes, cambios en la programación de evaluaciones, lineamientos para desarrollo de proyectos y trabajos de la asignatura, definición de equipos de trabajo, etc.
- Inexistencia de mecanismos que faciliten la realización de trámites académicos y administrativos con la jefatura del departamento, así como cualquier otra tarea que implique la comunicación con esta entidad.

4. Cuerpo Estudiantil.

- Poca efectividad por parte de las herramientas de comunicación actuales docente – alumno, como: pizarra informativa, avisos en carteles, llamadas telefónicas, mensajes de texto, correos electrónicos, incluso páginas Web; para establecer un modelo de comunicación bidireccional, práctico, dinámico y efectivo.

Falta de una herramienta unificada que brinden a los estudiantes la capacidad de comunicarse de forma directa con otros estudiantes compartiendo documentación, conformando equipos de trabajo, estudio, etc.; aspectos que actualmente no son contemplados

5. Docente Director de Trabajos de Graduación.

- Carencia de herramientas y mecanismos que faciliten dar seguimiento al desarrollo de cada uno de los trabajos de grado; registro de avances, programación de reuniones y evaluaciones, elaboración y control de bitácoras de trabajo, etc. Lo que permitiría facilitar la evaluación de los trabajos de grado.
- Inexistencia de un registro histórico de los trabajos de grado realizados con un seguimiento de tallado de los mismos.
- Inexistencia de mecanismos que faciliten la comunicación adecuada entre el docente director y los alumnos en proceso de grado. Que faciliten la coordinación de actividades, consultas, revisiones, concertación de reuniones, etc.

6. Docente Coordinador de Proyección Social.

- Desconocimiento, por parte de alumnos aspirantes a realizar su servicio social, sobre los procesos, documentación, formatos y actividades necesarias para el cumplimiento satisfactorio de su servicio social.
- Inexistencia de mecanismos que apoyen o faciliten llevar un seguimiento adecuado del servicio social realizado por cada alumno. Desde todas las perspectivas, en la elaboración y entrega de documentación por parte del alumno; así como en la recepción, registro, control y revisión por parte del Docente Director del Servicio Social y el Docente Coordinador de Proyección Social.

2.3 Planteamiento de la solución.

Luego de analizar los servicios que presta el Departamento de Ingeniería y Arquitectura, la infraestructura y las deficiencias que cada una de las entidades de este presenta, se diseñará un aplicativo el cual logre eliminar varias de estas debilidades.

Para lograrlo, se elaborará un Aplicativo orientado a trabajar en red con el nombre de Sistema de Comunicación Integral Punto a Punto (SCIP2P), en el cual se permitirá a todas las entidades que conforman el Departamento una comunicación personalizada con quienes así lo requiera, por ejemplo: Dentro del aplicativo en general abra una sección que permitan crear grupos de trabajo, para que de esta manera la comunicación con los integrantes a este sea oportuna y eficaz. Así también se mostraran todos los usuarios que hagan

uso del Sistema de comunicación Integral P2P (SCIP2P) para que estén a la disposición de entablar comunicación, siempre y cuando ellos acepten la invitación. También, se creará una manera personalizada de entablar comunicación con otros usuarios mediante grupos personales. Este mecanismo de comunicación funcionará tanto en tiempo real (para efectos de chat) como también cuando el usuario con quien se quiere notificar alguna nota no esté conectado, por medio de un control de notificaciones.

Con el fin de eliminar la debilidad, en lo que respecta a la necesidad de documentos u otro tipo de archivos que no están a disposición todo el tiempo, se creará un aplicativo en el cual al instalarse se cree una carpeta compartida, en la que el usuario podrá tener guardados archivos y estos a su vez estarán a disposición de todos los usuarios del SCIP2P que estén conectados a la misma red, los tenga agregados o no en su listas de contacto o en sus grupos. El usuario que necesite un archivo ingresara al aplicativo de transferencia y ahí podrá escribir el nombre del archivo que busca así como también podrá verificar los archivos que tenga guardados en su propia carpeta compartida.

Para contribuir con la secretaría del departamento en la elaboración repetitiva de formas para los diferentes tramites que ahí se realizan, se elaborará en el mismo SCIP2P un aplicativo que permita elaborar formatos haciendo uso de un gestor de documentos este poseerá un formato propio del aplicativo, para que en casos futuros la secretaría solo tenga que rellenar las formas con los datos cambiantes y luego poderlos mandar a imprimir o enviársela al usuario que la necesite. Estos con un envió personalizado (no en la carpeta compartida).

Con respecto a la problemática de los trabajos de Grado y de Proyectos Sociales se creará un aplicativo en el que los responsables de estas áreas puedan ir archivando todos los documentos que cada alumno tiene que ir entregando para estos trámites para que de esta manera puedan llevar una bitácora de su trabajo, así también los alumnos que comenzaran con estos trámites pueda tener una guía de lo que se tiene que hacer en todo el proceso.

Con el fin de eliminar el uso de papel y búsquedas tediosas se sistematizara el proceso de préstamos de libros en la biblioteca del departamento y del equipo de Multimedia, mediante este aplicativo los usuarios podrán ingresar todos los libros, tesis, CDs, (estos en el caso de la biblioteca) cañones, Laptops, baterías (en el caso del equipo multimedia) e identificarlos por un código especifico para así poder llevar un inventario de estos y con solo poner en cualquiera de estos dos sistemas el nombre de lo que se busca saber si se tiene a disposición o no. Y en caso que no esté en ese mismo momento se pueda hacer reservaciones para días futuros.

Así con todo este sistema integrado se tendrá distintos tipos de usuario para lo cual se crearan distintos tipos de instaladores, ya que como se puede notar no todos hacen uso de los mismos aplicativos. Los tipos de usuario que se tendrán son:

- Jefatura
- Secretaria
- Docentes
- Docentes Directores de Trabajo de Grado.
- Docentes Encargados Proyectos Sociales
- Estudiantes Encargados de Biblioteca
- Estudiantes

Debido a lo amplio de este sistema y su facilidad para ampliarlo se tendrá la opción de instalar nuevas versiones si este es modificado, esto siempre en red, así de esta manera no se tendrá la necesidad de ir a cada PC a desinstalar el que ya se tiene y volver a instalar la nueva versión.

CAPITULO III

ANALISIS DE REQUERIMIENTOS DEL PROYECTO

Capítulo III.- Análisis de Requerimientos del Proyecto

3.1 Definición de requerimientos

3.1.1 Requerimientos de Información

Esta fase consta de dos partes, requerimientos de información que se necesita conocer acerca de los procesos que actualmente se llevan a cabo en el departamento, que son los expuestos en la parte anterior de este documento el cual es la situación actual, como también la parte de requerimientos de información que se ha tenido que investigar para el desarrollo e implementación de este proyecto.

3.1.1.1 Información de Investigación de Campo

En la primera Fase se ha necesitado recurrir a entrevistas frente a frente con las personas involucradas en los procesos como también entrevistas hechas a papel *(aquí se mandara a anexos a ver las entrevistas*). Para esta fase también se ha recurrido a documentación preexistente en la biblioteca del departamento de Ingeniería y Arquitectura en la que se describen estos procesos.

3.1.1.2 Información Técnica

Luego en la segunda fase de la información que ha sido necesaria para la puesta en marcha de este proyecto es toda la documentación bibliográfica a la que se ha tenido que recurrir para ampliar los conocimientos acerca de los conceptos que se utilizaron en la elaboración de este sistema. Esta ha sido tanto libros como documentos en Internet.

En general, los conceptos los cuales se han estudiado son los que se presentan a continuación:

3.1.1.2.1 Programación Orientada a Objetos

La programación orientada a objetos OOP, expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener, y reutilizar.

- Conceptos Fundamentales

  - ✓ **Objeto**: Es un conjunto de funciones y procedimientos, todos ellos relacionados con un determinado concepto estos a su vez son

entidades que combinan *estado (atributo)*, *comportamiento (método)* e *identidad*.

- **El *estado*** está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- **El *comportamiento*** está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.
- **La *identidad*** es una propiedad de un objeto que lo diferencia del resto, en otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

✓ **Clase**: Definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

✓ **Herencia:** (por ejemplo, herencia de la clase D a la clase C) Es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables públicas declaradas en C. Los componentes registrados como "privados" (private) también se heredan, pero como no pertenecen a la clase, se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Esto es así para mantener hegemónico el ideal de OOP.

✓ **Método**: Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las

propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

- ✓ **Evento**: Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento, a la reacción que puede desencadenar un objeto, es decir la acción que genera.

- ✓ **Mensaje**: una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

- ✓ **Propiedad o atributo**: contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

- ✓ **Estado interno**: es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.

- ✓ **Componentes de un objeto**: atributos, identidad, relaciones y métodos.

- ✓ **Identificación de un objeto**: un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

En comparación con un lenguaje imperativo, una "variable", no es más que un contenedor interno del atributo del objeto o de un estado interno, así como la "función" es un procedimiento interno del método del objeto.

- Características

Las más importantes son las que se presentan a continuación:

- ✓ **Abstracción**: Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar *cómo* se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción.

- ✓ **Encapsulamiento**: Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema.
- ✓ **Principio de ocultación**: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una *interfaz* a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del

objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

- ✓ **Polimorfismo**: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama *asignación tardía* o *asignación dinámica*.

- ✓ **Herencia**: las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en *clases* y estas en *árboles* o *enrejados* que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay *herencia múltiple*.

- ✓ **Recolección de basura**: la Recolección de basura o Garbage Collector es la técnica por la cual el ambiente de Objetos se encarga de destruir automáticamente, y por tanto desasignar de la memoria, los Objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación

o liberación de memoria, ya que el entorno la asignará al crear un nuevo Objeto y la liberará cuando nadie lo esté usando. En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de Programación Orientada a Objetos como C++ u Object Pascal, esta característica no existe y la memoria debe desasignarse manualmente.

3.1.1.2.2 SQLite

Sistema de gestión de base de datos relacional que está contenida en una relativamente pequeña (225 KB) biblioteca en C.

A diferencia de los sistemas de gestión de base de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

SQLite usa un sistema de tipos inusual. En lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales. Por ejemplo, se puede insertar un *string* en una columna de tipo entero (a pesar de que SQLite tratará en primera instancia de convertir la cadena en un entero). Algunos usuarios consideran esto como una innovación que hace que la base de datos sea mucho más útil, sobre todo al ser utilizada desde un lenguaje de encriptamiento de tipos dinámicos. Otros usuarios lo ven como un gran inconveniente, ya que la técnica no es portable a otras bases de datos SQL. SQLite no trataba de transformar los datos al tipo de la columna hasta la versión 3.

Varios procesos o hilos pueden acceder a la misma base de datos sin problemas. Varios accesos de lectura pueden ser servidos en paralelo. Un acceso de escritura sólo puede ser servido si no se está sirviendo ningún otro acceso concurrentemente. En caso contrario, el acceso de escritura falla devolviendo un código de error (o puede automáticamente reintentarse hasta que expira un timeout configurable). Esta situación de acceso concurrente podría cambiar cuando se está trabajando con tablas temporales.

3.1.1.2.3 ADO.net

Este concepto es una evolución del modelo de acceso a datos de ADO que controla directamente los requisitos del usuario para programar aplicaciones escalables.

Utiliza algunos objetos ADO, como **Connection** y **Command**, y también agrega objetos nuevos como son: **DataSet**, **DataReader** y**DataAdapter**.

La diferencia más importante entre esta fase evolucionada y la arquitectura de datos anteriores es el objeto **DataSet,** el cual funciona como una entidad independiente, ya que se considera como un conjunto de registros que siempre están desconectados y por lo tanto no sabe nada sobre el origen y el destino de los datos que contiene. Dentro de este objeto al igual que en una Base de Datos hay tablas, columnas, relaciones, restricciones, vistas, etc.
El **objeto DataAdapter** administra objetos **Command** y **DataReade** y simplifica el proceso de sincronización, ya que transfiere datos entre una base de datos y DataSet.

El **objeto DataReader,** proporciona una secuencia de datos sin búfer que permite a la lógica de los procedimientos procesar eficazmente y de forma secuencial los resultados procedentes de un origen de datos. DataReader es la mejor opción cuando se trata de recuperar grandes cantidades de datos, ya que éstos no se almacenan en la memoria caché.

Un objeto **DataTable** representa una sola tabla en la base de datos. Tiene un nombre, filas, y columnas.

Un objeto **DataView** , este trabajo sobre un DataTable y ordena los datos (como una cláusula "order by" de SQL). Filtra los registros (como una cláusula "where" del SQL). Para facilitar estas operaciones se usa un índice en memoria. Todas las DataTables tienen un filtro por defecto, mientras que pueden ser definidos cualquier número de DataViews adicionales, reduciendo la interacción con la base de datos subyacente y mejorando así el desempeño.

Un **DataColumn** representa una columna de la tabla, incluyendo su nombre y tipo.

Un objeto **DataRow** representa una sola fila en la tabla, y permite leer y actualizar los valores en esa fila, así como la recuperación de cualquier fila que esté relacionada con ella a través de una relación de clave primaria.

Un **DataRowView** representa una sola fila de un DataView, la diferencia entre un DataRow y el DataRowView es importante cuando se está interactuando sobre un resultset.

Un **DataRelation** es una relación entre las tablas, tales como una relación de clave primaria - clave ajena. Esto es útil para permitir la funcionalidad del DataRow de recuperar filas relacionadas.

Un **Constraint** describe una propiedad de la base de datos que se debe cumplir, como que los valores en una columna de clave primaria deben ser únicos. A medida que los datos son modificados cualquier violación que se presente causará excepciones.

3.1.1.2.4 Protocolos de Comunicación

Los protocolos de comunicaciones definen las reglas para la transmisión y recepción de la información entre los nodos de la red, de modo que para que dos nodos se puedan comunicar entre si es necesario que ambos empleen la misma configuración.

- **Funciones básica de un protocolo**
  - ✓ **Control de Llamada,**

Establecimiento de conexión entre origen y destino, esta función lleva a cabo el mantenimiento y monitoreo de la conexión y los procedimientos de conexión y desconexión de una llamada, transferencia de datos, videoconferencia, etc.

- ✓ **Control de Errores,**

Verificación y control de errores durante la transmisión a través de algoritmos de verificación y control de error tales VRC, LRC, Checksum, CRC, etc.

- ✓ **Control de Flujo,**

  - Manejo de contención de bloques
  - Regulación del tráfico
  - Retransmisión de bloques
  - Convenciones para direccionamiento }
  - Control por pasos y de extremo a extremo (el error puede verificarse en cada paso o al final del enlace depende del algoritmo de control de error).

- ♦ **Protocolos Orientados a Conexión y No Conexión**

Los orientados a conexión, las entidades correspondientes mantienen las información del **estatus** acerca del **dialogo** que están manteniendo. Esta información del estado de la conexión soporta control de error, secuencia y control de flujo entre las correspondientes entidades. Es decir, La entidad receptora le avisa a la entidad transmisora si la información útil llego correctamente, si no es así también le avisa que vuelva a retransmitir.

El control de error se refiere a una combinación de detección de error (y corrección) y reconocimiento (acknowledgment). El control de secuencia se

refiere a la habilidad de cada entidad para reconstruir una serie de mensajes recibidos en el orden apropiado. El control de flujo se refiere a la habilidad para que ambas partes en un dialogo eviten el sobre flujo de mensajes entre sí.

L**os protocolos orientados a conexión operan en tres fases:**

- ✓ **Configuración de la conexión**, durante la cual las entidades correspondientes establecen la conexión y negocian los parámetros que definen la conexión.
- ✓ **Fase de transferencia de datos**, durante la cual las entidades correspondientes intercambian mensajes (información útil) bajo el amparo de la conexión.
- ✓ **Fase de liberación de la conexión**, en la cual ambas entidades se ponen de acuerdo para terminar la conexión.

Los protocolos orientados a no-conexión, difieren bastante a los orientados a conexión, ya que estos (los de no-conexión) no proveen capacidad de control de error, secuencia y control de flujo**. Los protocolos orientados a no-conexión, están siempre en la fase de transferencia de datos,** y no les interesa las fases restantes de configuración y liberación de una conexión.

Los protocolos orientados a no-conexión se emplean en aplicaciones donde no se requiera mucha precisión. Tal es el caso de la voz, música o el video. Pero en cambio en aplicaciones donde se requiera mucha precisión, ejemplo:. Transacciones electrónicas bancarias, archivos de datos, comercio electrónico, etc. se utilizarían los protocolos orientados a conexión.

3.1.1.2.5 Sockets

Un socket es una abstracción a través de la cual una aplicación puede enviar y recibir datos. En muchos, de la misma forma en la que se abre un archivo permite a una aplicación leer y escribir datos en un almacenamiento estable.

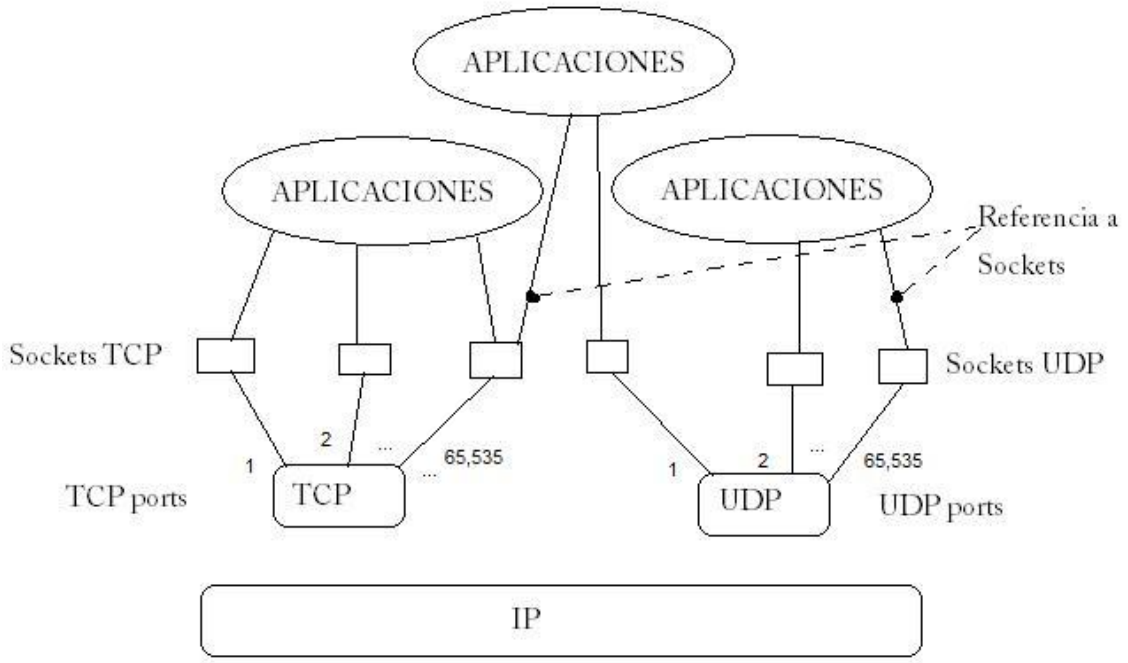
Un socket permite a una aplicación conectarse a la red y comunicarse con otras aplicaciones que también están conectadas a la misma red. La información escrita por el socket por una aplicación en una maquina puede ser leída por una aplicación en una maquina diferente y viceversa.

Los principales tipos de sockets en TCP/IP son los *stream sockets* y los *datagrama sockets.*

Los Socket stream utilizan TCP como el protocolo punto a punto (con la IP por debajo) y de ahí que provee un fiable y seguro servicio de flujo de bytes.

Los sockets datagrama usan UDP (de nuevo un punto a punto con una ip por debajo) y así proveen una buena gestión de servicio de datagramas que las aplicaciones pueden usar para mandar mensajes individuales no mayores a 65,500 bytes (equivalente a 64 kb) de tamaño.

Los sockets stream y datagramas son también soportados por otras suites de protocolos. Pero en este caso trataremos con sockets TCP (stream) y UDP (datagram). Un TCP/IP socket es identificado únicamente por una dirección de internet, un protocolo punto a punto (TCP o UDP) y un número de puerto.



Sockets, Protocolos y Puertos

Diagrama 5

En el diagrama 5 se dibujan las relaciones lógicas entre aplicaciones, abstracciones de sockets, protocolos y números de puerto dentro de un único host, se puede notar que una única abstracción de socket puede estar referida por múltiples programas de aplicación. Cada programa que posee una referencia (llamada descriptor) a un socket en particular puede comunicarse puede comunicarse a través de ese socket.

Un puerto identifica una aplicación en un host. Actualmente un puerto puede identificar a un socket en un host. En la figura se muestra como los múltiples programas en un host pueden accesar al mismo socket. En la práctica, programas separados que accesan al mismo socket podrían pertenecer usualmente a la misma aplicación (por ejemplo múltiples copias de un programa de web server), aunque en un principio podrían pertenecer a diferentes aplicaciones.

3.1.1.2.5.1 Sockets en C#

Una de las ventajas del lenguaje de programación C# es usa el Microsoft .Net Framework el cual provee de una poderosa biblioteca de APIs para programación, entre las que se encuentran los espacios de nombre *System.Net* y *System.Net.Sockets.* Estos espacios de nombres da una clara distinción entre el uso de TCP y UDP, definiendo un separado set de clases para ambos protocolos.

- **Socket Addresses**

IPv4 usa 32 bit binarios en direcciones para identificar la comunicación con hosts. Un cliente debe especificar la dirección IP de los hosts que ejecutan el programa servidor cuando inician una comunicación. La infraestructura de red utiliza direcciones de destino de 32 bit ara dirigir información del cliente a la maquina adecuada.

.Net encapsula la abstracción de las direcciones IP (192.168.2.5) en la clase IPAddresses la cual puede tomar una dirección IP *integer long* como argumento en su constructor o procesar s*tring* con la representación de una dirección IP utilizando la nomenglatura de un cuarteto punteado a través del método parse ( ).

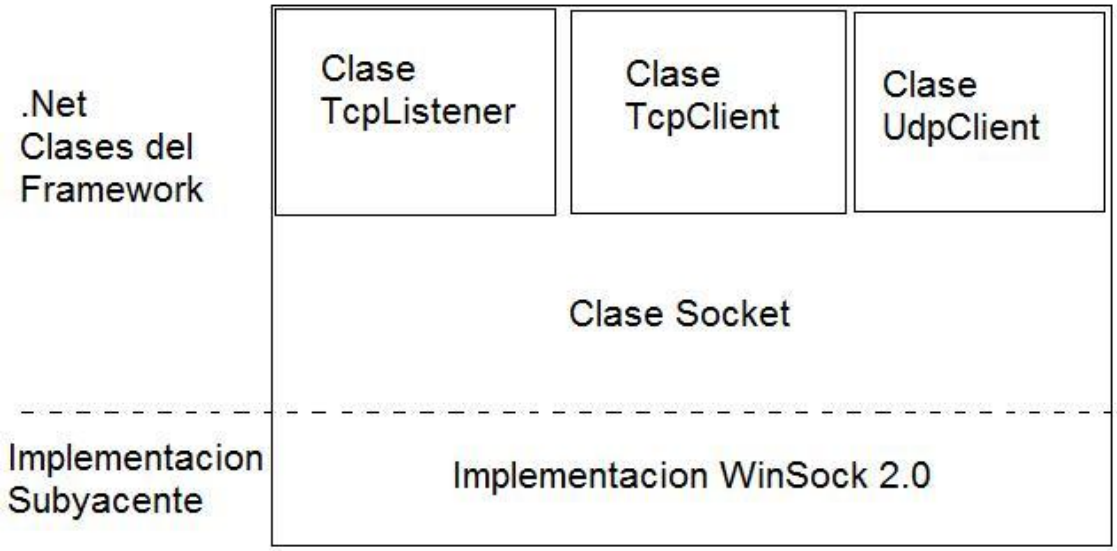
La clase *Dns* también posee diversos métodos para resolver direcciones IP y nombres sobre las cuales devulve información. El método *GetHostName ( )* no toma argumentos y retorna una cadena de string que contiene el nombre del localhost.

IphostEntry, contenedor que es retornado por el método *GetHostEntry( ), Resolve ( ) y GetHostbyAddresses ( ).* Este contenedor incluye información acerca del host como hostname, arreglo de direcciones IP y un arreglo de alias.

Al utilizar estas clases y si no se tiene acceso a una red la IP devuelta por *Dns* es el loopback 127.0.0.1

- **Implementación de Sockets en .Net**

Los sockets fueron inicialmente creados por *Berkeley Software Distribution (BSD) de Unix.* Una versión de sockets para Microsoft Windows llamada *WinSock 1.1* fue lanzada inicialmente sobre la versión 2.0. Con algunas diferencias menores, *WinSock* provee el estándar de funciones de sockets disponibles en el Berkely Socket C, que es la interfaz para C.



Relaciones de las Clases Socket

Diagrama 6

En 2002 Microsoft liberó el estandarizado Framework API conocido como .Net, el cual provee una unificada biblioteca de clases, a través de todos los lenguajes de programación que Microsoft ofrece.

Entre las características de la biblioteca, es que son clases de alto nivel que esconden mucho de los detalles de implementación y simplifican mucho las tareas de programación. Aun así, la abstracción puede algunas veces esconder mucho de la flexibilidad y el poder de las interfaces de bajo nivel.

En este orden para permitir el acceso a la subyacente interfaz socket, Microsoft implementa la clase .Net Socket la cual es una capa alrededor de las

funciones de *sockets de winsock* y posee más de la versatilidad y complejidad (Diagrama 6).

Entonces tres clases de alto nivel socket, *TcpClient, TcpListener y UdpClient* fueron implementadas usando la capa .Net con *su clase socket* . En efecto, esas clases poseen una propiedad protegida que es una instancia de la clase *socket* que ellas están usando. Gráficamente eso puede ser representado como lo muestra la figura anterior.

- **TCP Sockets**

El Framework .Net provee dos clases específicamente para TCP.

- TCPClient
- TCPListener

Estas clases proveen una abstracción de alto nivel de la clase *socket,* pero como se verá más adelante, existen instancias con avanzada funcionalidad que está disponible solo a través del uso directo de la clase *socket.*

Una instancia de cualquiera de estas clases representa un nodo final de una conexión TCP. Una *conexión TCP* es una abstracción de dos vías, cuyos finales son identificados cada uno por una *dirección IP y numero de Puerto.*

.Net usa la clase *EndPoint y su subclase IPEndPoint* para abstraer este concepto. Antes de ser usada para la comunicación, una conexión TCP debe pasar a través de una *fase de configuración* la cual *inicia enviando una solicitud TCP* del cliente al *TCP servidor.*

Una instancia de *TCPListener* escucha por solicitud de conexiones TCP y crea un nuevo socket (en la forma de un *TCPClient* o una instancia de Socket) para manejar cada conexión entrante.

- ✓ Cliente TCP,

Inicia la comunicación con un servidor que espera pasivamente a ser conectado. El típico cliente TCP pasa a través de tres pasos:

Construcción de la Instancia TCPClient

Una conexión TCP puede ser creada implícitamente en el constructor especificando el host y puerto remoto, o explícitamente utilizando el método *Connetc ( )*

Comunicarse utilizando el Stream del Socket

Una instancia Tcpclient conectada contiene un *Networkstream* que puede ser usado igual que cualquier otro stream de entrada o salida de .Net

Cerrar la Conexión

Llamar al método Close ( ) de TcpClient.

Cómo funcionan los pasos anteriores:

Los sockets TCP envían y reciben secuencias de bytes. Por tanto, los datos que deseemos enviar deben antes er convertidos a bytes. Un método estático de *ASCIIEndconding.* ASCII es *GetBytes ( )* retorna un arreglo de bytes de la representación de una cadena string pasada como argumento.

El servidor determina el puerto de echo que utilizara, aunque se tiene que por defecto es el 7, este se puede especificarse y ser convertido de un string a un entero con int32.parse ( ).

Pasos:

1- Creación de un Socket TCP

El constructor de la clase *Tcpclient* crea un socket e implícitamente establece una conexión con el servidor, especificado e identificado por una dirección IP o un nombre y un puerto especifico. Si se conoce con un nombre la implementación usa *DNS* para resolver la correspondiente dirección IP. Si cualquier error ocurre accesando al socket, el constructor lanza una *SocketException.*

2- Obteniendo el Stream del Socket

Asociado con cada *TcpClient* conectado, hay una instancia *NetworkStre*am, la cual esta heredada del *Stream y* provee una abstracción para una vista genérica de una secuencia de bytes. O sea, se envían datos en el socket, escribiendo bytes en el NetworkStream justo como se haría en cualquier otro stream y se recibirían los datos leyendo bytes del NetworkStream.

3- Enviando datos al Servidor

El método write ( ) del NetworkStream transmite el arreglo de bytes dado en la conexión con el servidor.

Los argumentos del método write ( ) son:

- El buffer de bytes conteniendo los datos a ser mandados.
- La distancia de bytes dentro del buffer donde los datos serán enviados inicialmente.
- El total de bytes a enviar.

4- Recibiendo Datos

Una de las sobrecargas del método Read ( ) de las clases NetworStream toma tres parámetros:

- Buffer de bytes a recibir.
- Distancia de bytes dentro del buffer donde el primer byte recibiendo se estaría colocando.
- El máximo de bytes colocados en el buffer.

El método Read ( ) bloquea, hasta que los datos están disponibles. Lee hasta el número de bytes especificados. Entonces retorna el numero de bytes

actualmente colocados en el el buffer (el cual podría ser menor al máximo dado). Si el servidor se cierra abruptamente el método read devolverá 0.

Es necesario manejar los errores que se podrían dar al utilizar estos métodos, estas pueden ser: S*ocketException* por el contructor de la clase *Tcpclient y IOException* por los métodos write ( ) y Read ( ) de la clase NetworkStream. Es por eso el uso de clase base de *Exception,* desde la cual se heredan las otras clases de excepciones, las cuales pueden ser manejadas por el usuario.

5- Cerrando el Stream,

El bloque final de la estructura Try/catch siempre será ejecutada. Si un error ocurre y fue capturado o si el cliente ha finalizado satisfactoriamente. El bloque finalmente es ejecutado y cierra el NetworkStream y el TcpClient.

- Clases y Métodos importantes en TcpClient

  - ✓ Constructores
    - TcpClient ( )
    - TcpClient (IPEndPoint local EP)
    - TcpClient (String hosname, int port)

Crea una nueva instancia de la clase TcpClient, el contructor posee parámetros opcionales para que una interfaz local se enlace a un IPEndPoint o al servidor al cual se conectara.

  - ✓ Métodos

Public void close ( )

Es el método que cierra de una conexión TCP pero hay que tener en cuenta que si se está usando un NetworkStream es preferible cerrar a este método ya que implícitamente cerrara a TcpClient, si se hace lo contrario no se cerrara el NetworkStream.

Public void Connect (String hostname, int port);

Conecta a un host remoto usando los parámetros de destino especificados y lanza excepciones como *ArgumentNullException, ArgumentOutOfRangeException, SocketException y ObjectDisposeException.*

Public NetworkStream GetStream( ) ;

Retorna una instancia NetworkStream usada para mandar y recibir datos y lanza excepciones como: *InvalidOperationException y ObjectDisposedException.*

Public Protected Socket Client (get;set);

Retorna o asigna el subyacente socket. Desde el cliente es una propiedad protegida y puede ser únicamente accedida heredándola de la clase TcpClient. Esto es útil para acceder a las opciones del socket que no son accesibles desde el *API TcpClient.*

IPEndPoint

Esta clase se hereda de la clase *EndPoint* y representa un nodo final de una conexión TCP/IP o sea una dirección IP y un número de puerto.

- **Servidor TCP**

El trabajo de este es establecer un punto final de clientes para conectarse y pasivamente esperara conexiones.

Un servidor TCP atraviesa dos pasos:

1- *Creación del Socket Servidor,* El escuchador TcpListener se inicializa con una IpAddress.any (0.0.0.0) lo que indica que estará escuchando sobre cualquier interfaz local. Y específica el número de puerto. El TcpListener escucha por peticiones de conexiones entrantes de clientes, sobre el puerto específicado, es importante tomar en cuenta no hacer uso de un puerto que está siendo utilizado por otro aplicativo, de lo contrario se lanzara una *SocketException.* Finalmente el método *Start (),* inicializa el socket subyacente enlazando el punto final local e iniciando la escucha por posibles conexiones entrantes.

2- *Se crea un Bucle infinito para el manejo de las conexiones,*

- *Aceptando una conexión entrante,*

  El único propósito de la instancia de un TcpListener es suministrar una nueva instancia TcpClient conectada a cada nueva conexión TCP. Cuando el servidor esta listo para manejar o procesar un cliente el TcpListener llama a AcceptTcpClient () y acepta únicamente una conexión entrante por el el puerto de TcpListener. Luego que es escuchada AcceptTcpClient () retorna una instancia TcpClient ya conectada al socket remoto y lista para leer y escribir. De la misma manera se puede utilizar el método

- *Obteniendo el NetworkStream,*

  Esto se hace por medio del método *GetStream ()* de la clase TcpClient y retorna una instancia de NetworkStream, el cual es usado para lectura y escritura.

- *Recibiendo Datos,*

  El bucle lee bytes desde el NetworkStream y los escribe hasta que el cliente cierra la conexión, lo cual es indicado por el retorno con valor de cero en el método Read ().

- *Finalmente se cierra el Cliente NetworkStream y el Socket.*

✓ Constructores

Public TcpListener (int port)
Public TcpListener (IPEndPoint)
Public TcpListener (Ipaddress, IP, int port)

El constructor posee tres formas, únicamente el puerto, una instancia de IpEndPoint o una dirección Ipaddress y un numero de puerto, cuando una dirección es especificada, esta representa la interfaz local sobre la que se escuchara.

✓ *Métodos*

- *Public AcceptSocket ();*

  Acepta una solicitud de conexión pendiente y retorna un socket que usara para enviar y recibir datos, si da algún error lanza la una excepción *InvalidOperationException.*

- *Public TcpClient AcceptClient ();*

  Acepta una solicitud pendiente de conexión y retorna un TcpClient, utilizado para recibir y enviar datos.

- *Public bool Pending ();*

  Retorna verdadero si existe una conexión entrante pendiente, que puede ser aceptada .

- *Public void Start ();*

  Comienza la inicialización del socket subyacente enlazado e inicia a escuchar por las solicitudes de la red.

- Public void Stop ();

  Detiene la escucha de las conexiones entrantes y cierra el TcpListener. Cualquier TcpClient o Socket aceptado debe ser cerrado separadamente.

♦ **Stream**

Este concepto se refiere a la secuencia ordenada de bytes. En .Net ya sea que se esté trabajando como cliente o servidor, las instancias de los métodos TcpClient o TcpListener poseen una instancia NetworkStream (clase heredada de Stream).

Cuando se escribe en el Stream de un TcpClient, los bytes pueden ser leídos desde el stream del TcpListener del objeto final de la conexión. Si se tiene un error al enviar o recibir por el Stream, se enviara la excepción *IOException.*

- ✓ Métodos
  - Public Virtual Close ();

    El método close del NetworkStream, lo cierra y asi también el socket subyacente, si posee uno.

  - *Public abstract int Read (bytes [] buffer, int offset, int length);*

    El método Read lee datos desde el stream a la red dentro del argumento de bytes buffer. El inicio (offset) dentro del buffer y el número de bytes a leer también están especificados. Read (); retorna el numero de bytes leidos. Al producir un error puede mandar las siguientes excepciones : *ArgumentNullException, ArgumentExcetion o IOException*
  - *Public Abstract Void Write (bytes buffer, int offset, int length);*

    Este método envía el contenido de bytes suministrado por el buffer.

    .

- ♦ **UDP sockets**

Este provee un servicio punto a punto diferente al que ofrece el TCP y ejecuta solamente dos funciones:

1- *Agregar otras capas de direccionamiento (puertos) a las IP*
2- *Detectar la corrupción de los datos, que puede ocurrir al transitar y descargar cualquier mensaje.*

Los sockets UDP tienen características diferentes a los Sockets TCP entre las cuales se puede mencionar:

- ✓ Los sockets UDP no tienen que estar conectados antes de ser utilizados.
- ✓ La información que viaje por estos sockets es llamada Datagramas, y posee su propia información de direccionamiento.
- ✓ Tan pronto es creado un socket UDP está preparado para enviar y recibir mensajes.
- ✓ El servicio de transporte punto a punto de UDP provee su mejor esfuerzo por lo que no garantiza que los mensajes lleguen asu destino.
- ✓ Los mensajes pueden ser entregados en un orden diferente la que se han mandado, por lo que un aplicativo que hace uso de Sockets UDP tiene que estar preparado para tratar con perdidas y reordenamiento.

Razones por las que se utiliza UDP:

- ✓ Eficiencia, cuando una aplicación intercambia una pequeña cantidad de datos , es decir un único mensaje de solicitud al servidor y un único mensaje de respuesta al cliente.
- ✓ Flexibilidad

El .Net Framework provee sockets UDP funcionales usando la clase *UdpClient* o *Sockes* para opciones más avanzadas.

- ♦ **Cliente UDP**

Un cliente UDP inicia por el envió de un datagrama al servidor que esta pasivamente esperando ser contactado. Este pasa a través de tres pasos:

1- Construye la instancia de la Clase UdpClient, opcionalmente especificando la dirección IP y el número del Puerto.
2- Usa los métodos Send () y Receive () de UdpClient para envio y recepción de datagramas.
3- Cuando ha finalizado se libera el socket con la ayuda del método Close ().

Como se puede observar a diferencia del TcpCliente, este no tiene que estar conectado con una dirección especifica de destino.

Funcionamiento en TcpClient:

1- *El método Connect () del UdpClient,* permite la especificación de la dirección remota y puerto. Este es enviado y no genera ningún tipo de mensaje de confirmación.

2- Creación del Socket UDP, La instancia del UdpClient puede mandar datagramas a cualquier socket UDP. El destino host y numero de puerto puede ser colocado en el constructor llamando al método Connect () o directamente con el método Send ().

3- Mandando Datagramas, La llamada Send () toma el datagrama (arreglo de bytes) y el número de bytes a mandar como argumento, este también puede tomar parámetros como la dirección o hostname y puerto como una instancia de la clase IPEndPoint. Si los argumentos de destino ha sido omitidos, tienen que estar incluidos en el constructo de la instancia. UdpClient o el método Connet ().

4- *Creando una IP y Punto Remoto para la Recepción,* La clase EndPoint específica una combinación de dirección IP y numero de puerto. Esa instancia IPEndPoint puede ser pasada por referencia al método s *Recieve (),* el cual se llenara con la información de IP y numero de pueto de enviador remoto.

5- *Manejo de la Recepcion del Datagrama,* Recieve () se bloquea hasta que recibe un datagrama. Cuando retorna la instancia de la clase IPEndPoint mandada como referencia contendrá la información dirección IP y numero de puerto de host remoto que envía el paquete que se recibió.

- **Servidor UDP**

Al igual que un servidor TCP, la función de este es establecer comunicación con un punto final, y esperar pasivamente a que un cliente inicie una comunicación. En UDP se inicia la comunicación cuando un datagrama es enviado por un cliente sin pasar por una etapa de configuración. Por lo que, recibir un datagrama de un servidor no es diferente a recibirlo de un cliente. Por lo anterior, es la misma clase (UdpClient) la que se utiliza para implementar a ambos.

- **Clases UdpClient**

  - ✓ Constructores

Public UdpClient ();
Public UdpClient (int port);
Public UdpClient (IPEndPoint local EP);
Public UdpClient (string hostname, int port);

Se crea una nueva instancia de la clase UpdClient y se tienen argumentos opcionales para el puerto, la interfaz local a la cual se esta enlazada (IPEndPoint) o al servidor al cual se conectara (string IP, hostname, numero de puerto). Si el destino no es colocado en el constructor debe ser colocado en una llamada a Connect (); o en el método Send (); sino lanzara las siguientes excepciones*: ArgumentNullException, ArgumentException o SocketException*

✓ Métodos

- Public void Close ();
  Cierra la conexión UDP.
- *Public void Connet (IPEndPoint end point);*
- *Public void Connet (IpAddress addr, int port)*
- *Public void Connet (string hostname, int port);*
  Estos métodos colocan el destino por defecto para un UdpClient.
- *Public byte Receive (ref IPEndPoint remote EP);*
  Retorna un datagrama UDP mandado por un host remoto como un arreglo de byte, y llena el IPEndPoint de referencia con la información del punto final del host remoto que está enviando el paquete.
- *Public int Send (byte[ ] dgram, int length);*
- *Public int Send (byte [ ] dgram, int length);*
- *Public int Send (byte[ ] dgram, int length, IPEndPoint end point);*
- *Public int Send (byte[ ] dgram, int length, string hostname, int port);*
  Envía un datagrama UDP a un host remoto, el cual esta especificado por el arreglo de bytes como argumento, y el numero de bytes a ser mandados en un argumento integer. Se pueden incluir argumentos opcionales para especificar el destino del datagrama usando una instancia de la clase IPEndPoint o una cadena string con el hostname/IP y un argumento integer que contenga el numero de puerto.

✓ Propiedades

- Protected Socket {get ; set}

  Obtiene o asigna el subyacente socket de red. Desde el echo que está en una propiedad protegida. Solo puede ser utilizada por las clases heredadas de UdpClient. Esto es útil para accesar a opciones del socket que no son directamente accesibles desde el API UdpClient.

♦ **Enviando y Recibiendo con Socket UDP**

Una diferencia importante, como se mencionaba anteriormente, es que UDP preserva las fronteras de los mensajes. Cada llamada a *UdpClient.receive ();* retorna información desde al menos una llamada de *UdpClient.Send ();.* Además, diferentes llamadas a UdpClient.Recieve (); nunca retornan datos desde la misma llamda UdpClient.Send ();

Cuando una llamada a Write (); en los sockets TCP retorna un stream, todo llamador sabe que esa información ha sido coiada dentro del buffer de transmisión y esta puede ser transmitida en ese momento o no. UDP, sin embargo, no provee recuperación por errores de la red, y por tanto no existe un buffer de datos para una posible retransmisión. Esto significa que por el tiempo de retorno a llamada Send (); el mensaje tiene que haber pasado el subyacente canal de transmisión.

Entre el tiempo que un mensaje llega de la red y el tiempo que su información retorna vía Read (); o Receive (); la información es guardada en una estructura FIFO (primera en llegar, primera en salir).

Con un socket TCP conectado, todos los datos recibidos son tratados como una secuencia continua de bytes.

Por otra parte, todos los datos recibidos de un socket UDP pueden haber tenido que venir desde diferentes envidadores, estos son mantenidos en una cola de mensajes y cada uno está asociado con la información que identifica su origen.

Una llamada a Receive (); nunca retorna mas de un mensaje. La máxima cantidad de datos que pueden ser transmitidos por un UdpClient son 65,507 bytes (63.97 Kb), o sea la máxima cantidad de dts que pueden ser cargados en un Datagrama UDP.

3.1.1.2.6 Procesamiento Múltiple (Hebras)

El principal objetivo del uso de hebras o procesamiento Múltiple es mejorar el rendimiento del sistema. De hecho, el uso de hebras se usa a menudo para reducir el tiempo de respuesta de una aplicación.

Y ya que .Net provee la capacidad de utilizarla, se ha hecho uso de este término.

La clase principal para el trabajo con subprocesos es Thread, y pueden ser en *primer plano o en segundo plano.*

El espacio de nombres utilizado es system.threading

- Propiedades

  - ✓ IsAlive, de tipo booleano e indica si el proceso esta activo.
  - ✓ IsBackground, tipo booleano y establece e indica su ka hebra es un sub proceso en segundo plano.
  - ✓ IsThreadPoolThread, indica si el subproceso pertenece a un grupo de subprocesos administrados.
  - ✓ ManagedThreadId, Estable un identificador único al subproceso.
  - ✓ Name, Obtiene o establece un nombre al subproceso (string).

- ✓ Priority, indica o establece una prioridad al subproceso con respecto al programa en ejecución. Pueden ser: *BelowNormal, highest, Normal, AboveNormal y Howest.*
- ✓ ThreadState, establece el estado actual del subproceso. Pueden ser: *AbortRequested, Background, Running, Sttoped, StopRequested, Suspended, SuspendeRequested, Unstarted, WaitSleepJoin.*

♦ Métodos

- ✓ Abort,

  Permite finalizar un subproceso, siempre que se utiliza genera una excepción, se puede omitir con el método ResetAbort (); desde la hebra que se ha abortado.
- ✓ Suspend,

  Suspende la actividad de un subproceso. Luego, con el método Resume (); se reanuda la actividad de este.

- ✓ .Sleep,

  Bloqueo en espera de un subproceso. Se le envía como parámetro el numero de milisegundos que se bloqueara el subproceso. Si se le pasa como parámetro cero se suspende.
  Si se envía como parámetro System.Threading.Timeout.Infinite el proceso se bloqueara de forma indefinida.

- ✓ Interrupt,

  Interrumpe el proceso actual si esta bloqueado.

- ✓ Start,

  Indica al programa que el proceso esta listo para ejecutarse.

- ✓ Join,

  Bloquea un subproceso hasta que otro proceso finalice.

- Constructores

Primero se tiene que crear una instancia de la clase ThreadStart, cuyo constructor, recibe como parámetro un delegado que apunta a un procedimiento que NO recibe parámetros.

Ejemplo:

ThreadStart mydelegado=new ThreadStart (Proc1);

Donde *Proc1* es un procedimiento sin parámetros.

Luego, ya se puede crear una nueva instancia de la clase Thread. Mandando como parámetro a su constructor la instancia de la clase ThreadStart que se creó anteriormente.

Ejemplo:

Thread Myhebra=new thread (mydelegado);

- Tareas de sincronización y Contención de Subprocesos

  - ✓ Clave Lock

Marca un bloque de instrucciones como una sección crucial, para lo cual utiliza el bloqueo de exclusión mutua de un objeto, la ejecución de una instrucción y, posteriormente, la liberación del bloqueo. La instrucción presenta la siguiente forma:

```
Object thisLock = new Object();
lock (thisLock)
{
   // Sección critica de código.
}
```

Y garantiza que un subproceso no va a entrar en una sección crítica del código mientras otro subproceso se encuentre ya en esta sección. Si otro subproceso intenta entrar en un código bloqueado, esperará, o se bloqueará, hasta que el objeto se libere.

- ✓ Exclusion Mutua Mutex

Un subproceso llama al método *WaitOne* de una exclusión mutua para solicitar su propiedad. La llamada se bloquea hasta que la exclusión mutua queda disponible o hasta que transcurre el intervalo de tiempo de espera opcional. Si una exclusión mutua no pertenece a ningún subproceso, el estado de la misma se señaliza.

Un subproceso libera una exclusión mutua llamando a su método *ReleaseMutex*. Las exclusiones mutuas cuentan con afinidad de subproceso; es decir, el subproceso que posee la exclusión mutua es el único que puede liberarla. Si un subproceso libera una exclusión mutua que no posee, se produce una excepción *ApplicationException* en el subproceso.

Como la clase **Mutex** se deriva *de WaitHandle*, también puede llamar al método estático *WaitAll* o *WaitAny* de **WaitHandle** para solicitar la propiedad de **Mutex** en combinación con otros controladores de espera.

Si un subproceso posee un objeto **Mutex**, dicho subproceso puede especificar el mismo **Mutex** en llamadas repetidas de solicitud de espera sin bloquear su ejecución; no obstante, deberá liberar **Mutex** las veces que sean necesarias para liberar su propiedad.

Hay dos tipos de exclusiones mutuas:

- Exclusiones mutuas locales
- Exclusiones mutuas del sistema con nombre.

Si crea un objeto **Mutex** mediante el uso de un constructor que acepta un nombre, quedará asociado a un objeto del sistema operativo con ese nombre. Las exclusiones mutuas del sistema con nombre son visibles en todo el sistema operativo y pueden utilizarse para sincronizar las actividades de los procesos. Puede crear varios objetos **Mutex** que representen la misma exclusión mutua del sistema con nombre y puede utilizar el método *OpenExisting* para abrir una exclusión mutua del sistema con nombre existente.

Una exclusión mutua local sólo existe dentro de su proceso. Puede utilizarla cualquier subproceso del proceso que tenga una referencia al objeto **Mutex** local. Cada objeto **Mutex** es una exclusión mutua local independiente.

3.1.1.2.7 Serialización Binaria

La serialización se puede definir como el proceso de almacenamiento del estado de un objeto.. Durante este proceso, los campos público y privado del objeto y el nombre de la clase, incluido el ensamblado que contiene la clase, se convierten en una secuencia de bytes que, a continuación, se escribe en una secuencia de datos. Cuando, después, el objeto se deserializa, se crea una copia exacta del objeto original.

Al implementar un mecanismo de serialización en un entorno orientado a objetos, es necesario equilibrar la facilidad de uso y la flexibilidad. El proceso se puede automatizar en gran medida, suponiendo que se tenga suficiente control sobre el proceso. Por ejemplo, pueden surgir situaciones en las que no basta con la simple serialización binaria o en las que puede existir un motivo específico para decidir qué campos se deben serializar. En las secciones siguientes se examina el robusto mecanismo de serialización que proporciona .NET Framework y se resaltan varias características importantes que le permiten personalizar el proceso para ajustarlo a sus necesidades.

- Clases

  - ✓ MemoryStream

Crea secuencias que utilizan como almacén de respaldo la memoria en lugar de un disco o una conexión de red. Esta encapsula los datos almacenados como una matriz de bytes sin signo que se inicializa al crear un objeto *MemoryStream;* también se puede crear una matriz vacía. Es posible obtener acceso directamente a los datos encapsulados en la memoria. Las secuencias de memoria pueden reducir la necesidad de archivos y búferes temporales en una aplicación.

Current position de una secuencia es la posición donde se llevará a cabo la siguiente operación de lectura o escritura. La posición actual puede recuperarse o establecerse mediante el método *Seek*. Al crear una nueva instancia de **MemoryStream**, la posición actual se establece en cero.

Las secuencias de memoria creadas con una matriz de bytes sin signo proporcionan una vista de secuencia de los datos que no es de tamaño variable y sólo es posible escribir en ellas. Al utilizar una matriz de bytes, no es posible anexar la secuencia ni reducirla, aunque tal vez sea posible modificar el contenido existente dependiendo de los parámetros pasados al constructor. Las secuencias de memoria vacías son de tamaño variable y se puede escribir y leer en ellas.

Si se agrega un objeto **MemoryStream** a un archivo ResX o un archivo .resources, llame al método *GetStream* en tiempo de ejecución para recuperarlo.

Si un objeto **MemoryStream** se serializa en un archivo de recursos, su serialización se realizará realmente como si fuera *UnmanagedMemoryStream.*

Este comportamiento proporciona un mejor rendimiento, así como la capacidad de obtener directamente un puntero a los datos, sin necesidad de pasar por los métodos de *Stream.*

- ✓ BinaryFormatter

Esta clase permite Serializar o deserializar un objeto o todo un gráfico de objetos conectados, en formato binario. Utiliza el **Espacio de nombres:**

System.Runtime.Serialization.Formatters.Binary

- ✓ FileStream

Esta se Utiliza para leer, escribir, abrir y cerrar archivos en un sistema de archivos, así como para manipular otros identificadores del sistema operativo relacionados con archivos, incluidos los de canalizaciones, entrada estándar y salida estándar. Puede especificar que las operaciones de lectura y escritura sean sincrónicas o asincrónicas. **FileStream** almacena en el búfer la entrada y la salida para obtener un mejor rendimiento.

Los objetos **FileStream** admiten el acceso aleatorio a los archivos mediante el método *Seek*, el cual permite que se mueva la posición de lectura/escritura a cualquier posición dentro del archivo. Esto se realiza mediante parámetros de punto de referencia de desplazamiento de byte. El desplazamiento de byte es relativo al punto de referencia de búsqueda, que puede ser el comienzo, la posición actual o el final del archivo subyacente, según lo representado por las tres propiedades de la clase *SeekOrigin.*

3.1.1.2.8 Checksum

Este término es utilizado para hablar del mecanismo que sirve para proteger la integridad de datos, verificando que no hayan sido corruptos es empleado para comunicaciones.

El proceso consiste en sumar cada uno de los componentes básicos de un sistema (generalmente cada byte) y almacenar el valor del resultado. Posteriormente, al llegr a su destino,se realiza el mismo procedimiento y se compara el resultado con el valor almacenado. Si ambas sumas concuerdan se asume que los datos no han sido corrompidos.

La forma más simple de checksum no detecta una variedad de corrupciones; particularmente no cambiará si:

- Se cambia el orden de los bytes de la información.
- Se agregan o eliminan bytes de valor igual a cero.
- Múltiples errores que se cancelan unos con otros.

Los tipos de control de redundancia más sofisticados, incluyendo el checksum de Fletcher, Adler-32 y el control de redundancia cíclica (CRC) los cuales son diseñados para tratar estas deficiencias, considerando no sólo el valor de cada byte sino también el de su posición. El costo de la capacidad de detectar más tipos de error aumenta junto con la complejidad del algoritmo de comprobación.

Estos tipos de control por redundancia son útiles en la detección de las modificaciones accidentales como corrupción de los datos o los errores de almacenamiento en un canal de comunicaciones.

3.1.1.2.9 Redes Punto a Punto.

Son estas redes más conocidas como redes P2P y se refieren a redes que *no tienen clientes ni servidores fijos,* sino una serie de *nodos* que se comportan simultáneamente como *clientes* y como *servidores* de los demás nodos de la red.

Este modelo contrasta con el modelo *cliente-servidor,* el cual se rige por una arquitectura monolítica donde no hay distribución de tareas entre si, sino una simple comunicación entre el cliente y el servidor donde se pueden intercambiar roles.

Las redes P2P son redes que a*provechan, administran y optimizan* el uso del ancho de banda que acumulas los demás usuarios de la red por medio de la conectividad entre los nodos de la red de forma directa.

Por tanto brindan mayor *rendimiento* en las *conexiones y transferencias* que con algunos métodos centralizados donde unos pocos servidores provee el total de ancho de banda y recursos compartidos. En estas redes generalmente para comunicar dos nodos, se conectan en gran parte vía *"ad hoc"* directamente con dos interfaces de rede conectadas .

Las redes P2P brindan facilidades para compartir: *audio, video, texto, datos y software* en cualquier formato digital también es utilizado en la tecnología *VoIP,* para hacer más eficiente la transmisión de datos en tiempo real.

Cualquier nodo puede iniciar, detener o completar una transacción compatible. La eficiencia de los sistemas P2P depende de varias variables:

- La velocidad del proceso
- Capacidad de Almacenamiento
- Configuración Local de la Red (firewall, Nat, Router, Ancho de Banda)

- **Filosofía de P2P**

Se basa principalmente en la que todos los usuarios deben compartir, y entre mas comparten mas privilegios y acceso más rápido a la información poseerán.
En una red P2P los usuarios que no comparten se conocen como *leechers.*

- **Características de las Redes P2P**

  - ✓ Escalabilidad

En general entre mayor sea el numero de nodos correctamente conectados a una red P2P, mejor será su funcionamiento. Esto difiere de una arquitectura cliente-servidor tradicional, con un sistema de servidor fijo, donde la adición de más clientes podría significar una baja velocidad de transferencia para todos estos.

- ✓ Robustez

La naturaleza distribuida de las redes P2P también incrementa la robustez en casos de existir fallas en la replicación excesiva de los datos hacia múltiples destinos, que en sistemas P2P puros, permitiendo hacer búsquedas de información a los peers sin necesidad de hacer peticiones a ningún servidor centralizado. En el último caso, no hay ningún punto de falla.

- ✓ Descentralización

Estas redes por definición son descentralizadas y todos los nodos son iguales, por lo tanto, ningún nodo es imprescindible para el funcionamiento de la red. En realidad algunas redes comúnmente llamadas P2P no cumplen con estos requerimientos como por ejemplo: *Napster, eDonkey o BitTorrent.*

- ✓ Los Costos están Repartidos

Esto se refiere a que se donan recursos a cambio de recursos. Según la aplicación de la red estos recursos pueden ser múltiples como: archivos, ancho de Banda, etc.

- ✓ 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. Algunas industrias DRM (Digital Rights Management) Gestión de Derechos Digitales.

- **Seguridad de las Redes P2P**

La seguridad en las redes P2P es de las más deseable pero menos implementada. Esta seguridad tiene como objetivo:

- Identificar y evitar a 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.

Algunos mecanismos de Seguridad que se están implementando actualmente son:

- Cifrado Múltiple
- Caja de Arena
- Gestión de Derechos de Autor
- Reputación
- Comunicaciones Segura
- Comentarios sobre Ficheros

3.2 Requerimientos de desarrollo

3.2.1 Entorno de Desarrollo.

Para el desarrollo de la aplicación se han tomado en cuenta dos aspectos básicos, la plataforma en la que se desarrollaría y las herramientas que se utilizarían como complemento para que se logre acoplar a las necesidades del usuario y así mismo tenga una apariencia que llame al usuario a hacer uso de esta.

3.2.1.1 Plataforma.

En un primer momento se tenían dos opciones de plataforma para desarrollar esta aplicación las cuales eran ASP.net y C#.net. Todas tenían sus ventajas y desventajas pero al estudiar detenidamente el tipo de aplicativo que se pretendía desarrollar el primer lenguaje en salir de las opciones fue ASP.net ya que este lenguaje ayuda a crear páginas Web y lo que se pretendía crear era un aplicativo de escritorio para ser utilizada en red por medio de protocolos de comunicación, esto se explica más adelante. Por lo que se decidió por la plataforma C#.net ya que esta provee de un lenguaje nuevo y a la vez maduro. Maduro porque la misma plataforma .Net está hecha con C#.

Para crear C# se mezclo a Visual Basic, C++ y Java, y se ha escogido lo mejor de cada uno y mejorado aquello en lo que éstas fallaban.

C# es un lenguaje orientado a objetos simples, elegantes y con seguridad en el tratamiento de tipos, que permite crear una gran variedad de aplicaciones.

C# también proporciona la capacidad de generar componentes de sistema duraderos, en lo que respecta a:

- Total compatibilidad entre COM y plataforma para integración de código existente.

- Gran robustez, gracias a la recolección de elementos no utilizados (liberación de memoria) y a la seguridad en el tratamiento de tipos.
- Seguridad implementada por medio de mecanismos de confianza intrínsecos del código.

Además, es posible interaccionar con otros lenguajes, entre plataformas distintas, y con datos heredados, en virtud de las siguientes características:

- Plena interoperabilidad por medio de los servicios de COM+ 1.0 y .NET Framework con un acceso limitado basado en bibliotecas.
- Compatibilidad con XML para interacción con componentes basados en tecnología Web.
- Capacidad de control de versiones para facilitar la administración y la implementación.

Además de tener la gran ventaja de ser *sencillo e intuitivo* y lo mejor ser un lenguaje *moderno* y sobretodo muy *eficiente*.

3.2.1.2 Herramientas.

La herramienta que se han utilizado en la elaboración del aplicativo es una suit de componentes que se instalaron para darle al aplicativo una mejor apariencia y de esta manera hacer al usuario un ambiente agradable. Esta es DevXperience la cual es un completo conjunto de controles y herramientas destinados al uso del desarrollo de aplicaciones en distintos lenguajes de programación como son Vb.net y C#.net la cual se instala en la maquina donde se está programando, luego se importa desde C#.net y ya está lista para hacer uso de todos los componentes que esta trae. Ya al momento de ejecutar el aplicativo en una maquina solo es necesario instalar el DLL de este software según sea la maquina o sea de 32 bit o 64 bit y todos los componentes son reconocidos, estos DLL ya iran incluidos en el instalador de la aplicación para que el usuario no tenga ningún problema.

Otra herramienta que ha sido utilizada es otra suit llamada DotnetBar esta es es una suite de 41 componentes para el Visual Studio 2005 / 2008, que nos proporciona -entre otros- todos los controles que aparecieron con Office 2007.

3.2.2 Aspectos legales

Para el caso de los aspectos legales para el desarrollo del Aplicativo es necesario obtener las licencias de los diferentes software a utilizar, para evitar así el uso ilegal de licencias, en este caso particular solamente se hará uso de la plataforma .net ya que el gestor de programación que se utilizara es C# y esta ya la posee el Departamento de Ingeniería y Arquitectura. Para lo que respecta el manejo de Base de Datos se hará uso de SQLite el cual es un gestor que no necesita licencia para utilizarlo.

Así mismo, el Aplicativo como tal, posee requerimientos operativos legales que cumplir, uno de ellos son los derechos de autor, este derecho se encuentra regulado por la Ley de Fomento y Protección de Propiedad Intelectual, en la cual el Art. 32 recalca que los programas de ordenador están protegidos los Derechos de Autor y se encuentran incluidos en el Régimen de Protección del Capítulo II Art. 13 de la referida Ley. Anexo 1

Además de esta ley que regula los Derechos de autor, existen otras, entre las cuales se encuentra el Reglamento General de Procesos de Graduación de la Universidad de El Salvador, en donde el Art. 29 establece, que son de propiedad exclusiva de la Universidad los trabajos de Graduación y solo ella puede disponer de los mismos, y autorizar a otros para que puedan hacer uso de los trabajos.

Por otra parte existe en el país una entidad reguladora de las copias ilegales de software la cual es Bussines Software Alliance (BSA), esta institución presenta una Denuncia a la Unidad de Protección de la Propiedad Intelectual de la Fiscalía General de la República, para que ellos se encarguen

de realizar el proceso de aprehensión de las personas u organizaciones que cometen el delito contra la propiedad intelectual. Este delito es penalizado con 4 años de prisión, de acuerdo con el Artículo 227 del Código Penal. Sin embargo la legislación establece una salida alternativa de la conciliación, el pago a cambio de cárcel.

3.3 Requerimientos operativos

3.3.1 Hardware.

La PC en la que se instalara el aplicativo tiene que tener ciertas características especiales para su correcta utilización en lo que respecta al Hardware estas son:

- Disco duro de 40 Gb
- Memoria Ram 512 Mb
- Una tarjeta de red.
- Tarjeta de video compatible con directx 9.

3.3.2 Software.

Para poder hacer uso del aplicativo las PC tienen que tener el Sistema Operativo Windows desde su versión XP en delante, también para mayor protección en los archivos que se estarán compartiendo es importante tener activo un antivirus en la PC.

3.3.3 Usuarios.

El sistema está elaborado con un ambiente amigable para el usuario, para que de esta manera no se haga engorroso su uso. Se puede mencionar como requerimientos mínimo conocimientos básicos del uso de una computadora, como son:

- Instalación de un software
- Uso de Internet
- Creación de Carpetas
- Uso de la Herramienta Word de Microsoft Office

3.3.4 Red.

Para que el sistema funcione con la finalidad para el cual ha sido creado, se necesita estrictamente acceso a la red, no necesariamente Internet pero sí que en el lugar donde se utilice este instalada una red de trabajo en la que las estaciones de trabajo puedan hacer comunicación entre sí. Puede ser esta por wireless o con cable UTP.

Con respecto al puerto que utiliza el aplicativo, este por defecto utiliza el 4545 en caso se tenga bloqueado dicho puerto, este puede ser configurado por el usuario y cambiarlo al que desee.

La computadora que haga uso del aplicativo, tiene que utilizar el protocolo TCP/IP y usar IPv4.

3.4 Factibilidad del Proyecto

3.4.1 Factibilidad de Recursos Humanos.

El recurso humano que se necesitará en el desarrollo de la Aplicación .NET para Apoyar a la Gestión Administrativa del Departamento de Ingeniería y Arquitectura de la Facultad Multidisciplinaria de Occidente que se ha titulado Sistema de Comunicación Integral P2P, es el que se detalla a continuación.

- Analistas Programadores:

Los integrantes del proyecto, egresados de la carrera de Ingeniería en Sistemas Informáticos, como trabajo de graduación, para optar al título de Ingenieros en Sistemas Informáticos, jugaran el papel de Analistas Programadores.

Los Analistas programadores estarán a cargo del desarrollo de todo el proyecto. Encargados de realizar el análisis, diseño, programación, implementación e instalación de la aplicación .NET, como producto final.

- Docente Coordinador de Trabajo de de Grado:

Profesional que brinda orientación académica integral y personalizada, durante el desarrollo y evaluación del Proyecto de Tesis, para optar a la carrera de Ingeniero en Sistemas Informáticos.

- Futuros Usuarios de la Aplicación:

Personal que labora en el Departamento de Ingeniería y Arquitectura como futuros usuarios del sistema de información a desarrollar interactuarán con el equipo de desarrollo a fin de determinar sus necesidades de información y garantizar que el sistema desarrollado las satisface.

3.4.2 Factibilidad de Recursos Bibliográficos.

Se ha recurrido a literatura con contenidos que describen el funcionamiento de las diferentes actividades que se realizan en el Departamento de Ingeniería y Arquitectura, así también a libros con el fin de entender la terminología y el uso de esta para el momento de la elaboración del aplicativo.

También se ha hecho uso de documentación publicada en internet con los fines antes descritos.

Todo la información bibliográfica que se ha ido necesitando siempre se ha encontrado ya sea en las bibliotecas de la Facultad multidisciplinaria de Occidente como también en internet, en sitios donde la descargas son gratuitas, por lo que no se ha incurrido en ningún gasto en lo que respecta a la recolección de información.

3.4.3 Factibilidad Técnica

En la factibilidad técnica se consideran los conocimientos técnicos que poseen cada una de las personas encargadas de utilizar el sistema, además de la disponibilidad de hardware y software capaz de tener en funcionamiento el sistema.

Tomando en cuenta lo anterior, se han analizado los siguientes puntos:

- Actualmente se cuenta con el hardware adecuado capaz de que el sistema funcione de forma correcta, *Tabla 3*.

DESCRIPCIÓN	CARACTERÍSTICAS MINIMAS	* CARACTERÍSTICAS RECOMENDADAS
Procesador	**1.8 MHZ**	2.4 GHZ
Memoria RAM	**512 MB**	512 MB
Espacio Libre Disco Duro	**40 GB**	2 GB
Tarjeta de Video	**Compatible con directx 9**	1280 * 1024
Tarjeta de Red	**Ethernet**	Ethernet

Tabla 3

- El sistema operativo Windows XP, Vista y Seven se acopla perfectamente al Aplicativo.
- Los futuros encargados del SCIP2P tienen los conocimientos técnicos necesarios para utilizar el sistema.

3.4.3.1 Software

Para poder llevar a cabo el desarrollo del software propuesto, para poder llegar a la automatización de los procesos del Departamento de ingeniería y Arquitectura, se evaluaron los cuatro principales requerimientos de desarrollo a nivel de plataforma tecnológica *tabla 4* y cuatro a nivel de gestor de base de datos *tabla 5*, esta evaluación se realizo en base a una calificación de 0 a 5.

Lenguaje de Programación	Conocimiento por Parte de Equipo de Programación	Disponibilidad de Licencia	Velocidad	PUNTAJE FINAL
C#.NET	5	5	4	14
JAVA	3	5	4	12
PHP	3	5	5	13
ASP.NET	4	5	3	12

Tabla 4

SGBD	Conocimiento por Parte de Equipo de Programación	Disponibilidad en la Institución a Implementar	Menor Consumo de Recursos	PUNTAJE GLOBAL
SQL Server	4	5	4	13
MySQL	4	5	5	14
Access	4	5	1	10
SQLite	5	5	5	15

Tabla 5

En base a la información anterior se elige a C# .Net Express Edition 2005 y SQLite, para el Desarrollo del SCIP2P.

3.4.3.2 Hardware

En el desarrollo del aplicativo se necesitan varias maquinas, por ser este orientado a que trabaje en red. Las maquinas que se utilizaran para este fin tienen que tener las siguientes características *tabla 6*:

Tipo de Equipo	Descripción con Respecto a Hardware	Sistemas Operativos
Laptop.	Motherboard: Intel Pentium 4 Procesador : 2.8 GHz RAM : 512 Mb Disco Duro: 80 Gb Tarjeta/Red: Si (una)	**S. O. : XP Profesional Service Pack 3**
Laptop	Motherboard: AMD Turion 64 Procesador : 2.1 GHz RAM : 4 Gb Disco Duro: 250 Gb Tarjeta/Red: Si	**S. O. : Vista Ultimate 32 bit Service Pack 3**

Tipo de Equipo	Descripción con Respecto a Hardware	Sistemas Operativos
Laptop	**Motherboard: AMD Turion 64** **Procesador : 1.8 GHz** **RAM : 1 Gb** **Disco Duro: 100 Gb** **Tarjeta/Red: Si**	**S. O. : XP Profesional Service Pack 2**
Laptop	Motherboard: Intel Core 2 Duo Procesador : 11.8 GHz c/u RAM : 2 Gb Disco Duro: 240 Gb Tarjeta/Red: Si	**S. O. : XP Profesional Service Pack 3**

Tabla 6

3.4.4 Factibilidad Operativa

Uno de los principales inconvenientes que surge es la resistencia al cambio por parte de los usuarios cuando se implanta un nuevo sistema. Esta se tratará de desaparecer al hacer un plan de implementación que permita a los usuarios interactuar con el sistema.

- Seguridad
  - ✓ En lo que respecta a la seguridad, se plantean recomendaciones divididas en los siguientes ámbitos:
- Espacio Físico y Ambiente
  - ✓ El sitio donde se encuentren los equipos en los que estará instalado el aplicativo deberán poseer paredes, cielos falsos y tejado libre de derrumbes.
  - ✓ El área en la cual se ubiquen los equipos deberá contar con la adecuada ventilación e iluminación, para que las condiciones ambientales de enfriamiento del equipo sean las óptimas.

- Instalaciones Eléctricas

  - ✓ Deberá proporcionarse una infraestructura eléctrica segura, empotrada y polarizada.
  - ✓ Utilizar UPS por ordenador, esto garantizara un tiempo suficiente para salvaguardar la información que se esté procesando y apagar el equipo correctamente.

- Información

  - ✓ Deberá implementarse una política de respaldos o backup de la información contenida en el archivo .dir por cualquier problema que surja con la maquina en la que ejecuta normalmente el sistema.

- Otras Generalidades
  - ✓ Mantener instalado y actualizado un antivirus en cada equipo, tanto servidores como terminales de usuario, para evitar cualquier amenaza de virus que puedan afectar los sistemas en ejecución.
  - ✓ Deberá proporcionarse en el área cercana a los equipos un extintor de polvo tipo **ENPA06** para contrarrestar el riesgo de incendio.

3.4.5 Factibilidad Económica

En esta parte se muestra el costo del aplicativo que se ha desarrollado *Tabla 7*. Así como también el costo de la puesta en marcha de este *Tabla 8*.

El sistema brindará muchos beneficios como mejorar la comunicación, ahorro de tiempo, en la búsqueda de información gracias al compartimiento de esta, lo que conlleva a un gran ahorro de dinero por parte de los usuarios *Tabla 9*.

Para la institución, es viable debido a que el costo será mínimo, esto debido a que ya cuentan con el equipo de cómputo que cumple con los requisitos para que el sistema funcione correctamente, además de la red, e licencias de software para el desarrollo de este.

- **COSTOS DE DESARROLLO**

1. Energía eléctrica

El costo de KWH según la empresa AES CLESA es de $0.151715 IVA incluido y tomando en consideración que los equipos se utilizan 8 horas diarias y consumen 0.600 KWH, se obtiene:

$0.151715 KWH x 0.600 KWH = $0.09 consumo por hora

$0.09 por hora x 8 horas diarias x 30 días del mes x 26 meses = $ 561.6

Por lo que se tiene un costo de energía Eléctrica de $561.60

2. Costos de personal

El presente aplicativo como se viene mencionando a lo largo del documento ha tenido una duración de 2 años con 2 meses exactos y han trabajado dos personas.
Por lo general, un programador tiene un sueldo mensual de $800.00.

El costo de mano de obra es de:

500 por trabajador x 12 meses = $ 6000.00 Anuales

Teniendo un costo total de $12,500.00

3. Depreciación

Según el art 30, capítulo IV, Titulo IV de la ley del impuesto sobre la renta, se puede depreciar un equipo con un máximo de 20% anual.

Los equipos que se han utilizado 4 computadoras, y cada una tuvo un costo de $700.00:

$700 cada equipo x 4 equipos x 0.20 depreciación = $560 anuales

Y se ha trabajado en el proyecto 2 años y 2 meses por lo que el costo de depreciación ha sido

$1213.33

Descripciónón	Precio
*** SQLite**	Gratis
*** C#.Net 2005**	Gratis
Libros de Texto	$ 20.00
Electricidad	$ 561.00
Costos Personales	$ 12,500.00
Depreciación de Equipo	$ 1,213.33
TOTAL	$ 14,294.33

Tabla 7

- **COSTO DE PUESTA EN MARCHA**

Para sacar el valor de la puesta en marcha del aplicativo, se ha analizado los siguientes puntos:

- ✓ Computadoras (con los requerimientos mínimos antes mencionados)
- ✓ Red
- ✓ Capacitación para el uso

Y se ha encontrado que el Departamento cuenta con cada uno de estos, y con respecto a las capacitaciones que requieran los usuarios estas serán impartidas sin costo alguno por parte de los desarrolladores del aplicativo.

Por lo cual, el costo de la puesta en marcha es de $0.00 dólares, lo que hace al aplicativo totalmente Factible.

- **ANALISIS BENEFICIO-COSTO**

El Beneficio-Costo no solo se mide monetariamente hablando, esto también se puede apreciar desde el punto de vista en que el usuario se verá beneficiado con respecto a como se hacía el trabajo anteriormente.

Acción	Actualmente	Utilizando el SCIP2P	Beneficio
Comunicación entre diferentes entidades.	✓ Utilizando el Sin Messenger, sujeto al servicio de internet en la institución. ✓ Por Teléfono ✓ Mensajes de Texto	Los usuarios que estén conectados al Sistema, tendrán el acceso a localizar y comunicarse con la persona con quien desea ponerse en contacto.	• Reducción de costos por parte de las personas que hacían uso de mensajes de texto o llamadas telefónicas para la comunicación. • No se estará sujeto a la salida de internet para que todas las personas del Departamento puedan estar comunicados. • La comunicación se puede establecer desde cualquier computadora, siempre y cuando el usuario tenga listo el archivo .dir desde el cual se abre la sesión de cada usuario.

Compartimiento de Información	• Solicitando el documento al docente que lo proporciona, y luego sacarle copia. • Buscar el documento en la fotocopiadora. • Solicitar con compañeros de clases. • Se compra el libro de donde se ha sacado la información.	Los estudiantes iniciaran sesión en el SCIP2P, y desde la sección de transferencia de archivos, podrán: • Subir un archivo para compartirlo. • Eliminar un archivo que no quieren seguir compartiendo. • Buscar archivos • Descargar • Compartir	• Reducción de costos en fotocopias, beneficio no estimados por la diversidad de documentación que se mueven en las diferentes carreras. • Reducción de costo en la compra de libros que están en formato digital y pueden ser compartidos por medio del SCIP2P. • Reducción de tiempo de búsqueda del documento que se desea. • Una gran cantidad de documentación que se utiliza en las distintas materias estarán a la disposición de quien quiera hacer uso de ella.

		• Envió Personalizado	
Solicitud de formas en Secretaria	• La secretaria llena manualmente las Formas. • Se digitan la información en su totalidad, tantas veces los usuarios solicitan una misma forma.	La secretaria creara una plantilla por solicitud, en la que solamente será necesario ingresar datos que son variables para cada usuario. Una vez ingresada la forma, esta se podrá recuperar y volver a utilizar tantas veces sea necesario. Desde el aplicativo se podrá mandar a imprimir.	• Reducción de tiempo, en la creación de Formas necesarias.
Reserva de Equipo Audio Visual	Los alumnos o docentes que necesitan de este servicio, tienen que solicitar a la secretaria. Ella a su vez tiene que revisar un folleto en el que se muestra un	La secretaria sabrá si hay equipo disponible con solo ingresar la hora y día para la cual se está haciendo la reservación. Así como también se sabrá el cañón	• Los usuarios de este servicio se verán beneficiado ya que reducirán tiempo de espera en la reservación de un equipo. • La secretaria del departamento, reducirá tiempo en la asignación de

cuadro con la programación	que le será asignado.	equipo a los usuarios.

Tabla 8

Capítulo IV

Diseño de la Aplicación

Capítulo IV.- Diseño de la Aplicación

4.1 Esquema de Comunicación.

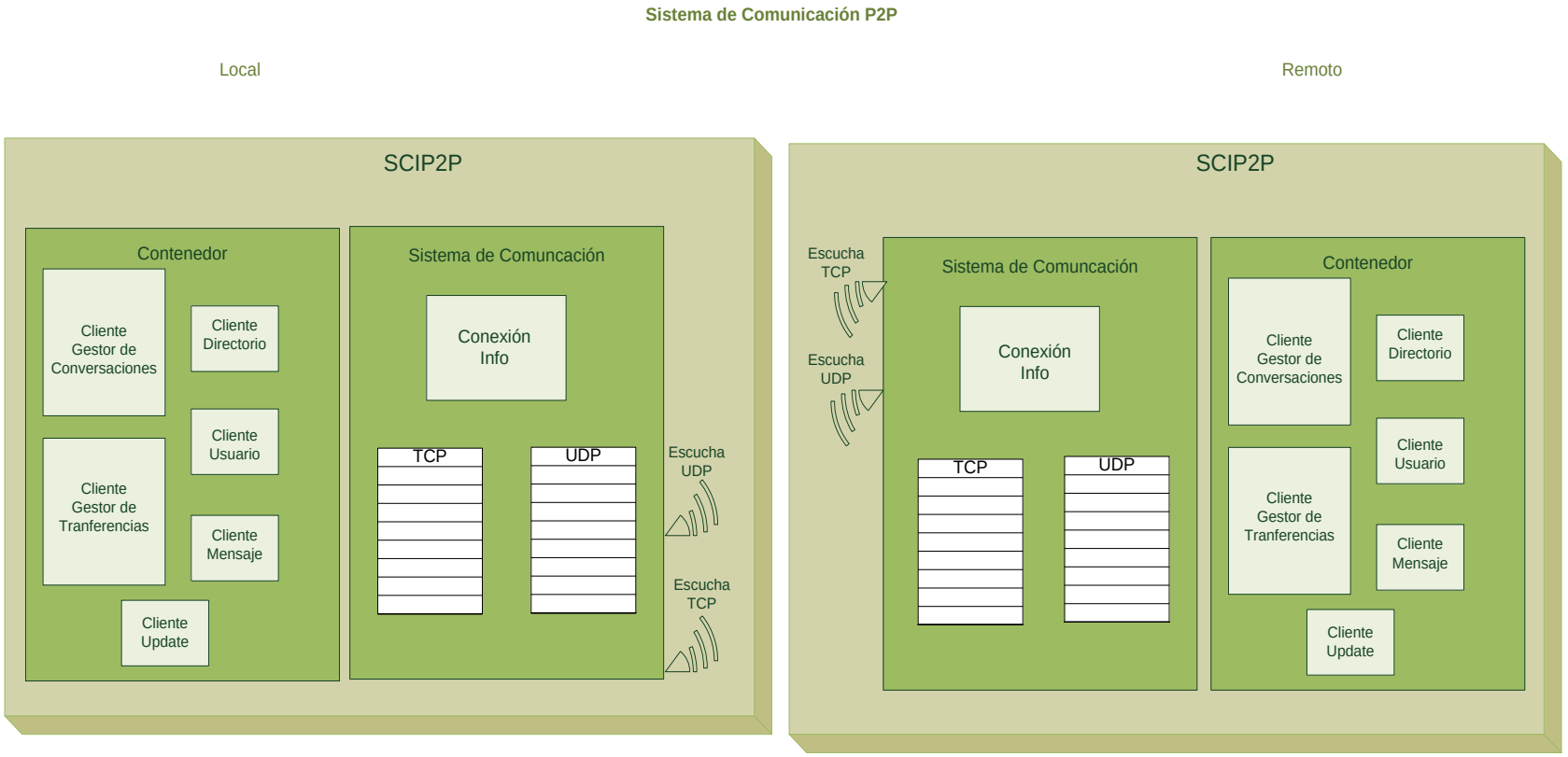


Diagrama 7

El esquema de trabajo dentro del **Sistema de Comunicación Integral Punto a Punto SCIP2P** (diagrama 7) hace uso de dos entidades principales el Sistema de *Comunicación* y el *Contenedor de Clientes*.

Siendo El Sistema de Comunicación, la clase más importante del SCIP2P, ya que es la que hace posible la comunicación entre dos estaciones de trabajo en la red, brindando servicios al contenedor de clientes para que la información procesada por cada cliente pueda ser enviada y/o recibida a través de la red, desde o hacia otra estación SCIP2P conectada. Es por eso que el Contenedor de Clientes posee una referencia al Sistema de Comunicación, lo cual garantiza que todos los Clientes (Clientes P2P o Clientes de Datos, (Ver diferencia entre Clientes P2P y Cliente de Datos)) del Contenedor de Clientes puedan hacer uso de los servicios de El Sistema de Comunicación.

4.1.1 Sistema de comunicación

Para poder brindar la conectividad requerida a una estación de trabajo SCIP2P, la clase Sistema de Comunicación (Diagrama 8) se encargada de gestionar todo lo referente a conexión o desconexión un equipo a una red local, a través una instancia de la clase ***ConexionInfo*** *(Diagrama 9)*.

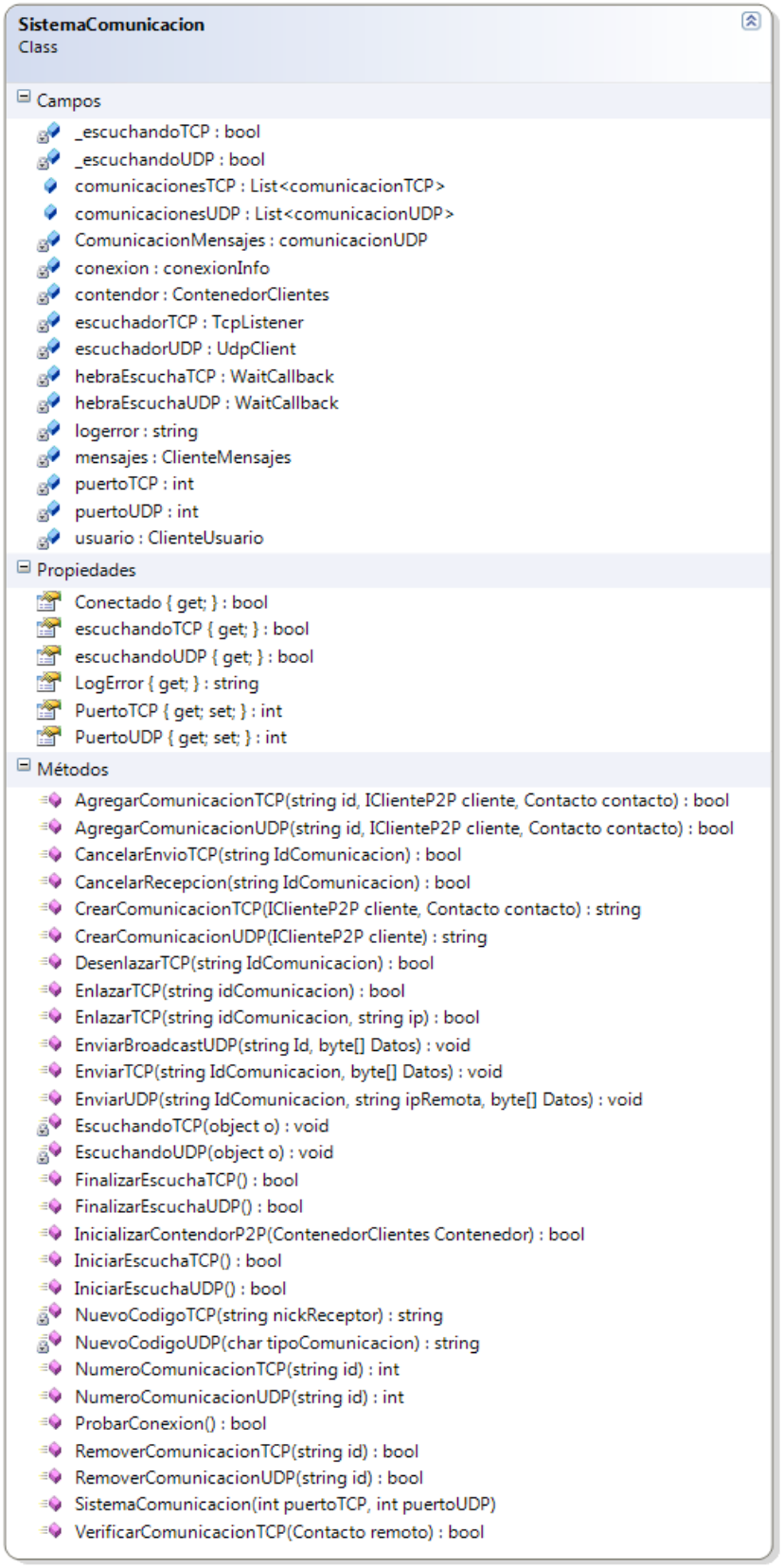

SistemaComunicacion
Class
Campos
_escuchandoTCP : bool
_escuchandoUDP : bool
comunicacionesTCP : List<comunicacionTCP>
comunicacionesUDP : List<comunicacionUDP>
ComunicacionMensajes : comunicacionUDP
conexion : conexionInfo
contendor : ContenedorClientes
escuchadorTCP : TcpListener
escuchadorUDP : UdpClient
hebraEscuchaTCP : WaitCallback
hebraEscuchaUDP : WaitCallback
logerror : string
mensajes : ClienteMensajes
puertoTCP : int
puertoUDP : int
usuario : ClienteUsuario
Propiedades
Conectado { get; } : bool
escuchandoTCP { get; } : bool
escuchandoUDP { get; } : bool
LogError { get; } : string
PuertoTCP { get; set; } : int
PuertoUDP { get; set; } : int
Métodos
AgregarComunicacionTCP(string id, IClienteP2P cliente, Contacto contacto) : bool
AgregarComunicacionUDP(string id, IClienteP2P cliente, Contacto contacto) : bool
CancelarEnvioTCP(string IdComunicacion) : bool
CancelarRecepcion(string IdComunicacion) : bool
CrearComunicacionTCP(IClienteP2P cliente, Contacto contacto) : string
CrearComunicacionUDP(IClienteP2P cliente) : string
DesenlazarTCP(string IdComunicacion) : bool
EnlazarTCP(string idComunicacion) : bool
EnlazarTCP(string idComunicacion, string ip) : bool
EnviarBroadcastUDP(string Id, byte[] Datos) : void
EnviarTCP(string IdComunicacion, byte[] Datos) : void
EnviarUDP(string IdComunicacion, string ipRemota, byte[] Datos) : void
EscuchandoTCP(object o) : void
EscuchandoUDP(object o) : void
FinalizarEscuchaTCP() : bool
FinalizarEscuchaUDP() : bool
InicializarContendorP2P(ContenedorClientes Contenedor) : bool
IniciarEscuchaTCP() : bool
IniciarEscuchaUDP() : bool
NuevoCodigoTCP(string nickReceptor) : string
NuevoCodigoUDP(char tipoComunicacion) : string
NumeroComunicacionTCP(string id) : int
NumeroComunicacionUDP(string id) : int
ProbarConexion() : bool
RemoverComunicacionTCP(string id) : bool
RemoverComunicacionUDP(string id) : bool
SistemaComunicacion(int puertoTCP, int puertoUDP)
VerificarComunicacionTCP(Contacto remoto) : bool

Diagrama 8

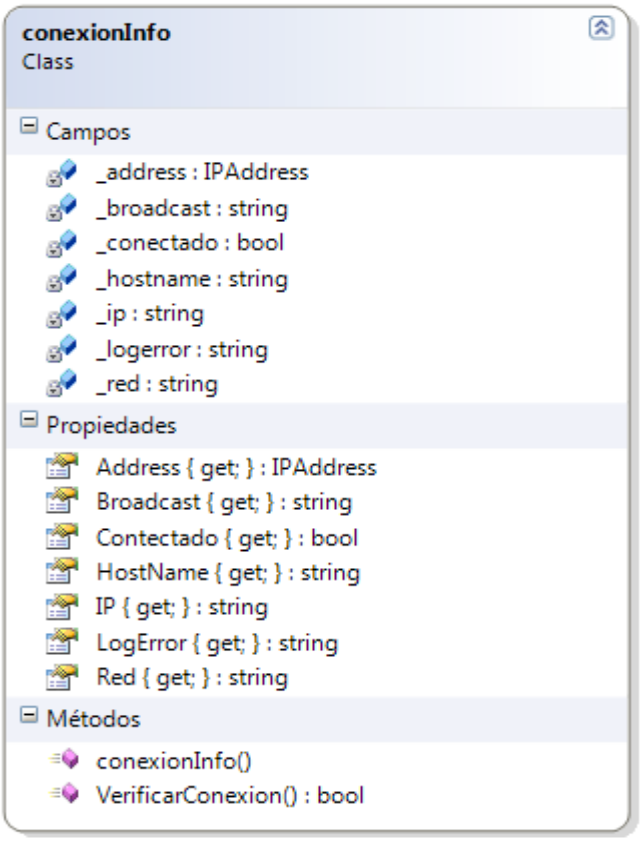


Diagrama 9

Esta clase se encarga obtener información sobre la conectividad de la computadora en la red. Una instancia de la clase **conexionInfo** es capaz de almacenar los datos básicos necesarios para poder identificar a una computadora en la red, como por ejemplo: su dirección IP, dirección de Brodcast, nombre de la computadora en la red, etc.

Volviendo al Sistema de Comunicación se puede decir que la función más importante es la de *administrar las Comunicaciones TCP y Comunicación UDP,* ya que dentro de esta clase se maneja todo lo referente a recibir conexiones entrantes así como Iniciar, Interrumpir o Finalizar una comunicación TCP o UDP. El sistema de comunicación así mismo se vale de Hilos o Hebras para lograr el funcionamiento multipropósito que le permiten funcionar como un servidor, procesando conexionen entrantes TCP y UDP, o administrando

conexiones clientes (comunicaciones TCP y UDP que serán escuchadas por el Sistemas de Comunicación en otras estaciones de trabajo), es decir creando, eliminando y transmitiendo información a través de flujos Stream (si utilizan el protocolo TCP, esto debido a que es un protocolo orientado a la conexión y trabaja estableciendo una conexión virtual entre dos y equipos en una misma red de área local), o para el envío de datagramas (Si utiliza el protocolo UDP, protocolo no orientado a la conexión).

Para poder capturar las conexiones entrantes el sistema de comunicación, este se vale de dos métodos de escucha que funcionan cada uno en hebra o hilos en segundo plano (garantizando que la estación de trabajo no detenga ni interrumpa ningún proceso realizado), de estas hebras, una es la encargada de escuchar las conexiones entrantes TCP (**EscuchaTCP**) y otra es la encargada de escucharlos datagramas UDP (**EscuchaUDP**) (Diagrama 10).

Funcionamiento de Escucha TCP y UDP

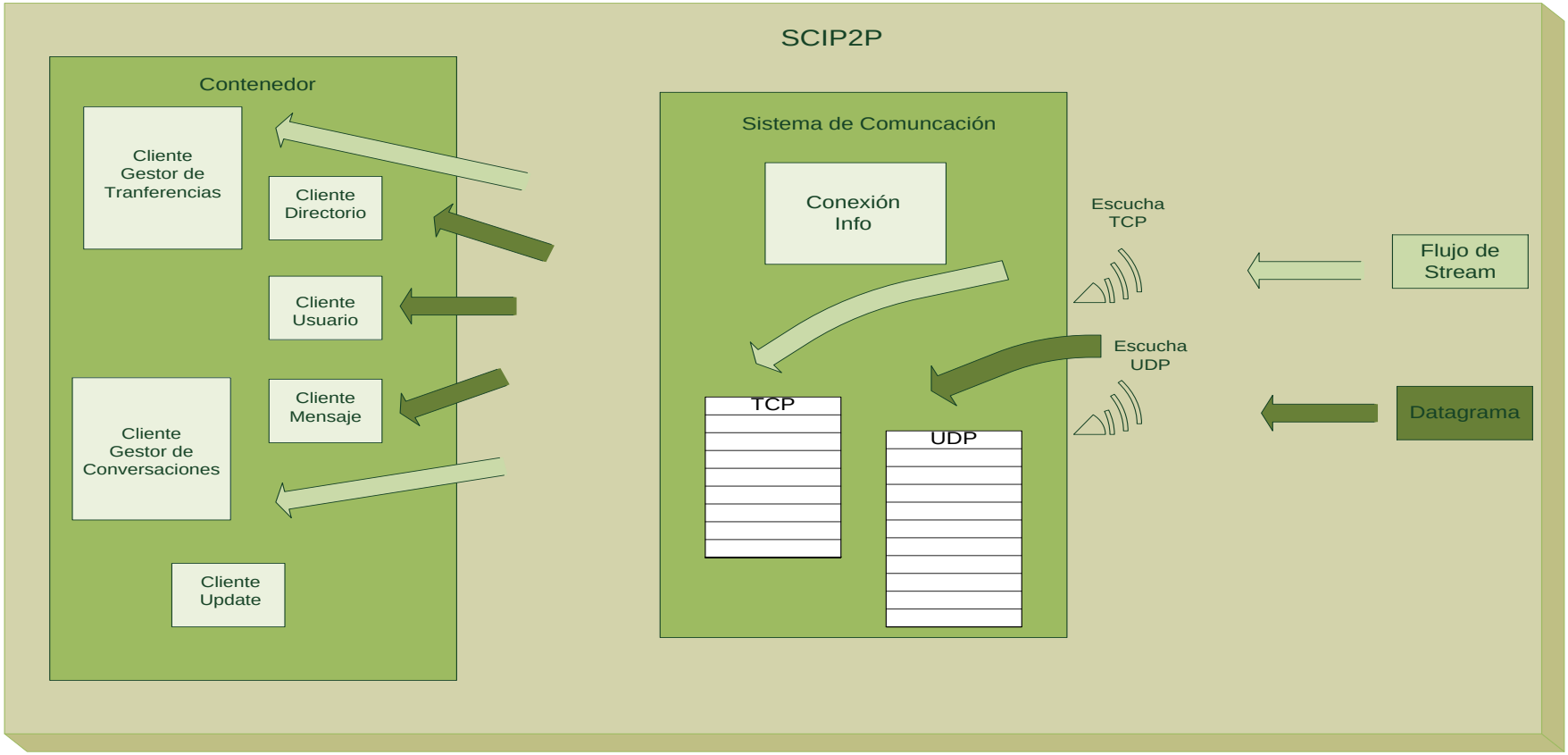


Diagrama 10

Estos métodos (Escucha TCP y UDP) están continuamente en espera de datagramas (UDP) o flujos Streams (TCP), al recibir uno de estos (independientemente del caso que sea) los métodos de escucha son los encargados de *leer la cabecera del Datagrama o Flujo de Bytes* (Stream) y así poder determinar a cuál de las comunicaciones administradas por el Sistema de comunicación (comunicación TCP o Comunicación UDP) le pertenece, para que ella sea quien lea los datos recibidos.

Tanto las conexiones remotas entrantes capturadas, como las conexiones creadas localmente son *recordadas* por el Sistema de Comunicación, utilizando *dos listas doblemente enlazadas*, cuya función *es almacenar temporalmente las Comunicaciones TCP y Comunicaciones UDP*, estas están asociadas a clientes del Contendor de Clientes. Básicamente, como se decía al inicio, el sistema de comunicación es el que da soporte al contenedor de clientes y a cada uno de los clientes (ClientesP2P) contenidos en el mismo cuando ellos necesitan crear o utilizar alguna de estas comunicaciones.

Es importante de mencionar que a pesar que sistema de comunicación crea las comunicaciones TCP o UDP a solicitud del Contendor de Clientes, es el sistema de comunicación el que se carga asignarle un identificador a través de la cual se puede identificar cada comunicación de forma única.

Las clases de Comunicación TCP y UDP, poseen una referencia al cliente de datos (clienteP2P) que hace uso de sus servicios*. Los métodos de lectura y escritura* de datos requieren cada uno de su propia hebra para garantizar que puedan ejecutarse la lectura y el procesamiento de los datos recibidos de forma simultánea.

4.1.1.1 ComunicacionUDP

La clase *ComunicacionUDP* es la clase que permite administrar una comunicación UDP utilizando una instancia de la clase ***UDPClient*****,** la cual permite crear un socket capaz de transmitir datagramas por la red.

En la familia de protocolos de Internet UDP proporciona una sencilla interfaz entre la capa de red y la capa de aplicación. UDP no otorga garantías para la entrega de sus mensajes y el origen UDP no retiene estados de los mensajes UDP que han sido enviados a la red. UDP sólo añade multiplexado de aplicación y suma de verificación de la cabecera y la carga útil.

+	Bits 0 – 15	16 - 31
0	Puerto origen	Puerto destino
32	Longitud del Mensaje	Suma de verificación
64	Datos	

Tabla 9

La cabecera de un Datagrama UDP (tabla 9) consta de cuatro campos, de los cuales dos son opcionales (con fondo rojo en la tabla). Los campos de los puertos origen y destino son campos de 16 bits que identifican el proceso de origen y recepción. Ya que UDP carece de un servidor de estado y *el origen UDP no solicita respuestas*, el *puerto origen es opcional*. En caso de no ser utilizado, el puerto origen debe ser puesto a cero. A los campos del puerto destino le sigue un campo obligatorio que indica el tamaño en bytes del datagrama UDP incluidos los datos. El valor mínimo es de 8 bytes. El campo de la cabecera restante es una suma de comprobación de 16 bits que abarca la cabecera, los datos y una pseudo-cabecera con las IP origen y destino, el protocolo, la longitud del datagrama y 0's hasta completar un múltiplo de 16, pero no los datos. El checksum también es opcional, aunque generalmente se utiliza en la práctica.

Una de las características más importantes de este protocolo, es que permite transmitir con gran velocidad aunque es muy mencionado que no se puede garantizar el hecho de que lleguen absolutamente todos los paquetes.

Los paquetes enviados a través de la clase *ComunicacionUDP* (Diagrama 11) utilizan un sector del Datagrama UDP correspondiente a los datos, por tanto un mensaje enviado a través de la comunicación UDP no puede exceder los 64kb de datos que son en consecuencia lo que puede contener un datagrama UDP.

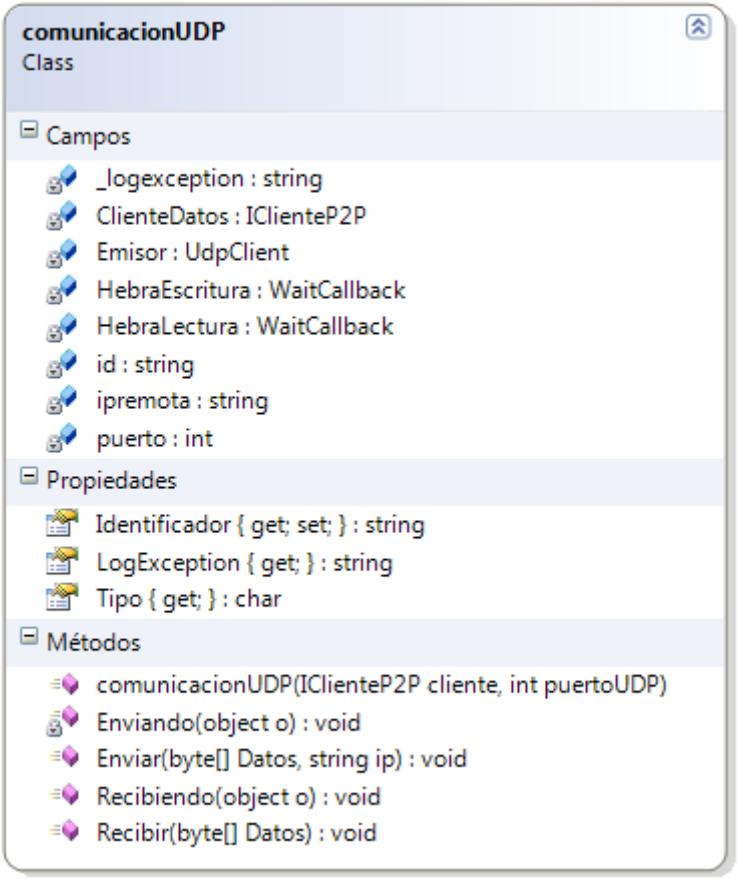


Diagrama 11

Cuando un Cliente dentro del Contenedor de Clientes necesita una nueva comunicacionUDP, es el Sistema de Comunicación quien crea esta ComunicacionUDP y le asigna un identificador (Tabla 10) el cual es transmitido al cliente que la solicito, para que la comunicación pueda ser ubicada de manera única dentro de la lista de comunicaciones UDP (lista doblemente

enlazada) en el Sistema de Comunicación y así poder estar recibiendo los datagramas capturados por el método de ***EscuchaUDP***, la vez que le servirá al Cliente que la solicito (quien la utiliza para comunicarse) para saber cómo referirse a ella y pueda enviar datagramas y procesar los datagramas recibidos.

El formato de este Identificador es el siguiente:

Identificador de Comunicación UDP					
Descripción	Tipo de Datagrama Mensaje o Datos	**:**	Emisor	**:**	Contador
Contenido	M | D	**:**	Nickname del Emisor	**:**	00
Tamaño	1 byte	1 byte	16 bytes	1byte	2 bytes
Tamaño Total de Cabecera de un Mensaje de Comunicación UDP					**19 bytes**

Tabla 10

Por ejemplo: M:Fredy:01, lo que correspondería a la cabecera de un datagrama UDP de tipo Mensaje enviado por Fredy.

4.1.1.2 ComunicacionTCP

La ComunicacionTCP (Diagrama 12) es la clase que permite establecer enlaces para la transmisión de flujos de bytes (Stream) entre dos estaciones de trabajo, utilizando el protocolo TCP.

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. Tales conexiones

pueden existir a través de una serie de conexiones punto a punto, por lo que estas conexiones extremo-extremo son llamadas **circuitos virtuales.**

La creación de estos circuitos virtuales está a cargo de una instancia de la clase *TcpCliente* dentro de la clase ComunicaciónTCP. Al crear una instancia de la clase *TcpClient* se intenta establecer este circuito virtual con un servidor *TCP remoto (identificado por la IP*) que se encuentre escuchando, si no se puede establecer este enlace es imposible crear la instancia, más aún enviar o recibir información por esa vía.

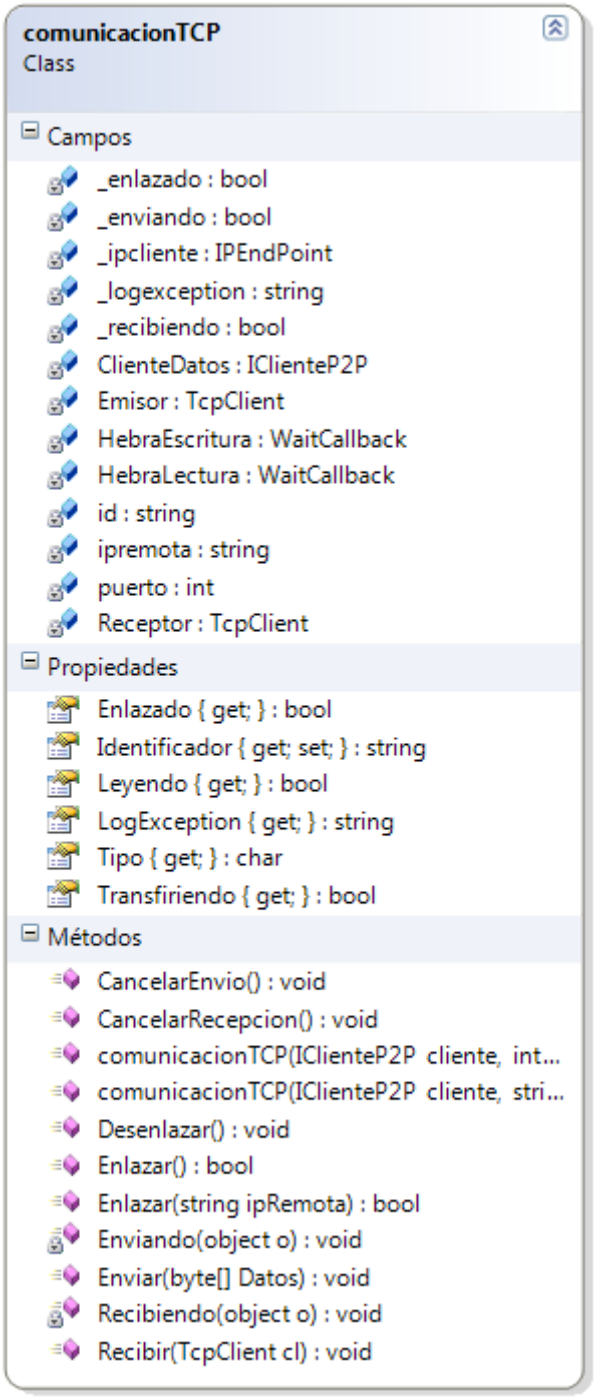


Diagrama 12

Por tanto, es necesario que esta clase posea métodos que permitan establecer o romper este enlace (circuito virtual), además de los métodos que ya se poseían en la comunicación UDP encargados del envió y recepción de datos siempre definidos en hebras en segundo plano.

Una vez establecido el enlace entre dos sockets TCP, el envío y recepción de datos debe ser realizado a través de la escritura y lectura, respectivamente, de un Stream (NetworkStream) que se comparte entre los dos sockets, quien recibe los datos realiza una lectura (destructiva) de ese Stream a medida que un emisor de los datos los está escribiendo en una referencia a este Stream. Por esta razón, la comunicación a través del protocolo TCP es considerada más segura o fiable que la comunicación a través de datagramas UDP, ya que en esta última no es necesario establecer ningún enlace para mandar el mensaje de datos, lo que implica desconocer si el receptor se encuentra preparado para recibir la información o no.

Por estas razones las comunicaciones TCP son utilizadas para clientes en los cuales conocemos al receptor y necesitamos garantizar integridad de la información enviada y saber si algún segmento de información ha llegado incompleto, corrupto o no llego, como es el caso de los clientes ***GestordeTransferencias*** *y* ***GestordeConversaciones*** que se tratarán mas adelante.

La *creación de una ComunicacionTCP está a cargo del sistema de comunicación,* a pesar de ser realizada a petición de **El Contenedor de Clientes.** Cuando un **cliente (clientep2p)** requiere una nueva comunicación TCP el Contendor de Clientes solicita al sistema de comunicación que cree una nueva **comunicación TCP,** *al crearla el sistema de comunicación le asigna un identificador único (Tabla 11)*, que permitirá marcar paquetes enviados a través de esta comunicación TCP, *colocando este identificador en la cabecera de cada flujo de bytes escrito*. De tal forma que el sistema de comunicación que reciba el flujo de bytes sepa a qué comunicación TCP de su lista de comunicaciones le corresponde al procesar estos datos.

El identificador de una comunicación TCP, creada por el sistema de comunicación posee la siguiente estructura.

Identificador de Comunicación TCP							
Descripción	TCP	:	Emisor	:	Receptor	:	Contador
Contenido	TCP	:	Nickname de del Emisor	:	Nickname del Receptor	:	00
Tamaño (bytes)	3	1	16	1	16	1	2
Tamaño Total de Cabecera de un Segmento enviado a través de una Comunicación UDP							40

Tabla 11

A pesar de que la *ComunicaciónTCP*, utiliza estas características de TCP para la transmisión de los datos, se han definido estructuras y tamaños (configurables) para los segmentos (Tabla 12) enviados a través de una comunicación TCP lo que le brinda una confiabilidad mayor, además de la seguridad que brinda el encapsulamiento de la clase que define un bloque de datos enviado.

Un segmento enviado por una comunicación TCP posee la siguiente estructura:

Identificador de Comunicación TCP (40byes)	
Checksum (32 bytes)	**Número de Segmento (3bytes)**
DATOS (restante del tamaño de segmento configurado)	

Tabla 12

Se incluye el identificador de la comunicación TCP que envía el segmento, el checksum de los datos enviados en el segmento para poder verificar integridad de los datos cuando son recibidos por el usuario remoto y un contador de tres caracteres (000) para indicar el numero de paquete enviado) . Finalmente se incluyen los bytes que corresponden a los datos enviados en ese segmento. La longitud de estos datos depende de cómo se ha configurado la transmisión de datos, este aspecto se ha dejado a la opción del cliente y puede ser configurado dentro de SCIP2P entre dos valores 512 kb o 1 Mb (como tamaño del segmento enviado). En cualquiera de los dos casos el segmento correspondiente a los datos, toma todos los bytes del segmento disponibles luego de colocar la cabecera **(Identificador de Comunicación TCP + Checksum + Número de Segmento)**.

4.1.2 Contenedor de clientes

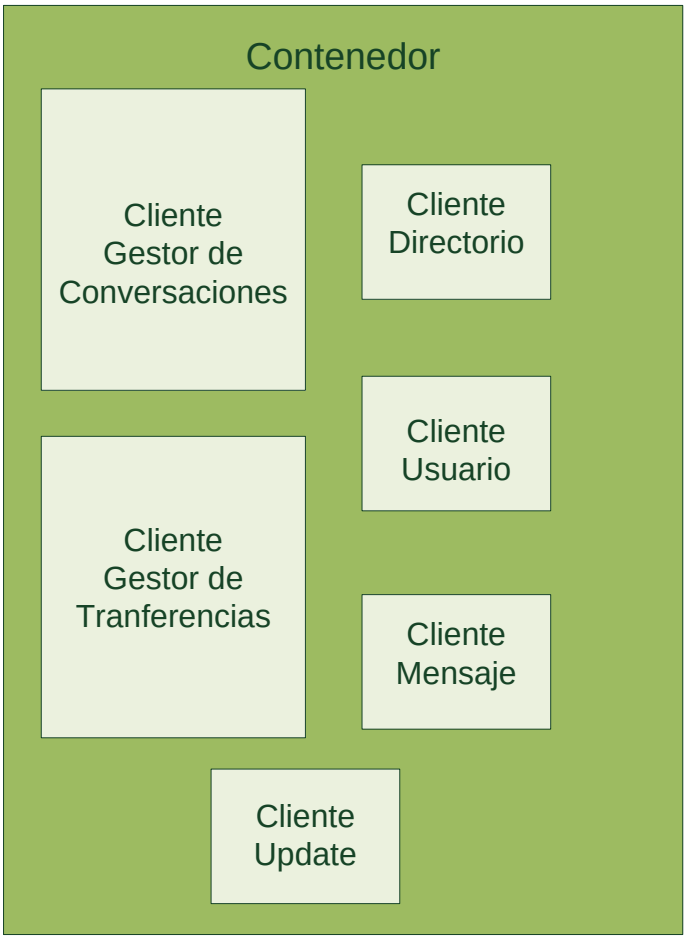


Diagrama 13

Esta clase constituye junto a la clase Sistema de Comunicación las dos entidades principales de SCIP2P. El **Contenedor de Clientes** (Diagrama 13) haciendo alusión a su nombre, *se encarga de contener y administrar los diferentes* ***Clientes (ClienesP2P)*** que permiten establecer diferentes mecanismos de comunicación entre dos usuarios en una misma red. *Para lograr esta función se utiliza una lista doblemente enlazad*a, donde se almacenan o recuerdan cada uno de estos clientes. Sin embargo, el contenedor de clientes posee referencias a los ***Clientes*** definidos como básicos (*ClienteUsuario, ClienteDirectorio, Cliente Mensajes, ClienteUpdate, GestordeTransferencias, GestordeConversaciones).*

Cada vez que un cliente es agregado *al Contendor de Clientes, este último es quien le asigna un identificador (Tabla 9)* , que permite ubicar dentro de la lista clientes, al cliente que se desea encontrar, para cualquiera de sus tareas de envió o recepción de datos, incluso el procesamiento del mismo.

Este identificador está constituido por una cadena String, que posee la estructura siguiente:

Identificador de ClienteP2P		
Descripción	Tipo Cliente	Contador
Contenido	**Tres Caracteres Identifican el tipo de clientep2p.** ✓ Cliente Usuario USU ✓ Cliente de Mensajes MSJ ✓ Cliente Directorio DIR ✓ Cliente Gestor de Conversaciones. GCO o Cliente Chat. CHT ✓ Cliente Gestor de Transferencias. GTA o Cliente Subida de Archivo. SUA o Cliente Descarga de Archivo. DEA ✓ Cliente Update. UPD	**Dos dígitos** 00
Tamaño (bytes)	3 bytes	2 bytes
Tamaño Total de Identificador de Cliente P2P		5 bytes

Tabla 13

El tipo de cliente hace referencia a la enumeración **TipoClienteP2P** (Diagrama 14) dentro del **Contenedor de Clientes.**

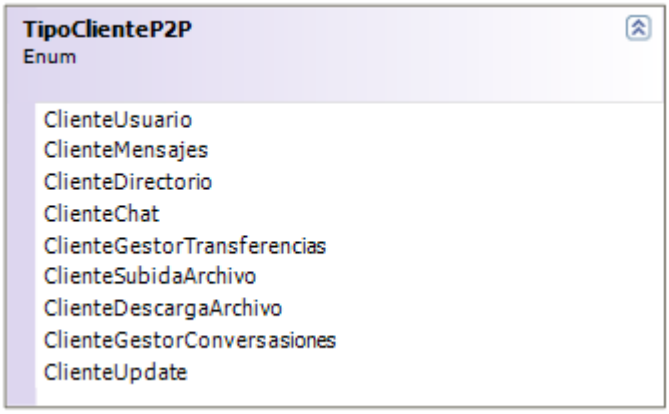


Diagrama 14

Un aspecto muy importante es que la clase **Contenedor de Clientes** (Diagrama 15) posee una referencia al **Sistema de Comunicación,** lo que le permite solicitar sus servicios en la creación, administración y finalización de las comunicaciones TCP o UDP necesarias o vinculadas al **cliente de datos** en el Contenedor de Clientes.

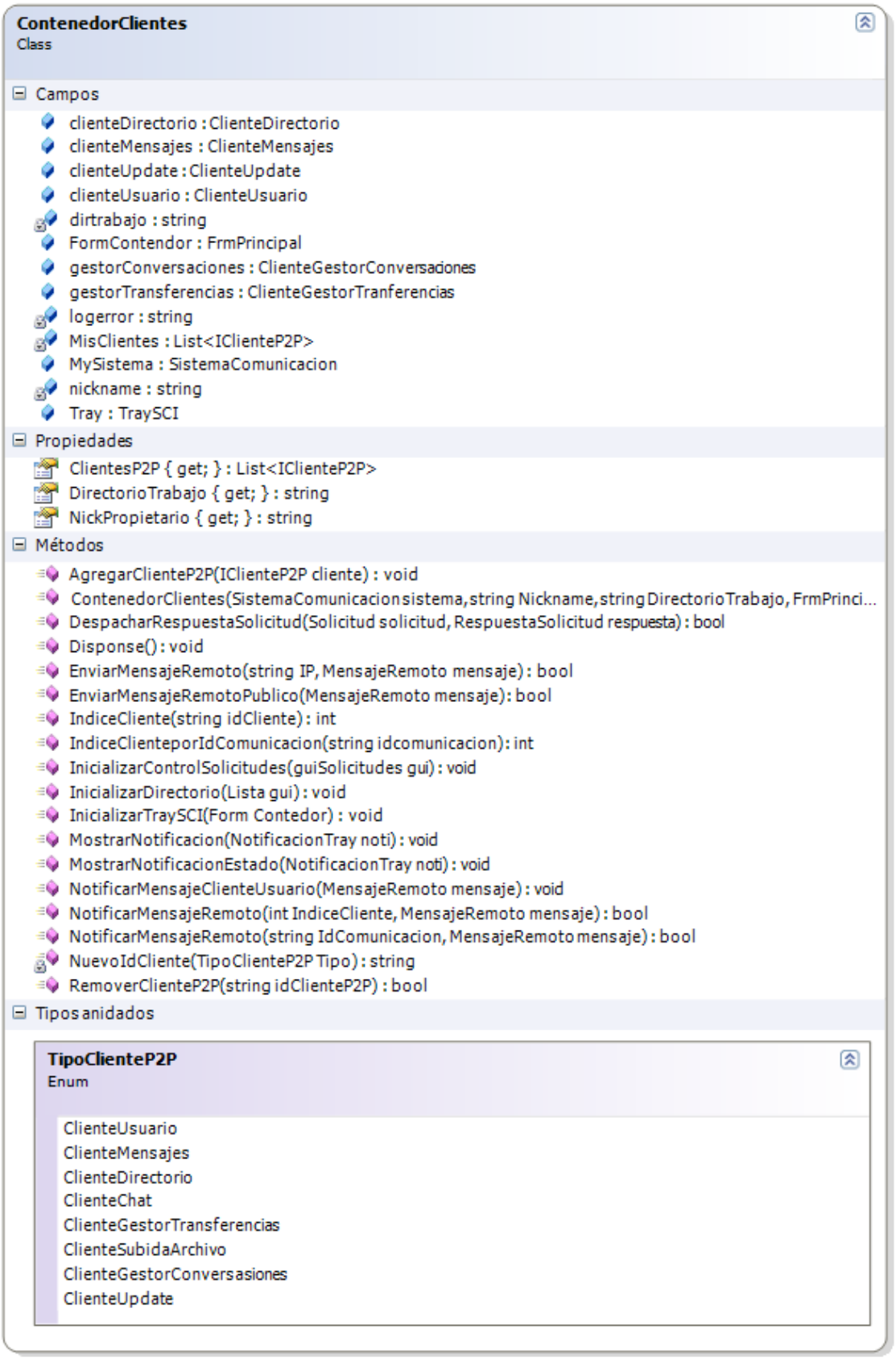

ContenedorClientes
Class
Campos
clienteDirectorio : ClienteDirectorio
clienteMensajes : ClienteMensajes
clienteUpdate : ClienteUpdate
clienteUsuario : ClienteUsuario
dirtrabajo : string
FormContendor : FrmPrincipal
gestorConversaciones : ClienteGestorConversaciones
gestorTransferencias : ClienteGestorTranferencias
logerror : string
MisClientes : List<IClienteP2P>
MySistema : SistemaComunicacion
nickname : string
Tray : TraySCI
Propiedades
ClientesP2P { get; } : List<IClienteP2P>
DirectorioTrabajo { get; } : string
NickPropietario { get; } : string
Métodos
AgregarClienteP2P(IClienteP2P cliente) : void
ContenedorClientes(SistemaComunicacion sistema, string Nickname, string DirectorioTrabajo, FrmPrinci...
DespacharRespuestaSolicitud(Solicitud solicitud, RespuestaSolicitud respuesta) : bool
Disponse() : void
EnviarMensajeRemoto(string IP, MensajeRemoto mensaje) : bool
EnviarMensajeRemotoPublico(MensajeRemoto mensaje) : bool
IndiceCliente(string idCliente) : int
IndiceClienteporIdComunicacion(string idcomunicacion) : int
InicializarControlSolicitudes(guiSolicitudes gui) : void
InicializarDirectorio(Lista gui) : void
InicializarTraySCI(Form Contedor) : void
MostrarNotificacion(NotificacionTray noti) : void
MostrarNotificacionEstado(NotificacionTray noti) : void
NotificarMensajeClienteUsuario(MensajeRemoto mensaje) : void
NotificarMensajeRemoto(int IndiceCliente, MensajeRemoto mensaje) : bool
NotificarMensajeRemoto(string IdComunicacion, MensajeRemoto mensaje) : bool
NuevoIdCliente(TipoClienteP2P Tipo) : string
RemoverClienteP2P(string idClienteP2P) : bool
Tipos anidados
TipoClienteP2P
Enum
ClienteUsuario
ClienteMensajes
ClienteDirectorio
ClienteChat
ClienteGestorTransferencias
ClienteSubidaArchivo
ClienteGestorConversasiones
ClienteUpdate

Diagrama 15

4.1.2.1 TraySCI

Dentro del Contenedor de Clientes se posee una instancia de la clase TraySCI (Diagrama 16) la cual *permite la utilización de un icono para la aplicación en la barra de herramientas de Windows junto a la fecha y hora del sistema* (Trayicon) (Diagrama 17), para indicar los momentos en que el sistema de comunicación se encuentra activo.

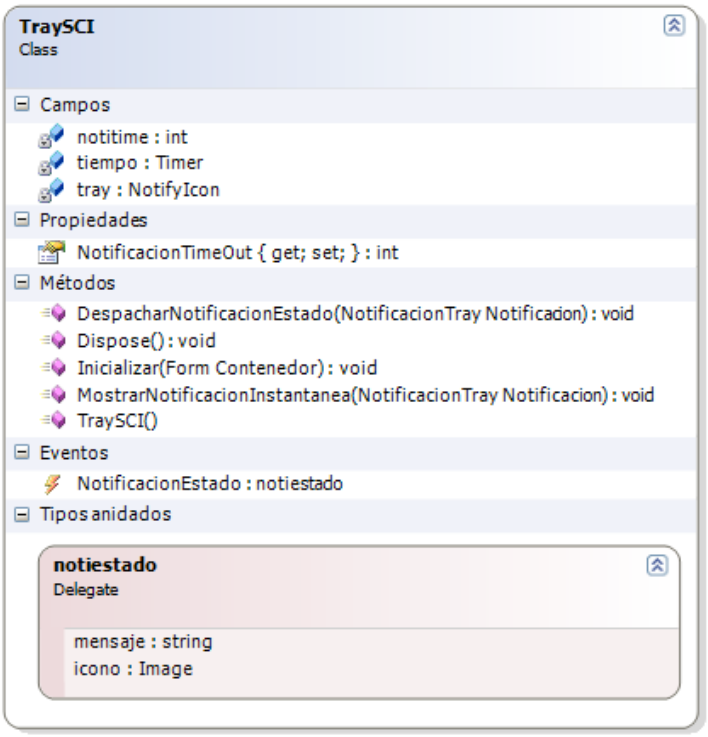


Diagrama 16

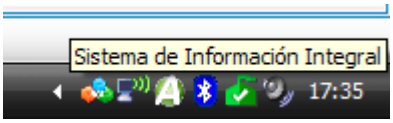


Diagrama 17

Además esta clase permite desplegar mensajes de notificación **NotificaciónTray** (Diagrama 18) para conocimiento del usuario de la estación de trabajo SCI, estas notificaciones son un servicio que presta el Contenedor de Clientes a todos los clientesP2P, cuando estos desean desplegar alguna notificación para el conocimiento del usuario del sistema.

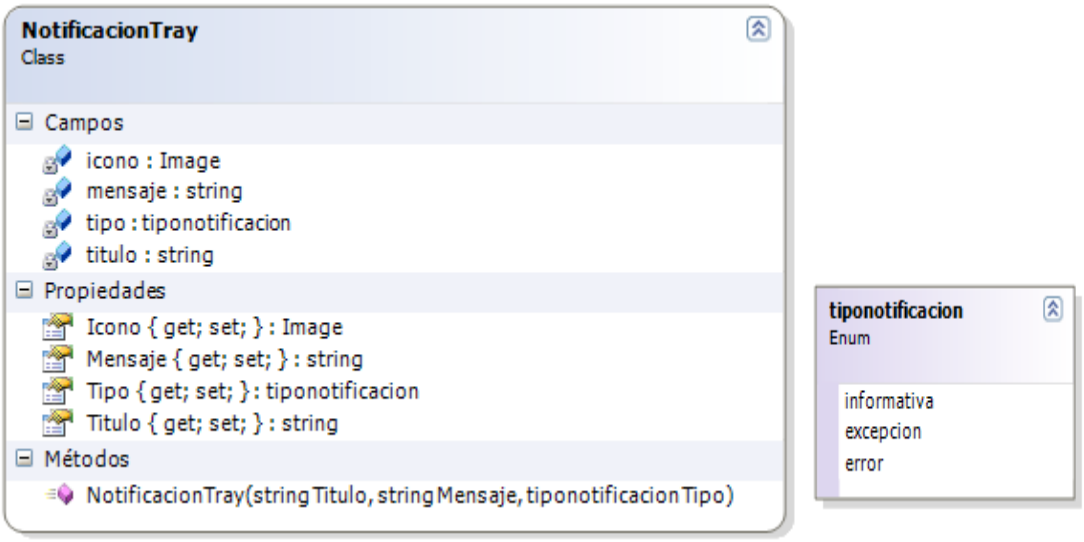


Diagrama 18

Cuando un cliente desea desplegar una notificación crea una nueva **NotificacionTray** definimiento el mensaje a desplegar, el icono a utilizar y el tipo de notificación (informativa, excepción, error) y solicita al Contenedor de Clientes que despliegue esa notificación, enviándole a él, la instancia de la clase **NotificacionTray.**

- ¿Qué es un clienteP2P o cliente de datos?

Un cliente P2P lo constituyen todas aquellas clases que implementan la interfaz **IClienteP2P** (Diagrama 19)**,** al implementar esta interfaz las clases se ven obligadas a definir los miembros citados en esta interfaz. Esto permite establecer una relación (fuera de la herencia) lo que permite de cierta forma referirse a un clientep2p llamado también cliente de datos sin importar su propósito específico, mensajes, archivos, etc.

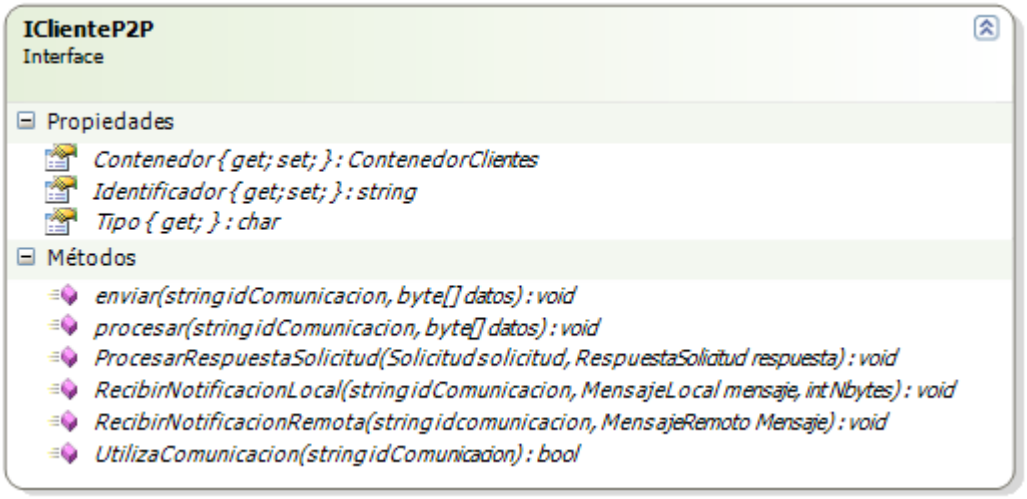


Diagrama 19

Esto a la vez ayuda a definir un estándar para los **clientesp2p**, ya que al implementar esta interfaz, sin importar el propósito del cliente, el Contenedor de Clientes y el Sistema de comunicación sabrán como referirse a ellos para enviar datos o procesar datos recibidos, por una comunicación TCP o UDP en el Sistema de Comunicación asociada al clienteP2P, incluso se determinan los métodos para procesar **Notificaciones Locales** (Entre el Sistema de Comunicación y el ClienteP2P) y **Notificaciones Remotas** (Entre el clienteP2P local y su homologo en una estación de trabajo SCIP2P remota).

La lista doblemente enlazada administrada por el Contendor de Clientes no es más que una lista de objetos **IClienteP2P,** es decir instancias de clases que implementan esta interfaz.

Como se puede apreciar la estructura de la interfaz, espera que todo cliente que la implemente se vea obligado a poseer una referencia al contenedor de clientes al cual pertenece, ya que muchas ocasiones un clienteP2P puede necesitar de servicios que brinda otro cliente, pero sin conocerlo, entonces cada cliente solicita al **Contenedor de Clientes** que gestione esta solicitud. Es decir que todos los clientes administrados por el contenedor implementan esta interfaz.

4.1.2.2 Cliente Usuario

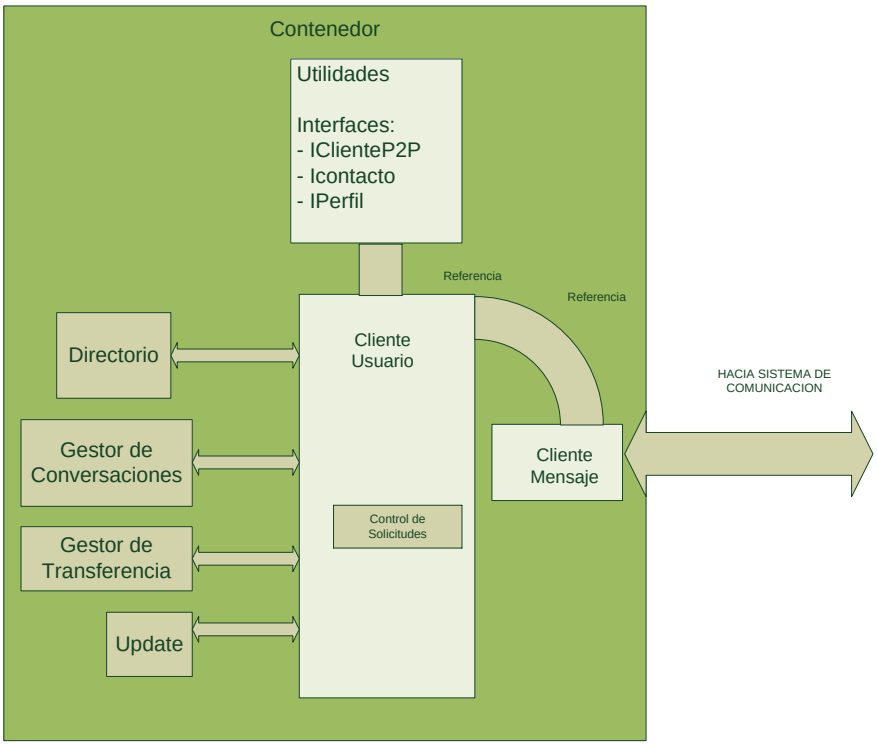


Diagrama 20

*El **Cliente Usuario*** (Diagrama 20) permite representar al usuario de la estación de trabajo SCIP2P en el **Contenedor de Clientes**. Como se mencionaba anteriormente todos los clientes implementa la interfaz IClienteP2P, pero este cliente en particular implementa dos interfaces más **IContacto, IPerfil** (Diagrama 21)**.**

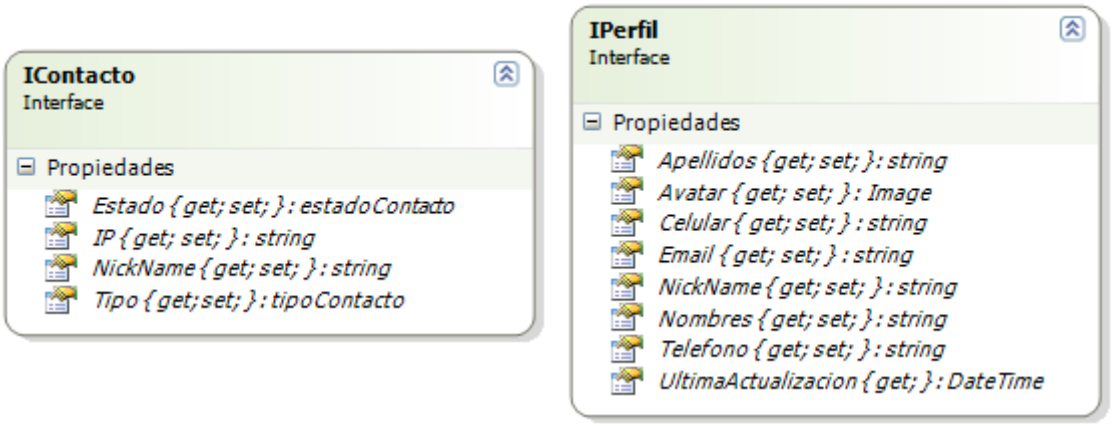


Diagrama 21

Por esta razón la clase Cliente Usuario (Diagrama 22) se ve obligado a **Contacto** o **Perfil de Contacto**, o sea, miembros que permitan identificar y comunicarse con un contacto en la red, como un **nickname,** dirección **IP**, o datos personales incluidos en un perfil de usuario como nombres, apellidos, teléfonos, foto, etc. De esta forma el Cliente Usuario es capaz de ser visto como un contacto dentro de un **Directorio de Contactos Remoto.**

Además, otra tarea importante del **Cliente Usuario** es, interactuar con todas las solicitudes y notificaciones, que desencadenan los demás clientes P2P, que requieren la intervención del usuario de la estación de trabajo, para aceptar o denegar alguna solicitud por lo general remota. Por tanto el Cliente Usuario posee una instancia de la clase **ControlSolicitudes**[1].

[1] El **ControlSolicitudes** se explica con mayor detalle en una sección posterior.

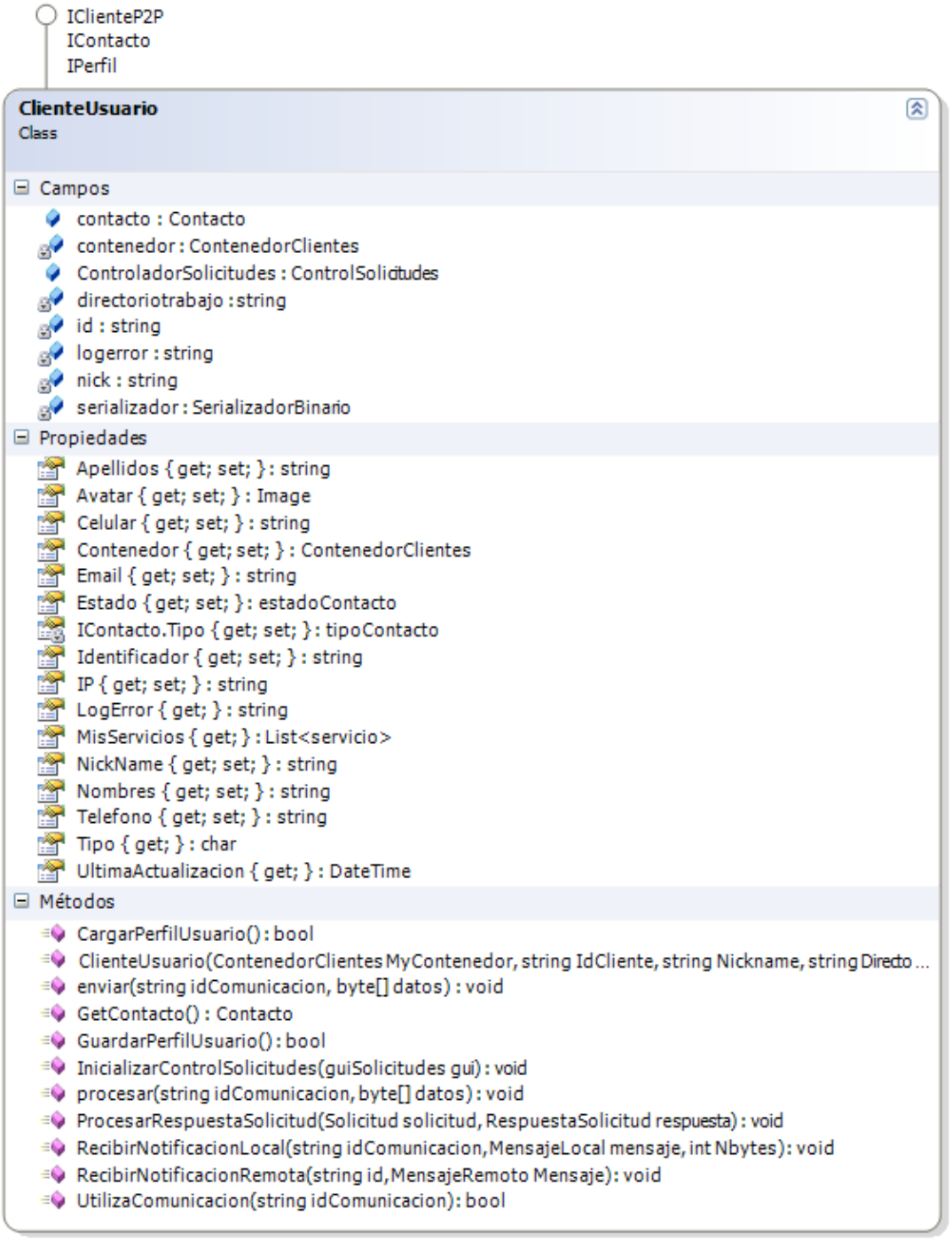


Diagrama 22

Es muy importante resaltar que el **ClienteUsuario,** no posee ninguna comunicación TCP o UDP asociada que le permita enviar o recibir datos directamente, casi todo envió y recepción de datos en este cliente se realiza a través del **Cliente de Mensajes**, cliente que explicaremos más adelante.

- Control de Solicitudes

La clase ControlSolicitudes (Diagrama 23), es la encargado de administrar todas las solicitudes enviadas y recibidas, por y desde cualquiera de los clientes en el **Contendor de Clientes**, a diferencia de las notificaciones administradas por la clase **TraySCI**, las solicitudes tienen la característica que requiere de una respuesta, por lo tanto requieren de la intervención del usuario de la estación de trabajo **SCIP2P.**

Todas las clases utilizadas para el **Control de Solicitudes** se encuentran en el espacio de nombre **P2P.Solicitudes** de nuestro proyecto. **ControlSolicitudes, Solicitud**, **PersistenciaSolicitudes, guiSolicitudes, Notifyform y DespachoPendiente (estas Clases se explican más adelante).**

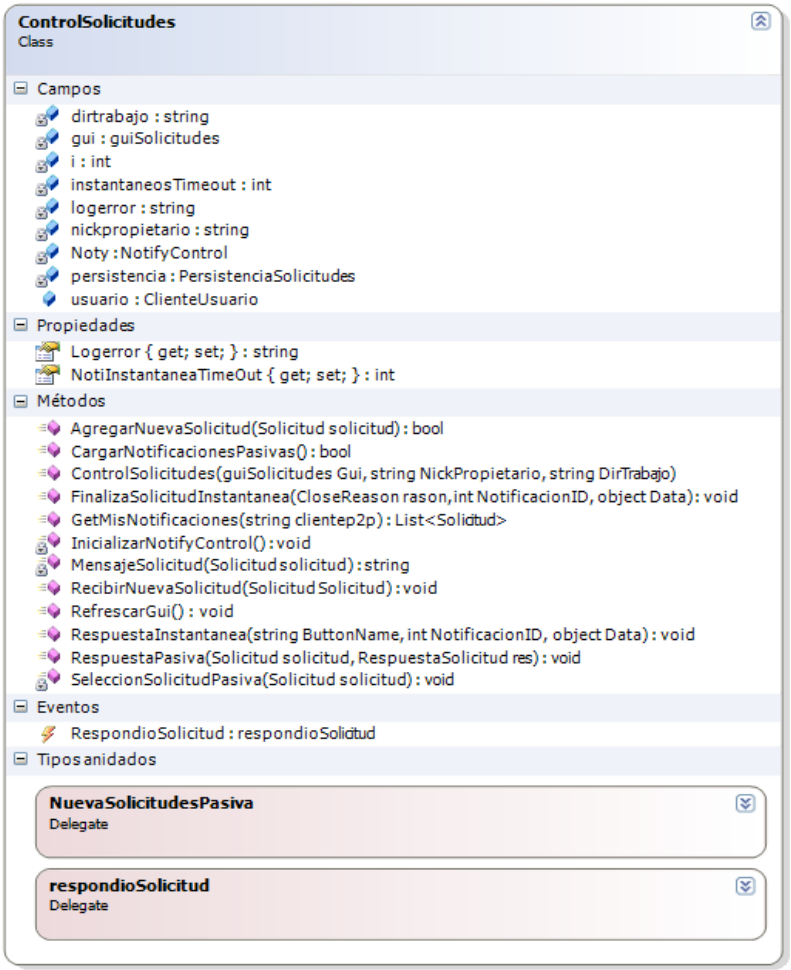


Diagrama 23

El **Control de Solicitudes** instanciado en el **Cliente Usuario**, brinda sus servicios a todos los clientes del contenedor de clientes, así, cuando un **Cliente** recibe una solicitud para transmitir, recibir o sincronizar información de su homologo remoto, dicha solicitud requiera la intervención del usuario para poder decidir como procesarla y dar respuesta a esta solicitud.

El **ControlSolicitudes** funciona como *el motor en el procesamiento de solicitudes,* posee referencias a las demás instancias que dan soporte a las solicitudes, para crear nuevas solicitudes, desplegarlas a través de las interfaces adecuadas, guardarlas y transmitir la respuesta de la solicitud al Cliente Usuario, para que luego el determine a que Cliente, le transmitirá la repuesta a la solicitud para que sea procesada.

Funcionamiento:

El cliente que recibe este mensaje proveniente de un cliente remoto, como primer paso determina que el mensaje recibido, es una solicitud y que requerirá de la intervención del usuario del sistema, es entonces cuando crea una nueva instancia de la clase **Solicitud** (Diagrama 24)**.**

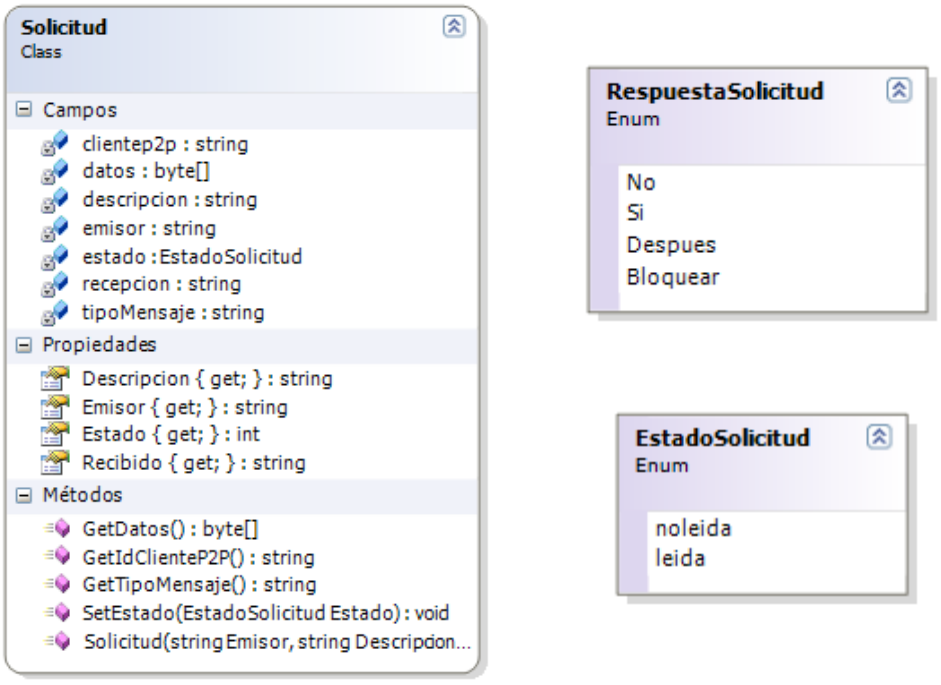


Diagrama 24

Una instancia de la clase **Solicitud**, además de contener toda aquella información necesaria para desplegar el mensaje al usuario, posee datos que hacen referencia directa al **Mensaje Remoto**[2] a partir de cual se ha creado, el identificador del cliente que ha creado la solicitud, el nickname del usuario que envió el **Mensaje Remoto** predecesor a la solicitud, las datos asociados, etc. esto permite que una vez capturada la respuesta a la solicitud, sea posible procesar o responder al cliente remoto de acuerdo al tipo de **MensajeRemoto** recibido (Diagrama 25).

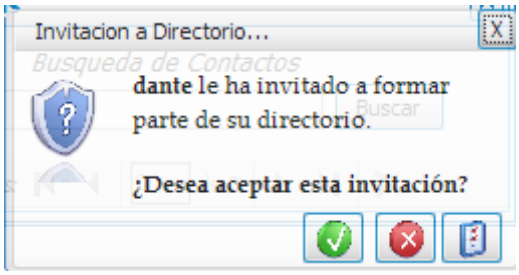


Diagrama 25

Una vez creada la instancia de la clase **Solicitud** cada cliente, envía esta nueva solicitud al **Cliente Usuario** para que, a través de su campo de ***ControlSolicitudes*** despliegue la solicitud, en forma de un mensaje emergente en la parte inferior derecha de la de la pantalla, solicitando una respuesta del usuario. Sin embargo, si después de pasados cinco segundos esta ventana emergente desaparece y la solicitud pasará a ser una **Solicitud Pendiente**.

*Las **Solicitudes Pendientes***, son de dos tipos solicitudes

- ✓ Perecederas
- ✓ Imperecederas

Esto de acuerdo al tipo de solicitud recibida, estás pueden ser almacenadas como una **Solicitud Pendiente** en el archivo de datos que contiene el **Directorio de Contactos**[3] asociado al Cliente Usuario que está utilizando la estación de trabajo SCIP2P. De esta forma estas solicitudes

[2] **MensajeRemoto**: clase en la cual se encapsulan los mensajes recibidos y que es tratado por el **Cliente de Mensajes,** este cliente se explica con detalle posteriormente.
[3] EL **Directorio de Contactos,** hace referencia al **Cliente Directorio** explicado posteriormente.

pueden ser respondidas en cualquier momento incluso sesiones diferentes, ya que toda la información necesaria para recrear la solicitud en cualquier momento queda almacenada en disco.

En esta parte entra el papel de la clase **PersistenciaSolicitudes,** encargada de gestionar todo lo referente guardar y recuperar Solicitudes Pendientes, internamente del archivo que contiene el **Directorio de Contactos**, las **Solicitudes y Despachos Pendientes**[4] trabaja con una base de datos **SQLite**[5]**.**

La clase **PersistenciaSolicitudes** (Diagrama 26)**,** utiliza una instancia de la clase **ConexionSQLite**[6]**,** para realizar todas las tareas de guardar, buscar, eliminar o cargar **Solicitudes y Despachos Pendientes**.

[4] Los **Despachos Pendientes** se generan cuando se responde una solicitud y el contacto remoto a quien debe enviarse la respuesta esta desconectado, se explica con mayor detalle en una sección adelante.
[5] En la sesión, se detallan las características de las base de datos **SQLite**.
[6] El detalle de las clases de Acceso a Datos se encuentra en la sección de **Clases Utilitarias.**

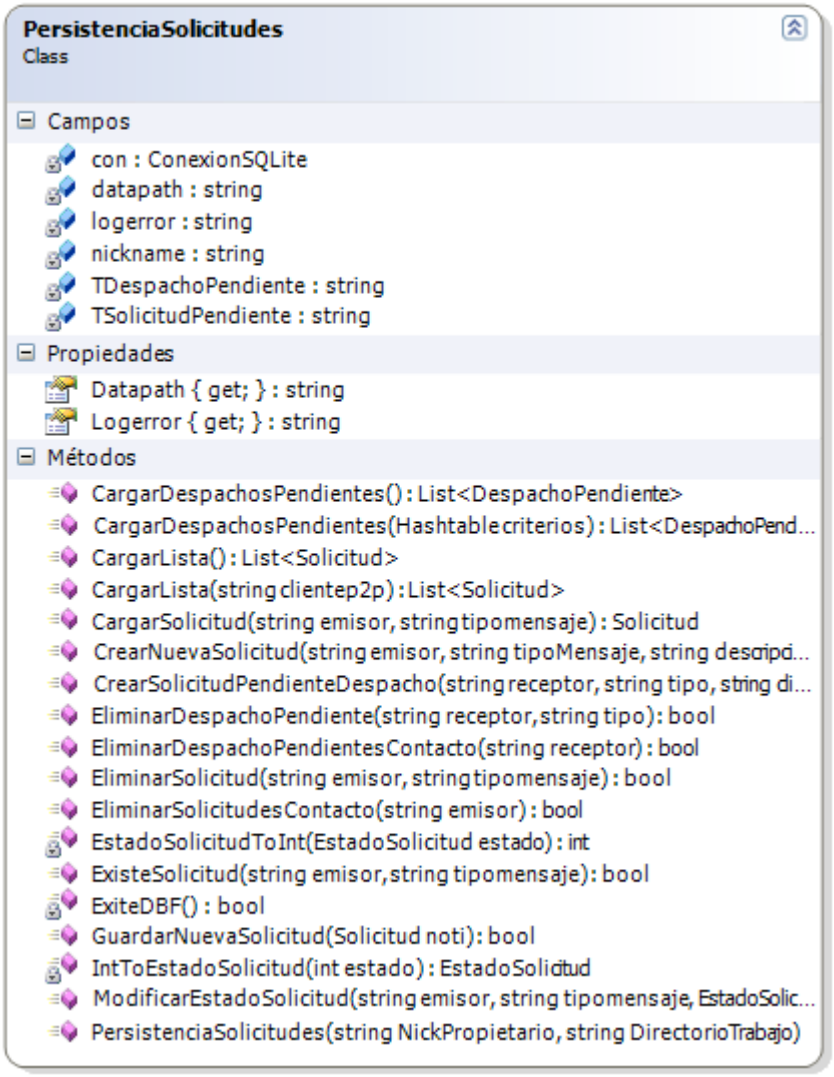


Diagrama 26

En el achivo **.dir** (archivo que contiene toda la informacion de un contacto) del SCIP2P, existe la tabla **SolicitudesPendientes** (Tabla 14), donde se almacenan las solicitudes hasta el momento en que son respondidas y procesadas. La tabla presenta la siguiente estructura:

SolicitudesPendientes			
Campo	Tipo	Llave	Requerida
emisor	Varchar (16)	Primaria	Si
tipomj	Varchar (4)	Primaria	Si
descripcion	Varchar (100)		
clientep2p	Varchar (5)		
estado	Integer		
recepción	Varchar (50)		
datos	BLOB		

Tabla 14

Se almacena el nickname del contacto emisor , tipo descripcion y datos del **Mensaje Remoto**, que desencadena la creación de la **Solicitud** en primera instancia , todo con el fin que la solicitud pueda ser recreada a partir de los datos almacenados y ser respondida y procesada en el momento deseado.

La clase **ControlSolicitudes** muestra la lista de solicitudes pendientes a través de una instancia de la interfaz de usuario **guiSolicitudes** (Diagrama 27). Esta interfaz (Diagrama 28) permite mostrar el listado de las solicitudes pendientes. Dando doble click sobre alguna de las solicitudes pendientes. En el **guiSolicitudes** se genera nuevamente el mensaje de solicitud que nos permite responder a la solicitud del contacto remoto.

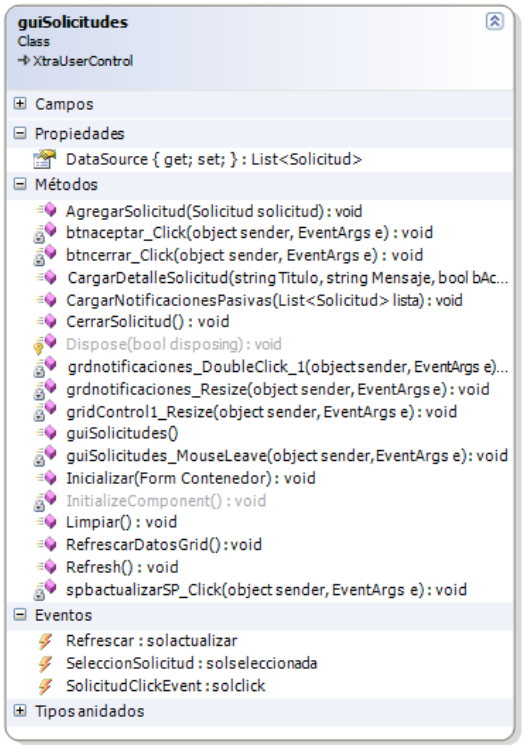


Diagrama 27

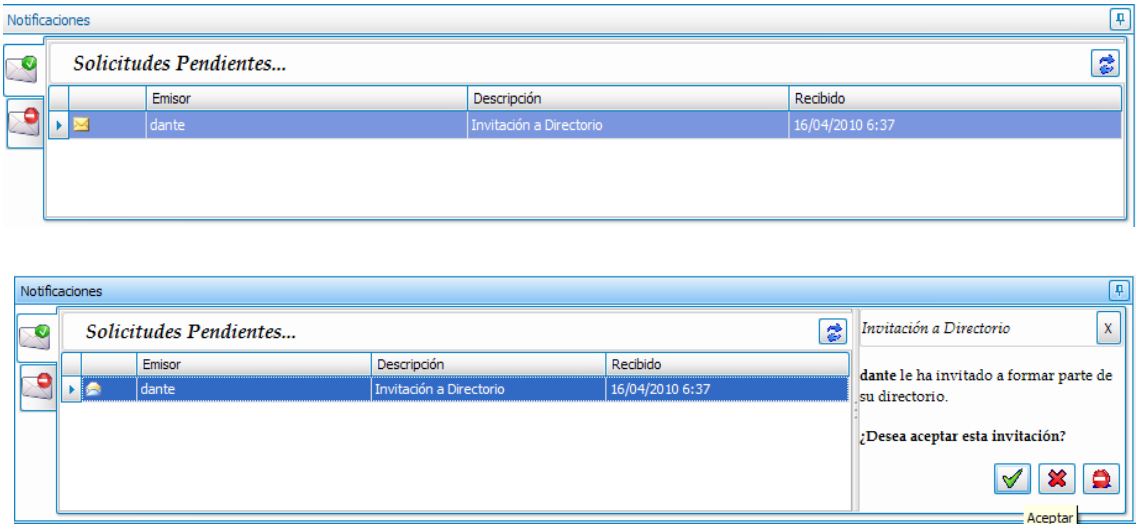


Diagrama 28

Cada vez que se responde a una solicitud dando click en Si o No de alguna de las solicitudes, ya sea en el mensaje emergente que se genera al momento de crear la solicitud o a través de las solicitudes pendientes en la **guiSolicitudes,** esta respuesta afirmativa o negativa es capturada por el **Control de Solicitudes** quien lo notifica al **Cliente Usuario** el cliente usuario es quien se encarga de enviar esa respuesta de solicitud al Cliente correspondiente. Es decir el cliente que inicialmente recibió el Mensaje Remoto y obligo a la creación de la **Solicitud**, ya que solamente este cliente sabe como deberá procesar la respuesta a este tipo de solicitud.

Esta comunicación entre los clientes se hace a través del método de la interfaz **IClienteP2P** llamado

ProcesarRespuestaSolicitud(*Solicitud solicitud, RespuestaSolicitud respuesta****)***

4.1.2.3 Cliente de Mensaje

La clase **Cliente de Mensajes** (Diagrama 29) es uno de los clientes más importantes y utilizados dentro del Contenedor de Clientes y la estación de trabajo SCIP2P en general. Ya que este cliente se encarga del envió, recepción y distribución **Mensajes Remotos**[7] desde y hacia todos los clientes dentro del contenedor de clientes. Cualquiera de los clientes que desee enviar un **Mensaje Remoto,** a su homónimo[8] remoto, lo hace a través del **Cliente de Mensajes.**

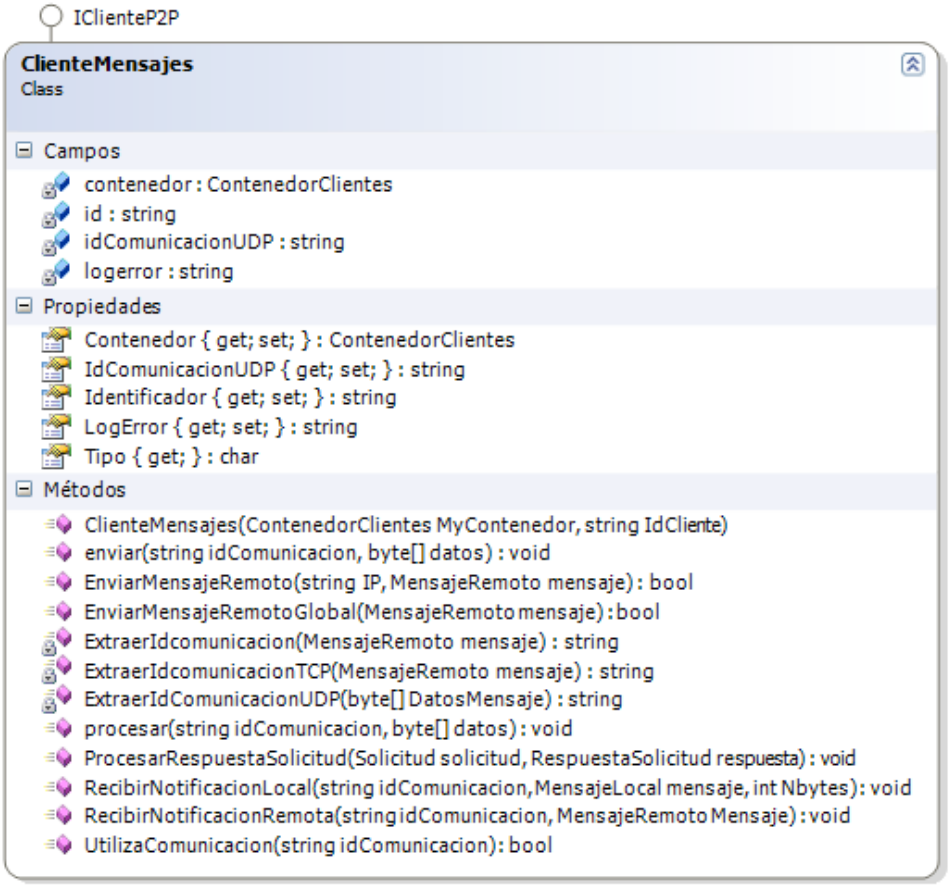


Diagrama 29

[7] La clase **MensajeRemoto** se explica con detalle en el apartado siguiente.
[8] La comunicación entre dos clientes remotos se realiza entre clientes del mismo tipo.

Al momento de crearlo en **Contenedor de Clientes** solicita al Sistema de Comunicación que cree una nueva **Comunicación UDP** asociada al **Cliente de Mensajes** que se ha creado, esta comunicación UDP le brinda soporte al cliente para poder realizar sus tareas de envió y recepción de mensajes remotos que será realizada a través de **Datagramas UDP**.

- ¿Qué es un Mensaje Remoto?

Los **Mensajes Remotos** son instancias de la clase **MensajeRemoto** (Diagrama 30), la cual se ha definido como la unidad de envió y recepción de mensajes a través del **Cliente de Mensajes**, todo cliente que desea enviar un Mensaje Remoto a su homónimo en otra computadora, creará una instancia de la clase **MensajeRemoto** y la enviará por el **Cliente de Mensajes**, todo mensaje remoto recibido desde el Cliente de Mensajes proveniente de un cliente en otra computadora será una instancia de la misma clase.

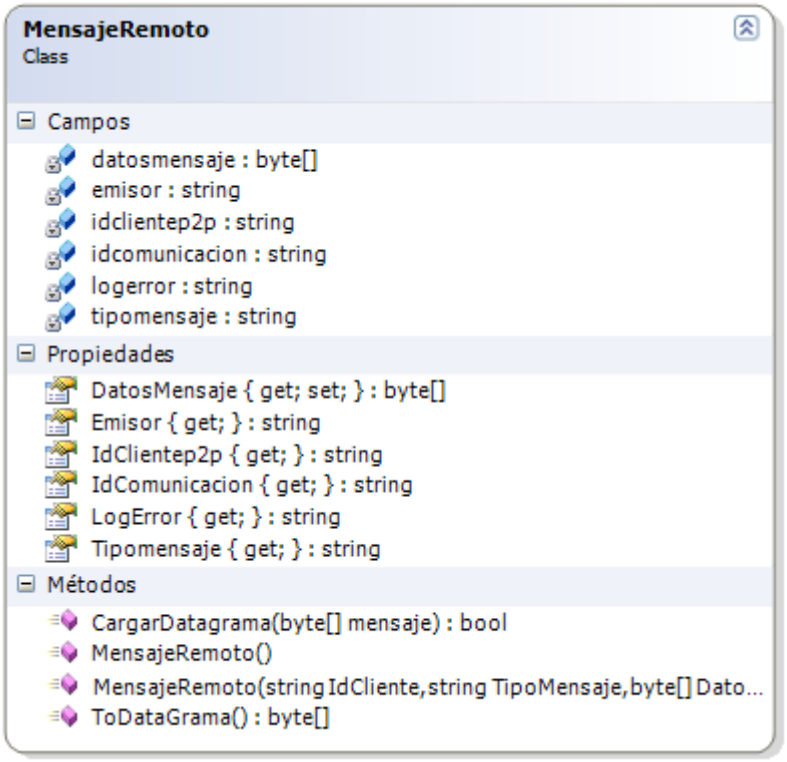


Diagrama 30

La clase **MensajeRemoto** posee diferentes propiedades algunas como *Emisor e IdComunicación,* las cuales están orientadas a garantizar la entrega y posible respuesta del mensaje remoto, así como otras propiedades como *Tipomensaje y DatosMensaje* que son propias del tipo de mensaje y datos que contiene, para ser procesados por el cliente Receptor del Mensaje.

Los mensajes remotos son recibidos desde la Comunicación UDP como un datagrama para poder crear un mensaje remoto a partir del mismo para esto es necesario conocer con que estructura se envía el mensaje remoto en un Datagrama UDP (Tabla 15).

Estructura de un Mensaje Remoto en un Datagrama UDP

Elemento	Descripción	Tamaño
IdComunicacionUDP	Identificador de la **Comunicación UDP**, designada al **Cliente Mensajes** al momento de ser creado.	21 bytes
IdClienteP2P	Código que identifica a cada cliente de manera única, dentro del **Contendor de Clientes.**	5 bytes
TipoMensaje	Conjunto de 4 caracteres que permiten identificar cual es el propósito del mensaje dentro del Cliente al que es enviado.	4 bytes
Datos	Conjunto de bytes restantes en el datagrama UDP, dispuestos para el envió de información	Restantes de 64Kb

Tabla 15

Lo más determinante en un **Mensaje Remoto** son los cuatro caracteres que hacen referencia al **Tipo de Mensaje**, ya que gracias a este el Cliente que recibe el mensaje remoto, sabe como procesar el mensaje, como deserializar los datos que puedan venir adjuntos al mensaje remoto.

Cada Cliente posee su propio protocolo de entendimiento con clientes del mismo tipo, cada uno posee un listado o enumeración de los diferentes tipos de mensajes que puede esperar de su cliente homónimo y los tipos de mensaje respuesta que podría despachar en cada caso. Así mismo, el objeto serializado adjunto en el bloque de los datos dependerá del tipo de mensaje enviado entre los clientes.

- Recepción de Mensajes Remotos a través del Cliente de Mensajes.

En el momento que los métodos de escucha del Sistema de Comunicación determinen que se ha recibido Datagrama UDP, que pertenece a la comunicación UDP asociada al cliente de mensajes, la comunicación transmitirá los datagramas leídos al Cliente de Mensajes para que este procese los bytes y genere una instancia de la clase **Mensaje Remoto**, luego el cliente de mensajes verificara hacia cual cliente va dirigido este mensaje recién generado y lo despacha al cliente respectivo.

Cualquier cliente es capaz de recibir un **Mensaje Remoto**, tarea que realizan a través del método declarado en la interfaz IClienteP2P, que por supuesto es implementada por Cliente de Mensajes.

RecibirNotificacionRemota(*String idComunicación, MensajeRemoto Mensaje*)

- Envío de Mensajes Remotos a través del Cliente de Mensajes.

Cualquier cliente, dentro del **Contenedor de Clientes** que desee enviar un mensaje a través del cliente de mensajes, creara una nueva instancia de la clase **MensajeRemoto**, encapsulando en la sección de datos el objeto (serializado) que desea enviar. Posteriormente, solicitará al Contenedor de Clientes el envió de un mensaje remoto, el contenedor de clientes le delega esta tarea al Cliente de Mensajes, por lo que este (El cliente de Mensaje) toma el mensaje remoto, lo transforma a un Datagrama UDP y lo envía a través de la comunicación UDP asociada al cliente.

Todos los clientes hacen uso del Cliente de Mensajes, desde clientes que funcionan totalmente a través de Mensajes Remotos como el caso del **Cliente Directorio** [9] o clientes TCP[10] que utilizan los mensajes remotos como una comunicación previa a la transmisión o como mensajes de control, como es el caso del **Gestor de Transferencia de Archivos[11].**

4.1.2.4 Cliente Directorio

También conocido como **Directorio de Contactos**, es el encargado de administrar todo lo referente a **Contactos[12], Perfiles, Etiquetas entendidas como grupos privados**, **Grupos Públicos[13]**, etc.

Además, la clase **Cliente Directorio** (Diagrama 32) brinda servicios a los demás clientes dentro del **Contenedor de Clientes**, en el momento en que cualquier cliente desee enviar información o establecer comunicación con un usuario; el cliente directorio es quien posee la lista de los usuarios que están

[9] **Cliente Directorio:** Es un clientep2p encargado de administrar los contactos, grupos y conectividad de los mismos, este cliente se desarrollara con detalle en la sección siguiente.
[10] **Clientes TCP**: clientesp2p que utilizan comunicaciones TCP para transferencia de datos.
[11] **Gestor de Transferencias de Archivos:** cliente que utiliza comunicaciones TCP para enviar y recibir trasferencias de archivos.
[12] **Contacto:** La clase contacto permite representar un usuario remoto que también utiliza SCIP2P.
[13] **Grupo Público**: Agrupación de contactos publica donde existe un anfitrión y uno o más invitados.

asociados a un Directorio Privado (**Etiqueta**[14]) o a un Grupo Público, y mantiene información actualizada sobre el estado de conexión de cada **Contacto**; cuando cualquier cliente desea que comunicarse con un Contacto remoto, solicita al cliente directorio una referencia a la información del contacto (nickname, IP, etc) para poder comunicarse con él.

El **Cliente Directorio** posee la peculiaridad que al ser creado en el Contenedor de Clientes, no se le asocia ninguna Comunicación del Sistema de Comunicación; debido a que todas las actividades de notificación y sincronización de información, realizadas por este cliente, son realizadas totalmente a través de **Mensajes Remotos** enviados a través del **Cliente de Mensajes.** La Clase **ClienteDirectorio**, al igual que todos los clientes, implementa la interfaz ICliente, por tanto se ve obligada a implementar todos los métodos de dicha interfaz, aunque en su mayor parte utilice la recepción de **Mensajes Remotos.**

La Clase **ClienteDirectorio** (Diagrama 32) es muy extensa, al igual que la gran cantidad de funciones y actividades que realiza para cumplir su papel, en el **Contenedor de Clientes**; las tareas que realiza este cliente, pueden dividirse en tareas de mantenimiento de **Contactos, Etiquetas, Grupos** y tareas de **Comunicación.**

[14] La Clase **Etiqueta** hace referencia a una agrupación de **Contactos** personal.


ClienteDirectorio
Class
Campos
agenda : List<Contacto>
ColaIngresos : List<IngresoContacto>
coleciones : List<GrupoDirectorio>
contactos : List<Contacto>
dirtrabajo : string
frmagregar : AgregarContacto
gruposbusqueda : List<GrupoDirectorio>
gui : Lista
id : string
InicioSesion : DateTime
logerror : string
mensajeupdate : bool
mycontetedor : ContenedorClientes
nickpropietario : string
persistencia : PersistenciaDirectorio
persistenciasolicitud : PersistenciaSolidtudes
serializador : SerializadorBinario
temporizador : Timer
temporizadorG : Timer
Propiedades
Contenedor { get; set; } : ContenedorClientes
DirectorioTrabajo { get; } : string
Identificador { get; set; } : string
Logerror { get; } : string
Tipo { get; } : char
Métodos
AceptarContacto(AgregarContacto Frm, string nickname, List<string> Colecciones) :...
AceptarInvitacionDirectorio(string nickname) : void
ActualizarChecksum(string NombreGrupo) : bool
ActualizarColecciones() : void
ActualizarContacto(string Nickname) : void
ActualizarPerfilUsuario(Contacto perfil) : bool
AgregarNuevoContacto(Form Frm, string nickname, List<string> Colecciones) : void
AplicarActualizacionGrupo(EdicionGrupoInfo einfo) : void
AsignarGrupoContacto(string nickname, string grupo) : bool
Boomerang(string emisor) : bool
Buscar(string nickname) : List<Contacto>
CambiarEstadoContacto(string nickname, estadoContacto estado) : bool
CambioMiEstado(estadoContacto e) : void
CancelarAceptarContacto(AgregarContacto Frm, string nickremoto) : void
CargarContacto(string nickname) : Contacto
CargarDirectorio() : void
CargarFrmDescGrupo(string nombregrupo) : void
CargarFrmDesGrupoInvitar(string nombregrupo) : void
CargarGrupoRemoto(string NombreGrupo, string NicknameOmitido) : GrupoRemoto
CargarListaGrupos() : List<GrupoDirectorio>
CargarMiembros(coleccion c, string grupo) : void
ClienteDirectorio(ContenedorClientes Mycontenedor, string IdCliente, Lista GUIDirect...
ContactoRemotoToContacto(ContactoRemoto cr) : Contacto
ContactoToContactoRemoto(Contacto c) : ContactoRemoto
ContestarSaludoPublico(string ip) : void
Dispose() : void
EdicionGrupo(string NombreGrupo, GrupoDirectorio grupo) : void
EliminarContacto(string nickname, string grupo) : void
EliminarGrupo(string Grupo) : void
EliminarGrupoPublico(string Grupo) : bool
enviar(string idComunicacion, byte[] datos) : void
EnviarCheksum(string Nickname, string NombreGrupo, string Checksum) : void
EnviarInvitacionaGrupo(string nickname, InvitacionGrupo invitacion) : void
EnviarInvitacionDirectorio(string nickname, string Mensaje) : void
EnviarListadoGrupoBusqueda(string Nickname, string CriterioBusqueda) : void
EnviarListaGrupos(string nickname) : void
EnviarMensajeBusquedaGrupos(string CriterioBusqueda) : void
EnviarMensajeDespedida() : void
EnviarPerfilActualizado(string nickname) : void
EnviarRespuestaAdmisionGrupo(AdmisionaGrupo admision, RespuestaSolicitud respu...
EnviarRespuestaInvitacionDiretorio(string Nickname, RespuestaSolicitud respuesta) :...
EnviarRespuestaInvitacionGrupo(string nickname, InvitacionGrupo invitacion, Respu...
EnviarSaludoPublico() : bool
EnviarSincronizacionGrupo(string Nickname, string NombreGrupo) : void
EnvioRespuestaSincronizacionSilenciosa(string nickname) : void
EnvioSincronizacionSilenciosa() : void
EstaConectado(string nickname, out Contacto contacto) : bool
ExisteContacto(string nickname) : bool
ExisteEnCola(string nickname) : bool
GetIP(string Nickname) : string
GetIpAgenda(string nickname) : string
GetSeleccionado() : string
IndiceAgenda(string nickname) : int
IndiceContacto(string nickname) : int
IngresaContacto(ContactoRemoto cr, string tipomsj) : void
Inicializar() : bool
InvitarContactos(string NombreGrupo, List<string> invitados) : void
MostrarFrmAgregar() : void
MostrarFrmAgregar(string nickname) : void
MostrarFrmEdicionEtiqueta(string Etiqueta) : void
MostrarFrmNuevoGrupo() : void
NotificacionExpulsionGrupo(string Nickname, string NombreGrupo) : void
NotificarActualizacionGrupo(string nickname, EdicionGrupoInfo edinf) : void
NotificarCambioEstado() : void
NotificarChecksum(string NombreGrupo, string Checksum) : void
NotificarRemocionGrupo(string Nickname, string NombreGrupo) : void
NotificarRetiroVoluntarioGrupo(string Anfitrion, string NombreGrupo) : void
NuevoGrupo(NuevoGrupo Frm, string NombreGrupo, string Contrasena, string Descri...
OrdenarEliminarGrupo(string NombreGrupo) : void
OrdenarExpulsarContacto(string Nickname, string Grupo) : void
PerteneceDirectorioPrivado(string Nickname) : bool
PoseePerfil(string nickname) : bool
procesar(string idComunicacion, byte[] datos) : void
ProcesarAdmisionaGrupo(AdmisionaGrupo admision, RespuestaSolicitud respuesta) :...
ProcesarDespachosPendientes(ContactoRemoto cr) : void
ProcesarRespuestaAdmisionGrupo(GrupoRemoto gr) : void
ProcesarRespuestaSolicitud(Solicitud solicitud, RespuestaSolicitud respuesta) : void
RecargarContactosDirectorio() : void
RecibirNotificacionLocal(string idComunicacion, MensajeLocal mensaje, int Nbytes) :...
RecibirNotificacionRemota(string id, MensajeRemoto Mensaje) : void
RemoverContactoAgenda(string nickname) : bool
RenombrarEtiqueta(ModificarEtiquetaFrm, string NombreAnterior, string NombreNu...
SincronizarListaGrupo(GrupoRemoto gr) : void
SolicitarAdmisionaGrupo(GrupoDirectorio grupo) : void
SolicitarAdmisionaGrupo(string anfitrion, string grupo) : void
SolicitarAdmisionaGrupo(string anfitrion, string grupo, string Password) : void
SolicitarCheksum(string Anfitrion, string NombreGrupo) : void
SolicitarListaGrupoActualizada(string Anfitrion, string NombreGrupo) : void
SolicitarListaGrupos(string nickname) : void
SolicitarPerfilActualizado(string nickname) : void
SolicitarPerfilUsuario() : Contacto
SolicitarSincronizacionLista(string Anfitrion, string NombreGrupo) : void
SolicitudesChecksum(string nickname) : void
Temporizador_Tick(object Sender, EventArgs e) : void
TemporizadorG_Tick(object Sender, EventArgs e) : void
UtilizaComunicacion(string idComunicacion) : bool
YaEstaInvitado(string nickname) : bool
Tipos anidados
LimpiarGui
Delegate

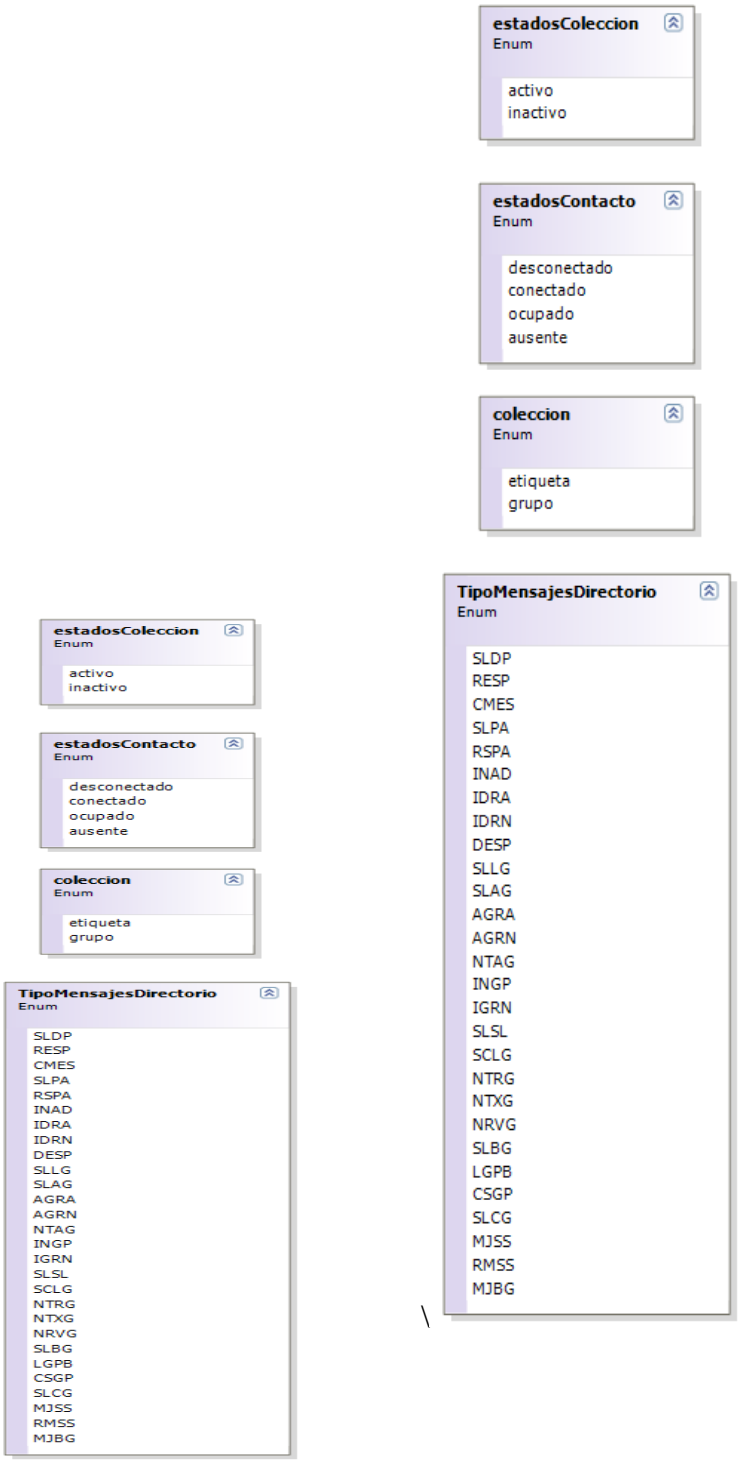

estadosColeccion
Enum
activo
inactivo
estadosContacto
Enum
desconectado
conectado
ocupado
ausente
coleccion
Enum
etiqueta
grupo
TipoMensajesDirectorio
Enum
SLDP
RESP
CMES
SLPA
RSPA
INAD
IDRA
IDRN
DESP
SLLG
SLAG
AGRA
AGRN
NTAG
INGP
IGRN
SLSL
SCLG
NTRG
NTXG
NRVG
SLBG
LGPB
CSGP
SLCG
MJSS
RMSS
MJBG

Diagrama 32

Para cualquiera de estas dos tareas, la clase **ClienteDirectorio** utiliza una referencia a la interfaz de usuario del Cliente Directorio (**Lista)** (Diagrama 33)**,** que permite hacer transparente al usuario de la estación de Trabajo SCIP2, la mayor parte de las tareas de mantenimiento y comunicación realizadas.

El usuario del sistema de comunicación Interactúa con el Cliente Usuario a través de esta interfaz y la interfaz envía las peticiones al Cliente Directorio asociado a la interfaz para que las realice.

Diagrama 33

- Tareas de Mantenimiento en el Cliente Directorio.

Gran parte de las tareas del **Cliente Directorio** son las de dar mantenimiento a la información almacenada en el directorio de un contacto; cada persona que crea un perfil o sesión para poder utilizar la estación de Trabajo SCIP2P, poseerá asociado un **Cliente Usuario** y un **Directorio de Contactos**; estos datos son persistentes en el tiempo y están almacenados en un archivo serializado de extensión .**dir. C**ada vez que se crea un nuevo perfil se crea un directorio de trabajo[15], con el nombre del **nickname** del contacto y archivo .**dir** en su interior**.**

La parte asociada al **Directorio de Contactos** en el archivo **.dir**, da soporte a la persistencia de las tres entidades principales dentro del Cliente Directorio: **Contactos, Etiquetas, Grupos Públicos.**

[15] Los directorios de trabajo locales se almacenan en /AplicacionPath/Data/clientes

- ✓ **Contactos:** Los contactos en el Cliente Directorio son la representación de otros usuarios utilizando la estación de Trabajo SCIP2P en la misma red. La estructura de un contacto está definida por la clase del mismo nombre **Contacto** (Diagrama 34) y las Interfaces **IContacto e IPerfil.**

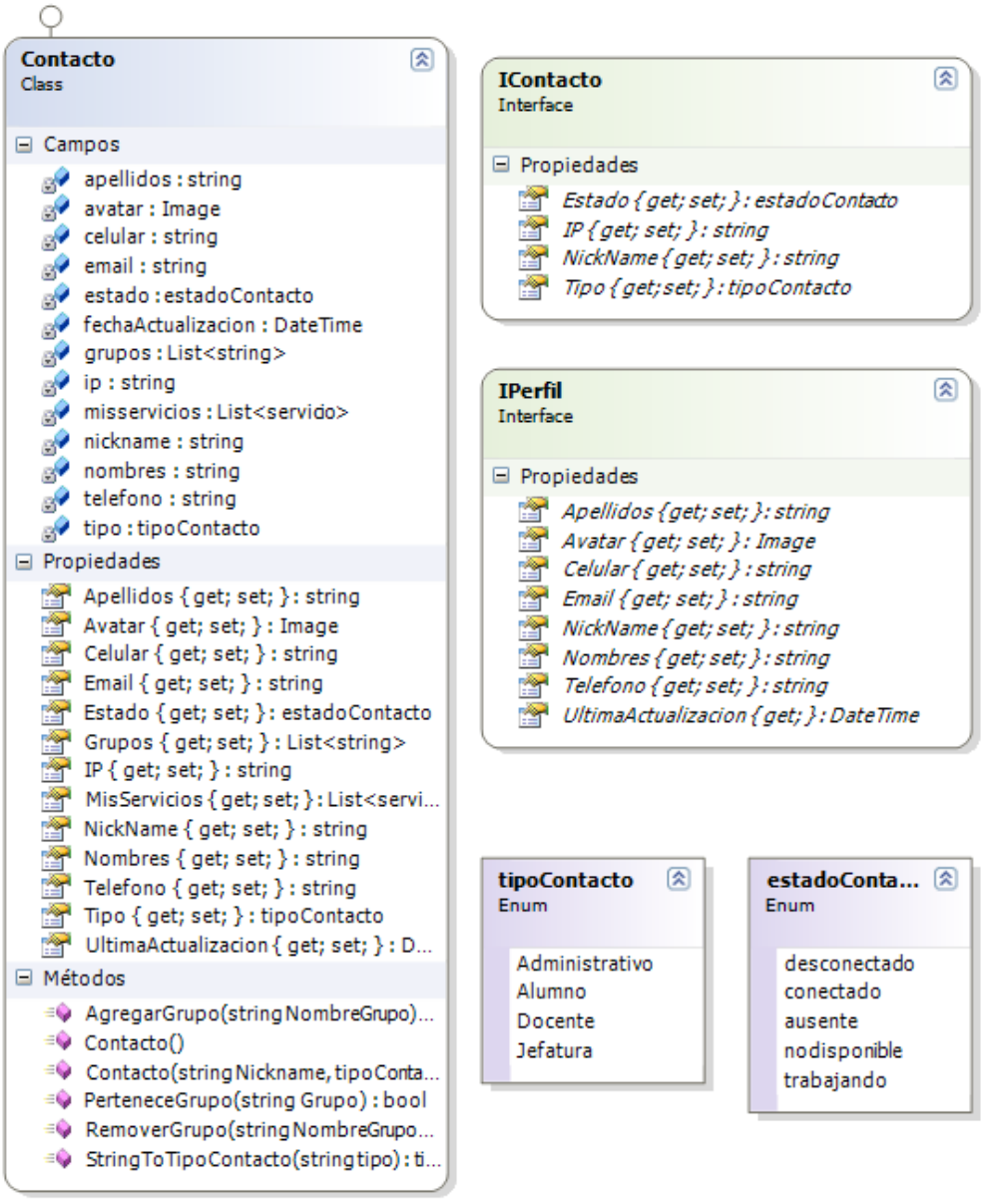


Diagrama 34

- ✓ **Etiquetas:** Son agrupaciones de contactos que nos permiten clasificarlos de acuerdo a criterios o etiquetas personales, por

ejemplo: amistad, trabajo, estudio, familia, etc. Un contacto puede estar asociado a más de una etiqueta o categoría.

- ✓ **Grupos Públicos:** Los grupos públicos, funcionan aparentemente de forma similar a una etiqueta**,** ya que permiten agrupar contactos en una categoría o criterio asociado a un nombre de grupo. Sin embargo su similitud llega hasta ahí, como su nombre lo dice estas agrupaciones son públicas, es decir que cualquier otro contacto en la red puede ver este grupo, y solicitar unirse a él.

  El creador del grupo se convierte en su anfitrión, y es quien limita el acceso al mismo; los demás contactos en la red, pueden localizar estos grupos y solicitar al anfitrión los admita como integrantes del grupo; al formar parte de un grupo, los miembros tendrán en todo momento una lista actualizada de los contactos que pertenecen al grupo, gracias a mecanismos de sincronización.

  La estructura de un grupo está definida por la clase **GrupoDirectorio** (Diagrama 35)**.**

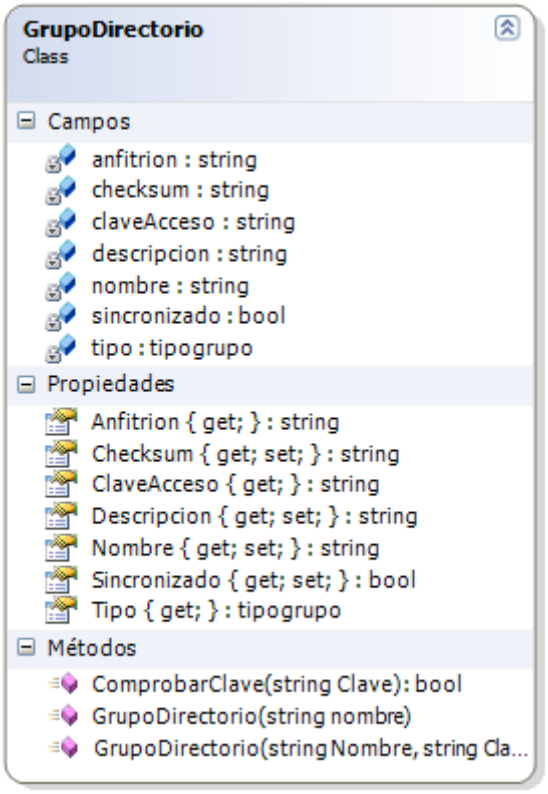


Diagrama 35

- Esquema de Comunicación del Cliente Directorio.

Como ya se ha mencionado, todas las tareas de comunicación en el **Cliente Directorio,** se realizan a través del envió y recepción de **Mensajes Remotos,** haciendo usos de los servicios que brinda el **Cliente de Mensajes**.

Si un **Cliente Directorio** desea enviar un mensaje a otro **Cliente Directorio**, lo primero que hace es crear una instancia de la clase **MensajeRemoto**; lo más importante es definir el tipo de mensaje remoto, con la finalidad que el Cliente Directorio que lo reciba sepa como procesarlo; luego, le notifica al Contenedor de Clientes, que desea enviar un **Mensaje Remoto,** y éste lo pasa al **Cliente de Mensajes** para que se encargue de convertir el mensaje remoto serializado en un Datagrama UDP, lo envíe por la Comunicación UDP en el **Sistema de Comunicación** asociada al **Cliente de Mensajes.**

En el **Sistema de Comunicación** Remoto, por el método de **Escucha UDP** se captura el paquete, y al leer la cabecera e identificar la **Comunicación UDP** a la que corresponde, se le despacha el Datagrama para que se encargue de leerlo y enviarlo al cliente, dentro del Contenedor de Clientes asociado a la comunicación; el **Cliente Mensajes** Remoto convierte el Datagrama UDP a un **MensajeRemoto,** lo despacha al cliente al que va dirigido, en este caso el **Cliente Directorio** Remoto, e identifica que tipo de mensaje es; de acuerdo a esto, deserializa los datos que están adjuntos al **Mensaje Remoto**, los procesa y de acuerdo al tipo de mensaje, puede notificar al **Cliente Usuario** a través del **Control de Solicitudes,** o Enviar un **MensajeRemoto** de Respuesta; así, el proceso se repite nuevamente.

Este esquema de comunicación se repite en todos los procesos que realiza el **Cliente Directorio**; de forma similar, en otros clientes que utilizan los **Mensajes Remotos** como mecanismos de control o de negociación previa, para una transmisión de datos por TCP, como son los casos de: **Cliente Gestor de Transferencia de Archivos** y **Cliente Gestor de Conversaciones.**

Esto permite que los clientes sean capaces de comunicarse con su homónimo en la estación de trabajo remota; cada cliente tiene definidos sus **maneras de entenderse o comunicarse**.

Es decir, qué acción, procedimiento, respuesta o notificación se realiza al recibir un **MensajeRemoto** de un tipo determinado; cada cliente posee una lista de tipos de **MensajesRemotos,** y los procesa de maneras diferentes. En el **Cliente Directorio** este protocolo de entendimiento se detalla en la siguiente tabla (Tabla 16):

Cliente Directorio - Protocolo de Entendimiento				
No	***Tipo de Mensaje***	***Descripción***	***Objeto Serializado Adjunto (Datos)***	***Posibles Mensajes de Respuesta.***
1	SLDP	Saludo Publico	ContactoRemoto	RESP
2	RESP	Respuesta a Saludo Publico	ContactoRemoto	-
3	CMES	Cambio de Estado	ContactoRemoto	-
4	SLPA	Solicitud de Perfil Actualizado	-	RSPA
5	RSPA	Respuesta de Solicitud de Perfil Actualizado	Contacto	-
6	INAD	Invitación de Admisión a directorio	-	IDRA/IDRN
7	IDRA	Invitación a Directorio Respuesta Afirmativa	ContactoRemoto	-
8	IDRN	Invitación a Directorio Respuesta Negativa	-	-
9	DESP	Mensaje de Despedida	-	-
10	SLLG	Solicitar Listado de Grupos Públicos	List<GrupoRemoto>	LDGP
11	SLAG	Solicitud de Admisión a Grupo	AdmisionaGrupo	AGRA/AGRN
12	AGRA	Admisión a Grupo Respuesta Afirmativa	GrupoRemoto	-
13	AGRN	Admisión a Grupo Respuesta Negativa	-	-
14	NTAG	Notificación de Actualización de Grupo	EdicionGrupoInfo	-
15	INGP	Invitación a Grupo Público.	InvitacionGrupo	IGRN

Cliente Directorio - Protocolo de Entendimiento				
No	***Tipo de Mensaje***	***Descripción***	***Objeto Serializado Adjunto (Datos)***	***Posibles Mensajes de Respuesta.***
16	IGRN	Invitación a Grupo Respuesta Negativa	-	-
17	SLSL	Solicitud de Sincronización de Lista	-	SCLG
18	SCLG	Sincronización de Listado de Grupo	List<ContactoRemoto>	-
19	NTRG	Notificación Remoción Grupo.	NombreGrupo	-
20	NTXG	Notificación de Expulsión de Grupo	NombreGrupo	-
21	NRVG	Notificación de Retiro Voluntario de Grupo	NombreGrupo	-
22	SLBG	Solicitar Listado Búsqueda de Grupos	-	LGPB
23	LGPB	Listado de Grupos Públicos Búsqueda.	List<GrupoRemoto>	-
24	SLCG	Solicitud de Checksum de grupo	NombreGrupo	CSGP
25	CSGP	Checksum de Grupo Público	Checksum de List<Contactos>	-
26	MJSS	Mensaje de Sincronización Silencioso	ContactoRemoto	RMSS
27	RMSS	Respuesta de Mensaje de Sincronización Silencioso	ContactoRemoto	-
28	MJBG	Mensaje de Búsqueda de Grupos	List<GrupoRemoto>	LGPB

Tabla 16

4.1.2.5 Cliente Gestor de Conversaciones

La clase ClienteGestorConversaciones (Diagrama 36) se encargar de crear, administrar, cancelar y finalizar conversaciones de Chat, entre dos usuarios con estaciones de trabajo SCIP2P. Específicamente, este cliente se encarga de la Gestión de **ClientesChat**[16].

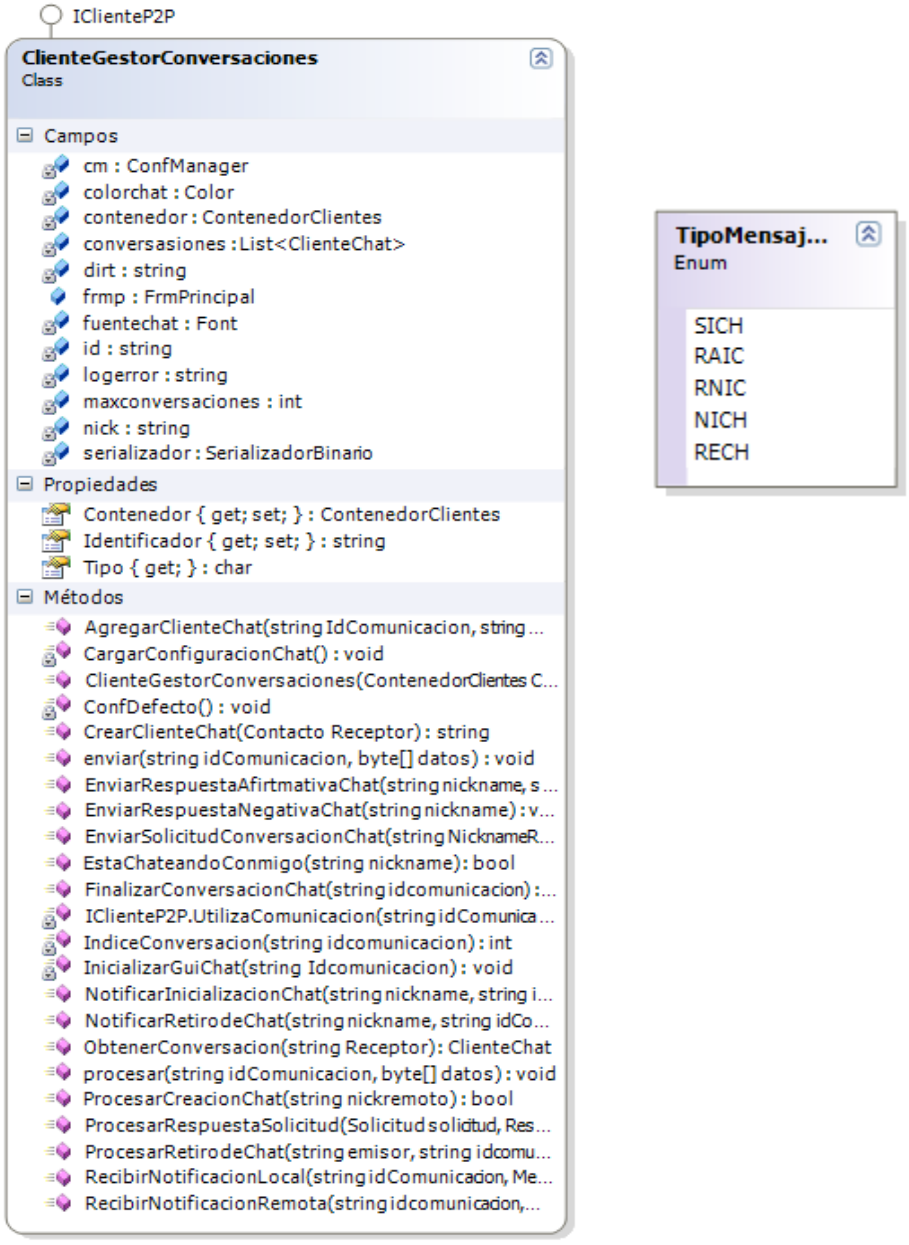


Diagrama 36

[16] **Cliente Chat:** Es el cliente especifico que gestiona la conversación entre dos usuarios, explicado con detalle en el apartado siguiente.

Los Clientes Chat son clientes TCP, es decir, que utilizan Comunicaciones TCP para la transmisión de mensajes. Por tanto se requiere de una comunicación previa para establecer el enlace de comunicación, esta gestión previa queda a cargo del **Clientes Gestor de Transferencias**.

Todos los mensajes previos necesarios para el establecimiento de conversación de Chat, los mensajes de control y los mensajes para la finalización de una conversación de chat son realizados por el **Cliente Gestor de Transferencias,** a través del envío y recepción de **Mensajes Remotos** utilizando el **Cliente de Mensajes**. Por eso no es de extrañarse que el este cliente no requiera de una comunicación TCP o UDP propia.

Este cliente maneja los **ClientesChat** en una Lista indexada por el identificador de comunicación TCP que posee cada ClienteChat. El número de conversaciones de chat activas depende de la configuración aplicada al cliente Gestor de Conversaciones.

El Gestor de conversaciones puede ser configurado a través de un archivo de configuración de nombre **cchat.conf** que es creado por cada cliente; almacena el número de conversaciones de chat permitidas así como la fuente predefinida para las conversaciones de chat.

- **Esquema de Comunicación del Gestor de Conversaciones**

El esquema de comunicación utilizado es similar al utilizado con el **Cliente Directorio** a través de Mensajes Remotos enviados por el **Cliente de Mensajes.**

El Gestor de Transferencias genera **Mensajes Remotos** definiendo: el identificador del cliente que lo envía, el tipo de mensaje (Tabla 17) (de acuerdo al protocolo de entendimiento entre Gestores de Conversación), y los datos que puedan ir adjuntos al mensaje; el mensaje se envía a través del **Cliente de**

Mensajes; su homónimo remoto recibe el mensaje y lo despacha al **Gestor de Conversaciones Remoto** quien lo procesa.

Cliente Gestor de Conversaciones - Protocolo de Entendimiento				
No	***Tipo de Mensaje***	***Descripción***	***Objeto Serializado Adjunto (Datos)***	***Posibles Mensajes de Respuesta.***
1	SICH	Solicitud para iniciar una conversación de Chat	-	RAIC / RNIC
2	RAIC	Respuesta Afirmativa de Invitación a Chat	IdComunicacionTCP	-
3	RNIC	Respuesta Negativa de Invitación a Chat	-	-
4	NICH	Notificación Inicialización Chat	IdComunicacionTCP	-
5	RECH	Retiro de chat	IdComunicacionTCP	-

Tabla 17

Para que pueda existir un enlace entre dos clientes TCP, es necesario que ambas comunicaciones TCP tengan el mismo identificador. Por eso al aceptar una solicitud para una nueva conversación de Chat, se envía el identificador de la comunicación con la cual debe crearse la comunicación en el otro extremo, de tal forma que puedan comunicarse los **ClienteChat.**

4.1.2.5.1 Cliente Chat

Este es un cliente TCP, es decir, que utiliza una comunicación TCP para la transmisión de sus datos, en este caso Mensajes de Chat. Por tanto, para su creación es necesaria una comunicación previa entre los

usuarios que desean iniciar una nueva conversación de Chat, tarea que queda a cargo del Gestor de Conversaciones. Una vez realizada esta comunicación previa, se crea un nuevo **ClienteChat** (Diagrama 37) en las estaciones de SCIP2P de los dos usuarios, ambas con el mismo identificador de ComunicacionTCP.

El **ClienteChat** como cualquier otro cliente implementa la interfaz IClienteP2P, por tanto posee todos los métodos necesarios para administrar su comunicación TCP, enviar y recibir datos por esta vía.

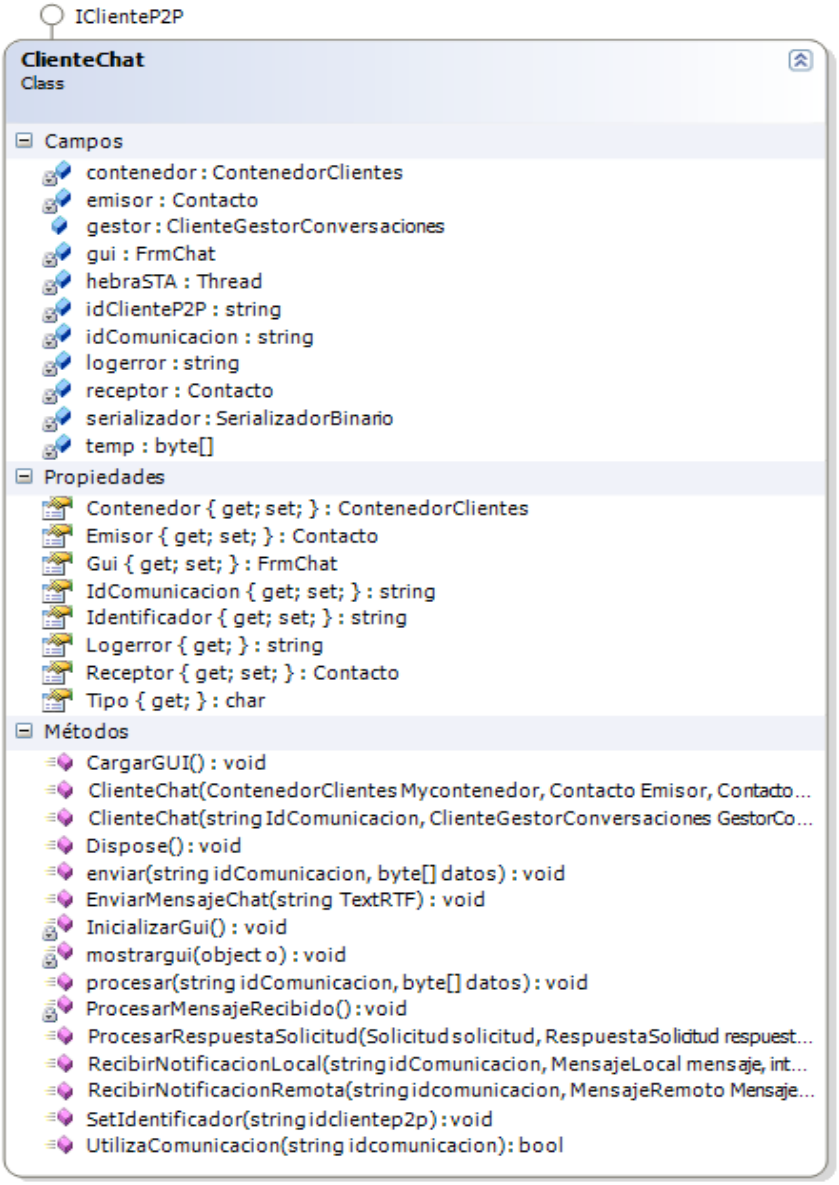


Diagrama 37

Además, cada **ClienteChat** posee la referencia a una instancia de la interfaz de usuario utilizada para las conversaciones de chat, **FrmChat** (Diagrama 38). Cada vez que se inicializa la interfaz, se inicia en una hebra **Simple Statement** para garantizar que no interrumpa ningún proceso.

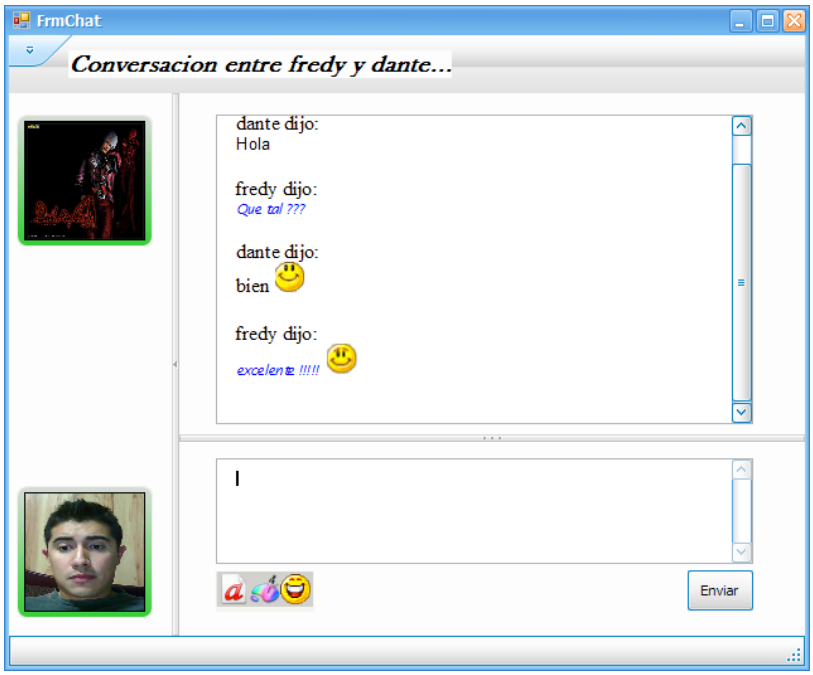


Diagrama 38

Como se observa en la interfaz, los mensajes de chat no son texto plano, sino RTF, lo que permite enviar en cada mensaje Chat , información de la fuente, colores, imágenes, además del texto del mensaje.

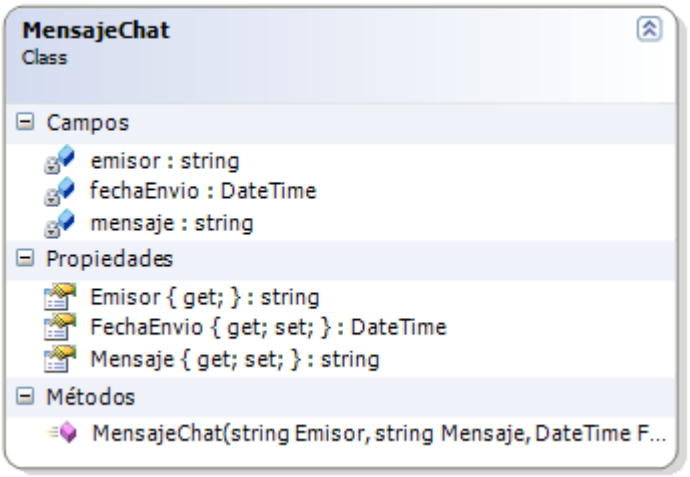


Diagrama 39

Cada Mensaje de Chat enviado, a pesar de enviarse a través de un Stream de Datos en la Comunicación TCP, va encapsulado en la estructura de un objeto de la clase **MensajeChat** (Diagrama 39). Lo que permite encapsular información extra como fecha y hora en que se envió el mensaje**.**

4.1.2.6 Cliente Gestor de Transferencia de Archivos

Este cliente es un administrador de **Clientes TCP**, al igual que lo es el **Gestor de Conversaciones**; sin embargo, el **Cliente de Transferencia de Archivos** es el cliente más complejo del **Contenedor de Clientes**; este cliente permite gestionar la transferencia de archivos a través de dos estaciones de trabajo SCIP2P.

Al igual que en el cliente anterior, que es tarea del Gestor de Conversaciones realizar la conversación previa al establecimiento de los Clientes TCP, así es en el caso del Gestor de Transferencia. Pero la tarea de este cliente no termina ahí, ya que el **Gestor de Conversaciones** tiene como funciones:

- Administrar el directorio de archivos compartidos.
- Gestionar las búsquedas de archivos en la red.
- Administrar la transferencia de archivos.
  - ✓ Transferencias de Subida de Archivos.
  - ✓ Transferencias de Descarga de Archivos.

La configuración del Gestor de Transferencia de Archivos es almacenada en el directorio de trabajo, del usuario que ha iniciado sesión, con nombre **carchivos.conf,** guarda aspectos como número de descargas y subidas máximas permitidas, y el tamaño del bloque de transferencia enviado por la comunicación TCP configurable, entre bloques de 512kb y bloques de 1,024kb y la ruta del **directorio compartido**[17].

[17] **Directorio Compartido:** es la carpeta designada para las búsquedas y descargas del Gestor de Transferencias, se explica con detalle más adelante.

- **Esquema de Comunicación Cliente Gestor de Transferencia de Archivos.**

Este cliente no posee de una comunicación TCP o UDP asociada a él en el **Sistema de Comunicación**; debido a que todas sus tareas de comunicación para el establecimiento, control, interrupción y finalización de una transferencia de archivos, se realiza a través de **Mensajes Remotos**, enviados entre clientes **Gestores de Transferencias de Archivos** (Diagrama 40), utilizando el **Cliente de Mensajes** (Tabla 18)**.**

Cliente Gestor de Transferencias - Protocolo de Entendimiento				
No	***Tipo de Mensaje***	***Descripción***	***Objeto Serializado Adjunto (Datos)***	***Posibles Mensajes de Respuesta.***
1	BUSC	Búsqueda de Archivos.	BusquedaArchivo	-
2	RBUS	Resultado de Búsqueda.	ResultadoBusqueda	RBUS
3	SDES	Solicitud de Descarga	ArchivoInfo	ACTF
4	ACTF	Acuerdo de Transferencia.	AcuerdoTransferencia	-
5	NOCD	Notificación de Cancelación de Descarga.	CancelacionTransferencia	-
6	NOCS	Notificación de Cancelación de Subida	CancelacionTransferencia	-
7	SLEA	Solicitud Envió de Archivo.	SolicitudEnvioArchivo	ACTF

Cliente Gestor de Transferencias - Protocolo de Entendimiento				
No	***Tipo de Mensaje***	***Descripción***	***Objeto Serializado Adjunto (Datos)***	***Posibles Mensajes de Respuesta.***
8	SLTS	Solicitud de Transferencia de Segmento	SolicitudTSegmento	-

Tabla 18

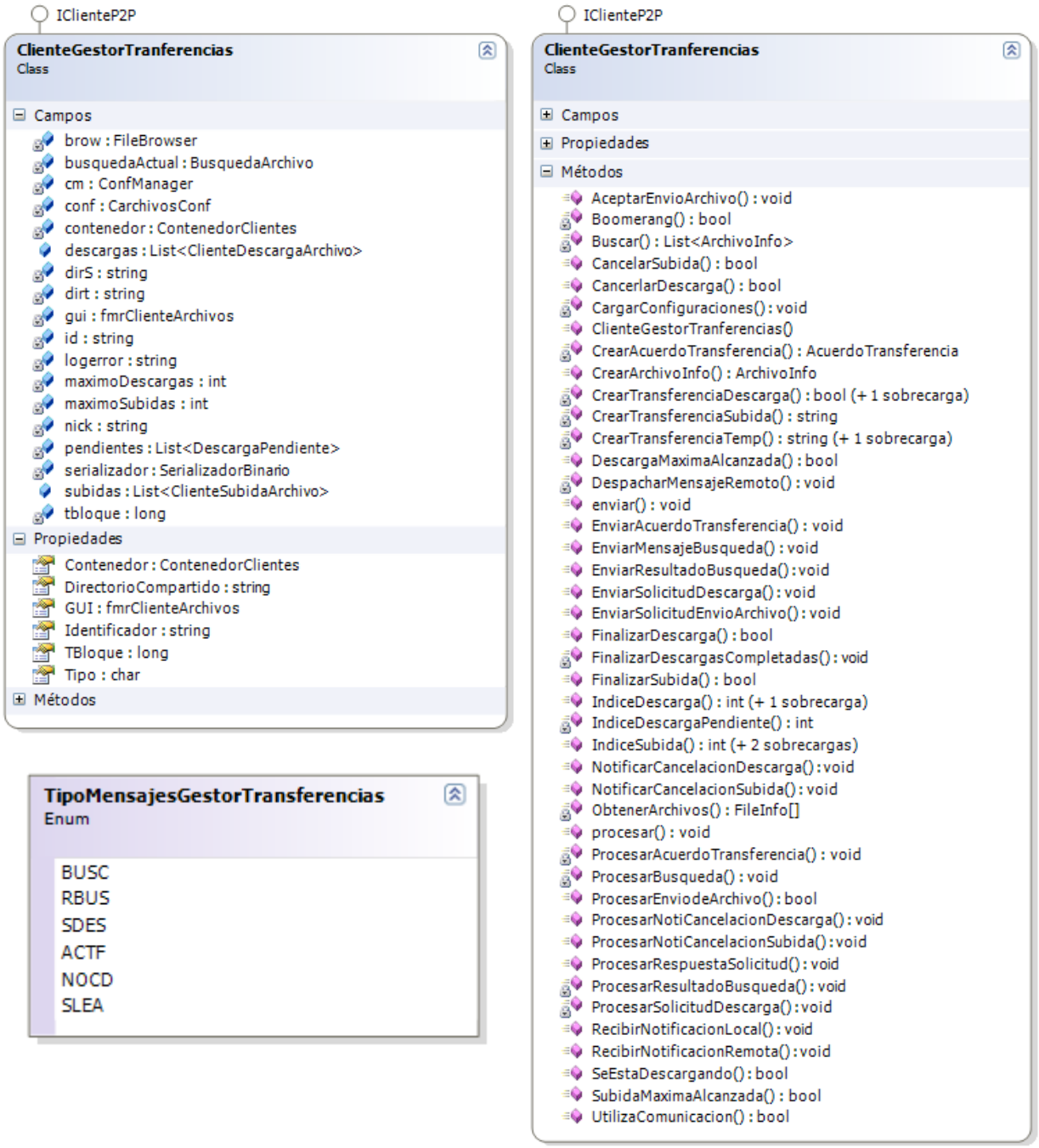

IClienteP2P
ClienteGestorTranferencias
Class
Campos
brow : FileBrowser
busquedaActual : BusquedaArchivo
cm : ConfManager
conf : CarchivosConf
contenedor : ContenedorClientes
descargas : List<ClienteDescargaArchivo>
dirS : string
dirt : string
gui : fmrClienteArchivos
id : string
logerror : string
maximoDescargas : int
maximoSubidas : int
nick : string
pendientes : List<DescargaPendiente>
serializador : SerializadorBinario
subidas : List<ClienteSubidaArchivo>
tbloque : long
Propiedades
Contenedor : ContenedorClientes
DirectorioCompartido : string
GUI : fmrClienteArchivos
Identificador : string
TBloque : long
Tipo : char
Métodos
TipoMensajesGestorTransferencias
Enum
BUSC
RBUS
SDES
ACTF
NOCD
SLEA
IClienteP2P
ClienteGestorTranferencias
Class
Campos
Propiedades
Métodos
AceptarEnvioArchivo() : void
Boomerang() : bool
Buscar() : List<ArchivoInfo>
CancelarSubida() : bool
CancerlarDescarga() : bool
CargarConfiguraciones() : void
ClienteGestorTranferencias()
CrearAcuerdoTransferencia() : AcuerdoTransferencia
CrearArchivoInfo() : ArchivoInfo
CrearTransferenciaDescarga() : bool (+ 1 sobrecarga)
CrearTransferenciaSubida() : string
CrearTransferenciaTemp() : string (+ 1 sobrecarga)
DescargaMaximaAlcanzada() : bool
DespacharMensajeRemoto() : void
enviar() : void
EnviarAcuerdoTransferencia() : void
EnviarMensajeBusqueda() : void
EnviarResultadoBusqueda() : void
EnviarSolicitudDescarga() : void
EnviarSolicitudEnvioArchivo() : void
FinalizarDescarga() : bool
FinalizarDescargasCompletadas() : void
FinalizarSubida() : bool
IndiceDescarga() : int (+ 1 sobrecarga)
IndiceDescargaPendiente() : int
IndiceSubida() : int (+ 2 sobrecargas)
NotificarCancelacionDescarga() : void
NotificarCancelacionSubida() : void
ObtenerArchivos() : FileInfo[]
procesar() : void
ProcesarAcuerdoTransferencia() : void
ProcesarBusqueda() : void
ProcesarEnviodeArchivo() : bool
ProcesarNotiCancelacionDescarga() : void
ProcesarNotiCancelacionSubida() : void
ProcesarRespuestaSolicitud() : void
ProcesarResultadoBusqueda() : void
ProcesarSolicitudDescarga() : void
RecibirNotificacionLocal() : void
RecibirNotificacionRemota() : void
SeEstaDescargando() : bool
SubidaMaximaAlcanzada() : bool
UtilizaComunicacion() : bool

Diagrama 40

El Cliente **Gestor de Transferencias** posee una referencia a la interfaz **frmClienteArchivos** (Diagrama 41)**,** que le permite a los usuarios interactuar con las funciones que brinda este cliente.

- Directorio de Archivos Compartidos.

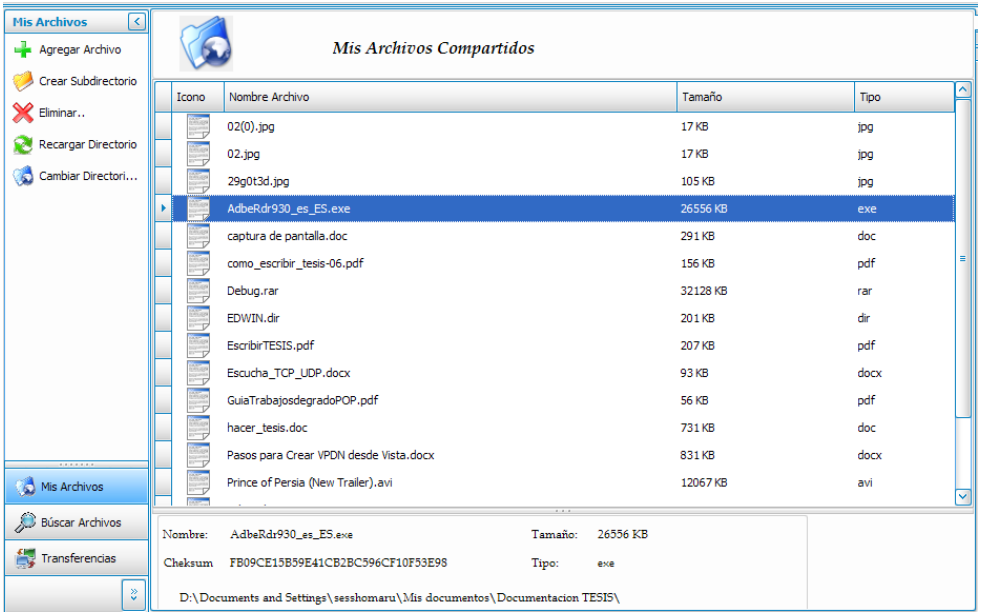


Diagrama 41

Una de las tareas de **Gestor de Transferencias** es administrar el directorio de archivos compartidos. Este directorio es almacenado en el archivo de configuración desde el primer inicio de sesión de un usuario de SCIP2P. Si por alguna razón no se encontrara este directorio se solicitará definirse nuevamente.

Para poder cargar la información de los archivos y subdirectorios ubicados en el directorio compartido, el **Cliente Gestor de Transferencias** utiliza una instancia de la clase **FileBrowser** (Diagrama 42), la cual permite cargar los listados de los objetos **DirectorioInfo y ArchivoInfo**; estas clases permiten la representación de un Subdirectorio y un Archivo respectivamente, dentro del directorio compartido y luego desplegarlas de forma visual en la

pestaña de **Mis Archivos Compartidos** de la interfaz de usuario **frmClienteArchivos.**

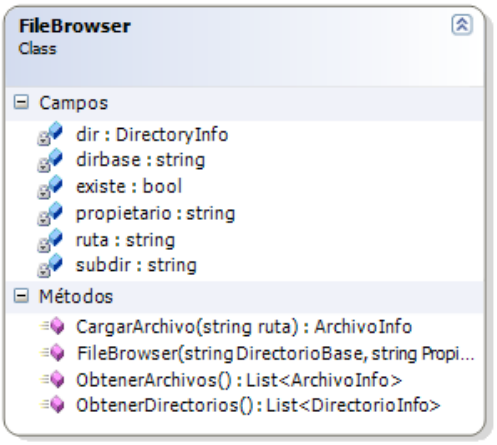


Diagrama 42

La Clase **IOinfo** constituye una clase abstracta, a partir de la cual de heredan la clase **ArchivInfo**, **DirectorioInfo e IOInfoBrowse** (Diagrama 43), que agrega las características necesarias para cargar la representación de un **Directorio** o **Archivo** de forma visual en la pestaña de Archivos Compartidos de **frmClienteArchivos.** Una vez cargado el directorio compartido el **Cliente Gestor de Transferencias**, permite realizar diversas tareas de mantenimiento, como: agregar o remover archivos, crear subdirectorios, escanear nuevamente el directorio compartido, incluso cambiar la ruta de nuestro **Directorio Compartido.**

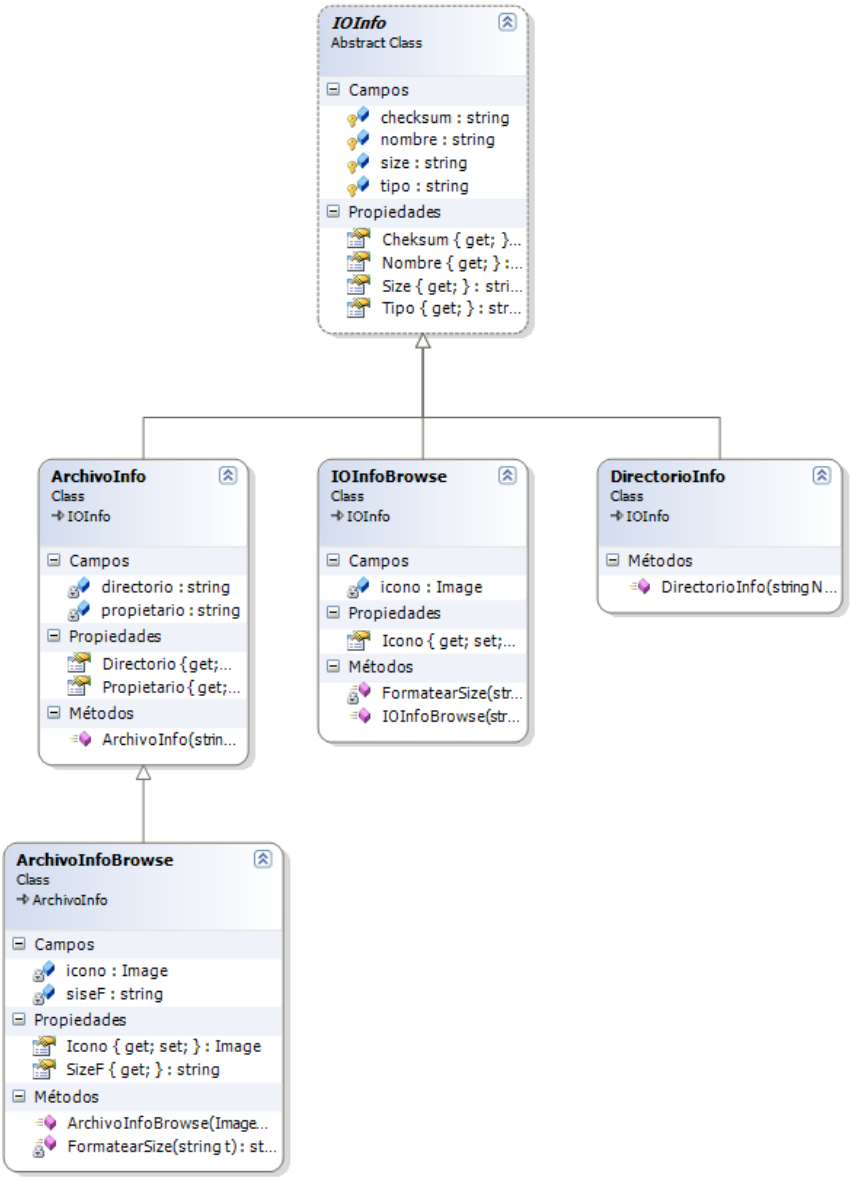

IOInfo
Abstract Class
Campos
checksum : string
nombre : string
size : string
tipo : string
Propiedades
Cheksum { get; }...
Nombre { get; } :...
Size { get; } : stri...
Tipo { get; } : str...
ArchivoInfo
Class
IOInfo
Campos
directorio : string
propietario : string
Propiedades
Directorio { get;...
Propietario { get;...
Métodos
ArchivoInfo(strin...
IOInfoBrowse
Class
IOInfo
Campos
icono : Image
Propiedades
Icono { get; set;...
Métodos
FormatearSize(str...
IOInfoBrowse(str...
DirectorioInfo
Class
IOInfo
Métodos
DirectorioInfo(string N...
ArchivoInfoBrowse
Class
ArchivoInfo
Campos
icono : Image
siseF : string
Propiedades
Icono { get; set; } : Image
SizeF { get; } : string
Métodos
ArchivoInfoBrowse(Image...
FormatearSize(string t) : st...

Diagrama 43

- Búsquedas de Archivos.

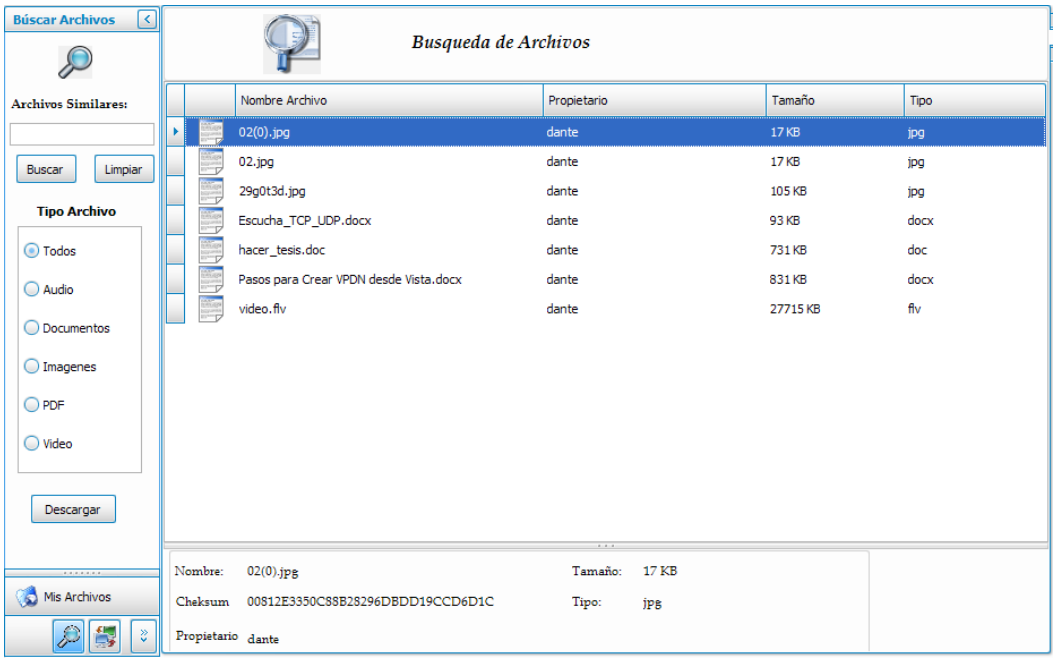


Diagrama 44

Otra de las Tareas del **Gestor de Transferencias** es la de generar y dar respuesta a búsquedas de archivos en la red (Diagrama 44). Una vez este cliente haya escaneado el directorio compartido, y tenga cargada la lista de objetos **DirectorioInfo** y **ArchivoInfo**, será capaz de dar respuesta a mensajes remotos de búsquedas de archivos, enviados por otra estación de trabajo SCIP2P, específicamente por otro **Cliente Gestor de Transferencias**.

Al realizar una búsqueda de archivos, el Cliente toma los criterios de búsqueda y crea una instancia de la clase **BusquedaArchivo** (Diagrama 45)**,** la cual es serializada y adjuntada a un **Mensaje Remoto** que es enviado a través del **Cliente de Mensajes** por toda la red.

Diagrama 45

Los Clientes Gestores de Transferencias, realizarán una selección de todos los archivos que cumplan con los criterios de búsqueda establecidos y genera un listado de objetos **ArchivoInfo,** que luego empaqueta en una instancia de la clase **ResultadoBusqueda** (Diagrama 46)**.**

Este objeto se adjunta como datos de un mensaje remoto resultado de una búsqueda, y es enviado por el **Cliente de Mensajes**.

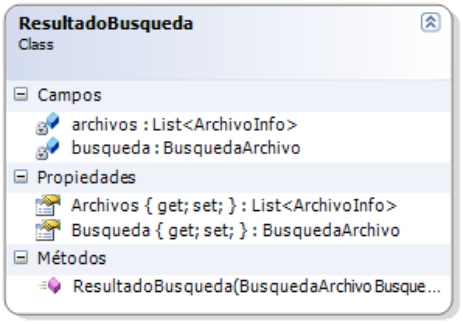


Diagrama 46

Una vez recibidos los resultados de la búsqueda de todos los usuarios que escucharon la solicitud, el **Cliente Gestor de Transferencias** genera un listado, que es desplegado en la pestaña de Búsqueda de Archivos de la interfaz de usuario **frmClienteArchivos.**

- **Transferencias de Archivos.**

Al obtener los resultados de una búsqueda, un usuario puede decidir descargar alguno de los archivos listado en este resultado. Para esto, el **Gestor de Transferencias** envía al dueño del archivo un **Mensaje Remoto** de solicitud de descarga del mismo, que posee adjunto el objeto **ArchivoInfo** del archivo a descargar, y que fue recibido como resultado de búsqueda.

Cuando el **Gestor de Transferencias Remoto** recibe el mensaje de Solicitud de Descarga de Archivo, crea un nuevo **ClienteSubidaArchivo**[18]**,**

[18] **ClienteSubidaArchivo:** clientep2p TCP que se da respuesta a la descarga de un archivo enviando los segmentos solicitados.

solicitando al **Sistema de Comunicación** una nueva **ComunicacionTCP** asociada a este cliente.

Una vez creado el **ClienteSubidaArchivos,** el **Gestor de Transferencias** responde al mensaje de Solicitud de Descarga del Archivo del solicitante, con un **Mensaje Remoto** que posee adjunta una instancia de la clase **Acuerdo de Transferencia** (Diagrama 47)**.**

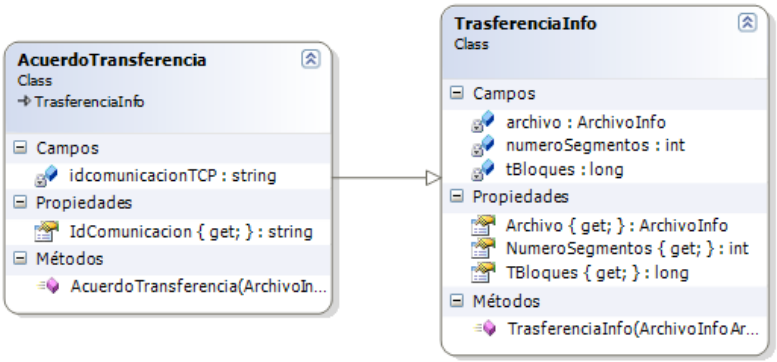


Diagrama 47

El acuerdo de transferencia posee toda la información necesaria para llevar a cabo la transferencia, y ayuda a garantizar la integridad de los archivos transferidos. Información como: el número de segmentos a transferir, el tamaño del bloque de transferencia, el objeto **ArchivoInfo** del archivo a transferir, y el **Identificador de la Comunicación TCP** que se utilizará para la transferencia.

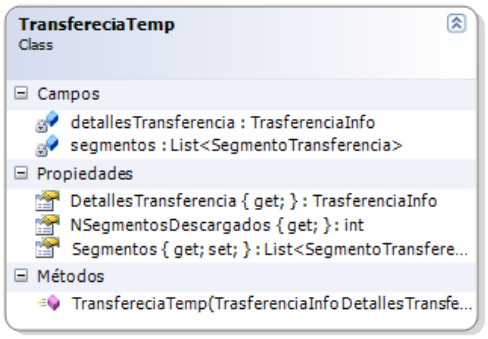


Diagrama 48

Cuando el **Gestor de Transferencias** recibe el Mensaje Remoto con el **AcuerdoTransferencia**, crea una instancia de la clase **TransferenciaTemp** (Diagrama 48), que servirá para almacenar temporalmente los segmentos descargados de un archivo en transferencia; esta instancia será serializada en un archivo en el Directorio Compartido con extensión **.tmp**.

Con la información del Acuerdo de Transferencia, el Gestor de Transferencias del cliente que desea descargar el archivo, crea un nuevo **ClienteDescargaArchivo**[19]; solicitando al Sistema de Comunicación que cree una nueva comunicación con el mismo Identificador de Comunicación TCP, que viene en el acuerdo de transferencia, con la finalidad de garantizar que ambas comunicaciones y ambos clientes puedan comunicarse. Finalmente se envía la orden al nuevo cliente para que inicie la descarga del archivo deseado.

4.1.2.6.1 Cliente de Descarga Archivos

Una vez iniciado el proceso de descarga, el **Cliente de Descarga de Archivos** se encarga de enviar **Mensajes Remotos** a través del Cliente de Mensajes, para solicitar en el otro extremo al **Cliente de Subida de Archivos**, que le envíe uno a uno los segmentos que constituyen el archivo. El número el tamaño de cada uno de estos segmentos fue el que previamente se definió en el **AcuerdoTranferencia**.

Cada vez que el **ClienteDescargaArchivo** (Diagrama 49), recibe un segmento, este es transmitido por la comunicación TCP, encapsulado en una instancia de la clase **SegmentoTransferencia** (Diagrama 50) y posee además de los datos el edificador del número de segmento.

[19] **ClienteDescargaArchivo:** Es el cliente que en una transferencia de archivos solicita los segmentos al ClienteSubidaArchivo hasta descargarlos completamente y generar el archivo en función de los segmentos.

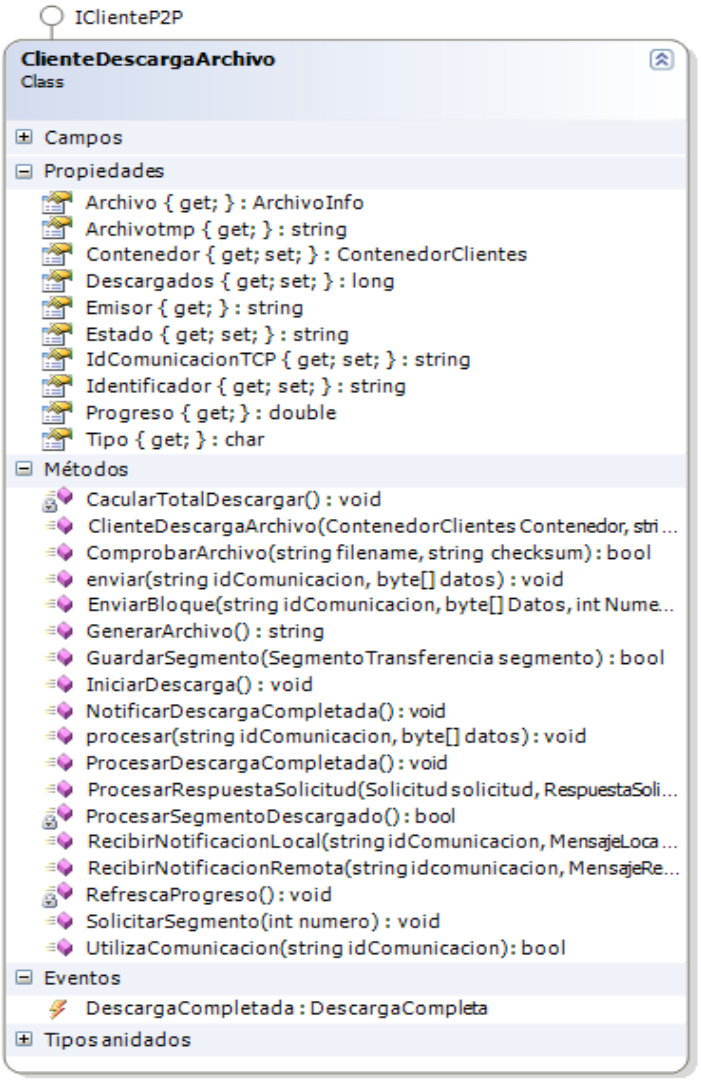

IClienteP2P
ClienteDescargaArchivo
Class
Campos
Propiedades
Archivo { get; } : ArchivoInfo
Archivotmp { get; } : string
Contenedor { get; set; } : ContenedorClientes
Descargados { get; set; } : long
Emisor { get; } : string
Estado { get; set; } : string
IdComunicacionTCP { get; set; } : string
Identificador { get; set; } : string
Progreso { get; } : double
Tipo { get; } : char
Métodos
CacularTotalDescargar() : void
ClienteDescargaArchivo(ContenedorClientes Contenedor, stri...
ComprobarArchivo(string filename, string checksum) : bool
enviar(string idComunicacion, byte[] datos) : void
EnviarBloque(string idComunicacion, byte[] Datos, int Nume...
GenerarArchivo() : string
GuardarSegmento(SegmentoTransferencia segmento) : bool
IniciarDescarga() : void
NotificarDescargaCompletada() : void
procesar(string idComunicacion, byte[] datos) : void
ProcesarDescargaCompletada() : void
ProcesarRespuestaSolicitud(Solicitud solicitud, RespuestaSoli...
ProcesarSegmentoDescargado() : bool
RecibirNotificacionLocal(string idComunicacion, MensajeLoca...
RecibirNotificacionRemota(string idcomunicacion, MensajeRe...
RefrescaProgreso() : void
SolicitarSegmento(int numero) : void
UtilizaComunicacion(string idComunicacion): bool
Eventos
DescargaCompletada : DescargaCompleta
Tipos anidados

Diagrama 49

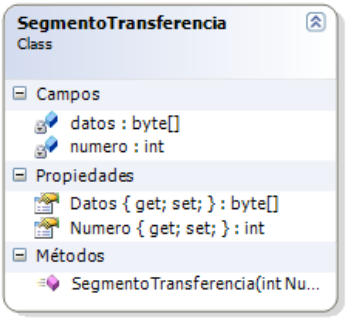


Diagrama 50

Cada vez que un segmento es descargado, se verifica su integridad comprobando y confrontando el **Checksum** en el bloque de transferencia TCP, y el Cheksum calculado de los datos recibidos. Si se comprueba que la Checksum es diferente, significa que la transferencia fue errónea y por tanto se pide de nuevo el paquete defectuoso; caso contrario, si la verificación es exitosa, se escribe el segmento descargado en el archivo temporal y se solicita el siguiente bloque, se repite sucesivamente hasta descargar todos los segmentos del archivo transferido.

Una vez se han descargados todos los segmentos el **ClienteDescargaArchivo**, se genera el archivo descargado, utilizando la instancia **TransferenciaTemp**, que esta serializada como archivo temporal.

Finalmente se realiza una verificación del cheksum del archivo ya ensamblado, y se envía un **Mensaje Remoto** al **ClienteSubidaArchivo** notificando que la descarga fue completada.

- **Cliente de Subida Archivos**

Este cliente (Diagrama 51), funciona una vez creado este cliente, su única tarea es estar expectante a la solicitud de envío de segmentos que puede realizar el **ClienteDescargaArchivo**, al que está asociado por la comunicación TCP.

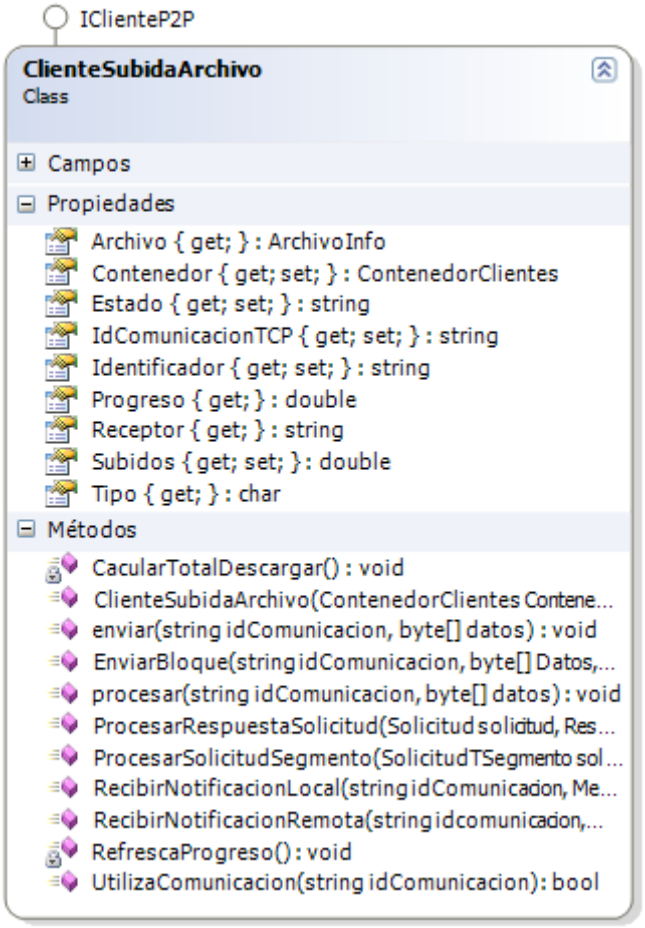


Diagrama 51

Finalmente recibe un mensaje de Descarga Completada que indica a este cliente, que debe liberar memoria y destruirse.

4.1.2.7 Módulo de Actualizaciones.

Con posterioridad a la fase de implementación del Sistema de Información Integral P2P, se impone la fase de mantenimiento.

No se puede dejar de considerar que el mantenimiento de sistemas es la fase más prolongada y costosa del ciclo de vida de los sistemas. Las implicaciones del volumen de trabajo para mantenimiento para los planes de implementación de una nueva tecnología de información en una organización es un tema que merece atención especial.

Desde el diseño del sistema y gracias a la utilización y una completa Programación Orientada a Objetos, SCIP2P brinda una gran flexibilidad para apoyar el mantenimiento necesario y, en consecuencia, reducir o disminuir los recursos y esfuerzos necesarios para llevar a cabo el mantenimiento de la aplicación.

Uno de las características que más ayuda a extender la vida útil del Sistema Integral de Comunicación P2P es su capacidad de instalar paquetes de actualización, que permitan mantener actualizada la aplicación sin que esto requiera el remplazar por completo la instalación de una versión inferior.

Pueden tratarse de pequeñas actualizaciones para corregir algunos problemas sencillos, mejorar un Cliente o Aplicativo, puede ser una gran actualización que implica un cambio de versión el mismo o simplemente actualizar repositorios de datos, etc. Todos aspectos son contemplados en los paquetes de actualización de SCIP2P.

.

4.2.1 Creación de Paquetes de Actualización (Diagrama 52).

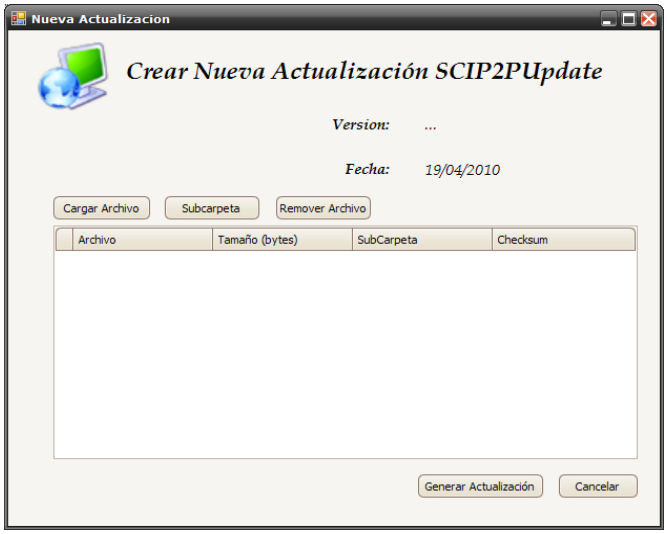


Diagrama 52

La creación de paquetes de actualización para el Sistema de Información Integral P2P está a cargo de una aplicación, externa. **SCIP2PUpdateMaker**.

Diagrama 53

Dentro de un paquete de actualización puede haber varios archivos (Diagrama 53) y especificar para cada uno la ruta o directorio donde se deberá ser instalado y un su checksum, que permitirá verificar la integridad del archivo una vez sea desempaquetado e instalado (Diagrama 54).

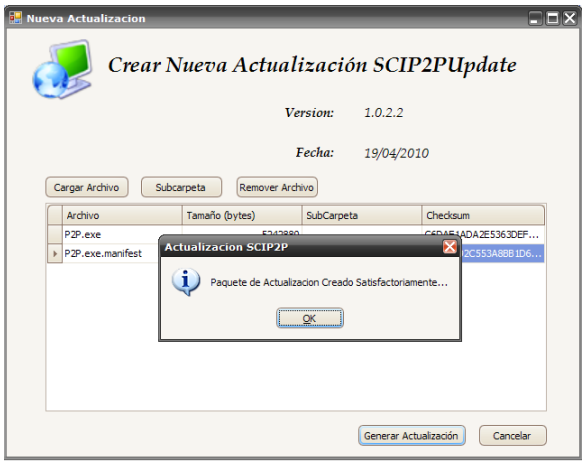


Diagrama 54

Los **Paquetes de Actualización SCIP2P** son en realidad instancias de la clase **SCIP2PUpdate**[20] (Diagrama 55) serializadas en archivos en archivos serializados con extensión **.update**. Por lo general con un nombre SCIP2PUPDATE_<versión>.update por ejemplo:

SCIP2PUPDATE_1.0.2.2.update.

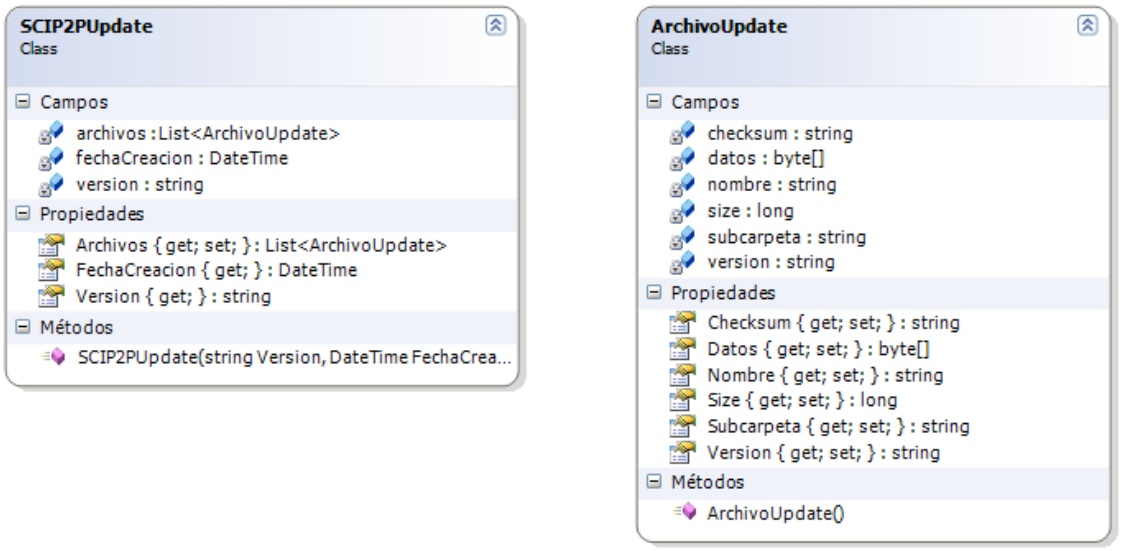


Diagrama 55

Una Instancia SCIP2PUpdate contiene una Lista de Objetos **ArchivoUpdate**[21]**,** uno por cada archivo que incluye el paquete de actualización.

Cada instancia **ArchivoUpdate** posee el nombre del archivo empaquetado, los datos del archivo serializados (bytes). Checksum de verificación, carpeta de instalación, tamaño. Etc y todo lo que permita generar el archivo físicamente en el disco duro al momento de la instalación del paquete.

[20]La clase **SCIP2PUpdate:** es la que contiene empaquetados todos los archivos de un paquete de actualización.
[21] La Clase **ArchivoUpdate** representa a uno de los archivos dentro del paquete de actualización.

4.2.2 Distribución de Paquetes de Actualización.

Debido a que las actualizaciones se encuentran empaquetadas en un único archivo ofrece una gran libertad de movilidad, puede transportarse en cualquier dispositivo de almacenamiento y transportarlo sin dificultad a la computadora donde se desea actualizar una estación de trabajo SCIP2P.

Incluso se puede aprovechar las mismas bondades de SCIP2P y transferir el archivo a través del **Cliente Gestor de Transferencias** a los contactos que lo deseen.

Extendiendo esta última posibilidad surge el **Cliente de Actualizaciones,** que se encarga de una distribución automática de actualizaciones a los contactos que la necesiten en una red de trabajo **SCIP2P**.

4.2.2.1 Cliente de Actualizaciones.

Al igual que los demás clientes en el contenedor de clientes, la clase **ClienteUpdate** (Diagrama 56) es un clienteP2P es decir un cliente que implementa la interfaz *ICLienteP2P.*

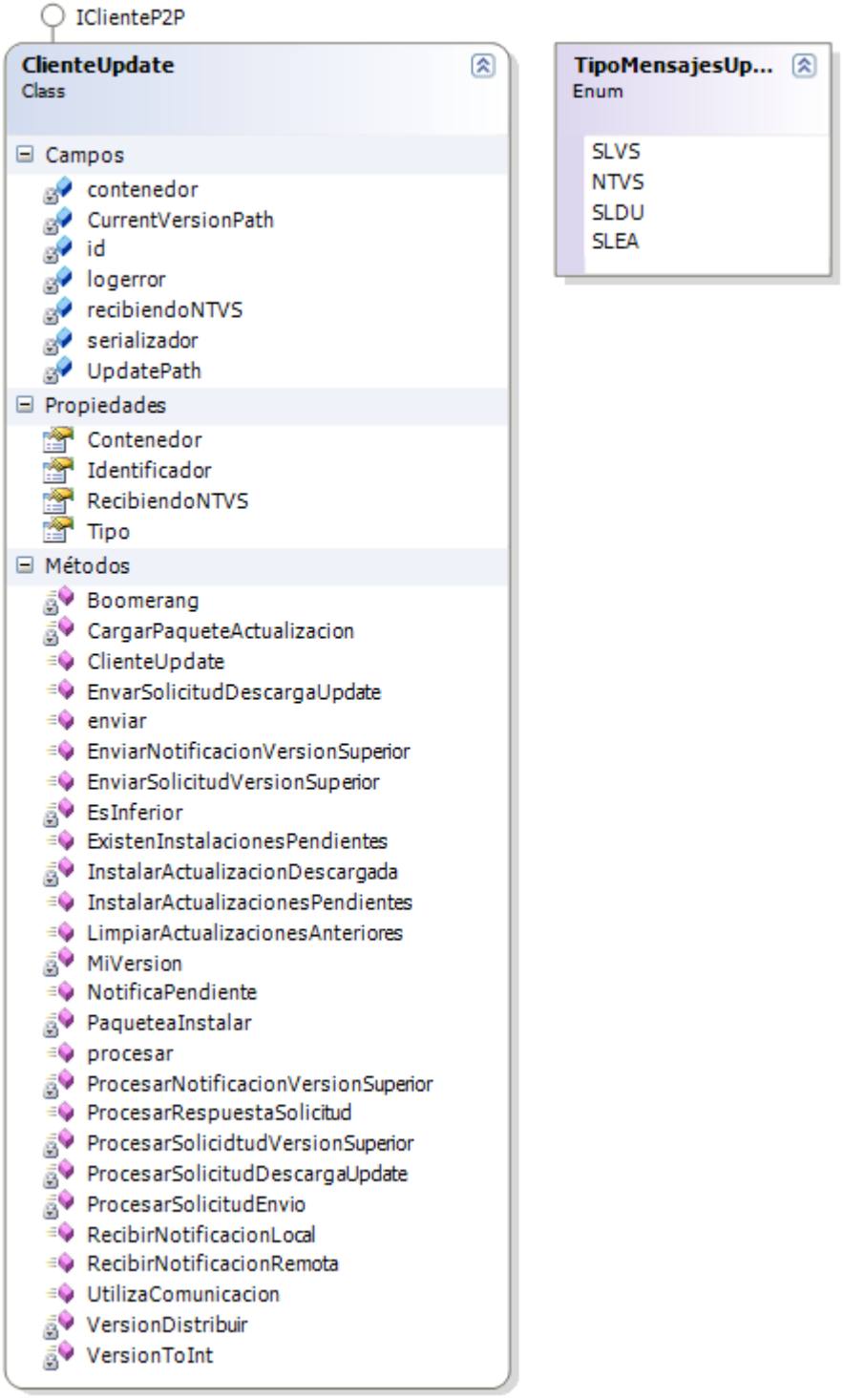


Diagrama 56

Este Cliente brinda de una gran versatilidad a las estaciones de trabajo en lo referente a la **Distribución de Paquetes de Actualización**. Ya que permite la transmisión automática de actualizaciones utilizando servicios de otros clientes en el **Contenedor de Clientes,** si son requeridos.

Esquema de Comunicación:

Este cliente en particular carece de una comunicación TCP o UDP propia ya que para sus tareas hace uso de los servicios que prestan otros clientes del **Contenedor de Clientes.** Los servicios del **ClienteMensajes** para el envió de **Mensajes Remotos** y los servicios del **Cliente Gestor de Transferencias** para las transferencias de los paquetes de actualización si es necesaria (Tabla 19).

ClienteUpdate – Protocolo de Entendimiento				
No	***Tipo de Mensaje***	***Descripción***	***Objeto Serializado Adjunto (Datos)***	***Posibles Mensajes de Respuesta.***
1	SLVS	Solicitud de Version Superior	MiVersion	-
2	NTVS	Notificación de Version Superior.	Miversion	NTVS
3	SLDU	Solicitud de Descargar Update	-	SLEA
4	SLEA	Solicitud de Envio de Archivo.	SolicitudEnvioArchivo	-

Tabla 19

Esquema de Funcionamiento:

Luego de 10 segundos de iniciada una sesión en una estación de trabajo **SCIP2P,** El **ClienteUpdate** envía un **MensajeRemoto** de Solicitud de Versión Superior a toda la red, los contactos con versiones superiores responden, si

existe al menos un contacto con una versión superior envía un **MensajeRemoto** de respuesta de tipo Notificación de Versión Superior.

Cuando el **ClienteUpdate** recibe este mensaje, notifica al **Cliente Usuario** (Diagrama 57) que cree una nueva solicitud para que el Usuario de la estación SCIP2P decida si desea descargar el paquete de Actualización.

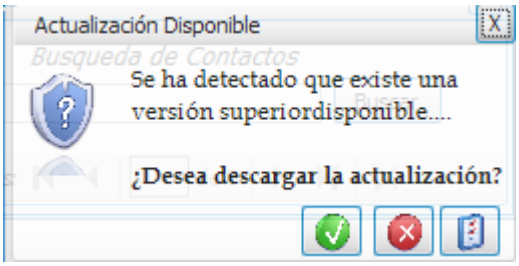


Diagrama 57

Si decide hacerlo responde con un **MensajeRemoto** de Solicitud de Descarga de Archivo. Este mensaje es recibido por el **ClienteUpdate** remoto que inicia el envió del paquete de actualización utilizando el **Gestor de Transferencias** (diagrama 58)**.**

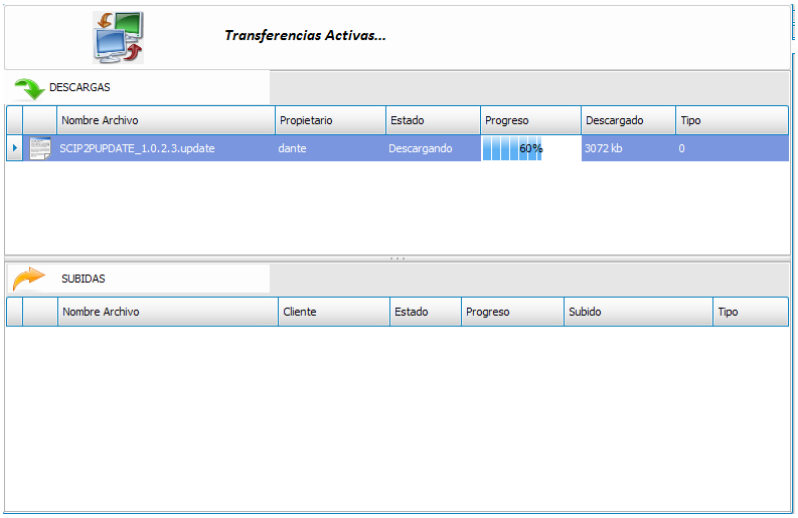


Diagrama 58

Cuando se finaliza la descarga del paquete de actualización, el **ClienteUpdate** notifica nuevamente al **ClienteUsuario** para que cree una nueva solicitud y el usuario decida si desea instalar las actualizaciones

descargadas. Si decide no hacerlo en ese momento, podrá hacerlo luego al intentar buscar una nueva actualización le aparecerá el mensaje indicando que existen actualizaciones pendientes de instalar (Diagrama 59).

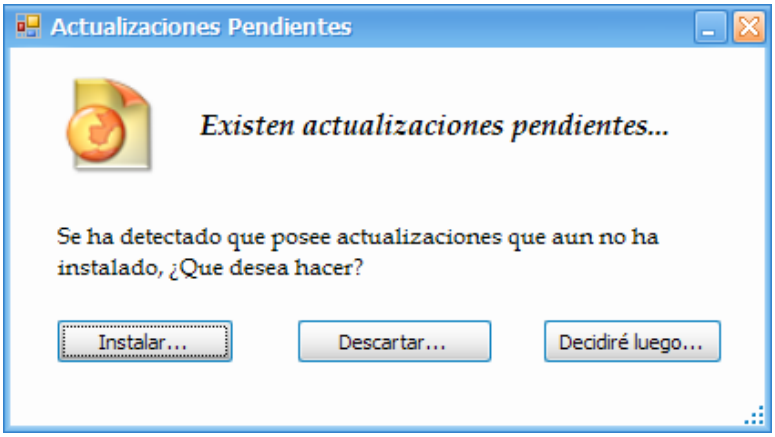


Diagrama 59

4.2.3 Instalación de Paquete de Actualización.

La instalación de los paquetes de actualización, está a cargo de una aplicación externa, a la estación de trabajo SCIP2P de nombre **SCIP2PUpdater** (Diagrama 60). Esto debido a que de ser capaz de cerrar la estación de trabajo, remplazar archivos de la aplicación y volver a cargarla al finalizar la instalación. Esto no sería posible hacerlo desde el mismo ensamblado en el que se ejecuta la estación de trabajo SCIP2P.

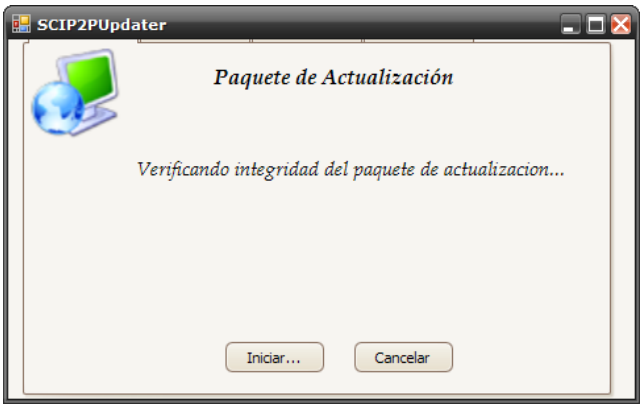


Diagrama 60

Cuando se decide instalar un paquete de actualización lo primero que se hace es notificar que para hacerlo deberá cerrarse la estación de trabajo SCIP2P, si se decide continuar, el siguiente paso es comprobar la integridad el paquete de actualización (Diagrama 61).

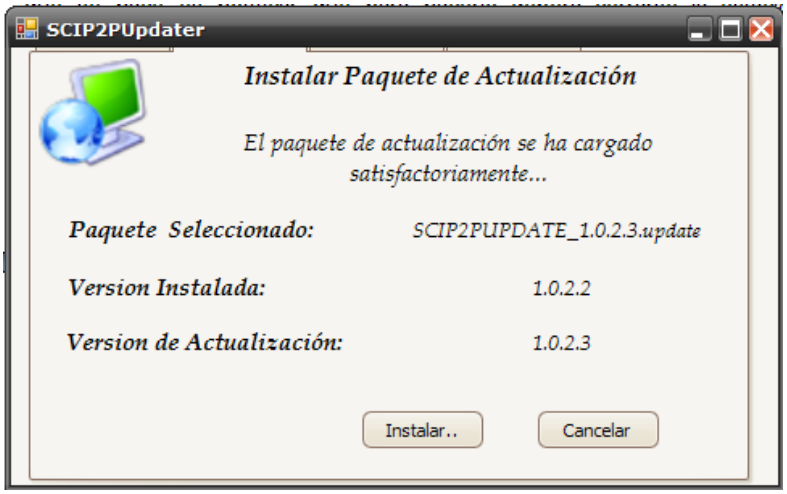


Diagrama 61

Una vez verificada la integridad del paquete y que la versión es una versión superior a la actualmente instalada, se permite proceder a instalar el paquete de actualización (Diagrama 62).

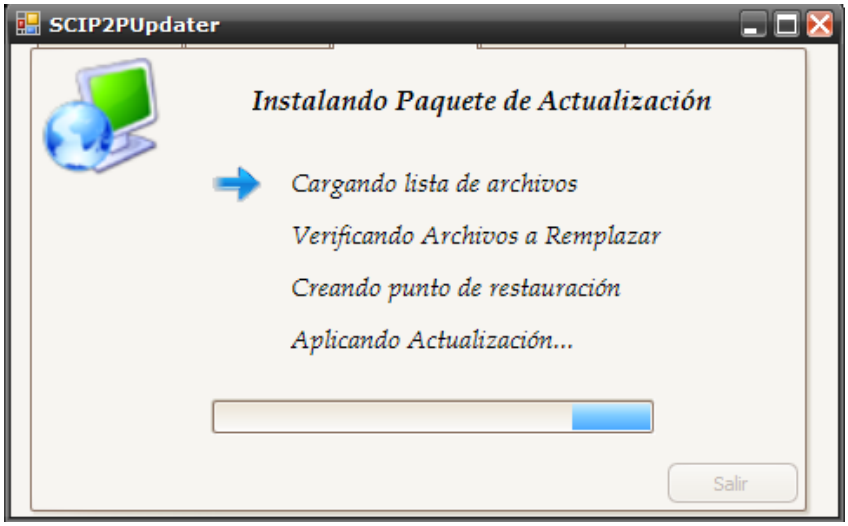


Diagrama 62

El proceso de instalación realiza varias tareas:

- ✓ **Cargando lista de archivos:** Carga el paquete de actualizaciones es decir la instancia de la clase **SCIP2PUpdate** y verifica la integridad de la lista de instancias **ArchivoUpdate.**

- ✓ **Verificando Archivos a Remplazar:** Genera un listado de los archivos que deberán ser remplazados.

- ✓ **Creando Punto de Restauración:** se genera un paquete de respaldo, similar a un paquete de actualización pero con los archivos actuales.

- ✓ **Aplicando Actualización:** Luego se inicia el proceso de generar y remplazar las instancias **ArchivoUpdate** en su posición correspondiente.

Finalmente al terminar el proceso de actualización se realiza una notificación al usuario que la actualización ha sido exitosa y que si desea cargar la estación de trabajo SCIP2P (Diagrama 63).

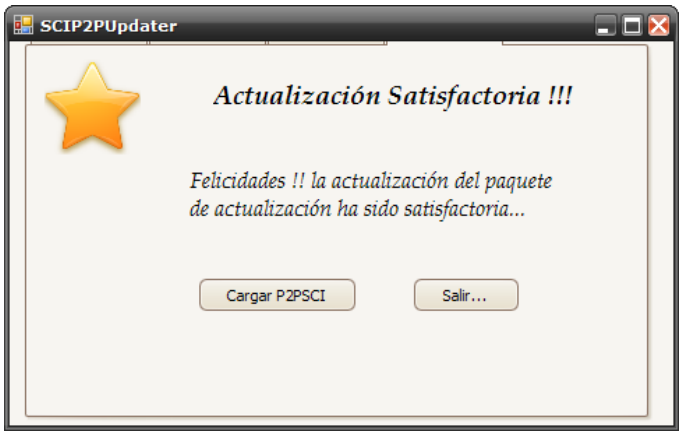


Diagrama 63

Además existe la opción de instalar un paquete de actualización desde un archivo, brindado la posibilidad distribuir el paquete de actualización por el medio que desee el usuario.

4.1.3 Aplicativos

4.1.3.1 Gestor de Documentos

Este aplicativo hace uso de las clases **Campo** (Diagrama 64)**, ConstructorDocumento** (Diagrama 65)**, Documento** (Diagrama 66)**, PersistenciaVariables** (Diagrama 67)**, Plantilla** (Diagrama 68) **y variable** (Diagrama 69). Y sirve para la creación de documentos. Para crear un documento este aplicativo lo guarda en la ubicación que el usuario asi lo desee y le coloca una extensión RTF

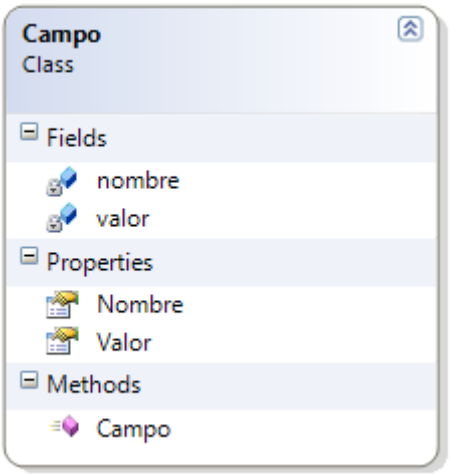


Diagrama 64

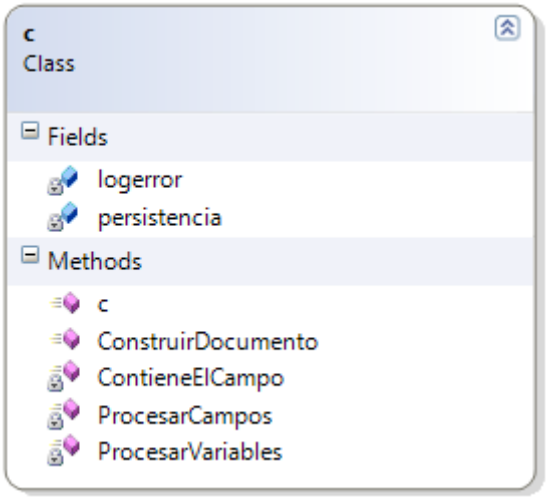


Diagrama 65

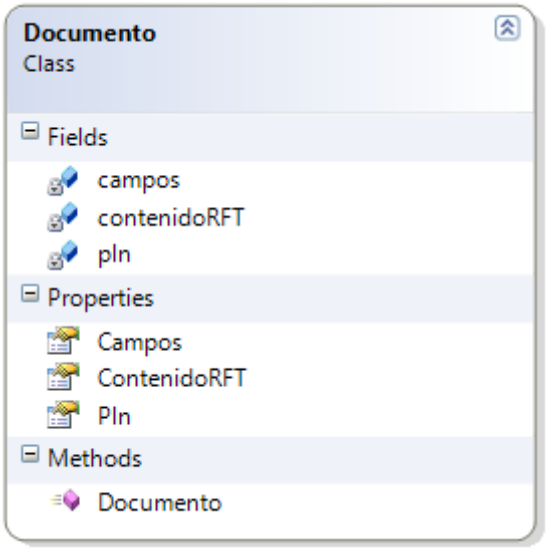


Diagrama 66

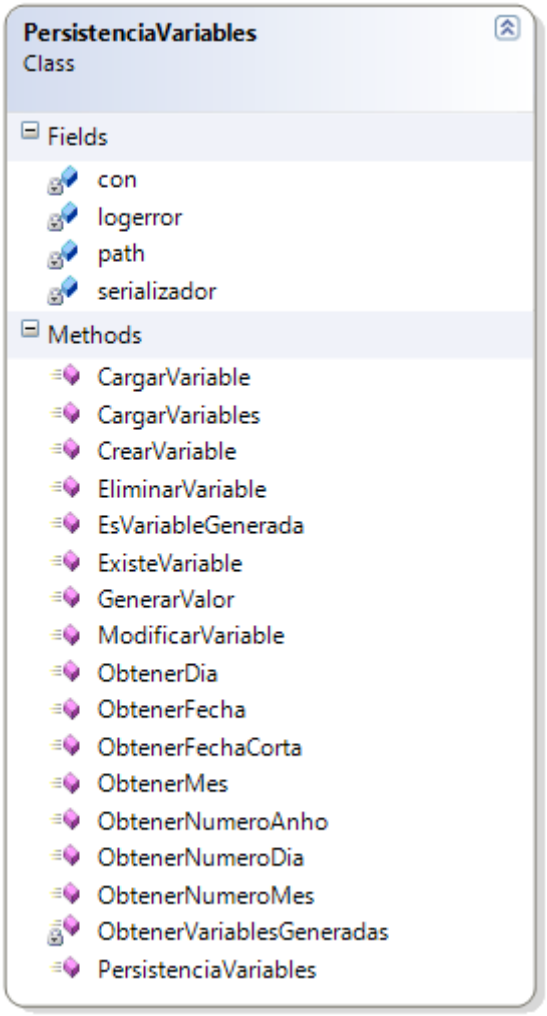

PersistenciaVariables
Class
Fields
con
logerror
path
serializador
Methods
CargarVariable
CargarVariables
CrearVariable
EliminarVariable
EsVariableGenerada
ExisteVariable
GenerarValor
ModificarVariable
ObtenerDia
ObtenerFecha
ObtenerFechaCorta
ObtenerMes
ObtenerNumeroAnho
ObtenerNumeroDia
ObtenerNumeroMes
ObtenerVariablesGeneradas
PersistenciaVariables

Diagrama 67

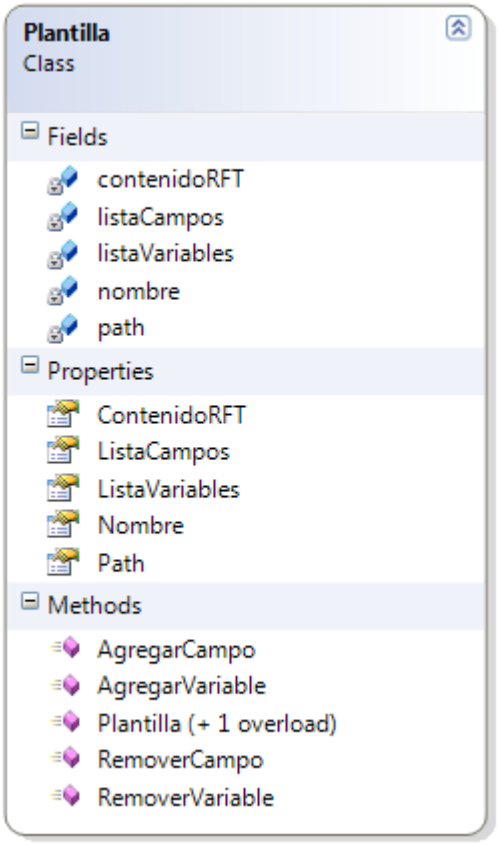


Diagrama 68

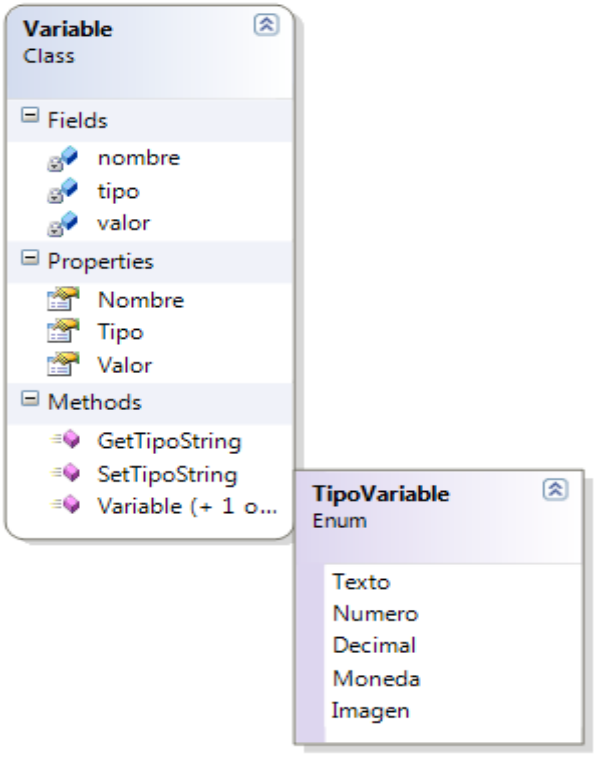


Diagrama 69

Utiliza una base de datos en la que se guardar la información con respecto a las variables que se van a estar utilizando (Tabla 20).

Variable			
Campo	Tipo	Llave	Requerida
nombre	Varchar (50)	SI	SI
tipo	Varchar (2)		NO
valor	Text		NO

Tabla 20

Funcionamiento:

Está dividido en tres elementos principales:

- Plantillas

Las clases que se utilizan es ***EditorPlantillas*** y se encarga de crear plantillas, estas contienen: nombre, contenido RTF dos listas doblemente enlazadas en las que se guardan los campos y las variables.

Además, se guarda la ruta de donde se cargo la plantilla. Esta se marca como serializable. Y esto permite crear una instancia de la clase platilla serializada como archivo extensión .pln

- Generación de Documentos.

Se utiliza la clase ***constructor documento***, y permite tomar una plantilla para generar el documento seleccionado, obteniendo los valores de las base de datos y solicitando los datos requeridos al usuario. Formando finalmente una instancia de clase ***documento***.

- Documentos en Si

Tiene referencia a la clase plantilla q sirvió como base para poderlo contruir, el contenido (texto rtf) y tiene la lista de los valores de los campos. Lo que permite generar el documento en cualquier momento.

4.1.3.2 Sistema de Recursos Audio Visuales

Este aplicativo está orientado a dar soporte al manejo de inventario de los recursos audiovisuales: proyectores, laptops, reguladores, etc. del Departamento de Ingeniería y Arquitectura (Diagrama 70).

Diagrama 70

El aplicativo se divide en dos módulos principales el de **Administración** orientado a tareas de gestión de los recursos audiovisuales Agregar, Modificar Registros, dar de baja los recursos, etc. (Diagrama 71).

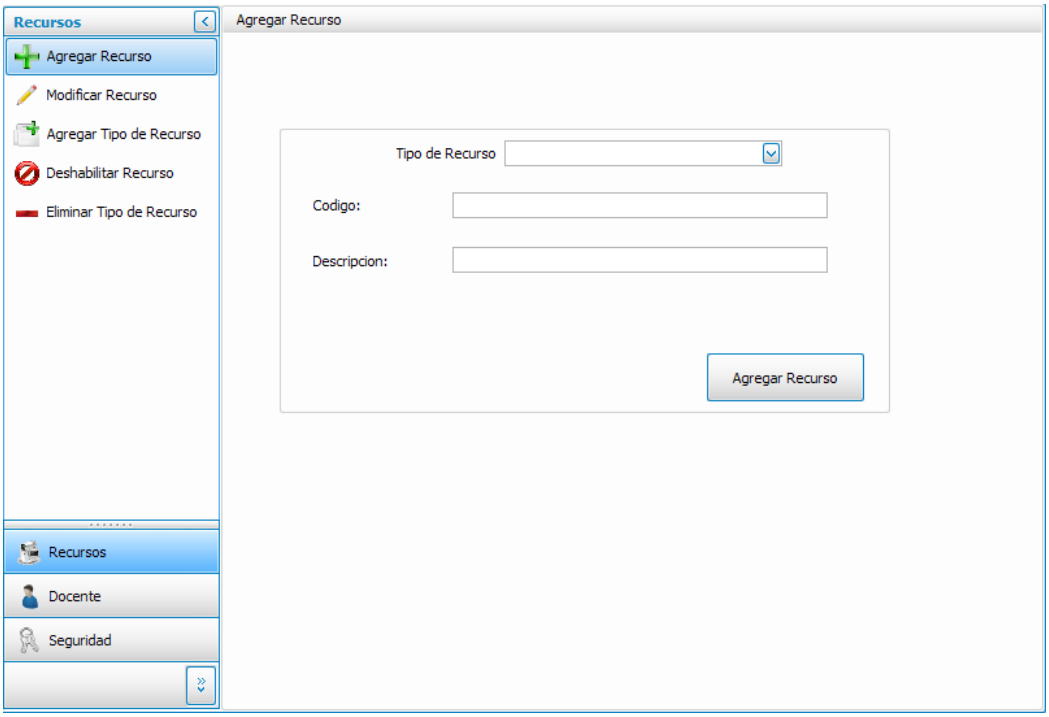


Diagrama 71

Es este modulo el encargado de llevar el inventario de los recursos audiovisuales del departamento.

El **Modulo de Reservas** (Diagrama 72), que permite manejar las reservas, entregas y devoluciones de los equipos audiovisuales que los docentes gestionan con la jefatura del departamento.

Esto permite llevar un control más adecuado, sencillo y practico de las reservas de equipo y el seguimiento en la entrega y devolución de los equipos.

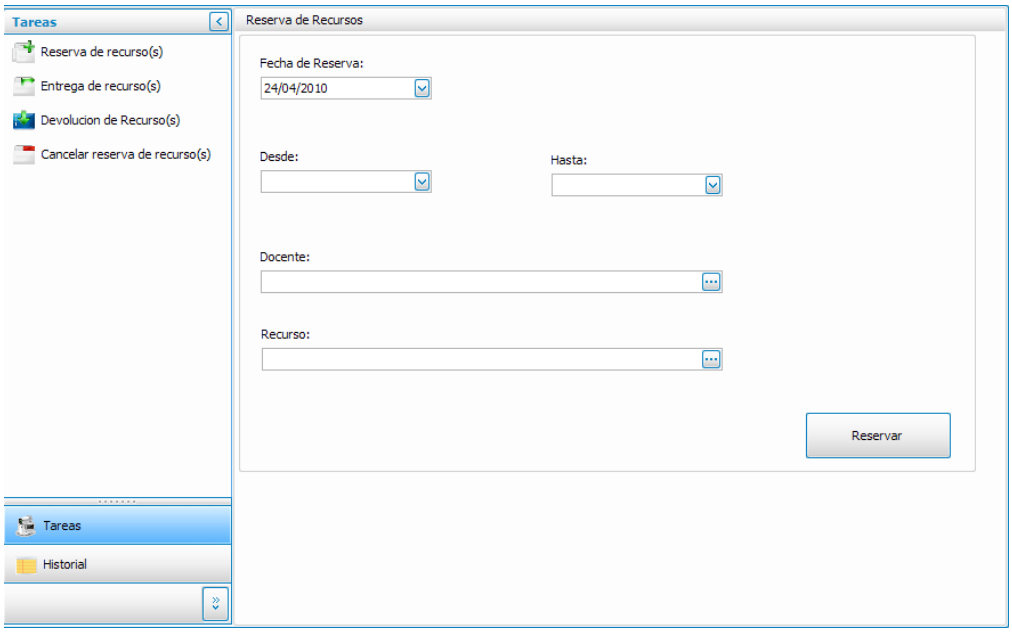


Diagrama 72

Para el manejo de toda la información que se introduce en los formularios que se presentan. Se lleva una base de datos (Tabla 21)

administrador			
Campo	Tipo	Llave	Requerido
password	NVarChar (10)	Primaria	SI

Carrera			
Campo	Tipo	Llave	Requerido
codigo	NVarChar (100)	Primaria	SI
descripcion	NVarChar (100)		NO

Docente			
Campo	Tipo	Llave	Requerido
nombre	NVarChar (100)		NO
apellido	NVarChar (100)		NO
cod	NVarChar (10)	Primaria	SI
carrera	Integer		NO

devolucion			
Campo	Tipo	Llave	Requerido
cod_dev	Integer	Primaria	SI
Cod_entrega	Integer		NO
hora	Time		NO

entrega			
Campo	Tipo	Llave	Requerido
Cod_entrega	Integer	Primaria	SI
Cod_reserva	Integer		NO
Hora_entrega	Time		NO
responsable	Text		NO

recurso			
Campo	Tipo	Llave	Requerido
Cod_recurso	NVarChar (10)	Primaria	SI
descripcion	NVarChar (100)		NO
cod_tipo	NVarChar (10)		NO
disponible	Boolean		NO

reserva			
Campo	Tipo	Llave	Requerido
cod_reserva	Integer	Primaria	SI
fecha	Date		NO
hora_ini	NVarChar (10)		NO
hora_fin	NVarChar (10)		NO
cod_docente	NVarChar (255)		NO
estado	NVarChar (10)		NO
cod_recurso	NVarChar (255)		NO

tipo			
Campo	Tipo	Llave	Requerido
cod_tipo	Integer	Primaria	SI
descripcion	Date		NO

Tabla 21

4.2 Diseño de la seguridad de la Aplicación.

Cada estación de trabajo SCIP2P puede contener información incluyendo todos los datos personales de un usuario, así como cualquier documentación o recurso de software que documentos, videos, programas, etc. Que se permiten contener para almacenar y hacer circular en la red. Por tanto como cualquier sistema de comunicación que trabaje conectado a una red deben estar protegidos.

Generalmente, la seguridad informática consiste en garantizar que el material y los recursos de software se usen únicamente para los propósitos para los que fueron creados y dentro del marco previsto.

La seguridad informática se resume, por lo general, en cinco objetivos principales:

- **Integridad:**

Garantizar que los datos sean los que se supone que son. Esto se logra con una programación complemente orientada a objetos, en donde cada mensaje enviado por la red, constituye un objeto, en nuestro caso concreto un **MensajeRemoto.**[22]

Con su estructura especifica, única y peculiar, permite encapsular cualquier mensaje enviado de una estación de trabajos SCIP2P. Y solamente los datos correctos podrán ser convertidos en la estación de trabajo receptora a un **MensajeRemoto**, para ser procesado. Ningún mensaje de terceros puede imitar un **MensajeRemoto u** objetos adjuntados como datos, con la misma estructura, tamaño y más importante aún con el mismo identificador de ensamblado, al cual pertenecen estas clases.

[22] Para mayor información sobre la clase **MensajeRemoto,** consultar el tema *¿Qué es un mensaje Remoto.* abordado anteriormente.

En él caso de las comunicaciones TCP gracias a las verificaciones de checksum en el envió de segmentos, se logra garantizar la integridad de los datos transferidos.

- **Confidencialidad:**

Asegurar que sólo los individuos autorizados tengan acceso a los recursos que se intercambian.

Esto se logra de tres formas en toda estación de trabajo SCIP2P:

a. **Encapsulamiento:** una de las virtudes de la programación orientada a objetos en donde cada unidad o modulo de la aplicación, en este casos sus clases, exponen al exterior simplemente los mecanismos para facilitar sus servicios, evitando ser vulnerados o alterados.

b. **Seguridad en la Persistencia:** Todas las bases de datos u archivos con información importante, datos personales, contraseñas, etc. Son encriptados utilizando una clase utilitaria principalmente orientada a dar seguridad a las estaciones de trabajo SCIP2P.

c. **Seguridad en Acceso a la Información**: este último aspecto principalmente orientado a la *autenticación*, garantizando que sólo los individuos autorizados tengan acceso a los recursos. Por tanto a pesar que claves de acceso a sesiones, además de estar contenidas en un **archivo encriptado**, posee su propio mecanismo de encriptamiento **HASH MD5** agregando un peldaño más de seguridad en el acceso a los datos.

Todos los aspectos de seguridad orientados a dar soporte a la **Confiabilidad**, son tarea de la clase **Encriptador** (Diagrama 73).

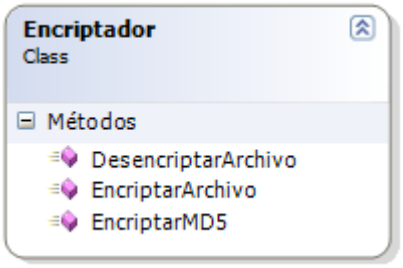


Diagrama 73

```
namespace P2P.AccesoDatos.Encriptamiento
{
   public class Encriptador
   {

      public String EncriptarMD5(String cadena)
      {
         try
         {
               MD5CryptoServiceProvider encriptadorMD5 = new
        MD5CryptoServiceProvider();
            byte[] pass = System.Text.Encoding.UTF8.GetBytes(cadena);
            byte[] encrip = encriptadorMD5.ComputeHash(pass);
            String Hash = BitConverter.ToString(encrip);
            string[] ArrayHash = Hash.Split('-');
            StringBuilder EncodedHash = new StringBuilder();

            for (int i = 0; i < ArrayHash.Length; i++)
            {
               EncodedHash.Append(ArrayHash[i]);
            }

            return EncodedHash.ToString();
         }
         catch (Exception ex) {
            return "";
         }
      }



      public bool EncriptarArchivo(String Origen, String Destino, String Clave) {

         try
         {

               FileStream fsInput = new FileStream(Origen, FileMode.Open,
               FileAccess.Read);
```

```
                FileStream fsEncrypted = new FileStream(Destino,
                FileMode.Create, FileAccess.Write);

            DESCryptoServiceProvider DES = new DESCryptoServiceProvider();
            DES.Key = ASCIIEncoding.ASCII.GetBytes(Clave);
            DES.IV = ASCIIEncoding.ASCII.GetBytes(Clave);
            ICryptoTransform desencrypt = DES.CreateEncryptor();
                CryptoStream cryptostream = new CryptoStream(fsEncrypted,
                desencrypt, CryptoStreamMode.Write);
            byte[] bytearrayinput = new byte[fsInput.Length - 1];
            fsInput.Read(bytearrayinput, 0, bytearrayinput.Length);
            cryptostream.Write(bytearrayinput, 0, bytearrayinput.Length);
            return true;
        }
        catch (Exception ex) {
            System.Windows.Forms.MessageBox.Show(ex.Message);
            return false;
        }

    }


    public bool DesencriptarArchivo(String Origen, String Destino, String
Clave) {
        try
        {
          DESCryptoServiceProvider DES = new DESCryptoServiceProvider();
              //Se necesita una clave de 64 bits y un IV para este proveedor.
              //Establecer la clave secreta para el algoritmo DES.
            DES.Key = ASCIIEncoding.ASCII.GetBytes(Clave);
            //Establecer el vector de inicialización.
            DES.IV = ASCIIEncoding.ASCII.GetBytes(Clave);

                //Crear una secuencia de archivo para volver a leer el archivo
cifrado.
                  FileStream fsread = new FileStream(Origen, FileMode.Open,
                  FileAccess.Read);
            //Crear un descriptor de DES desde la instancia de DES.
            ICryptoTransform desdecrypt = DES.CreateDecryptor();
                //Crear una secuencia de cifrado para leer y //realizar  una
                transformación de cifrado DES en los bytes de entrada.
                 CryptoStream cryptostreamDecr = new CryptoStream(fsread,
                desdecrypt,    CryptoStreamMode.Read);
           //Imprimir el contenido del archivo descifrado.
           StreamWriter fsDecrypted = new StreamWriter(Destino);
           fsDecrypted.Write(new
StreamReader(cryptostreamDecr).ReadToEnd());
           fsDecrypted.Flush(); fsDecrypted.Close();
```

```
            return true;
        }
        catch (Exception ex) {
            System.Windows.Forms.MessageBox.Show(ex.Message);
            return false;
        }

    }

  }
}
```

CAPITULO V

DESARROLLO Y PRUEBA DE LA APLICACION

5.1 Interfaces

5.1.1 *ICLienteP2P*

```
public interface IClienteP2P
    {

    //Metodo que recibi el conjunto de Bytes recibidos por una
    //Comunicacion TCP o UDP escuchada por el Sistema de Comunicacion
    //para ser procesada segun el proposito del clienteP2P.

    void procesar(String idComunicacion, byte[] datos);

    void enviar(String idComunicacion, byte[] datos);

 //Mensajes de Control entre el Sistema de Comunicacion y los Clientes P2P

void RecibirNotificacionLocal(String idComunicacion, MensajeLocal mensaje, int
Nbytes);

//Mensajes de Control Remotos que se dan entre clientesP2P remotos del mismo tipo
void RecibirNotificacionRemota(string idcomunicacion, MensajeRemoto Mensaje);


void ProcesarRespuestaSolicitud(Solicitud solicitud, RespuestaSolicitud
respuesta);

//Identificador del Cliente P2P para ser reconocido y
//diferenciado dentro de un contenedor de clientes

String Identificador { get; set;}

//Caracter que permite identificar el tipo de Cliente
//M - Cliente Mensajes
//U - Cliente Usuario
//D - Cliente Directorio

Char Tipo { get;}


//Referencia a la instancia Contenedora o adminintradora
//de los clientesP2P en una estacion de trabajo o Host

ContenedorClientes Contenedor { get; set;}

//retorna true si el cliente recibe un id de comunicacion utilizado por el .

Boolean UtilizaComunicacion(String idComunicacion);


}
```

5.1.2 *IContacto*

```
namespace P2P.ClientesP2P.Directorio
{
   public interface IContacto
    {
       String NickName { get;set;}
       tipoContacto Tipo { get; set;}
       string IP { get; set;}
       estadoContacto Estado { get; set; }

    }
}
```

5.1.3 *IPerfil*

```
namespace P2P.ClientesP2P.Directorio
{
   public interface IPerfil
    {
       string NickName { get; set;}
       DateTime UltimaActualizacion { get; }
       String Nombres { get; set;}
       String Apellidos { get; set;}
       String Email { get; set;}
       String Telefono { get; set;}
       String Celular { get; set; }
       Image Avatar { get; set; }
    }
}
```

5.2 Componentes

5.2.1 Lista - Interfaz de Cliente Directorio

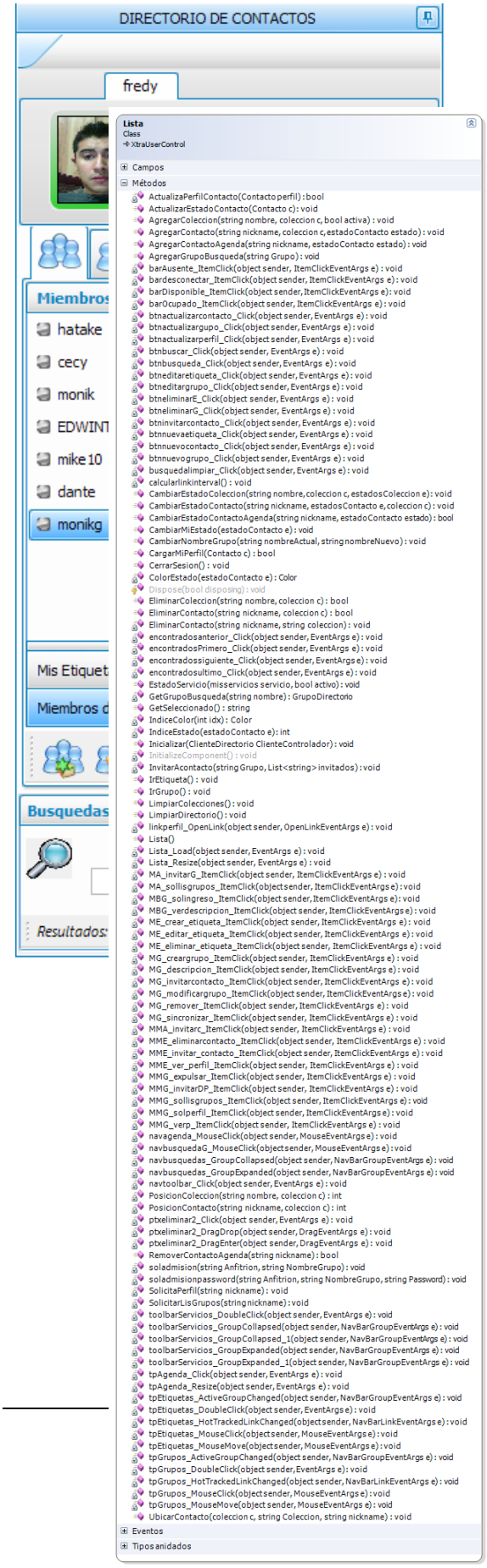


Diagrama 74

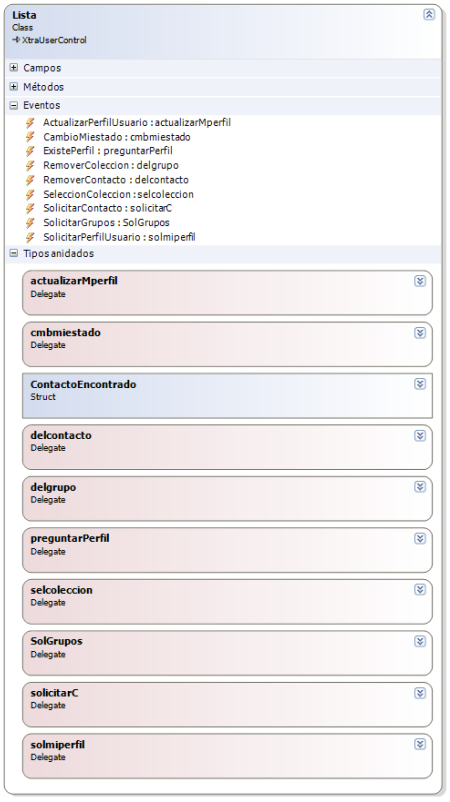

Lista
Class
XtraUserControl
Campos
Métodos
Eventos
ActualizarPerfilUsuario : actualizarMperfil
CambioMiestado : cmbmiestado
ExistePerfil : preguntarPerfil
RemoverColeccion : delgrupo
RemoverContacto : delcontacto
SeleccionColeccion : selcoleccion
SolicitarContacto : solicitarC
SolicitarGrupos : SolGrupos
SolicitarPerfilUsuario : solmiperfil
Tipos anidados
actualizarMperfil
Delegate
cmbmiestado
Delegate
ContactoEncontrado
Struct
delcontacto
Delegate
delgrupo
Delegate
preguntarPerfil
Delegate
selcoleccion
Delegate
SolGrupos
Delegate
solicitarC
Delegate
solmiperfil
Delegate

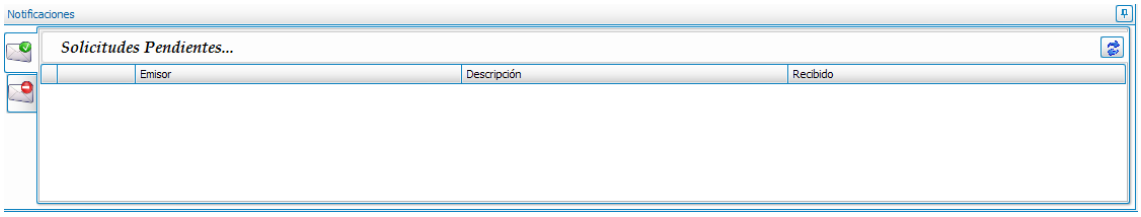
Diagrama 75

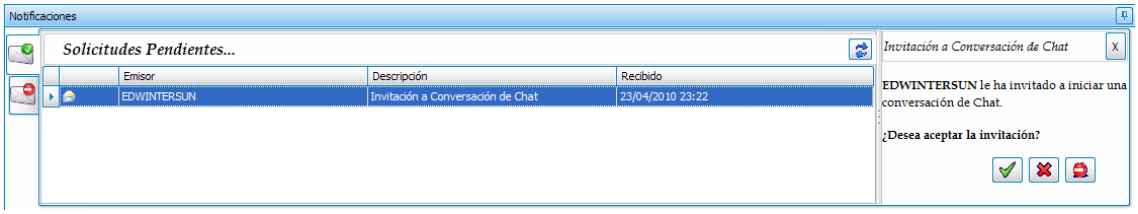
5.2.2 Control de Notificaciones – NotifyControl

```
namespace P2P.Notify
{
    public partial class NotifyControl : Component
    {
        private List<NotifyDevExpressForm> notificaciones;
        private int timeout;
        public int Timeout
        {
            get { return timeout; }
            set { timeout = value; }
        }
        private int y,x;
        private string logerror;

        public delegate void botonesclick(String ButtonName, int NotificacionId,
       Object Data);
        public event botonesclick BotonesNotificacionClickEvent;

        public delegate void initform(NotifyDevExpressForm Notificacion);
        public event initform ShowNotificacionEvent;

        public delegate void closeform(CloseReason rason, int NotificacionId,
       Object Datos);
        public event closeform CloseNotificacionEvent;

        public delegate void sinnotificaciones();
        public event sinnotificaciones InicializarRequerido;


        public NotifyControl()
        {
            InitializeComponent();
            timeout = 400;
            x= Screen.PrimaryScreen.WorkingArea.Width-260;
            y = Screen.PrimaryScreen.WorkingArea.Height-130;
            notificaciones = new List<NotifyDevExpressForm>();

        }

        public NotifyControl(IContainer container)
        {
            container.Add(this);
            timeout = 400;
            x = Screen.PrimaryScreen.Bounds.Width -260;
            y = Screen.PrimaryScreen.WorkingArea.Height- 130;
            notificaciones = new List<NotifyDevExpressForm>();
            InitializeComponent();
        }


        public void NuevaNotificacion(int IdNotificacion, String Titulo, String
       Mensaje, NotifyFormType tipo, Object Datos, String Skin) {

                   NotifyDevExpressForm noti = new
       NotifyDevExpressForm(IdNotificacion, Titulo, Mensaje, tipo,
       Datos,timeout, Skin);
            CorrerHaciaArriba();
            notificaciones.Add(noti);
            noti.CloseEvent += new NotifyDevExpressForm.closeform(this.AlCerrar);
            noti.ShowEvent += new NotifyDevExpressForm.initform(this.AlCargar);
             noti.BotonesClick += new
       NotifyDevExpressForm.botonesclick(this.AlClickear);
            noti.Mostrar(x, y);

        }



        private void CorrerHaciaArriba() {
```

```
                for (int i = 0; i < notificaciones.Count; i++)
                {
                    int actx, acty;
                    acty = notificaciones[i].Location.Y;
                    actx = notificaciones[i].Location.X;
                    notificaciones[i].Location = new Point(actx, acty -135);
                }

            }

            private void AlCerrar(CloseReason rason, int NotificacionID, Object Data)
            {
                try
                {

                    for (int i = 0; i < notificaciones.Count; i++)
                    {

                        if (NotificacionID == notificaciones[i].NotifyFormID)
                        {
                            notificaciones[i] = null;
                            notificaciones.RemoveAt(i);
                            break;
                        }
                    }

                    if (notificaciones.Count == 0) {
                        notificaciones = new List<NotifyDevExpressForm>();
                        InicializarRequerido();
                    }

                  CloseNotificacionEvent(rason, NotificacionID,Data);
              }
              catch (Exception ex)
              {
                  logerror = ex.Message;

              }
            }


            private void AlCargar(NotifyDevExpressForm Notificacion)
            {
                try {
                    ShowNotificacionEvent(Notificacion);
                }
                catch (Exception ex) {
                    logerror = ex.Message;
                }
            }


        private void AlClickear(String ButtonName, int  NotificacionID, Object Data) {
                try
                {
                    BotonesNotificacionClickEvent(ButtonName, NotificacionID, Data);

                    }
                    catch (Exception ex) {
                        logerror = ex.Message;
                    }
            }

        }

    }
```

5.2.3 guiSolicitudes – Interfaz de Control de Solicitudes.





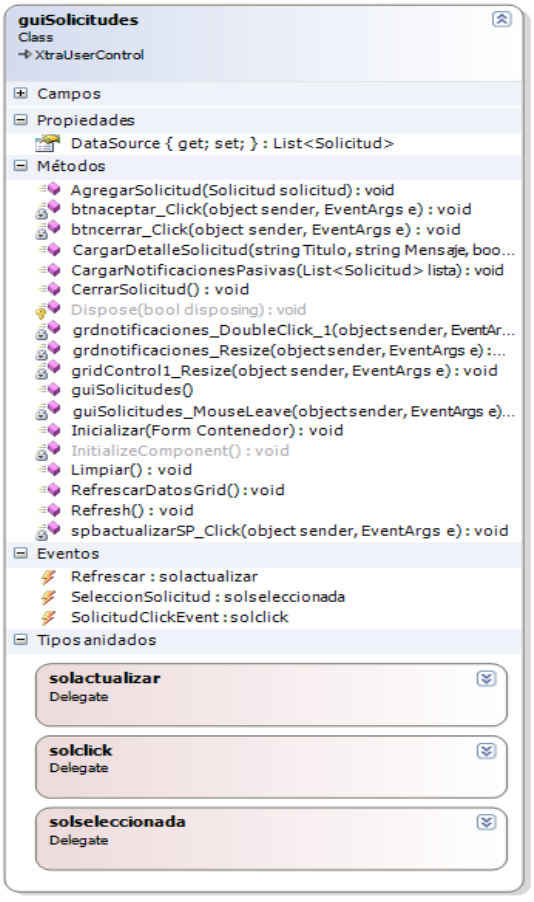


Diagrama 76

5.3 Espacios de Nombre de FrameworkSCIP2P.

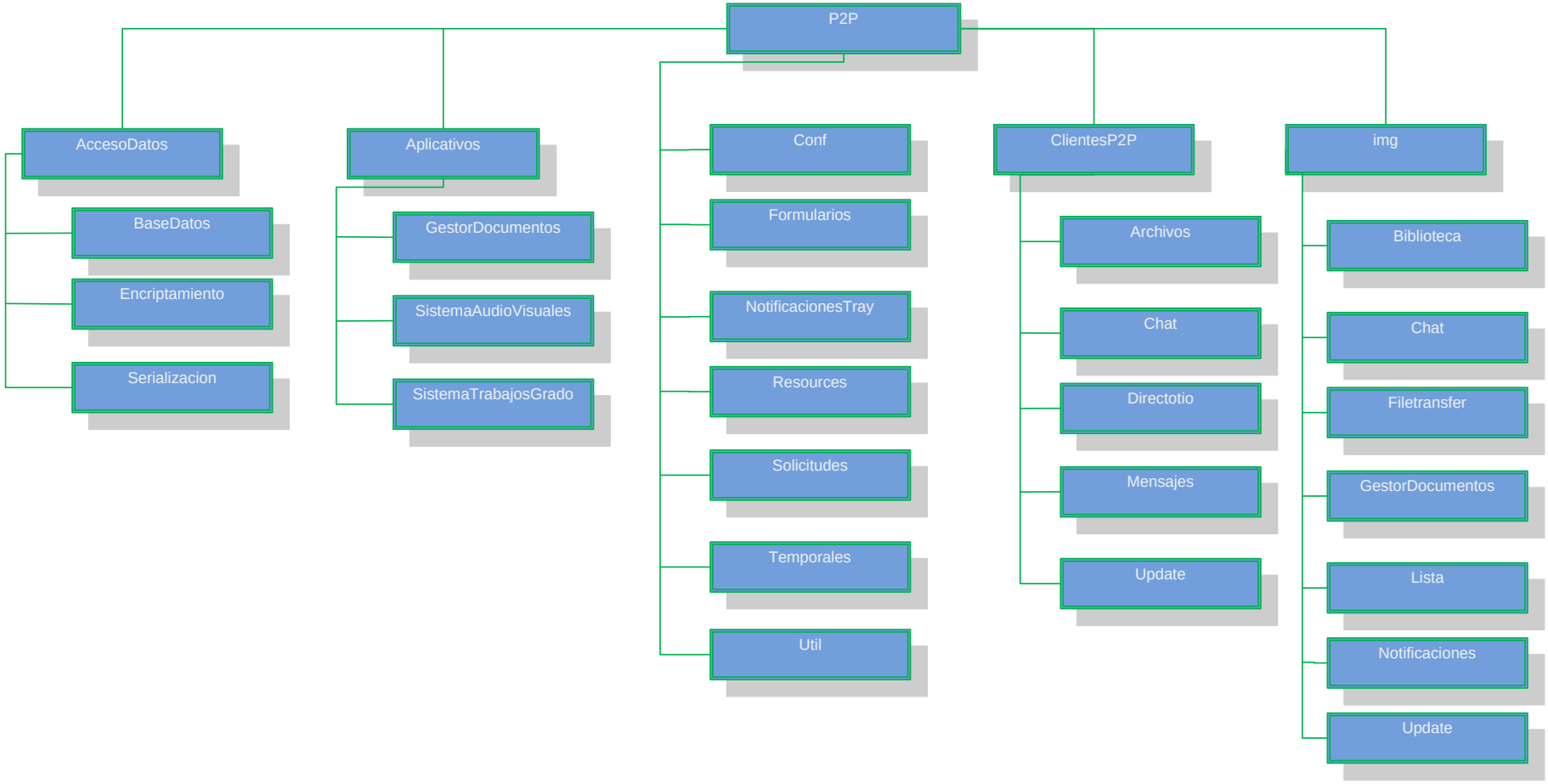
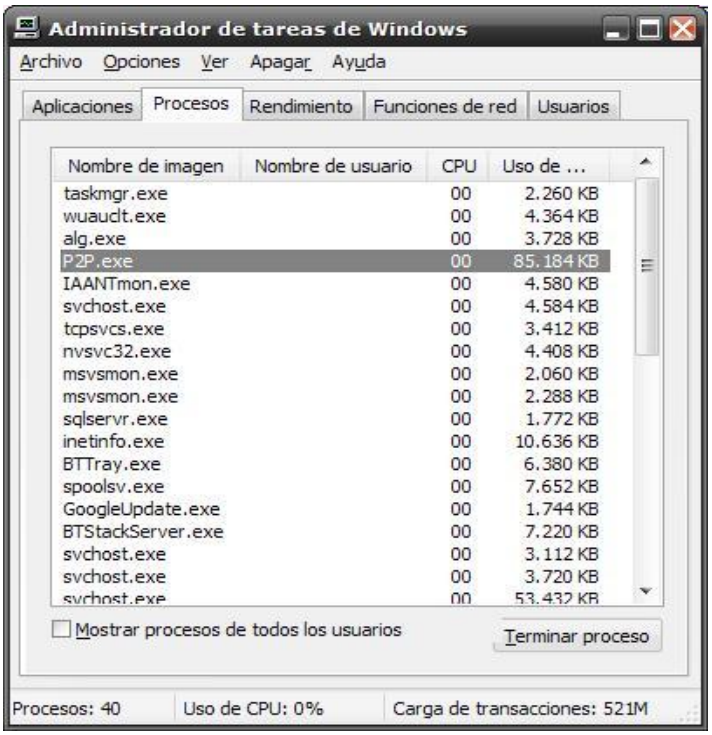


Diagrama 77

5.4 Pruebas de Rendimiento



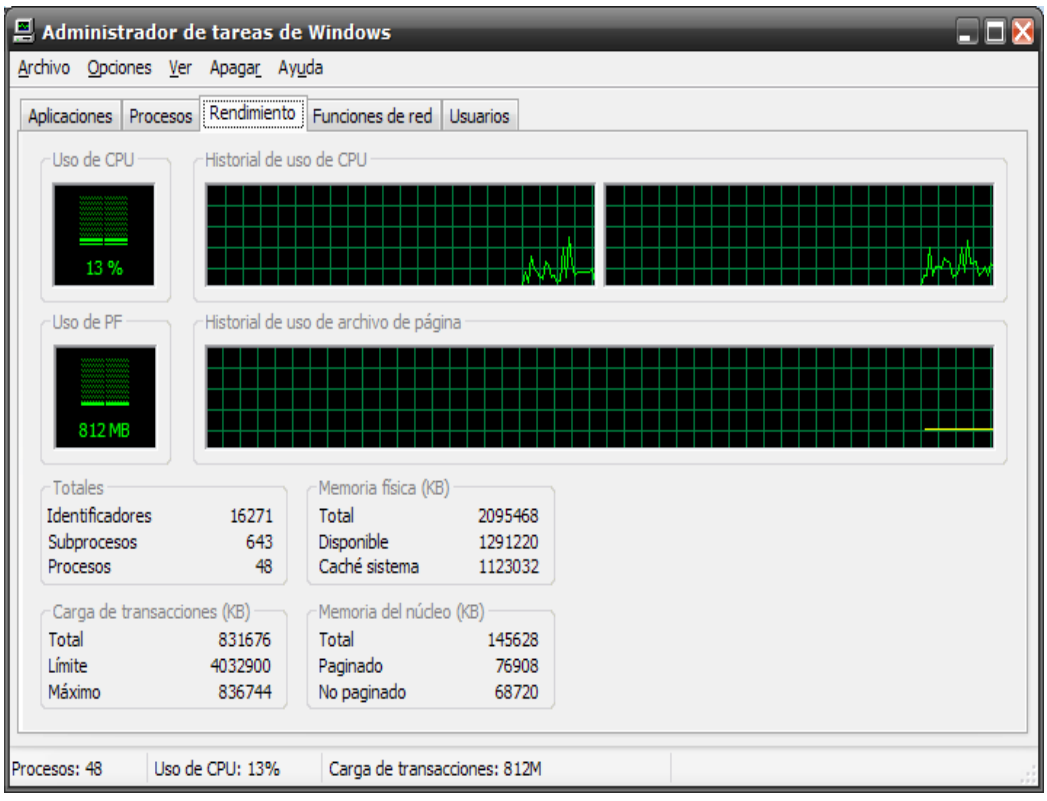


Diagrama 78

Un aspecto importante a contemplar, mas en un aplicativo que trabaja con hebras y subprocesos, es la cantidad de recursos que implica su ejecución y funcionamiento (Diagrama 78).

Por tanto se trató de identificar el estrés generado por el aplicativo al procesador y la cantidad de memoria RAM utilizada en algunos procesos considerados como los principales (Tabla 22).

La siguiente tabla, muestra los resultados de pruebas realizadas, un sistema operativo Windows XP SP3, procesador Inter Core 2 Duo 1.8 Mghz, y 2Gb RAM:

ACTIVIDAD EN UNA ESTACIÓN SCIP2P	Uso de CPU	Uso de Memoria RAM
Antes de Iniciar SCIP2P	0%	0
Cargando SCIP2P	20%	51
Luego de Iniciado SCIP2P (modo pasivo)	0%	51
Iniciando Sesión	22%	58
Luego de Iniciada Sesión (modo pasivo)	0%	37
Durante Envió y Recepción de MensajesRemotos	0%	70
Durante una conversación de Chat	5%	80
Durante la descarga de Archivos	2%	79
Durante el envió de archivos	2%	79
Utilizando Aplicativos	15%	71
Cerrando Sesión	3%	37
Descarga de SCIP2P	0%	0

Tabla 22

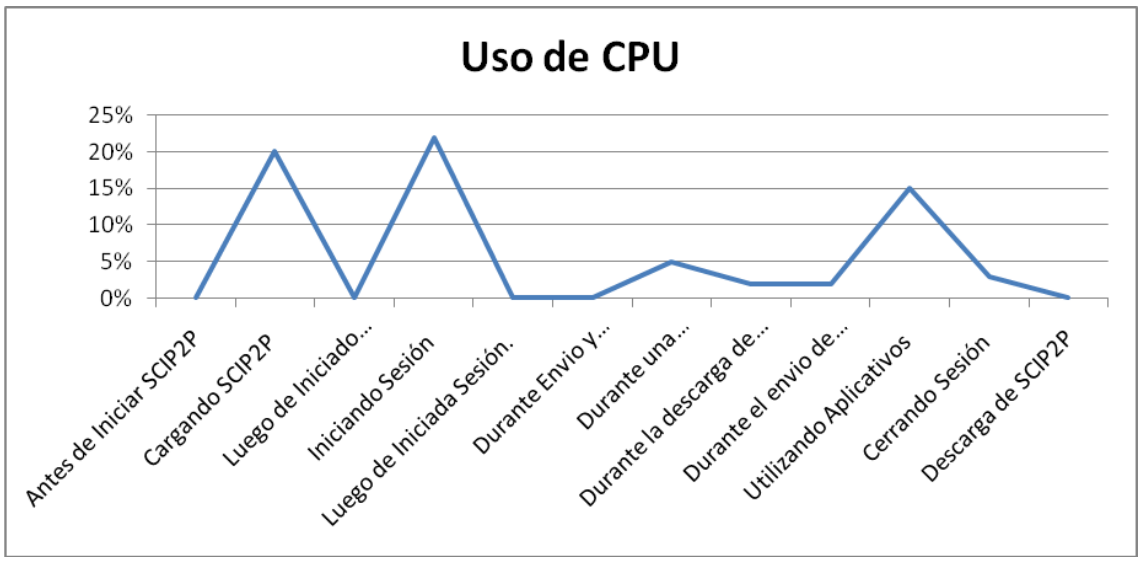


Diagrama 79

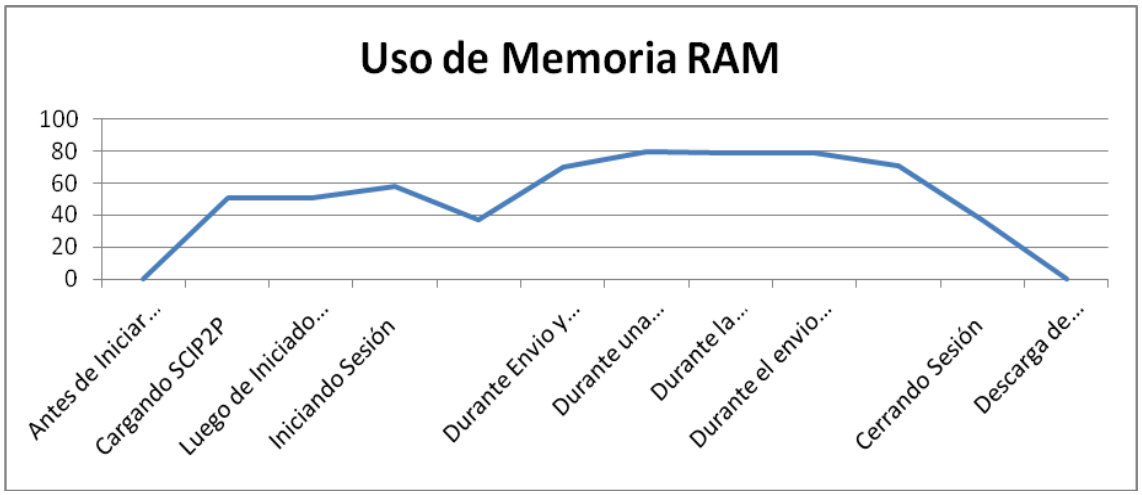


Diagrama 80

El promedio de uso se memoria RAM durante la ejecución de una estación de trabajo SCIP2P obtenido es de 61.3 MB con un promedio de 6% de estrés generado a las tareas del procesador.

Las implicaciones de un procesador con menos MHz, incidirá directamente en más memoria RAM durante la ejecución de la aplicación.

CAPITULO VI

PLAN DE IMPLEMENTACION DE LA APLICACION

Capítulo VI - Plan de Implementación de la Aplicación

6.1 Objetivos del plan de implementación

- Eliminar el temor al cambio, por parte de los usuarios del sistema.

- Ayudar al Departamento a hacer los procesos de comunicación más eficaces.

- Ayudar a las Entidades del Departamento a disminuir el tiempo de realización de las actividades que realizan, en lo que respecta a Formas que se proporcionan en secretaria como en la ayuda de equipo audio visual. Así también, ayudar a los docentes a llevar un orden por medio de bitácoras de los trabajos de grado que tienen asignados.

- Mostrar a los usuarios como se puede, reducir el tiempo de búsqueda de documentos, haciendo uso del aplicativo

6.2 Listado de actividades del plan de implementación

- Crear para los futuros usuarios del sistema, un Manual en el cual se describen las diferentes funciones que se pueden realizar con el SCIP2P.

- Brindar capacitaciones las veces y los días que sean solicitadas a los desarrolladores.

- Facilitar CDs que contengan el instalador del aplicativo.

6.3 Costo de Implementación

- Impresión del Manual de Usuario

Precio de cada impresión = 0.11 ctvs

N[umero de Paginas = 100

Número de Copias = 3

TOTAL = $33.00

- Copia de Cds

Precio de cada Cd ya quemado = 0.65

Número de Copias = 25

TOTAL = $16.25

- Capacitaciones

Hora de Trabajo de los Capacitadores = $3.00

Tiempo de Capacitacion = 3 horas

Cantidade de Capacitaciones, tentativas = 4

TOTAL = $45.00

- Propaganda por medio de carteles

Valor de cada cartel = $5.00

Se colocaran = 3

TOTAL = $15.00

Por lo que se tiene que la Implementación del Aplicativo tiene un Valor de $109.25

CAPITULO VII

CONCLUSIONES Y RECOMENDACIONES

Capítulo VII - Conclusiones y Recomendaciones

7.1 Conclusiones

Con el desarrollo del Sistema de Comunicación Integral Punto a Punto SCIP2P se ha logrado dar respuesta a la necesidad de una eficaz y eficiente comunicación, aspecto clave para el buen funcionamiento de toda organización.

Con las nuevas posibilidades de comunicación que brinda SCIP2P, el mantener una buena comunicación, crear grupos de trabajo o de estudio, distribuir información, archivos, etc.; dejará de ser una tarea difícil y engorrosa limitada en todo momento por el rendimiento de un servidor. Ya que con el desarrollo del presente sistema no se tendrá la necesidad de un servidor como intermediario y se logrará, por lo tanto, aprovechar mucho mejor las bondades de la infraestructura de red con que cuenta el Departamento de Ingeniería y Arquitectura.

Además de los beneficios implícitos en la puesta en marcha de este proyecto, se han sentado las bases para muchos otros proyectos que pueden hacer uso de este sistema de comunicación, gracias a su alta escalabilidad, las facilidades de integración, soporte para más funciones y aplicativos y que brinda un diseño orientado a objetos, abre una gran variedad de opciones.

Mucho más allá de todo beneficio obtenido, el desarrollo de este proyecto demuestra que es posible crear nuevas soluciones, desde cero incluso sin ningún precedente previo, dejar cualquier esquema o patrón de diseño predefinido y optar simplemente por ver con atención y descubrir la ilimitada gama de posibilidades.

Nuestros logros solamente están limitados, por el tamaño de nuestros sueños. Hay que soñar en grande y no permitir que nuestros sueños estén limitados por lo que nosotros pensamos que podemos o no hacer. El no puedo no existe, solo es una ilusión que no nos permite soñar y por tanto vivir con libertad.

7.2 Recomendaciones

Recomendaciones Para el Buen Uso del SCIP2P:

- ✓ Mantener actualizado los Sistemas Operativos Windows que se utilicen como estaciones de trabajo SCIP2P.

- ✓ Poseer instalados, actualizados e idealmente unificado antivirus en las estaciones de trabajo SCIP2P.

Recomendaciones para Futuros Trabajos de Grado:

- ✓ Permitir la unificación de los diferentes aplicativos y soluciones informáticas existentes, en una única herramienta, que incluya a SCIP2P como mecanismo de comunicación.

- ✓ Generar un mecanismo que permita al SCIP2P ser accesible entre las VLAN en las que esta segmentada la red de la FMO. Así también, ser accesible desde cualquier otra red.

- ✓ Buscar que futuros trabajos de grado orientados al desarrollo de aplicativos administrativos o de gestión para el Departamento de Ingeniería y Arquitectura se desarrollen integrados con el Sistema de Comunicación Integral P2P. Aprovechando todas las bondades de comunicación que ofrece y permitiendo al grupo del trabajo de grado dirigir sus esfuerzos en las funciones propias de los aplicativos en desarrollo.

- ✓ Incentivar los trabajos de grado orientados a ampliar la gama de Clientes de Comunicación, para el desarrollo de clientes de AudioLlamadas o VidieoLlamadas por citar ejemplos.

- ✓ Proponer como trabajo de grado el desarrollo de una estación de trabajo SCIP2P Linux Java, que permita la comunicación de estaciones de trabajo en lenguajes de programación diferentes C# y Java y más importante aun estaciones de trabajo Windows – Linux.

7.3 Bibliografía

- Elsevier - TCP IP Sockets in C#
  David B. Makofske
  Michael J. Donahoo
  Kenneth L. CAlvert

- Visual C# 2005
  ANAYA

- Ayuda del MSDN

ANEXOS

Anexos

Anexo 1

PROCESOS LEGALES SOBRE LA PROTECCIÓN A LA PROPIEDAD INTELECTUAL

El aplicativo SCIP2P , debido a la atribución que le confiere el Art. 29 del **REGLAMENTO GENERAL DE PROCESOS DE GRADUACION DE** L**A UNIVERSIDAD DE EL SALVADOR.** Esta protegido bajo la Ley de Fomento y Protección de la Propiedad Intelectual de la República de El Salvador, por lo cual está totalmente prohibido que sin el previo consentimiento de la Universidad se realicen aspectos como:

- Uso o venta del Software fuera de los procedimientos establecidos por la Universidad.
- Copiar o reproducir el Software pretendiendo comercializarlo.
- Usarlo en un equipo diferente al establecido sin el consentimiento del propietario.
- Esta Licencia incluye su uso en el equipo convenido.
- Modificar, cambiar o pretender utilizar el concepto vertido en el software con propósitos no acordados.

Hacer esto conlleva a ser penalizado por el Código Penal y la Ley de Fomento y Protección a la Propiedad Intelectual.

A continuación se detallan los Artículos que protegen al Autor del software:

LEY DE FOMENTO Y PROTECCION A LA PROPIEDAD INTELECTUAL

Art. 13
En las creaciones a que se refiere el artículo anterior, están comprendidas todas las obras literarias, científicas y artísticas, tales como libros, folletos y escritos de toda naturaleza y extensión, **incluidos los programas de ordenador**; obras musicales con o sin palabras; obras oratorias, plásticas, de arte aplicado; versiones escritas o grabadas de las conferencias, discursos, lecciones, sermones y otras de la misma clase; obras dramáticas o dramático-musicales y coreografía; las puestas en escena de obras gramáticas u operáticas; obras de arquitectura o de ingeniería, esferas, cartas atlas y mapas relativos a geografía, geología, topografía, astronomía o cualquier otra ciencia; fotografías, litografías y grabados; obras audiovisuales, ya sea de cinematografía muda, hablada o musicalizada; obras de radiodifusión o televisión, modelos o creaciones que tengan valor artístico en materia de vestuario, mobiliario, decorado, ornamentación, tocado, galas u objetos preciosos; planos u otras reproducciones gráficas y traducciones; todas las demás que por analogía pudieran considerarse comprendidas dentro de los tipos genéricos de las obras mencionadas.

PROGRAMAS DE ORDENADOR

Art. 32
Programa de ordenador, ya sea programa fuente o programa objeto, es la obra literaria constituida por un conjunto de instrucciones expresadas mediante palabras, códigos, planes o en cualquier otra forma que, al ser incorporadas en un dispositivo de lectura automatizada, es capaz de hacer que un ordenador, o sea, un aparato electrónico o similar capaz de elaborar informaciones, ejecute determinada tarea u obtenga determinado resultado.

Se presume que es productor del programa de ordenador, la persona que aparezca indicada como tal en la obra de la manera acostumbrada, salvo prueba en contrario.

Art. 33

El contrato entre los autores del programa de ordenador y el producto, implica la cesión ilimitada y exclusiva a favor de éste de los derechos patrimoniales reconocidos en la presente ley, así como la autorización para decidir sobre su divulgación y la de ejercer los derechos morales sobre la obra, en la medida que ello sea necesario par ala explotación de la misma, salvo pacto en contrario.

CAPITULO XI

VIOLACION Y DEFENSA DE LOS DERECHOS

Art. 89

Constituye violación de los derechos de autor, todo acto que en cualquier forma menoscabe o perjudique los intereses morales o pecuniarios de autor, tales como:

a) El empleo sin el consentimiento del autor, del título de una obra que individualice efectivamente a ésta, para identificar otra del mismo género, cuando exista peligro de confusión entre ambas.
b) La publicación por cualquier medio, de un escrito sin el consentimiento del autor, se haga o no a nombre de éste;
c) La impresión por el editor de mayor número de ejemplares que el convenido, salvo el exceso del cinco por ciento para dar cumplimiento a sus obligaciones con las autoridades públicas y efectos de propaganda;
d) La traducción, adaptación, arreglo o transformación de una obra, sin autorización del autor o de sus causahabientes;
e) La publicación de una obra con supresiones, modificaciones o alteraciones no autorizadas por el autor o sus causahabientes o con errores que constituyan una grave adulteración;
f) La publicación de antologías o recopilaciones, sin el consentimiento de los autores respectivos o de sus causahabientes;
g) La representación, ejecución, difusión, arrendamiento, comunicación o reproducción de obras en cualquier forma y por cualquier medio, con fines de lucro, sin la autorización del autor o de sus causahabientes;

h) La representación, ejecución, exhibición y exposición de la obra en lugares distintos de los convenidos;
i) La adaptación transformación o versión en cualquier forma de una obra ajena o parte de ella, sin consentimiento del autor respectivo o sus causahabientes;
j) La representación o ejecución de una obra con supresiones, modificaciones o alteraciones, no autorizadas por el autor o sus causahabientes;
k) Las adaptaciones, arreglos o limitaciones que impliquen una reproducción disimulada del original;
l) La retransmisión por cualquier medio alámbrico o inalámbrico, de una emisión de radiodifusión, sin el consentimiento del organismo de radiodifusión;
m) La reproducción, importación, exportación con fines convencionales, venta y alquiler de reproducciones o copias de las obras protegidas, en todo o en parte, sin autorización del titular de los derechos, incluyendo las actuaciones de los intérpretes o ejecutantes, fonogramas y emisiones de radiodifusión.

En ningún caso los dependientes, comisionistas o cualquier otra persona que desempeñe una actividad laboral de cualquier clase, bajo remuneración, para la persona que realice actos de violación de los derechos de autor, será responsable de tales actos, ni siquiera en forma subsidiaria.

Art. 90

Sin perjuicio de las acciones penales correspondientes los titulares de los derechos conferidos por esta ley, tienen acción para reclamar ante los tribunales competentes, el cese de la violación de cualquiera de sus derechos y la reparación de daños y perjuicios.

El cese de la violación de sus derechos comprende:

a) La suspensión inmediata de la actividad ilícita;
b) La prohibición al infractor de reanudarla;
c) El retiro del comercio de los ejemplares ilícitos;
d) La inutilización de los moldes, planchas, matrices, negativos y demás elementos utilizados predominantemente para la reproducción ilícita y en caso necesario, la destrucción de tales elementos; y
e) e) La remoción o la guarda bajo llave y sello, de los aparatos utilizados en la comunicación pública no autorizada.

El titular de derecho infringido podrá pedir la destrucción de los ejemplares ilícitos o la entrega de los mismos y del material utilizado para la reproducción, a precio de costo y a cuenta de la correspondiente indemnización por daños y perjuicios.

REGLAMENTO GENERAL DE PROCESOS DE GRADUACION DE LA UNIVERSIDAD DE EL SALVADOR

Derechos de Autor

Art. 29.

Los derechos de autor sobre los trabajos de investigación elaborados en los procesos de graduación, serán de propiedad exclusiva de la Universidad de El Salvador, la cual podrá disponer de los mismos de conformidad a su marco jurídico interno y legislación aplicable.

CODIGO PENAL DE LOS DELITOS RELATIVOS A LA PROPIEDAD INTELECTUAL VIOLACION DE DERECHOS DE AUTOR Y DERECHOS CONEXOS

Art. 226

El que reprodujere, plagiare, distribuyere o comunicare públicamente, en todo o en parte, una obra literaria, artística, científica o técnica o su transformación o una interpretación o ejecución artística fijada en cualquier tipo de soporte o fuere comunicada a través de cualquier medio, sin la autorización de los titulares de los correspondientes derechos de propiedad intelectual o de sus cesionarios, será sancionado con prisión de uno a tres años. En la misma sanción incurrirá quien no depositare en el Registro de Comercio, importare, exportare o almacenare ejemplares de dichas obras o producciones o ejecuciones sin la referida autorización.

VIOLACION AGRAVADA DE DERECHOS DE AUTOR Y DE DERECHOS CONEXOS

Art. 227

Será sancionado con pena de prisión de tres a cinco años quien realizare cualquiera de las conductas descritas en el artículo anterior, concurriendo alguna de las circunstancias siguientes:

- Usurpando la condición de autor sobre una obra o parte de ella o el nombre de un artista en una interpretación o ejecución.
- Modificando sustancialmente la integridad de la obra sin autorización del autor.
- Si la cantidad o el valor de la copia ilícita fuere de especial trascendencia económica.

D.L. N0 1030, del 26 de Abril de 1997
D.O. N0 105, Tomo 335 del 10 de Junio de 1997

CODIGO PROCESAL PENAL
Acciones públicas dependientes de Instancia Particular

Art. 26.

Para su persecución dependerán de instancia particular, los delitos siguientes:

1) Lesiones comprendidas en el Art. 142 del Código Penal.
2) Lesiones culposas.
3) Amenazas.
4) Inseminación artificial y experimentación.
5) Apropiación o retenciones indebidas y administración fraudulenta.
6) Hurto de uso.
7) Usurpaciones.
8) Daños.
9) Delitos relativos a la propiedad intelectual.
10) Delitos relativos a la propiedad industrial.

En estos casos no se perseguirá penalmente sino por petición de la víctima, o en caso de incapacidad, por quien ejerza su representación legal o por el guardador. . Sin embargo, la Fiscalía General de la República ejercerá la acción penal cuando el delito haya sido cometido contra un menor que no tenga padres ni tutor, contra un incapaz que no tenga tutor o cuando el delito fue cometido por uno de sus ascendientes o tutor, cuando haya perjudicado bienes del Estado, o cuando la víctima esté imposibilitada física o mentalmente para solicitar la investigación.

D.L. N° 904, del 8 de octubre de 1998
D.O. N° 206, Tomo N° 341, del 5 de noviembre de 1998

ANEXO 2

MANUAL DE USUARIO

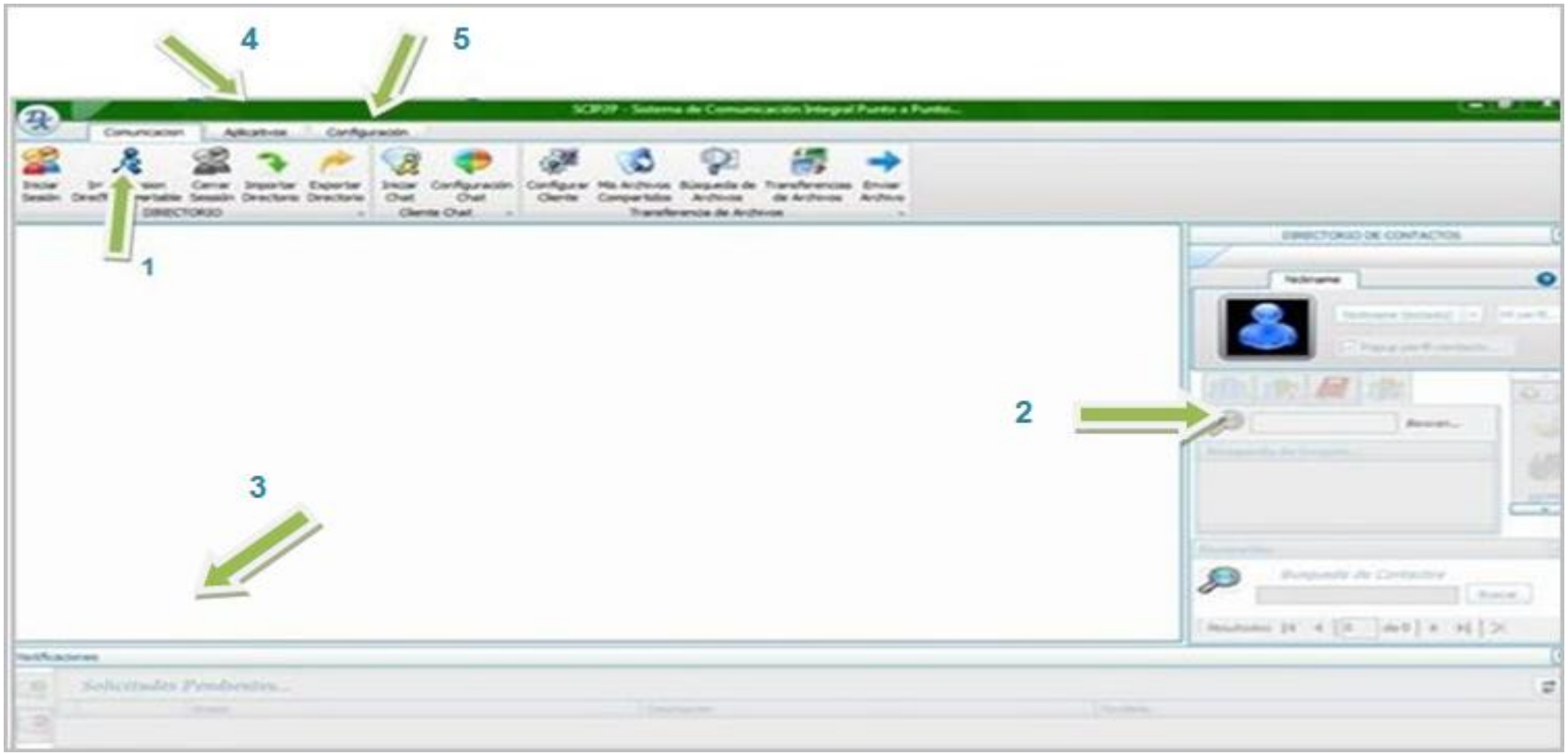
Sistema de Comunicación Integral Punto a Punto

SCIP2P

- **Iniciar SCIP2P**

  Al dar click en el icono que se creó al instalar el Sistema:

Se mostrara la pantalla del ambiente de trabajo con las opciones principales que el SCIP2P posee:



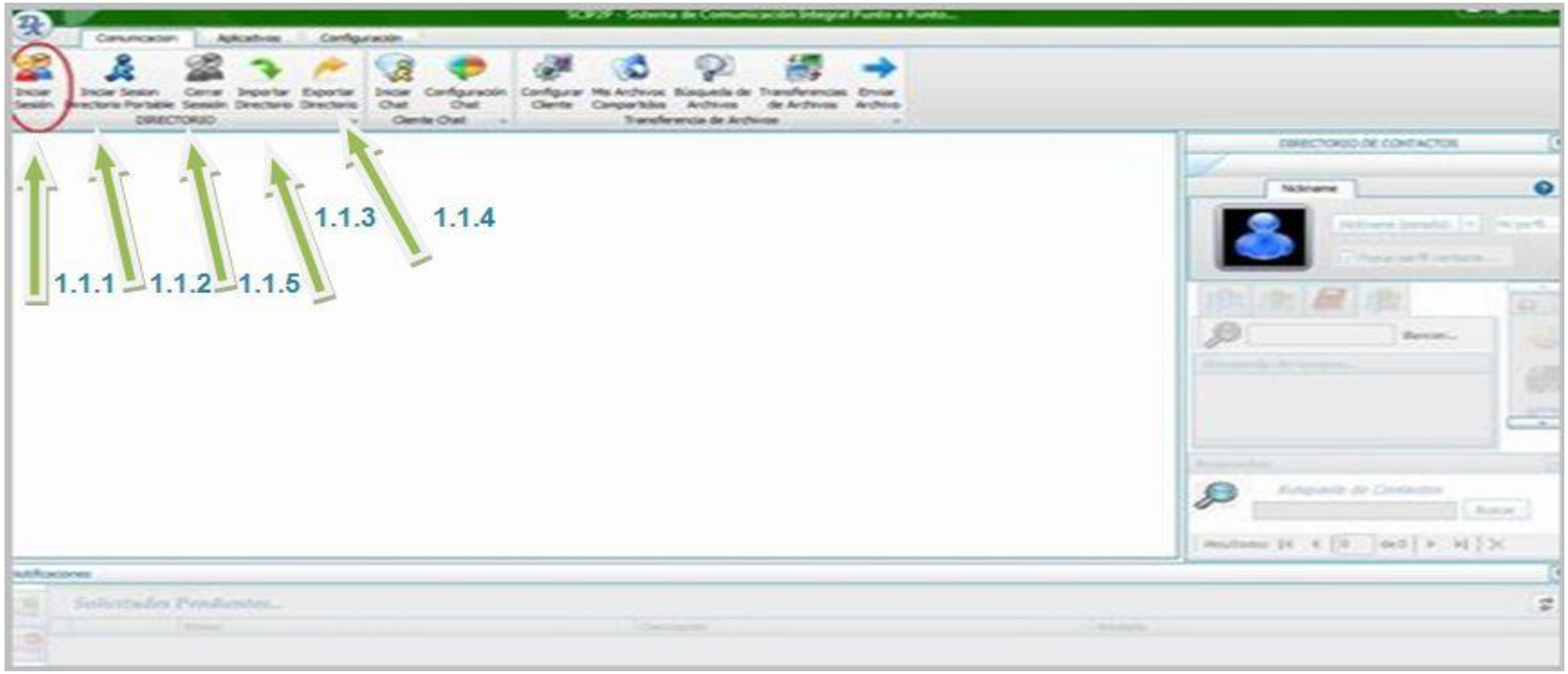
La cual esta está formada por 5 opciones de las cuales se utilizan para iniciar sesión solamente la Parte de Comunicación del menú Directorio (Ver Sección **“Iniciando Sesión”)**, las demás opciones se utilizan al haber Iniciado Sesión (Ver Sección **“Al haber Iniciado Sesión”).**

- ✓ **Iniciando Sesión**

1. Comunicación

1.1. Directorio

En esta categoría del menú de inicio de la pestaña de comunicación se encuentran las tres formas de iniciar sesión en una PC, así como también la opción de Exportar el perfil y finalizar una Sesión.



1.1.1. Desde Directorio Local

Al elegir esta opción el aplicativo mostrara una ventana en la el usuario podrá hacer uso de dos opciones

1.1.1.1. Si es un cliente Nuevo,

Da click en el Enlace *Crear Perfil …* y se llena el formulario con la información requerida para crear un nuevo perfil y poder iniciar sesión.

Las opciones que presenta son:

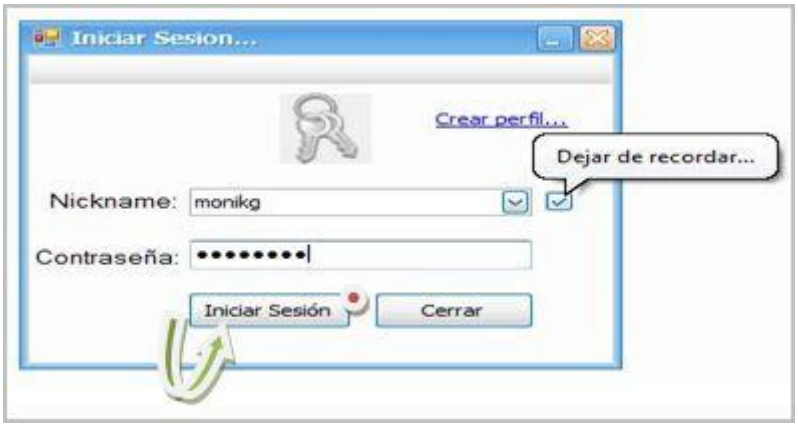
- *La opción de subir una imagen*, en la primera flecha se da click al botón que señala el punto rojo y se elige una imagen desde cualquier ubicación de la PC, esta imagen puede ser de cualquier formato y un tamaño menor a 50 Kb, si esta excede el tamaño el SCIP2P envía un mensaje en el cual advierte que no puede ser utilizada.
- Elegir un nickname el cual será único (ya que es con este con el cual se inicia la sesión)
- Tipo puede ser Docente, Alumno, Jefatura, Administrativo la parte más importante
- Luego se rellena con información propia del usuario, esta no es necesaria para crear el directorio
- Finalmente la contraseña, ya que sin esta el usuario no podrá ingresar al sistema.

Una vez ingresada toda la información, se da click en el botón *Crear Registro,* y si la información fue ingresada correctamente el sistema devolverá un mensaje.

Al dar click en el botón *OK* de la ventana anterior el usuario está listo para iniciar Sesión (ver Sección **"Si el usuario ya tiene creado su perfil previamente en la PC que se iniciara sesión")**

1.1.1.2. Si el usuario ya tiene creado su perfil previamente en la PC que se iniciara sesión,

Se ingresa el nickname y la contraseña, también aparece un checkbox en el que se elige si se desee que la información en el campo nickname se esté guardando.



Al terminar de llenar la información requerida, se da click en el botón Iniciar sesión, si la contraseña no coincide con el nickname el sistema devuelve el siguiente mensaje:

Se da click en el botón Ok y vuelve a la pantalla de ingresar nickname y Password. De manera contraria, si la información que se ingresa es correcta el sistema devuelve los siguientes mensajes:

Al dar click en el botón ok, se puede tener uno de dos casos:

- ***Ser usuario por primera del SCIP2P***, en este caso el sistema pedirá la ruta de la carpeta que se utilizara para compartir información con los demás usuarios del SCIP2P, y así mismo en la que se guardaran todos los archivos que se descargan de otros usuarios.

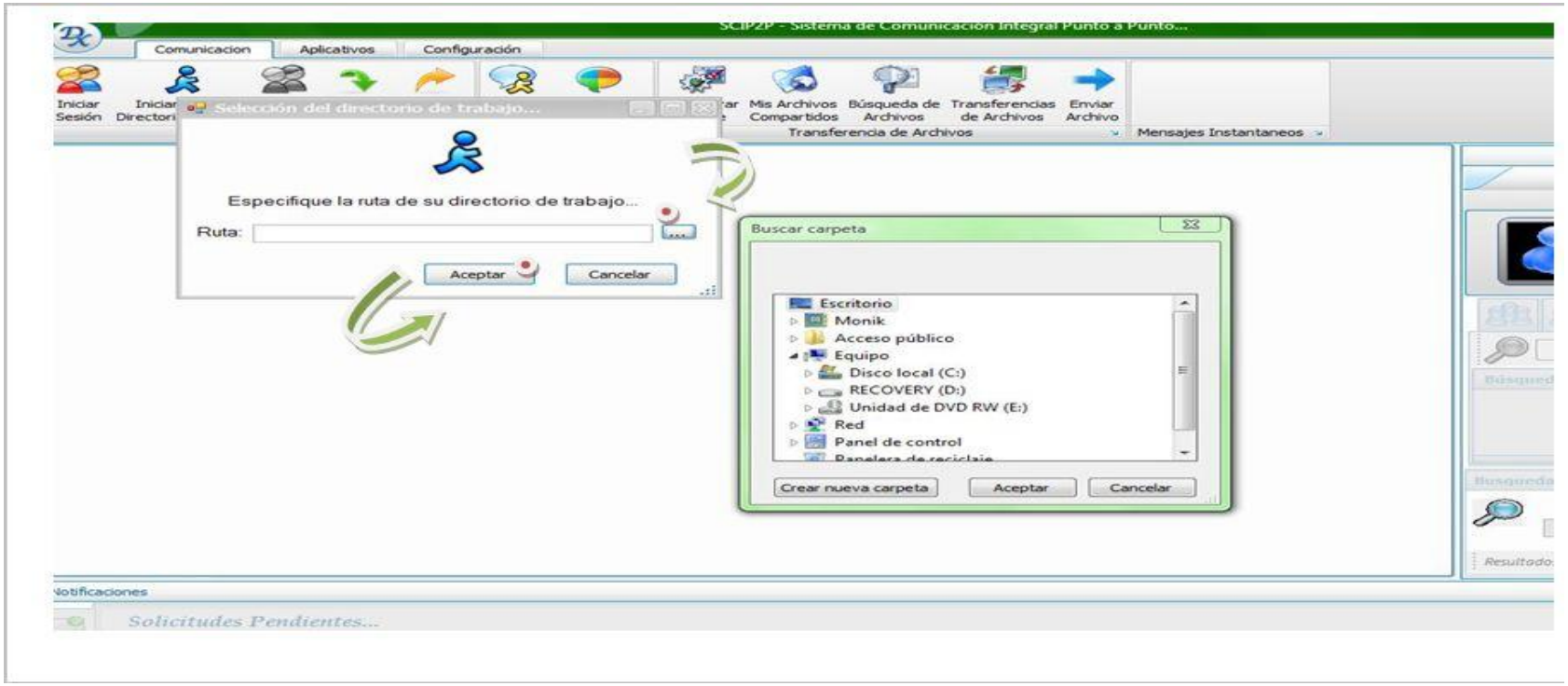
Para esto se da click en el botón que señala la flecha y aparecerá la ventana en la que se elija la carpeta que contendrá los archivos disponibles a compartir. Una vez seleccionada la carpeta se da clic en el botón *Seleccionar* y el

sistema abre la ventana principal en la que muestra todas las opciones que tiene el SCIP2P (Ver Sección **"Al haber Iniciado Sesión"**).

- ✓ ***Que el usuario ya haya hecho uso antes del sistema con el nickname y contraseña que ingreso***, en este caso el sistema abre automáticamente la pantalla principal (Ver Sección **"Al haber Iniciado Sesión"**).

1.1.2. Iniciar Sesión desde un Directorio Portable

Este modo de iniciar sesión es utilizado en momentos cuando el usuario ha exportado su perfil (ver **"Exportar Perfil"**), guardándolo en una ubicación de la PC diferente que la que el SCIP2P reconoce por defecto. Para ello se selecciona la opción *Iniciar Sesión Directorio Portable* en el menú directorio de la Pantalla Principal del SCIP2P, al elegir esa opción aparece la siguiente Ventana:



Para seleccionar la ubicación del archivo desde el cual se desea correr la aplicación, se da click en el botón que apunta la primera flecha y luego aparece la ventana en la que se elige el archivo que guarda toda la información del usuario que se desea conectar (este tiene que ser extensión .dir). Una vez ubicado el archivo, se carga en el SCIP2P e inicia de la misma manera en que se haría del modo ***Ser usuario por primera del SCIP2P*** *(ver).*

Al trabajar en este modo todo lo que el usuario realice, cualquier nueva configuración en su perfil, invitaciones de nuevos usuarios, etc. se guardara en ese archivo (extensión .dir).

En el caso que sea la misma PC la que se utilizara para correr el aplicativo una y otra vez, siempre se tendrá que llamar el perfil desde esa ubicación o importarlo (ver Sección **"Importar Perfil"**) de ambas maneras solamente la primera vez que inicie con este perfil le pedirá la ruta de la carpeta compartida ya que está ubicación está guardada en el archivo .dir y por estar utilizando la misma PC el sistema reconoce la ruta.

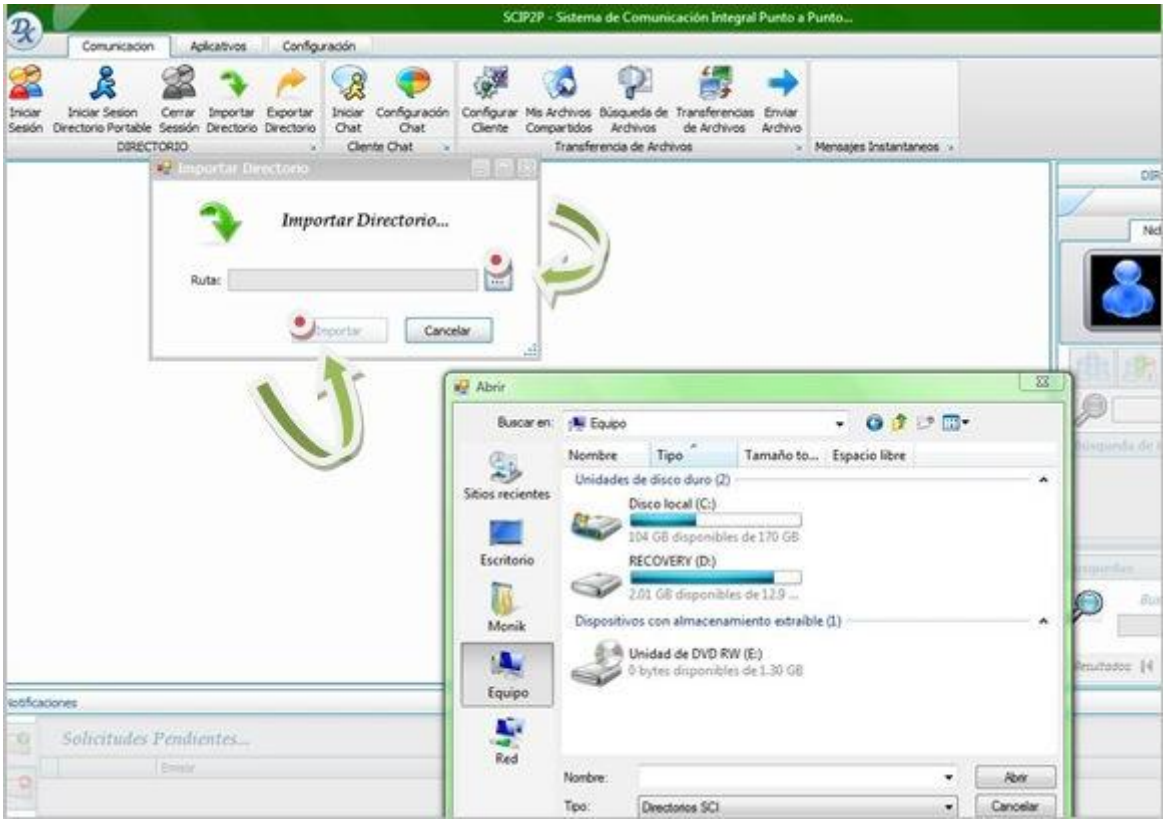
Si el usuario se conecta en diferentes PCs el usuario podrá exportar su perfil (ver Sección **"Exportar Perfil"**) o si ya tiene ubicado el archivo (.dir) puede guardarlo en un dispositivo de almacenamiento de la misma manera en que se guardar cualquier otro archivo, para poder cargarlo y utilizarlo en el momento en que lo necesite.

Una vez iniciada la sesión de este modo, se tendrán todas las opciones a disposición (Ver Sección **"Al haber Iniciado Sesión"**).

1.1.3. Iniciando Sesión por medio de Importar Directorio

Este modo de iniciar sesión es utilizado en momentos cuando el usuario ha exportado su perfil (ver **"Exportar Perfil"**) y lo tiene guardado en cualquier dispositivo de almacenamiento o ubicación de la PC pero desea tenerlo agregado en el archivo por defecto que el SCIP2P busca al iniciar sesión, para que en ocasiones futuras se pueda iniciar con la Opción de *Iniciar Sesión* (ver Sección

"Desde Directorio Local"). Para ello se selecciona la opción Importar Archivo en el menú directorio de la Pantalla Principal del SCIP2P, al elegir esa opción aparece la siguiente Ventana:



Al dar click en el botón… que señala la primera flecha aparecerá la ventana en la cual se elige el archivo (extensión .dir) que se va a copiar en la carpeta donde el SCIP2P guarda los perfiles de todos los usuarios que han creado su directorio en dicha PC. Una vez selecciona el archivo .dir Se da click en el botón *Importar* y ya está listo para Iniciar Sesión (Ver Sección **"Desde Directorio Local"**).

1.1.4. Exportar Directorio

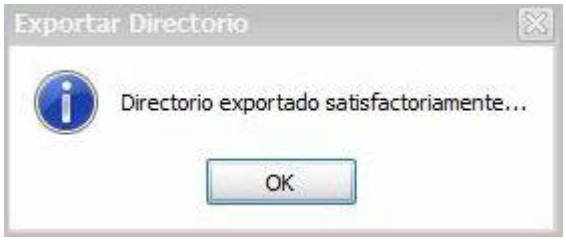
Esta opción se habilita al momento de Iniciar Sesión y es utilizada cuando el usuario desea hacer uso del SCIP2P desde diferentes PCs, pero quiere utilizar toda la configuración de su perfil siempre. Ya que este archivo incluye toda la información de la configuración que el usuario agrega a excepción de los

documento que el usuario tiene a disposición de ser compartida ya está solamente se guarda localmente. Para hacer uso de esta función (además de ya estar conectados con el SCIP2P), se da click en el botón Exportar Directorio que se encuentra en el menú Directorio de la Pantalla Principal, y aparece la siguiente ventana:

En esta ventana se primero se elige la ubicación de donde se irá a guardar el archivo .dir, una vez seleccionada se da click en el botón exportar, al dar click en dicho botón se abre la siguiente ventana:

En esta ventana el usuario debe escribir la contraseña de sesión de modo contrario no podrá finalizar la tarea de exportar el Directorio, esto asegura que un usuario no propietario del perfil abierto pueda robar un perfil. Una vez escrita la contraseña se da click en el botón Autentificar, si esta es errónea se manda el siguiente mensaje:

Se da click y aparece la pantalla anterior en la que permite corregir a la contraseña correcta, al escribirla correctamente se muestra un mensaje que dice:



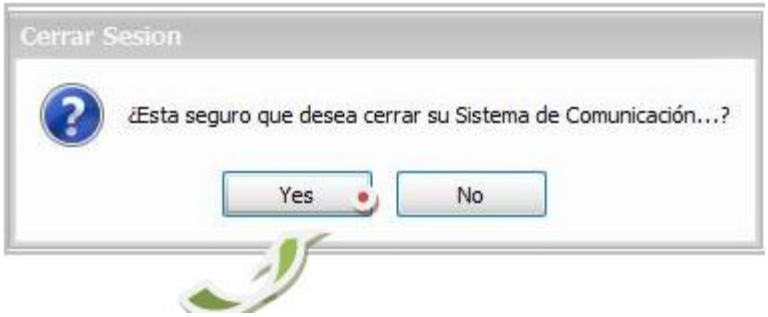
Luego se da click en *OK y ya e*stá listo para ser importado a otra PC, importarlo y tener la misma configuración que en la PC inicial.

1.1.5. Cerrar Sesión

Esta opción como su nombre lo describe sirve para cerrar la sesión que fue iniciada, esto independientemente del modo que se utilizo.

Al cerrar la sesión se guarda toda la configuración del usuario en el archivo .dir, ya sea en la carpeta que el SCIP2P utiliza por defecto (al haber iniciado “Desde Iniciar Sesión”) o en el archivo portable desde donde se está corriendo el aplicativo.

Para cerrar Sesión se da click en el botón Cerrar Sesión del menú del Directorio de la pantalla principal, y aparece la siguiente pantalla:

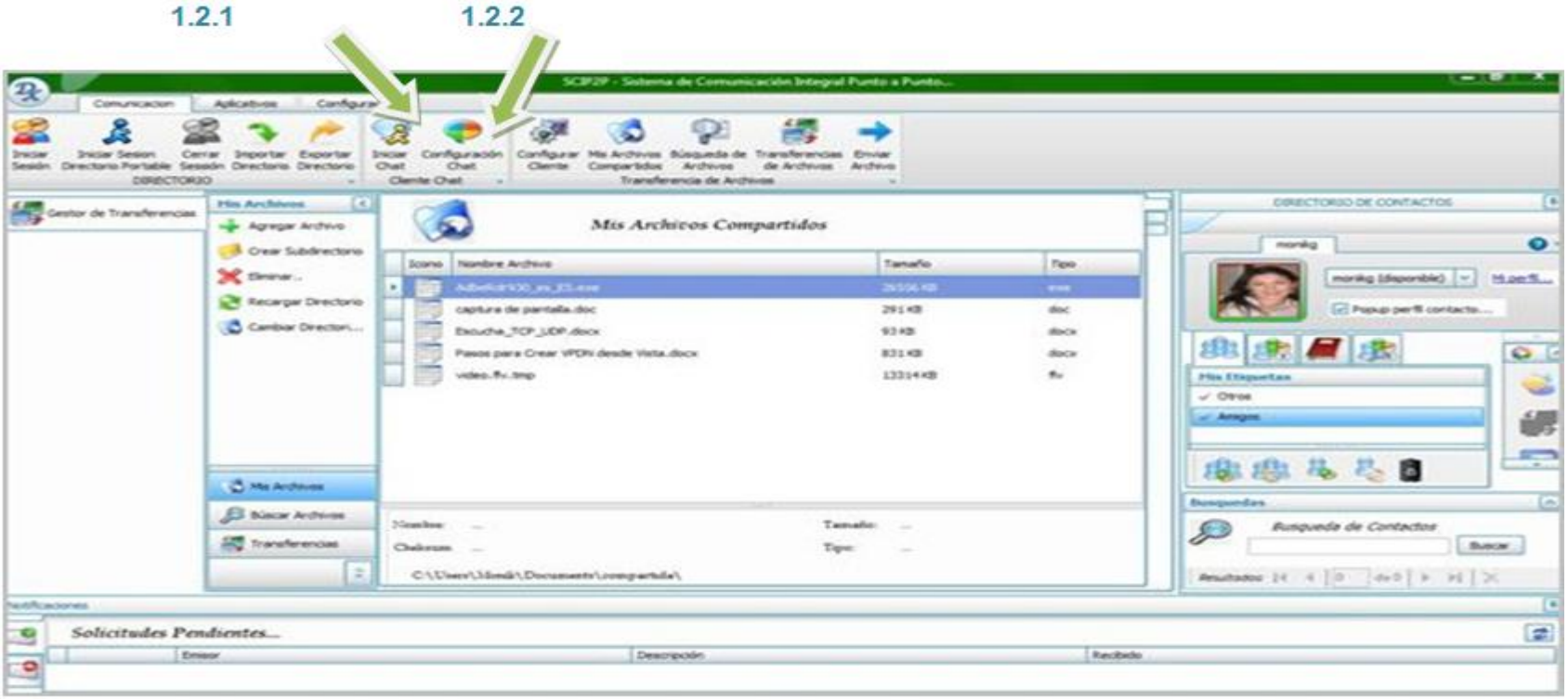


Esto a manera de comprobación, para que el usuario no cierre el aplicativo sin desearlo en realidad.

- **Al haber Iniciado Sesión**

1.2. Chat

Esta opción aparece una vez Iniciada Sesión y para hacer uso de ella es indispensable tener agregados contactos en el directorio (ver Sección **"Directorio, Agregar usuarios"**) ya que para iniciar se tiene que elegir primeramente el contacto con el que se establecerá la conversación.



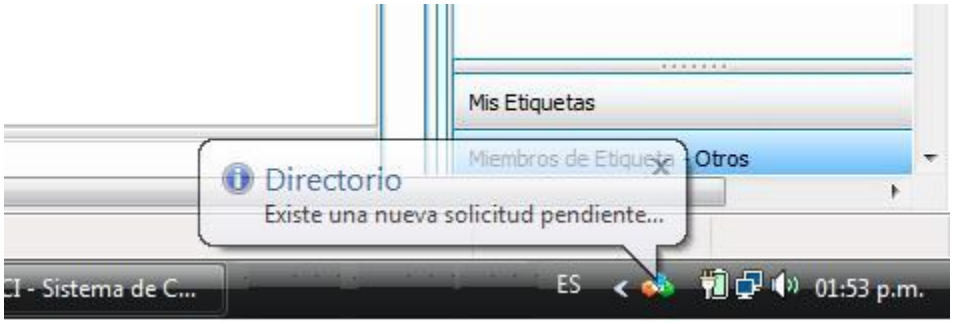
1.2.1. Iniciar Chat

Para hacer uso de esta función del SCIP2P se da click en el botón Iniciar Chat del menú de Chat de la pantalla principal. Si no se ha seleccionado el contacto con el que se desea establecer comunicación previamente aparcera la siguiente ventana:

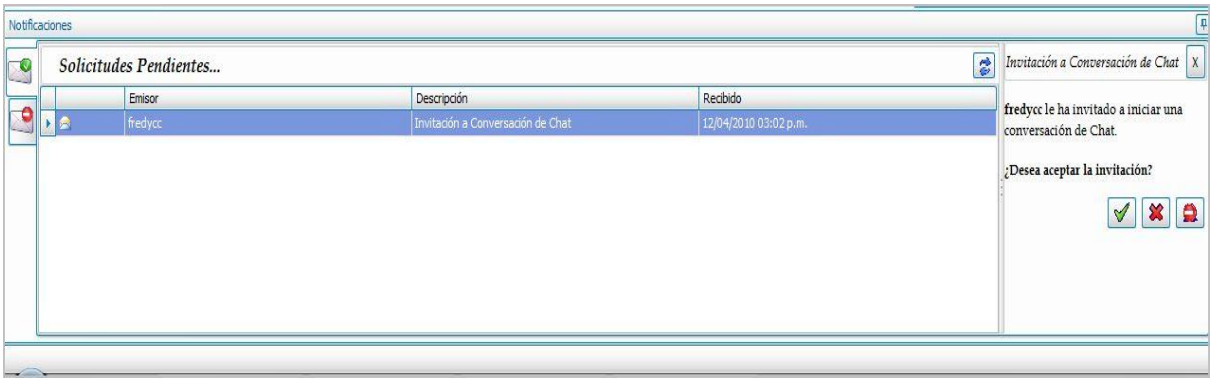
De lo contrario aparecerá la ventana que se muestra a continuación:

En el momento en que se da click en *Si* se envía una notificación al usuario que se está invitando, esta aparece sobre la barra de estado de su PC la siguiente manera:

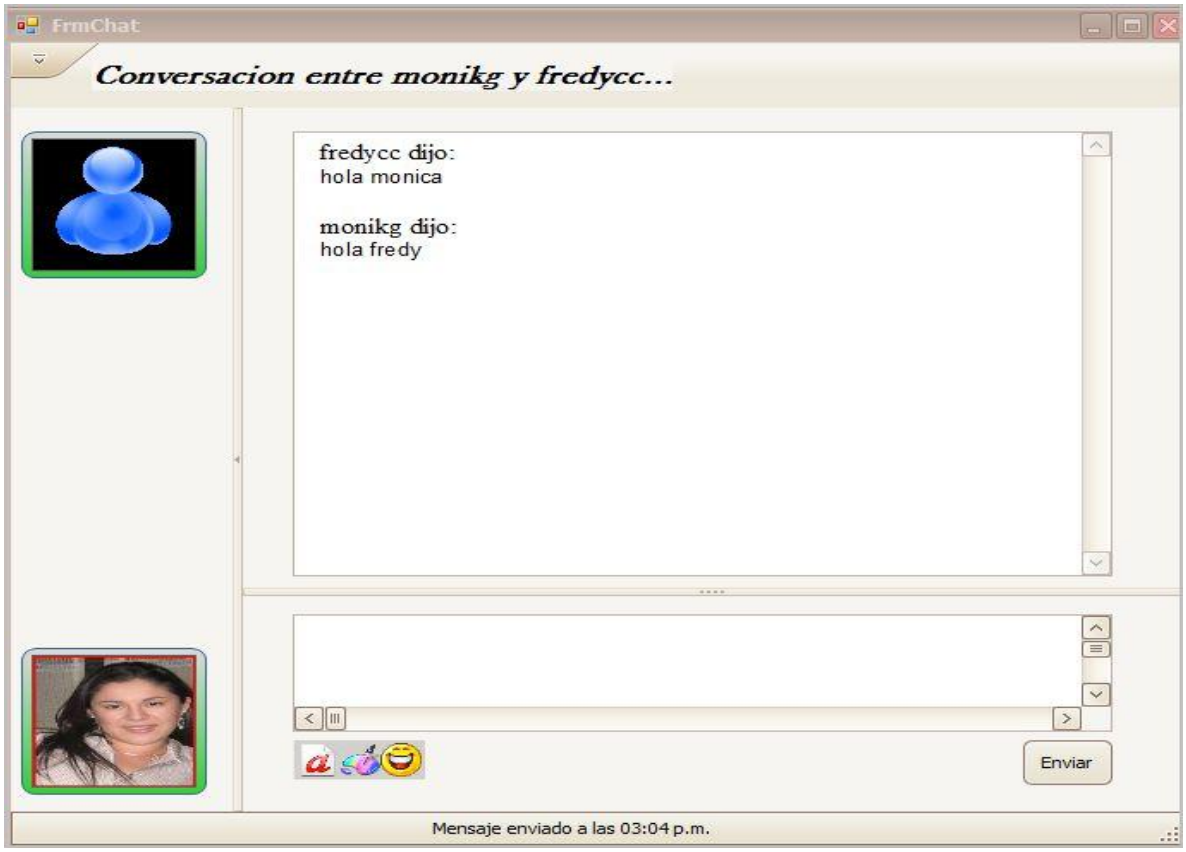
Si el usuario no contesta la invitación en los primeros 30 seg desde que se recibió el sistema le informa que la solicitud ha sido guardada en la sección de Solicitudes Pendiente (Ver Sección **"Manejo de Solicitudes Pendientes"**)



El usuario podrá retomar la invitación, desde dicha sección, dando doble click en la figura del sobre cerrado (amarillo de la parte izquierda de la ventana), al hacer esto la figura cambia a un sobre abierto y aparece una ventana al lado derecho en la que se tienen las opciones de aceptar, borrar y cancelar la invitación. Como se muestra:

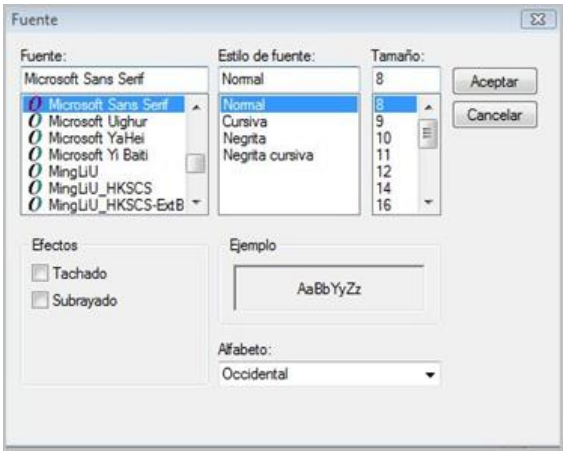


Si el usuario da clic en el botón aceptar, se abrirá automáticamente la ventana para chatear



Esta posee las opciones de:

- ✓ Cambiar la fuente, dando click en el botón  aparecerá la ventana

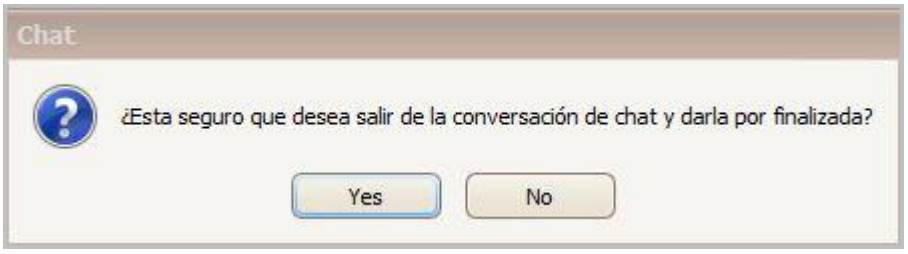


- ✓ Cambiar el color de la fuente, dando click en el botón  aparecerá la ventana:

- ✓ Insertar emoticos, dando click en el botón  aparecerán todas las opciones:

Esto toma efecto automáticamente se configuran las opciones y se puede apreciar así:

Para cerrar un chat, la persona que cierra recibe un mensaje de comprobación si está seguro o no de cerrar la conversación:

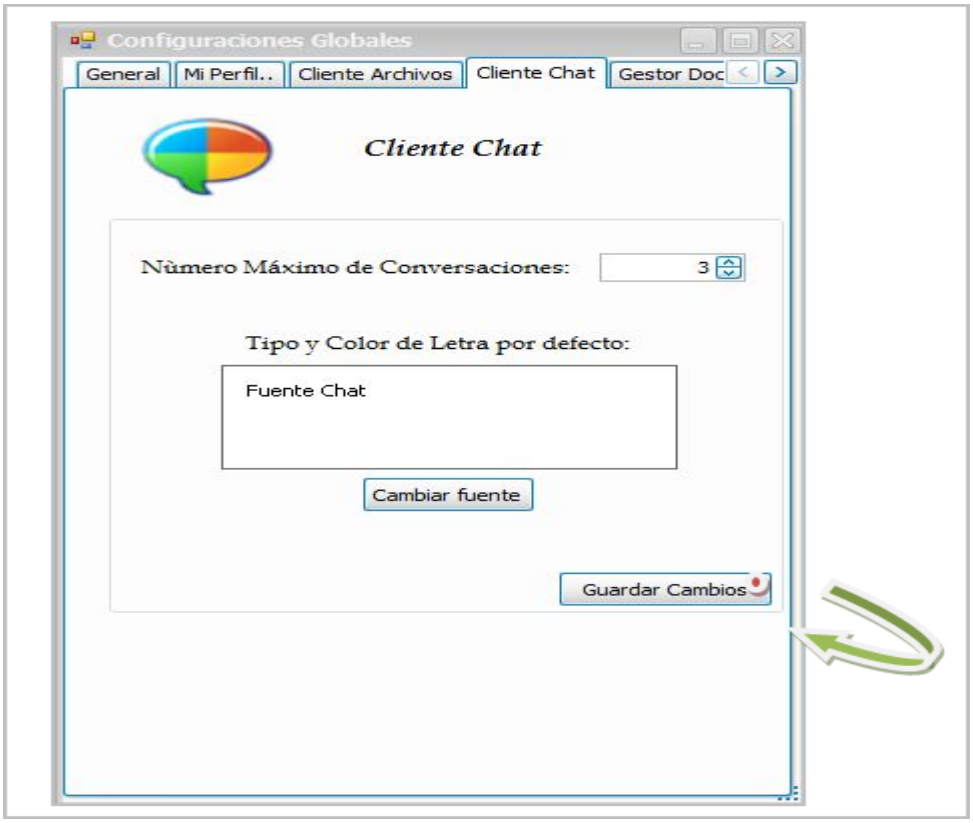


Al dar click en *Si* la persona con la que se había establecido el chat recibe el siguiente mensaje:

1.2.2. Configuración Chat

Esta opción permite al usuario configurar la cantidad de sesiones de chat que se podrán realizar al mismo tiempo así como también elegir la fuente por defecto que se quiere establecer.

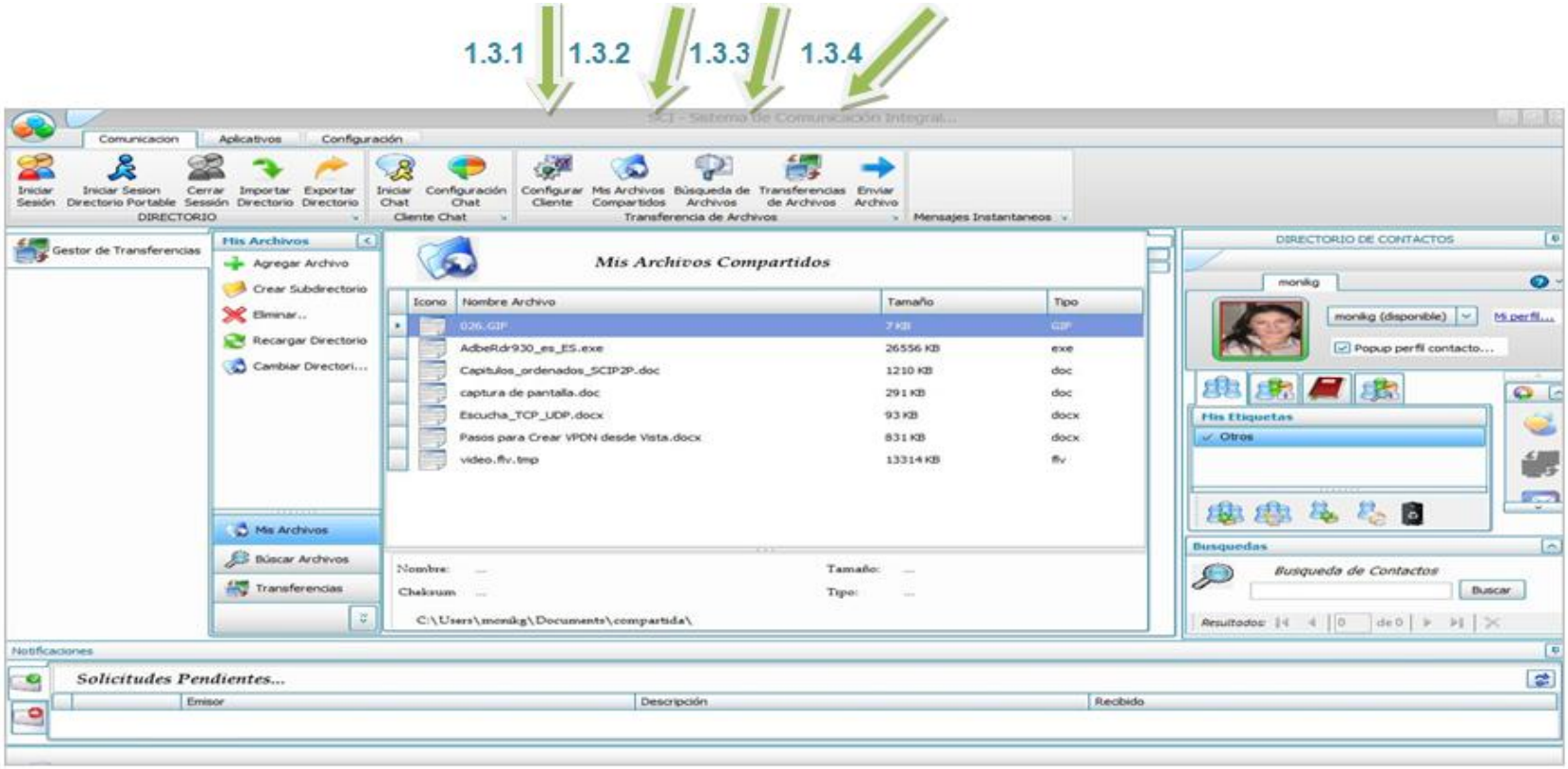
Para configurar esta opción, en el menú de Chat de la Pestaña de Comunicación de la Pantalla Principal del SCIP2P se da clic al botón configuración de chat, y aparece la siguiente pantalla:



Al dar click en el Botón *Guardar Cambios*, se guardan la configuración y se mantendrá en ocasiones futuras que se inicie sesión.

1.3. Transferencia de Archivos

Esta parte se encuentra cargada por defecto en la pantalla principal y está ubicada en parte central del SCIP2P. Sirve para que los usuarios que hacen uso puedan compartir los archivos que así deseen y así mismo descargar los que necesite de otros usuarios, estos estarán guardados en la carpeta que el usuario eligió al iniciar sesión por primera vez (ver Sección "**Si es un cliente Nuevo"**).



Las opciones que se tiene para esta función del SCIP2P son:

1.3.1. Configuración del Cliente

Esta parte permite al usuario configurar todo lo relacionado a la transferencia de Archivos, para esto se da click en el botón *Configurar Cliente* del menú de Transferencia de Archivos y aparece la siguiente ventana:

Esta pantalla contiene en un inicio toda la configuración que el SCIP2P posee por defecto, pero que el usuario puede modificar según prefiera:

- ✓ *Directorio Compartido,* En esta opción el usuario puede modificar la ruta de la carpeta en la cual están todos los archivos, ya sea que han descargado de otros

usuarios como los que él tiene a disposición, por la ubicación que se seleccione. Para ello se da click el botón… El cual abre la ventana del Sistema en la que presenta en forma de diagrama de árbol todas las ubicaciones. Para que el sistema tome efecto del cambio realizado se da clic en el botón de recargar Directorio (Ver Sección **"Archivos Compartidos"**).

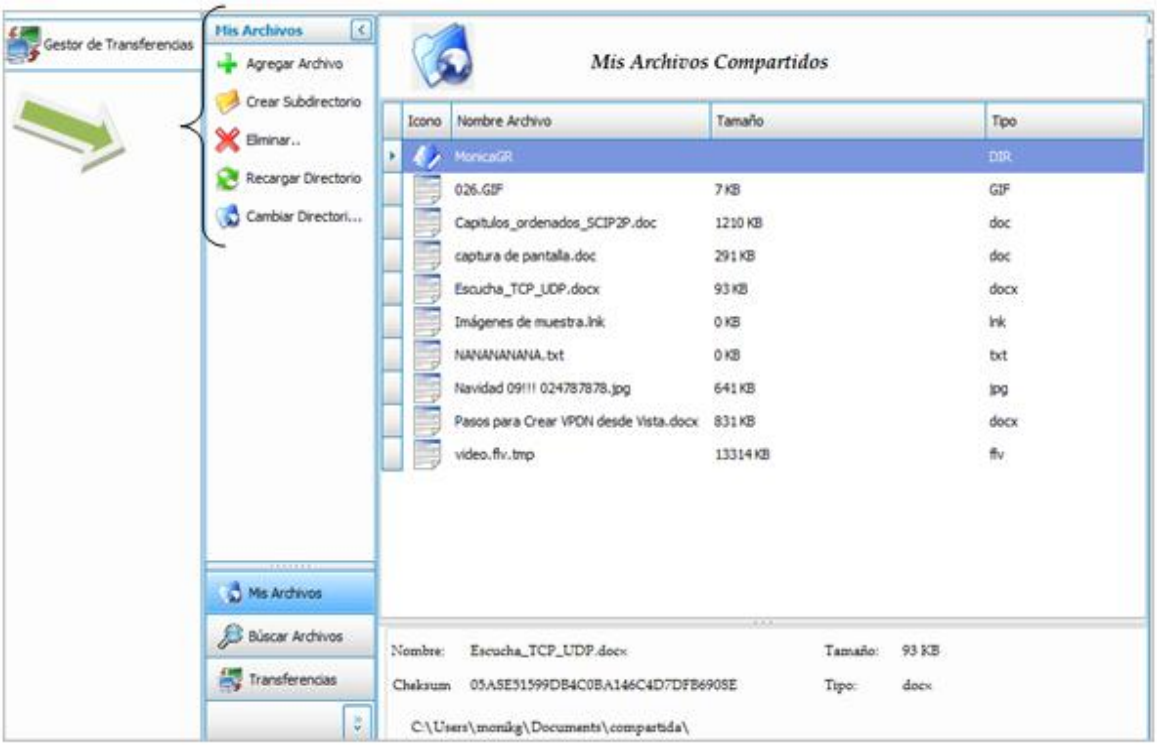
- ✓ *Tamaño Bloque de Transferencia,* Esta opción por defecto tiene configurado 512 Kb lo que significa que al momento de transferir un archivo este creara el stream con los datos y trasmitirá paquetes de 512 Kb por el sistema de comunicación, si el usuario desea que la trasferencia sea mas rápida puede configurar para que los paquetes que se trasmitirán sean de 1024 Kb.
- ✓ *Número Máximo Descargar,* Esta opción permite configurar la cantidad de descargas que el usuario tendrá permitidas en esa sesión, esta por defecto está configurada en 5 pero puede variar de 1 a 8.
- ✓ *Número Máximo Subida,* Esta opción permite configurar la cantidad de subidas que el usuario tendrá permitidas en esa sesión, esta por defecto está configurada en 3 pero puede variar de 1 a 8.

Luego que el usuario ha configurado todos los parámetros que se desea, se da click en el botón *Guardar Cambios.* Y estos toman efecto desde ese momento en adelanta al estar ejecutando el aplicativo.

1.3.2. Mis Archivos Compartidos

Esta parte del SCIP2P sirve, como su nombre lo indica, para compartir archivos de cualquier extensión. Esta opción aparece cargada en el sistema por defecto, pero si se ha trabajado en otras opciones del sistema y esta parte ha desaparecido del área de trabajo se puede accesar desde la Pestaña de Comunicación dando clic en el botón *Mis Archivos Compartidos* del Menú Transferencias de Archivos.

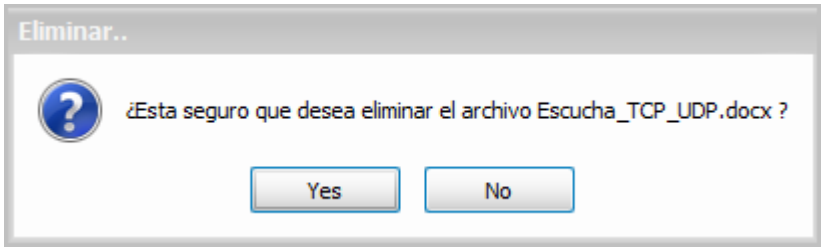
Aquí el usuario tiene una serie de opciones las cuales están ubicadas en la parte izquierda del área de trabajo, estas son:



- ✓ *Agregar Archivo,* al dar click a esta opción el sistema abre la ventana en la que muestra todas las ubicaciones del sistema para poder elegir el archivo que (internamente) se irá a guardar a la carpeta que el usuario a elegido como compartida, una vez seleccionado el archivo , este aparece automáticamente en el listado mostrando toda la información pertinente a el mismo como Nombre, tamaño, ubicación y la información que el sistema calcula para poder comprobar cuando este sea transferido y poder asegurar que el paquete llega completo a su destino.
- ✓ *Crear Subdirectorio,* al dar click en esta opción el sistema abre la siguiente ventana:

En esta ventana se escribe el nombre de la nueva carpeta que se quiere crear, siempre dentro de la carpeta compartida, se da click en *crear* y esta aparece automáticamente en el listado de los archivos compartidos. Luego para ingresar al contenido de la nueva carpeta se da doble click sobre el icono que la representa  y se muestra el contenido del subdirectorio. Para volver al directorio raíz de los archivos compartidos se da doble click en el mismo icono.

- ✓ *Eliminar,* esta opción funciona seleccionando el archivo que se quiere eliminar de la carpeta compartida y dando click en el botón *eliminar* , al hacer esto aparecerá la siguiente ventana:



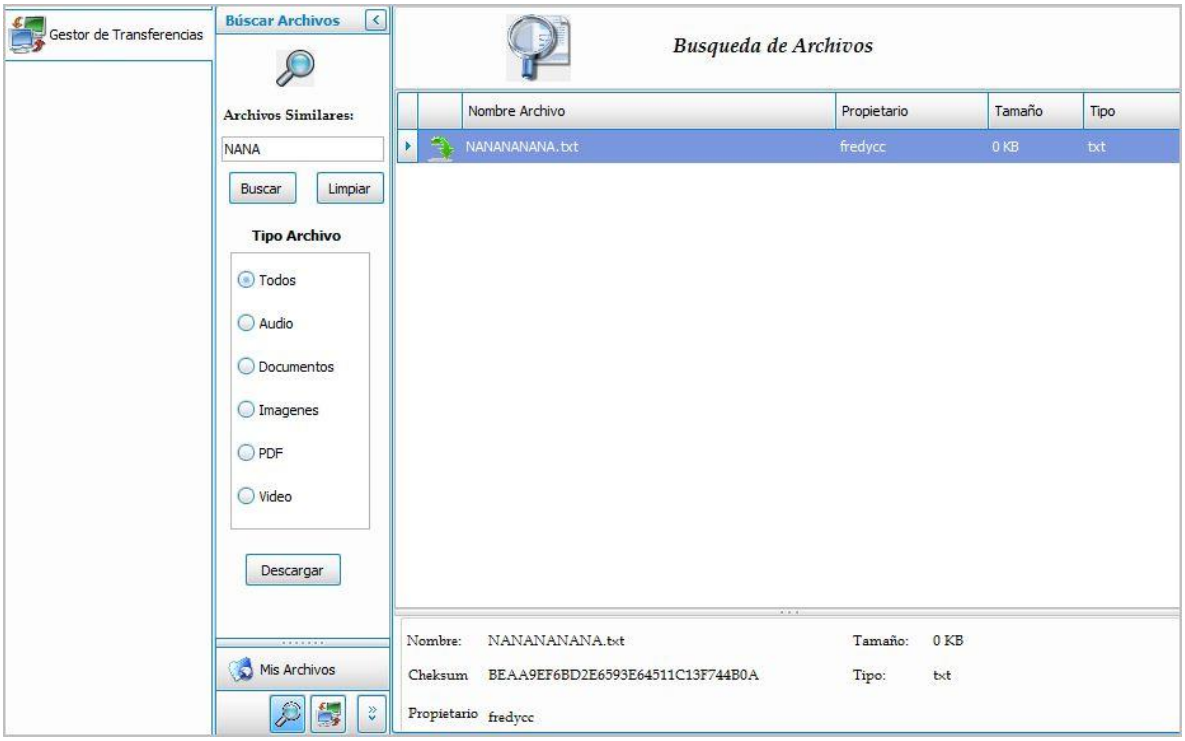
Que pregunta al usuario si en realidad quiere eliminar el archivo, dar click *Yes* el archivo desapareceré automáticamente del listado, click en *No* vuelve a la pantalla anterior.

- ✓ Recargar Directorio, Esta opción se utiliza cuando se ha descargado algún archivo y se quiere que este aparezca en el listado de archivos. Al dar click aparece una ventana que muestra que se está cargando el archivo.

- ✓ *Cambiar Directorio,* Esta opción permite cambiar la ruta de la carpeta compartida. Dando click en el botón *cambiar Directorio* y seleccionando la nueva ubicación. Al hacer el procedimiento anterior el SCIP2P automáticamente mostrara los archivos que contiene la nueva ubicación.

1.3.3. Búsqueda de Archivos

Esta opción permite buscar archivos en las carpetas compartidas de todos los usuarios que se encuentren conectados al SCIP2P en ese momento. Al dar click en el botón *Buscar Archivos* se muestra la siguiente pantalla en el área de trabajo:



Para realizar una búsqueda, se escribe el nombre del archivo que se está buscando, esta búsqueda también se puede delimitar según el tipo (extensión), luego de elegir lo que se está buscando se da click en el botón *buscar* y en la parte central del área de trabajo aparecerán todos los archivo que tienen la característica de lo que se busca junto con la información del usuario al que le

pertenece, tamaño, tipo y el checksum que es el número que el sistema maneja internamente para comprobar que este se descargue completo.

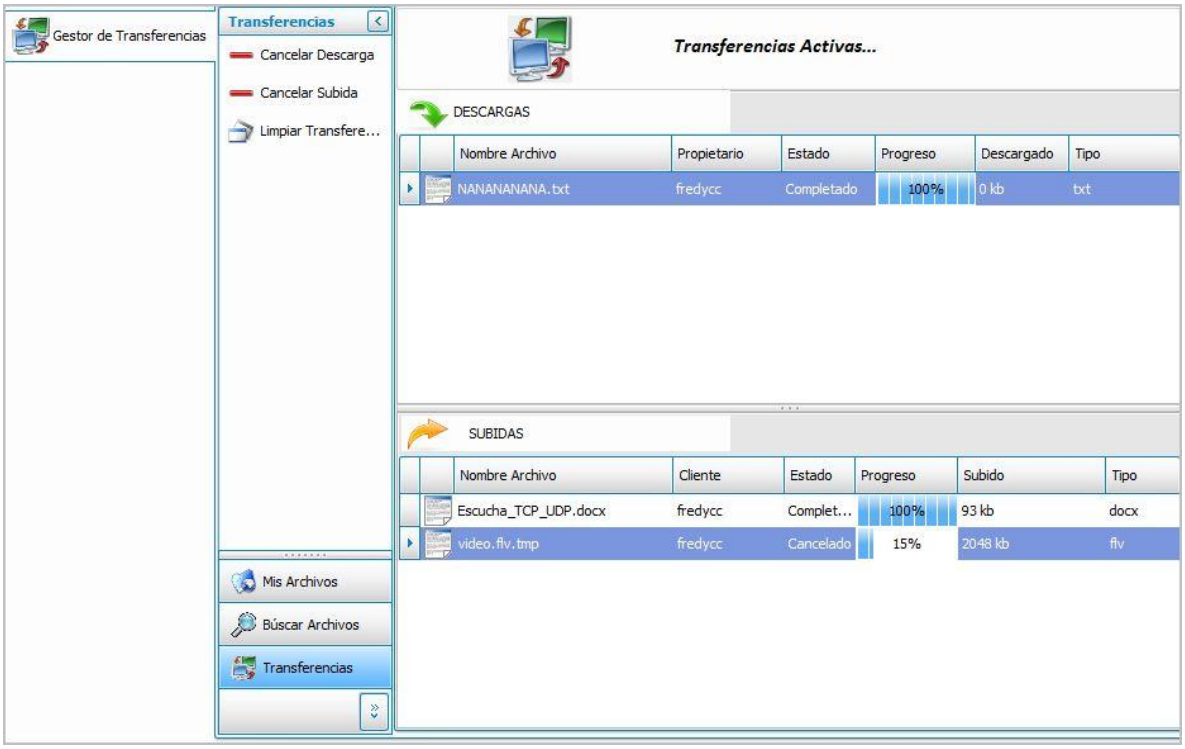
- ✓ Para Descargar el Archivo:

Se selecciona el archivo que se quiere descargar del listado que apareció y se da click en el botón *Descargar.* El avance de la descarga se puede observar en la parte de transferencia de archivos (Ver Sección **"Transferencia de Archivos"**).

Si se desea hacer otra búsqueda se da click en el botón Limpiar y se comienza con el mismo proceso de Realizar una búsqueda.

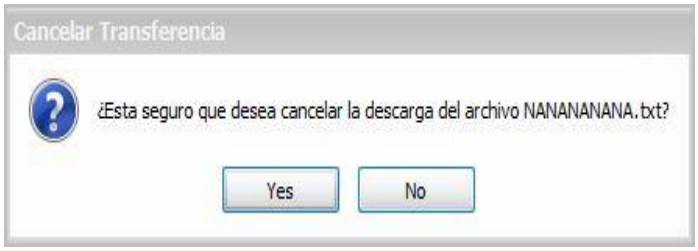
1.3.4. Transferencia de Archivos

Esta opción ayuda a mostrar al usuario el avance de las descargas que se hacen, así como también la subida de archivos, esto último tiene que ver con los archivos que otros usuarios de SCIP2P están descargando de su carpeta compartida. Al dar click en el botón *Transferencia de Archivos* aparece la siguiente ventana en el área de trabajo:



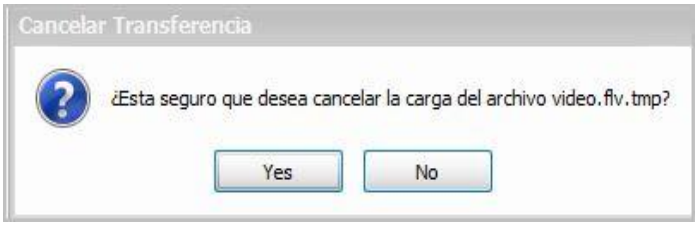
En esta opción del Menú de Transferencia de archivos permite:

- ✓ *Cancelar las Descargas*, para hacer una cancelación de descarga siempre el sistema pregunta al usuario si está seguro de hacerlo.



Si el usuario da click en el botón *Yes,* la descarga para instantáneamente. Si el usuario da click en *No* la descarga sigue su curso normal.

- ✓ *Cancelar la Subida,* esta opción para el envió del archivo al usuario remoto que la está haciendo (en este caso fredycc), pero al igual que en cancelar una descarga el SCIP2P pregunta al usuario local si desea cancelar la subida del archivo a fredycc.



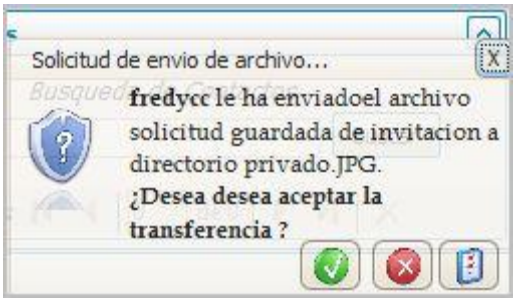
Al dar click en *Yes* el usuario remoto deja de recibir bits y la barra de avance para su secuencia en porcentaje, lo que le indica que le han cancelado la descarga. Si da click en *No* la subida sigue su curso normal.

1.3.5. Enviar

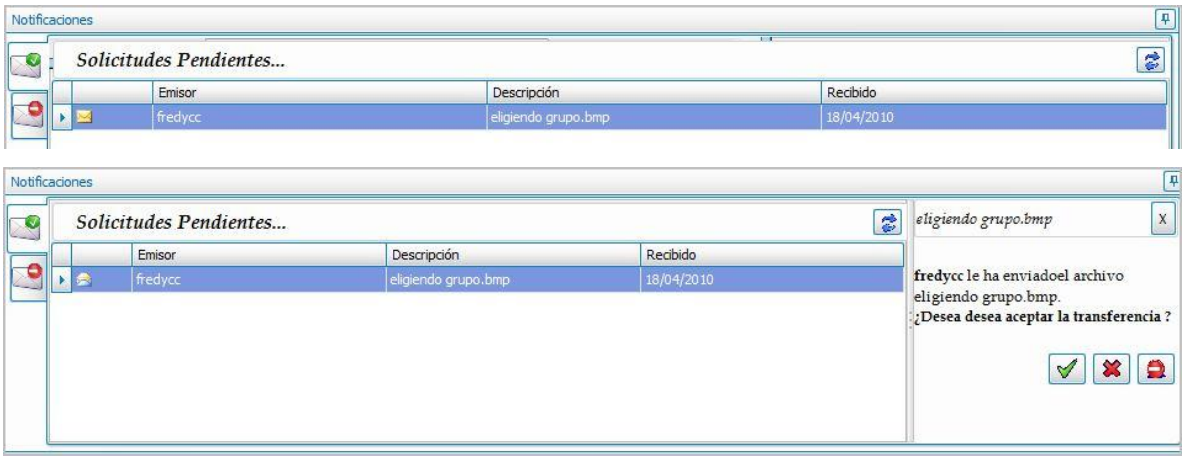
Esta opción permite al usuario del SCIP2P, enviar archivos de manera personalizada.

Para utilizar esta función se selecciona el contacto al cual se quiere enviar el archivo y se da click en el botón 

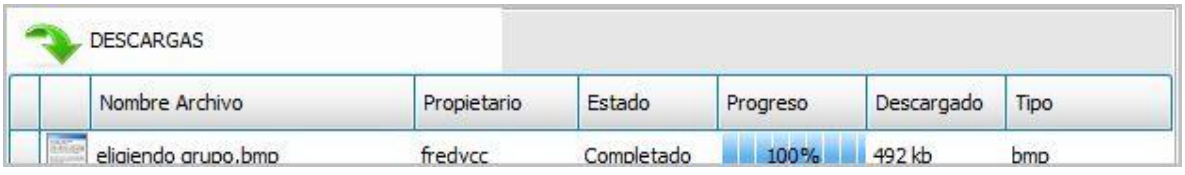
En la pantalla anterior se especifica la ruta que tiene el archivo que se enviará al contacto seleccionado. Una vez enviado el archivo, el contacto remoto recibe



Si no lo acepta en los primeros 5 segundos esta solicitud se almacenara en la sección de solicitudes pendientes.



Al dar doble click en el sobre amarillo que aparece en el control de solicitudes se puede aceptar, cancelar o denegar el archivo. Si se da click en aceptar. El progreso de descarga del archivo se observa en la parte de transferencias de archivos:

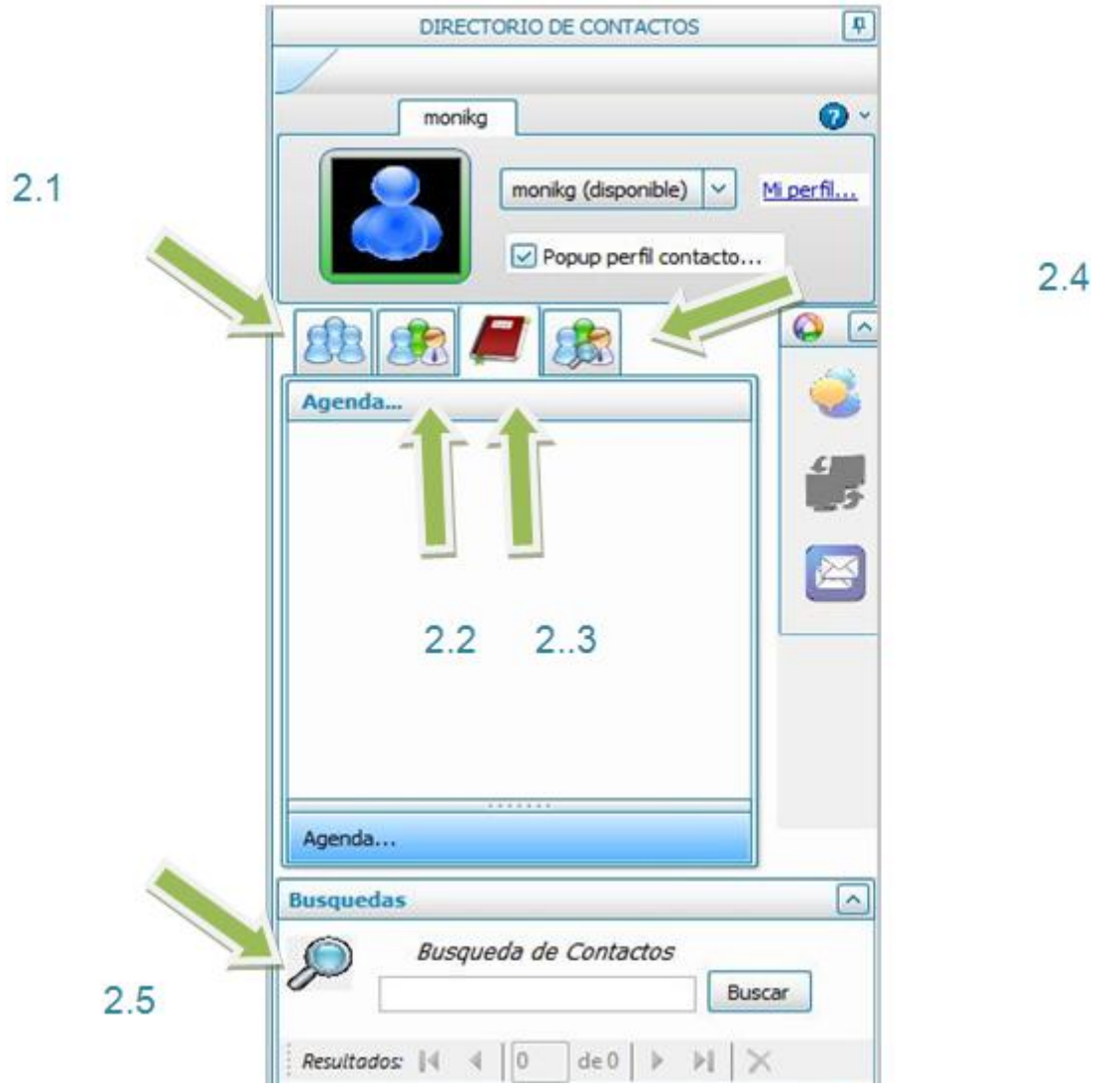


Una vez descargado el archivo, se envía una notificación a la persona que envió el archivo que este ha sido recibido satisfactoriamente por el contacto destino.


Busqueda de Contactos
Envio de Archivo.
eligiendo grupo.bmp se ha descargado satisfactoriamente....
de 0
Dibujo - Paint
ES

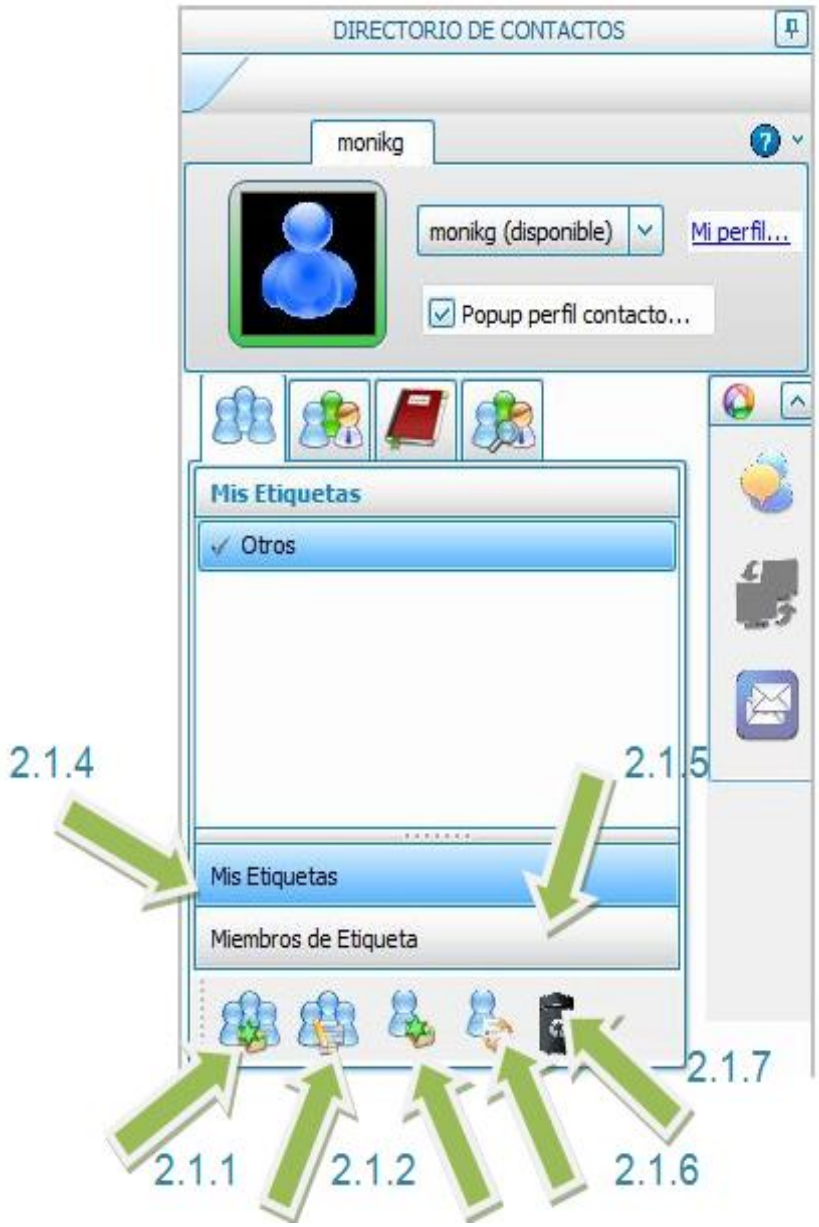
2. Manejo del Directorio

Esta parte del SCIP2P se carga al iniciar sesión y está ubicada en el lado derecho del área de trabajo, por defecto al cargase aparece abierta la pestaña de directorio (Ver Sección **"Opciones de Directorio"**) esta se muestra así:



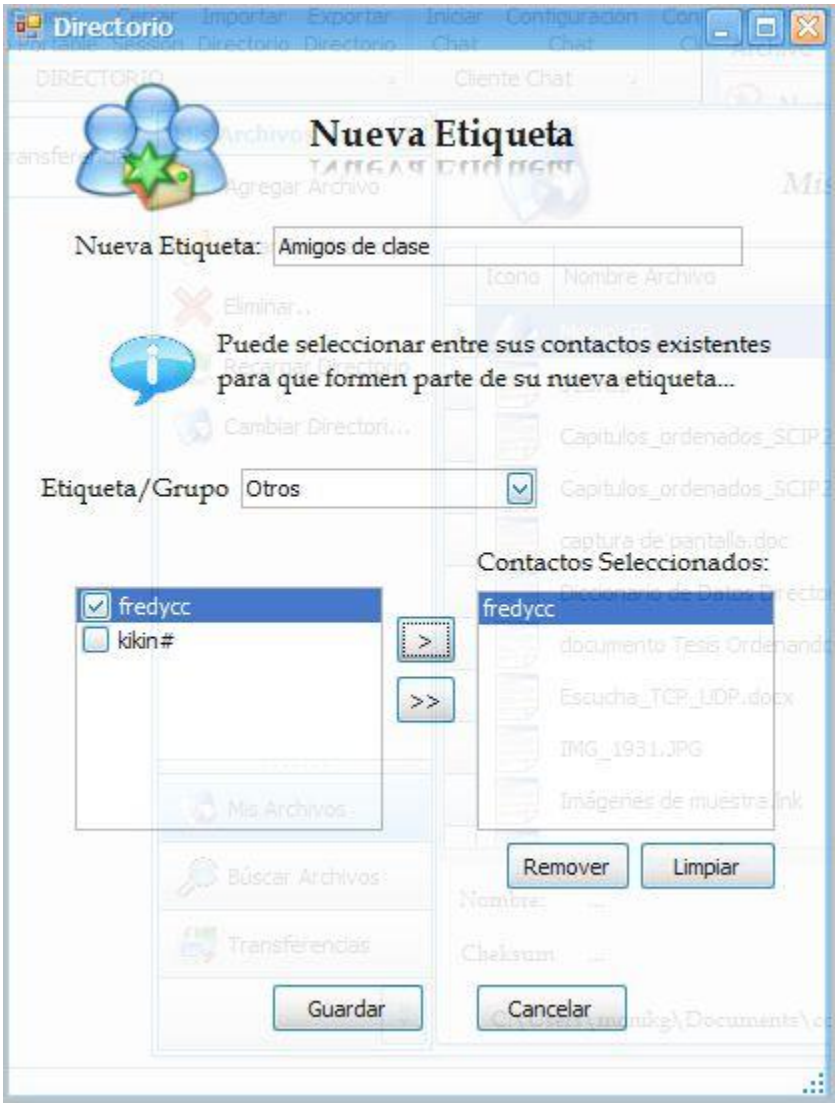
2.1. Opciones de Etiqueta

Cuando en el aplicativos SCIP2P se refiere a etiqueta, esto significa grupos privados que son propios del usuario. Estos pueden ser manipulados de la manera que a él le parezca mejor y no son observados por los contactos que pertenecen a esta. Se muestra de la siguiente manera:



2.1.1. Agregar Etiqueta

Esta opcion permite al usuario crear el titulo de sus propias etiquetas (grupos de contactos privados), lo que le ayudara a clasificar a sus contactos por categorias según a él le convenga. Al dar clic en el boton  mostrara la siguiente ventanta:



En esta pantalla aparecen pueden hacerse dos acciones: solamente crear una nueva etiqueta o crear la y ademas ingresarle contactos desde otras etiquetas o grupos.

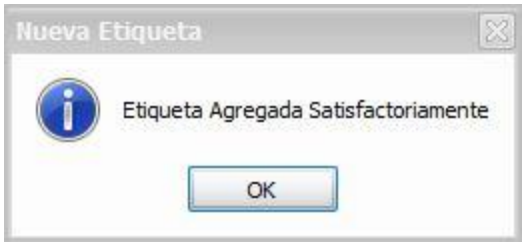
Si se desea solo crear una etiqueta, con solo escribir el nombre en el textbox nueva etiqueta y dar clic en *Aceptar* ya esta listo.

Si se quiere además guardar contactos en la nueva etiqueta, se selecciona del combobox el nombre de la etiqueta desde donde se copiaran contactos a la nueva, al elegirlo estos apareceran automaticamente al lado izquierto del cuado inferior. Para agregar un solo contacto a la nueva etiqueta, basta con seleccionar el contacto que se desea y dar clic en el boton de la parte central 

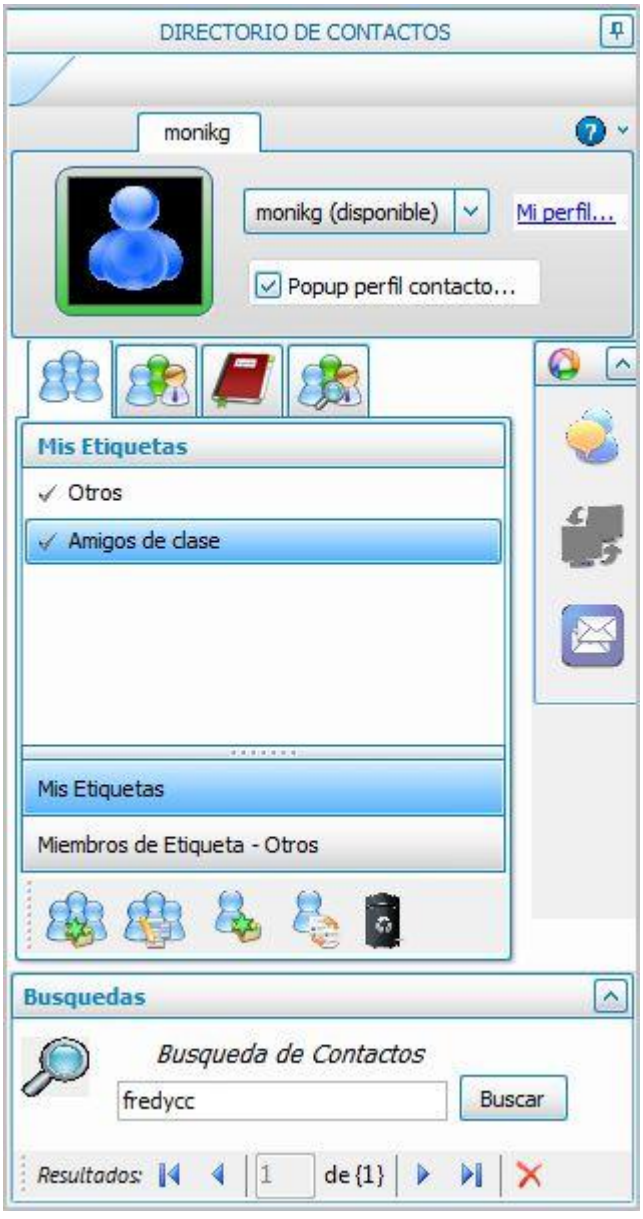
o ya sea agregar todos los contactos de la etiqueta seleccionada en el combobox dando clic en el boton 

y el/los contacto(s) se mostrara(n) automaticamente en el cuadro derecho indicando que el/los usuario(s) pertenecera(n) a la nueva etiqueta. Si por error se ingresaron contactos que no va en la nueva etiqueta, desde esta misma ventana se puede eliminar dando clic en el boton *Remover.* Cuando ya se esta seguro del nombre de la nueva etiqueta y de los contactos que formaran parte de ella se da clic en el boton *Guardar.* Si se produce algun error en el proceso, el sistema mostrará el siguiente mensaje:

De lo contrario, mostrará:



Luego, ya en la ventana de Directorio en el área de lista de etiquetas (Ver Seccion **"Lista de Etiquetas"**) mostrara el nombre de la etiqueta creada, asi:

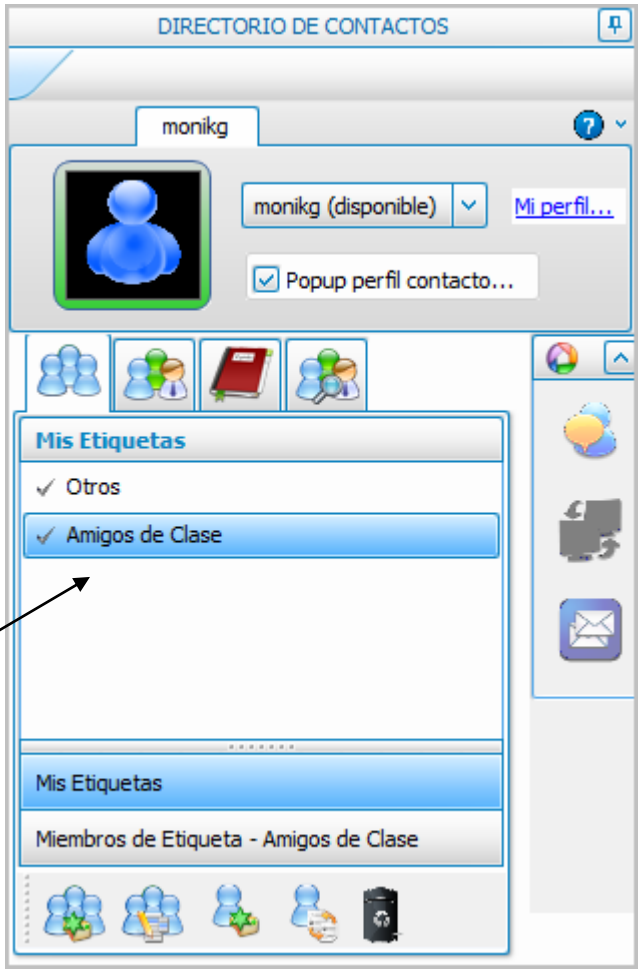


2.1.2. Modificar una Etiqueta.

Esta opcion permite al usuario modificar unicamente nombre de una etiqueta.

Se toma efecto al dar click en el boton  de la ventana de Directorio, seleccionando la pestaña Etiqueta en el menu que aparece en la parte inferior y se muestra asi:

En este ejemplo como se puede notar se ha modificado el nombre de la etiqueta Amigos de clase por el nombre Amigos de Clase con la letra C en mayúscula . Dar click en en boton *Guardar* se puede observar el cambio en el listado de etiquetas (Ver Sección **"Listado de Etiquetas"**). Y aparece así:



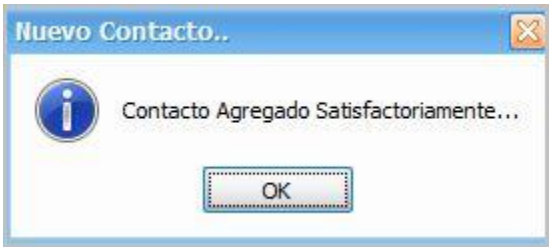
2.1.3. Agregar un Contacto.

Para realizar esta acción en el SCIP2P hay dos maneras:

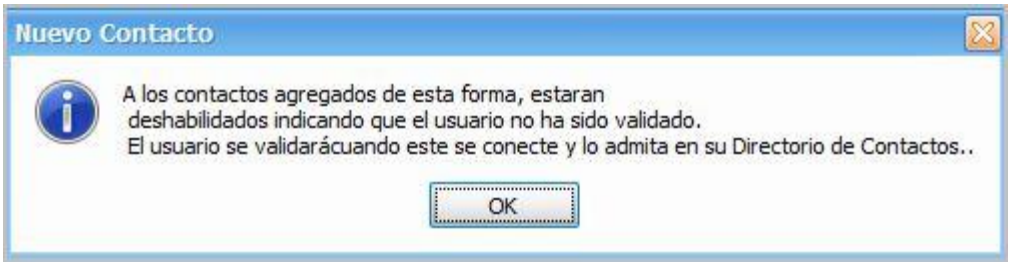
- *Desde la pestaña de Directorio* (Ver Sección **"Opciones de Directorio"**).
- *Desde el menú de Etiquetas*

Para agregar un contacto desde esta opción se da click en el botón  , y pararecera la siguiente ventana:

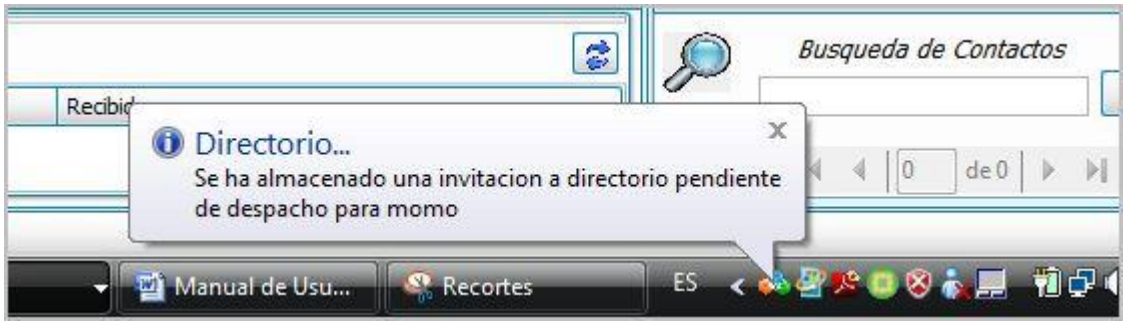
Para agregar al contacto es necesario saber en nickname del contacto que se desea agregar. Una vez escrito el contacto, se selecciona en la parte inferior a que etiqueta pertenecera y se da click en *Guardar.* Una vez guardado el cambio el sistema envia el siguiente mensajes.



Si la persona a la que se invita NO esta conectada, Muestra el mensaje:



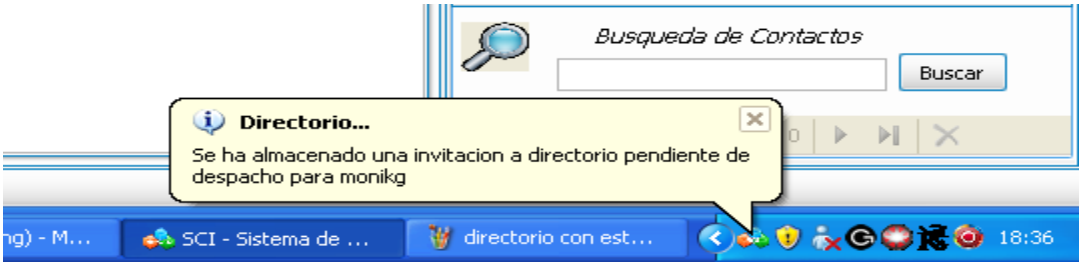
Y automaticamente el sistema informa al usuario que hizo la invitación que tiene una invitacion a directorio pendiente de ser despachada.



- ✓ **Recibiendo una Invitacion a Directorio Privado.**

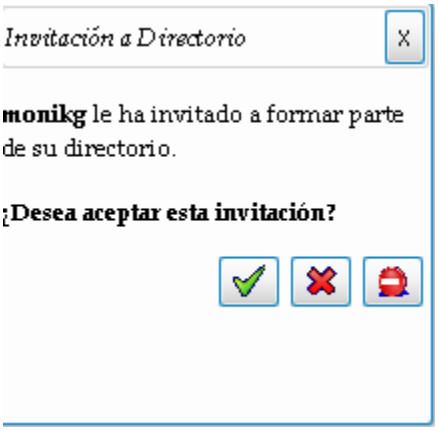
Al otro lado cuando el usuario remoto, que es al cual se le envio la invitación se conecta, recibe la siguiente solicitud:

Desde la cual se puede aceptar la invitacion inmediatamente, al pasar 30 seg y no se elige ninguna opción, el sistema envia la siguiente notificación:



La que indica que la invitación se guardara en la parte de *Solicitudes Pendientes.* Y se vera así:

Al dar doble click en el icono  se abre las opciones:

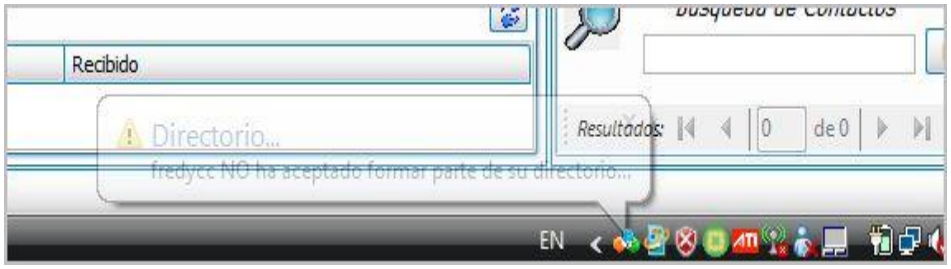


- *Aceptar*, en la cual el usuario remoto decidirá en que etiqueta agregarlo.

-

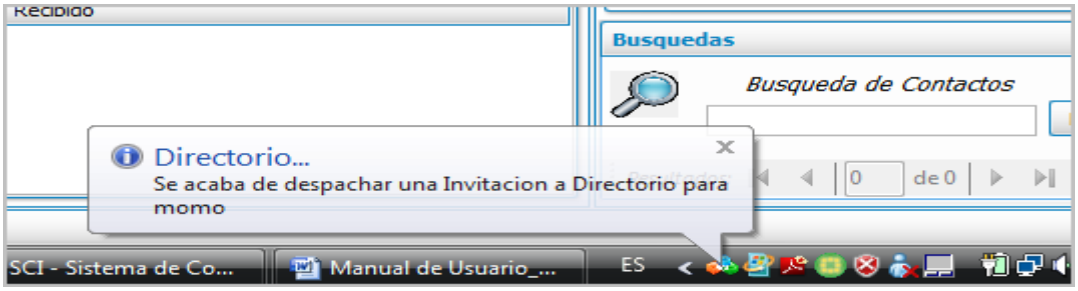
Una vez seleccionada la etiqueta en que se quiere guardar, ya se muestra el usuario según el estado que tenga.

- Denegar, si se da click en esta opcion el SCIP2P, se envia el siguiente mensaje al contacto que lo invito:



  Y borra de la lista de contactos de la etiqueta, el contacto que denego la invitacion. Y por consiguiente, el contacto que lo denego no lo agrega en ninguna etiqueta. Pero el usuario que lo denego puede ser invitado por el mismo contacto que lo invito de nuevo.
- Bloquear (lo que ya no permitira que el usuario remoto realice invitación a este).

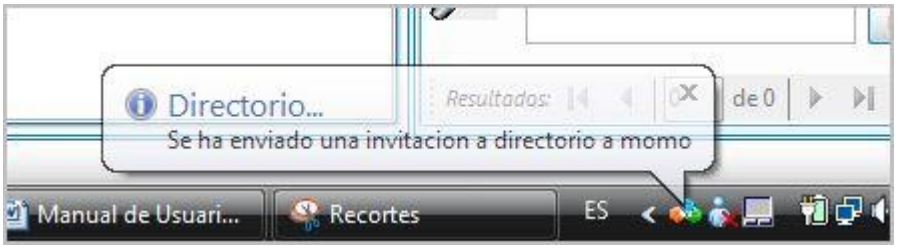
Una vez la acepta, el usuario que envio la invitacion, recibe la siguiente notificacion:



Y ya se puede apreciar el contacto conectado en el listado de contactos.

- *Si la persona que se invito SI esta conectada,*

Muestra la siguiente notificación:

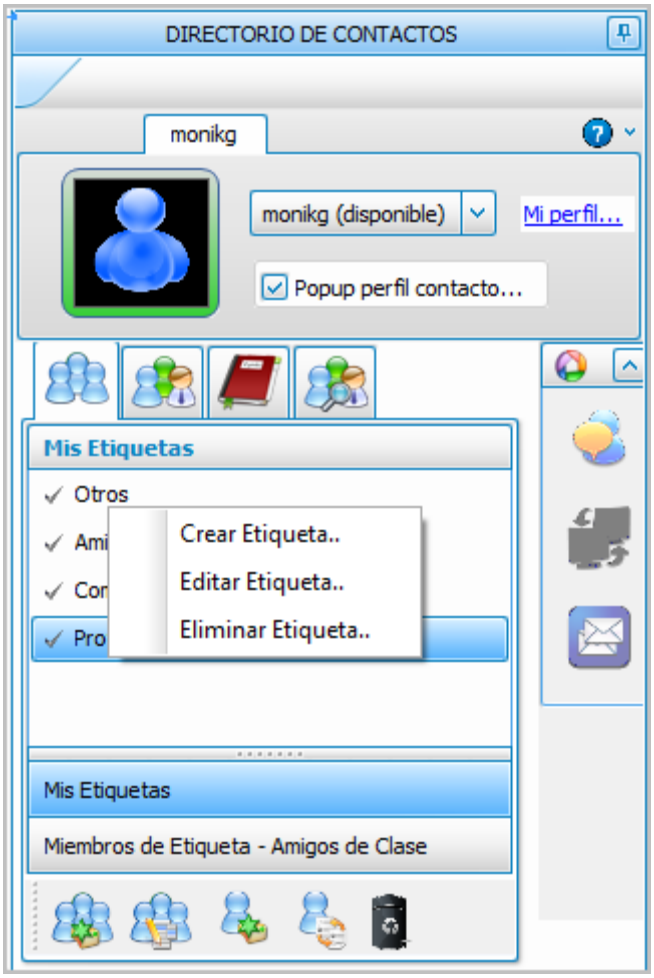


Y el cliente remoto recibe el mensaje, tal como se describe en el proceso de Recibiendo una Invitacion a Directorio Privado (Ver Sección **"Recibiendo una Invitacion a Directorio Privado"**).

2.1.4. Opciones de Ver Mis Etiquetas

Esta opción permite al usuario ver el listado de etiquetas (grupos privados) que ha creado.

Al dar click derecho a alguna de las etiquetas seleccionadas se tiene la opción de:



- *Crear Etiqueta,* (Ver Sección **"Crear Etiqueta"**)
- *Editar Etiqueta,* (Ver Sección **"Editar Etiqueta"**)
- *Eliminar Etiqueta* (Ver Sección de **"Eliminar Etiqueta"**)

2.1.5. Opciones de Miembros de Etiqueta

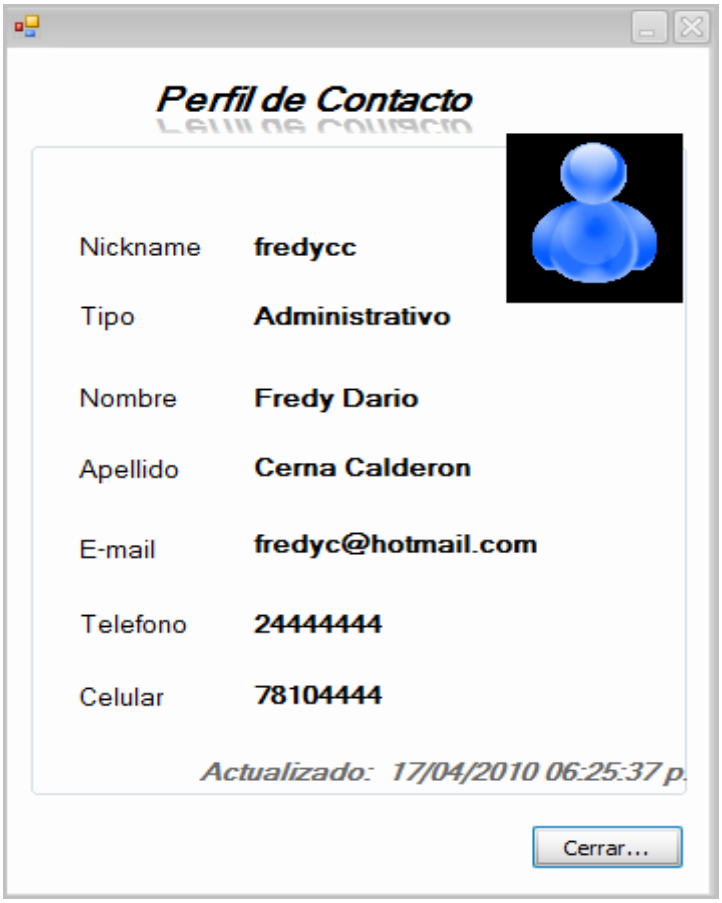
Esta opción permite al usuario ver los contactos que tiene agregados en la etiqueta. Para poder utilizar esta función, primeramente se tiene que seleccionar una etiqueta en la parte de listado de etiquetas, luego dar click en la parte de *Miembros de Etiqueta,* que se encuenta en la parte inferior de la ventana de Directorio. Una vez realizado el procedimiento se mostrara la lista de contactos, así:

Teniendo chequeada la opcion de *popup perfil de contacto* y posicionandose sobre el contacto se pueden mostrar dos opciones, dependiendo de si el cliente ya acepto la invitacion o no:

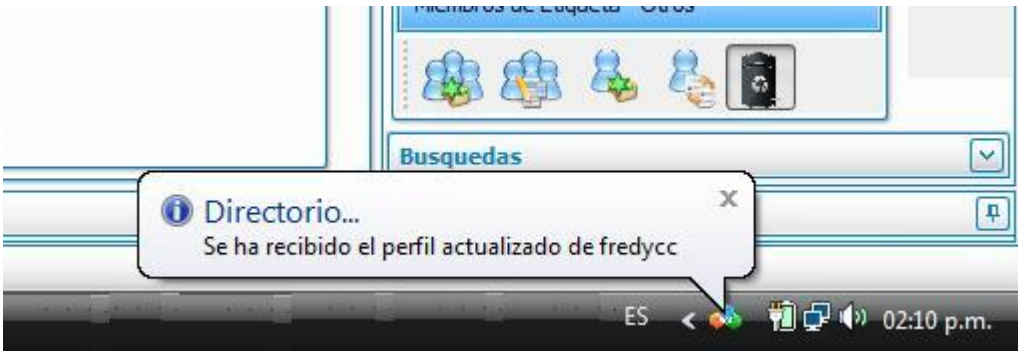
- Si ya Acepto la Invitación

Desde el cual se pueden:

- *Ver el perfil*, al dar click en esta opcion aparece la ventana:



- *Actualizar el Perfil*, al dar click en esta opcion se muestra la siguiente notificación:

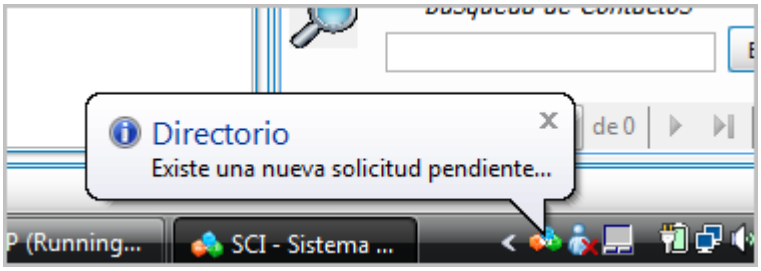


- Solicitar ver los grupos publicos (Ver Seccion **"Opciones de Grupos"**)

En esta opcion pueden aparecer dos opciones :

- ✓ Si el contacto NO tiene grupos publicos: y muesta la siguiente notificación:

- ✓ Si el contacto SI tiene grupos publicos: muestra la siguiente notificación:



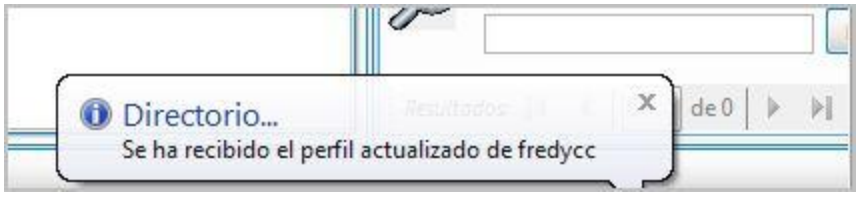
Y en la seccion de solicitudes pendientes se muestra el listado de grupos que posee el usuario que se selecciono, desde ahí se puede hacer una solicitud de invitacion al usuario remoto (Ver Seccion **"Recibiendo Invitacion a Grupo Publico"**)

- ♦ Si No ha aceptado la Invitación

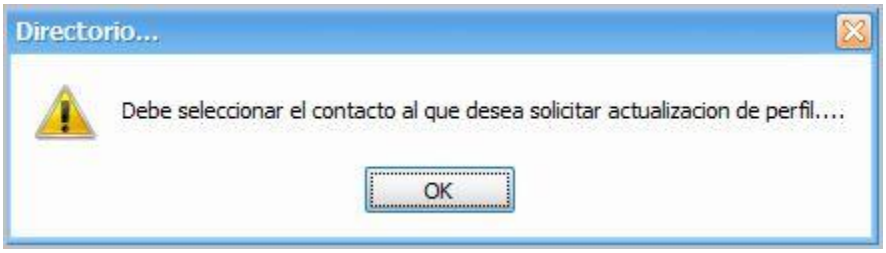
Al dar click al enlace que aparece en azul, *Eliminar Contacto,* el enlace es eliminado y cuando se conecte ya no recibira la invitación.

2.1.6. Opciones de Actualizar Perfil

Esta opción Permite al usuario recibir , información nueva de un contacto. Para realizar esta acción, el usuario debe seleccionar el contacto del cual se desea actualizar el perfil, una vez seleccionado (desde el listado de contactos), da clic en el boton  , y se muestra la siguiente notificacion:



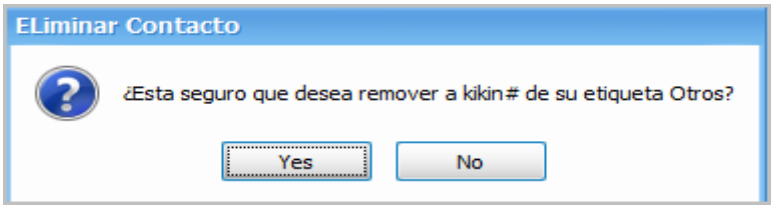
Si no se ha seleccionado un contacto, muestra el mensaje:



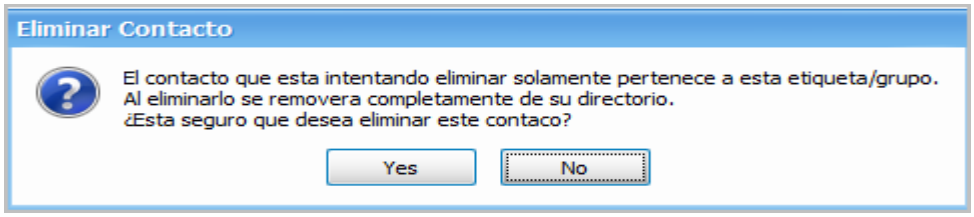
La actualizacion del perfil de un contacto también se puede realizar de la opcion de Miembros de Etiqueta (Ver Sección **"Opciones de Miembros de Etiqueta"**).

2.1.7. Eliminar

Esta opción permite al usuario borrar tanto contactos como etiquetas. Con la advertencia que para poder borrar una etiqueta esta tiene que estar vacia. Para el uso de esta funcion se selecciona el contacto/etiqueta que se desea eliminar, y se da click en el boton  , y muestra el siguiente mensaje:

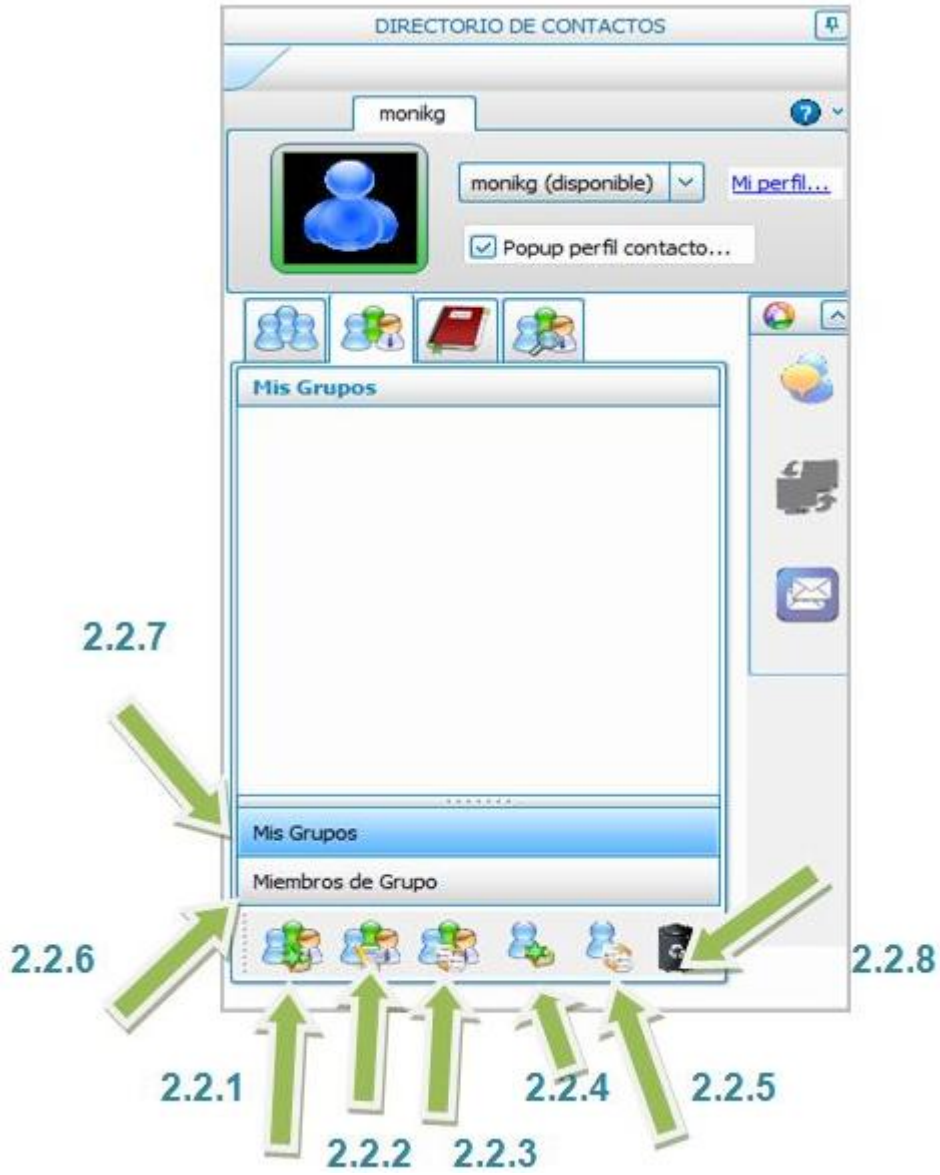


Si se da click en *Yes,* aparece la ventana, en la que se elige finalmente si se quiere eliminar o no, así:



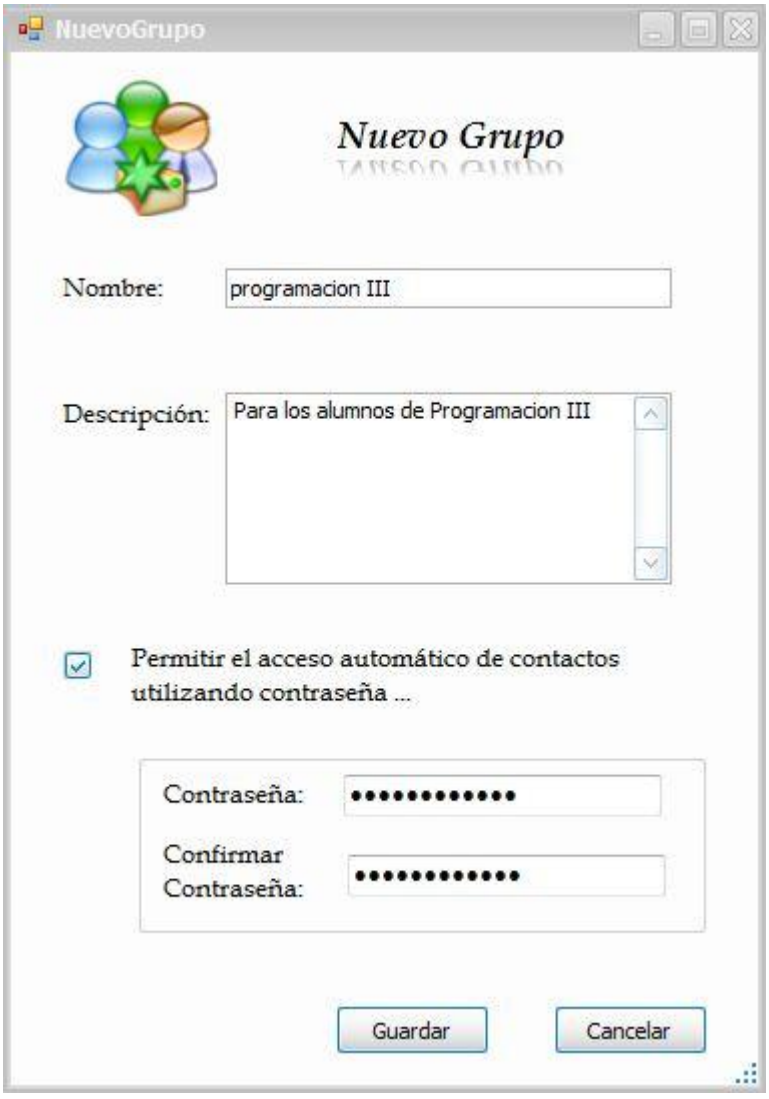
2.2. Opciones de Grupo

Está Opcin permite al usuario crear grupos de trabajo publicos, ya que todos los usuarios del SCIP2P pueden tener acceso a estos, siempre y cuando el anfitrion (usuario que crea el grupo) lo acepta. Al dar click en la pestaña Directorio, se muestran las siguientes opciones:

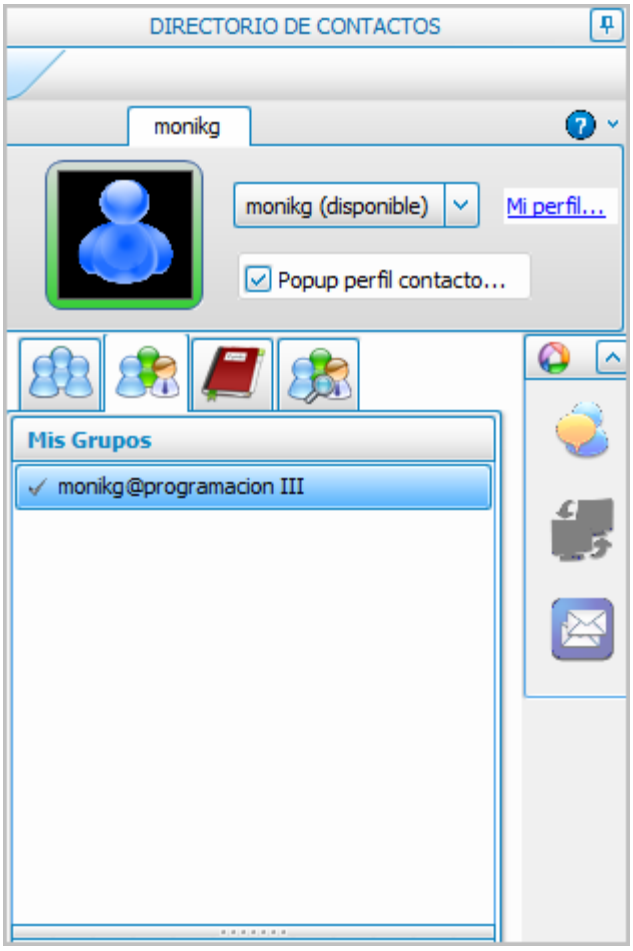


2.2.1. Agregar Grupo

Está opcion permite crear los grupos de trabajo de los cuales el usuario será anfitrión. Para agregar un grupo se da click en el boton  , y parece la siguiente ventana:

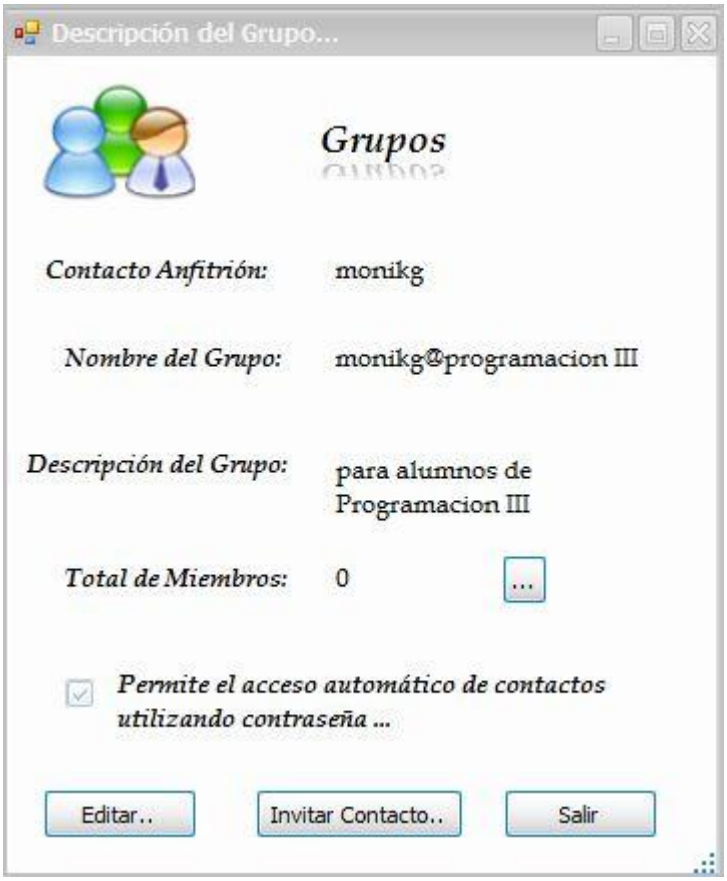


Desde esta pantalla se agrega el nombre del grupo que se esta creando, como también se elige si este grupo va a ser necesario autenticacion por medio de la contraseña. Una vez rellenada la informacion, se da click en *Guardar* y los cambios tienen efecto en la lista de grupos. Así:



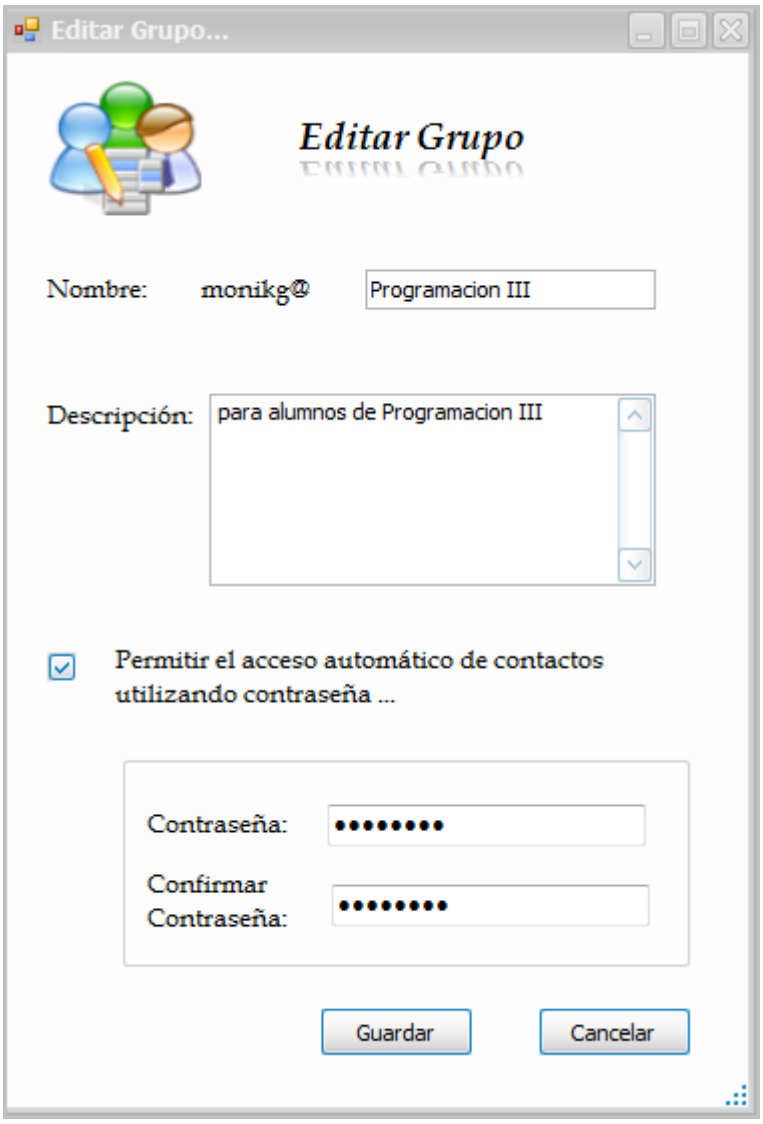
2.2.2. Detalle de Grupo

Desde esta opcion del menu del Directorio, se pueden realizar una serie de acciones. Para poder hacer uso de esta funcion se da clic en el botón  , el cual despliega la siguiente ventana:



En esta ventana el sistema muestra toda la informacion pertinente al grupo selecionado previamente. Y tiene las siguientes opciones:

- *Editar,* al dar click en esta opcion se muestra la ventana:



Desde esta ventana el usuario anfitrión tiene la posibilidad de cambiar el nombre, quitar la opcion de *Permitir el acceso automatico de contactos utilizando contraseña* o si se quiere conservar cambiarla.

- *Invitar contacto a grupo Publico,* (Ver Seccion **"Agregar Contacto a Grupo Publico"**).

2.2.3. Sincronizar listado de Miembros

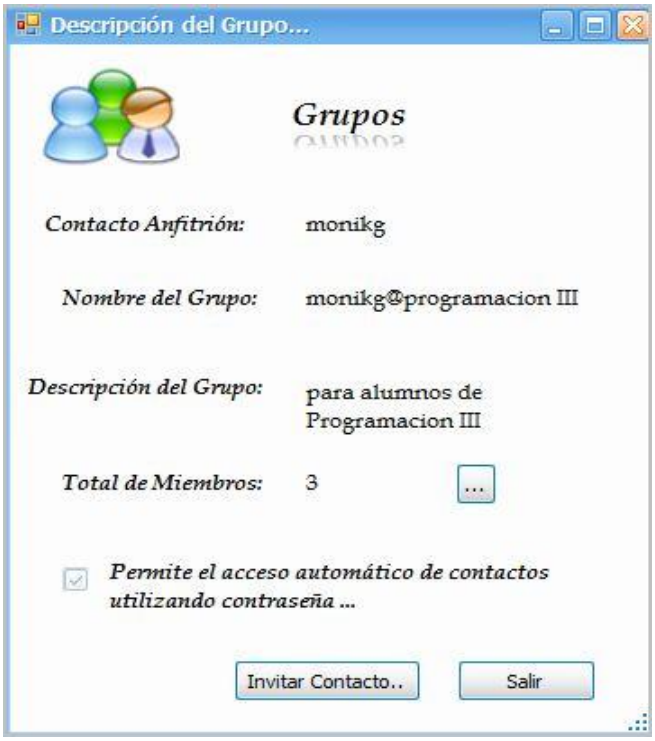
Esta opción permite al usuario, como su nombre lo indica sincronizar el listado de miembros que pertenecen a un grupo publico en caso que no sea el anfitrion.

El usuario se conecta y para saber si hay mas contactos nuevos en un grupo publico, se selecciona el grupo publico que se desea actualizar, luego se da click en el boton  , al dar click en este si hay clientes conectados que pertenecen al grupo publico se actualizara el listado de contactos de dicho grupo publico, de lo contrario no se hará nada.

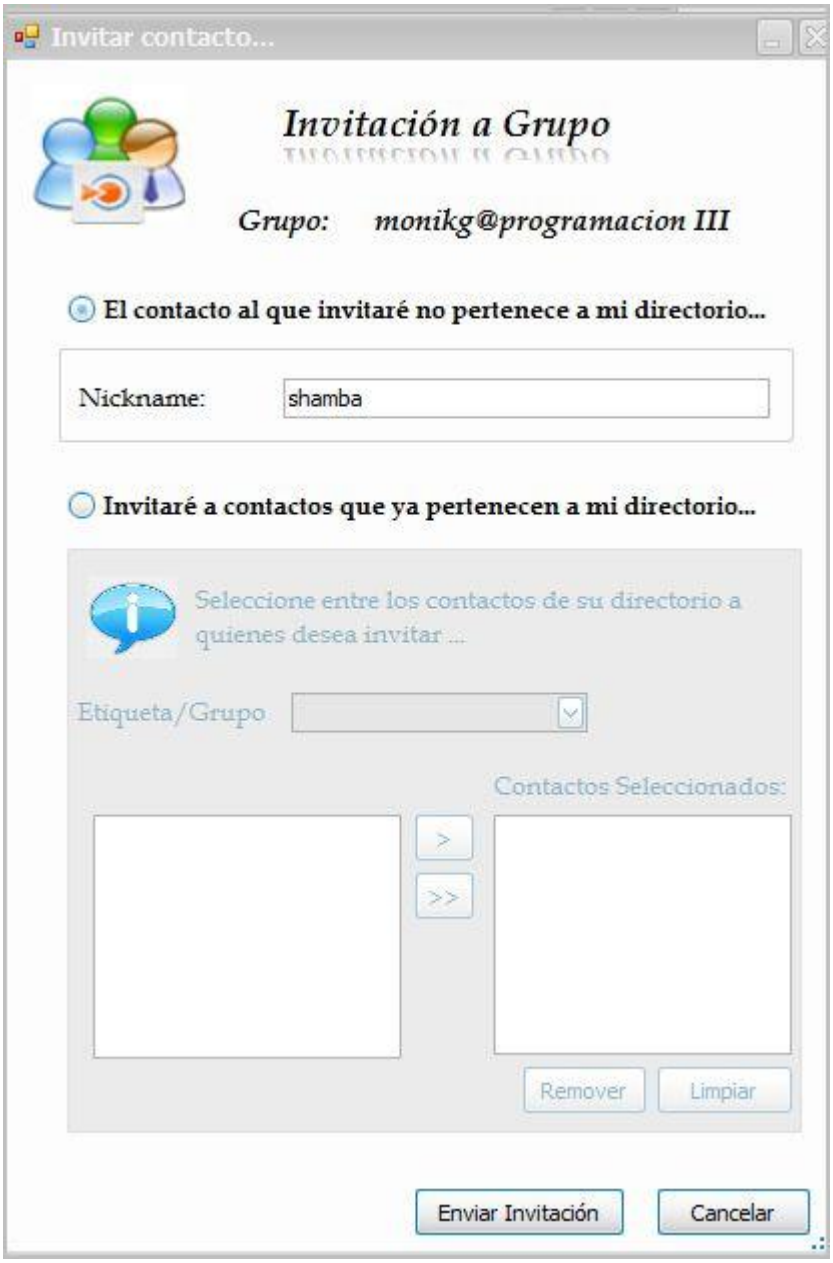
2.2.4. Agregar Contacto a Grupo Público.

Para realizar esta función, hay varias maneras:

- Dando click en el botón  se abre la siguiente ventana:



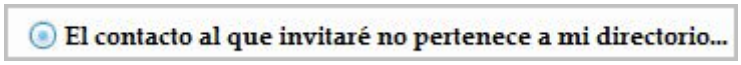
En esta pantalla aparece la información más importante del grupo, para agregar al contacto, se da click en el botón *Invitar Contacto..* y aparece la siguiente ventana:



En la que se tienen dos opciones, invitar a un contacto que no se tiene previamente agregado en ninguna etiqueta o grupo e invitar a un contacto que ya pertenece a otra etiqueta o grupo del mismo directorio.

- *Para agregar a un contacto que no está previamente en ninguna etiqueta o grupo*

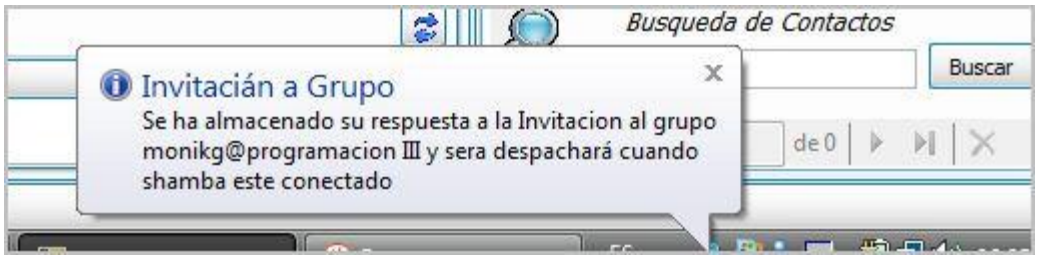
  Se selecciona el chequesito en

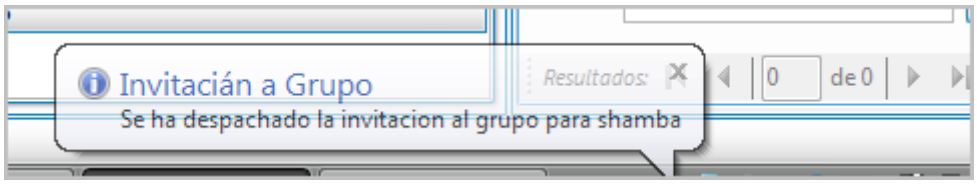
  Para esta opción es necesario saber el nombre del contacto. Una vez agregado el contacto , se da click en *Invitar a Contacto …*

  - **Procesos que se dan al Invitar a un contacto**

  Si este no está conectado se muestra la siguiente notificación:

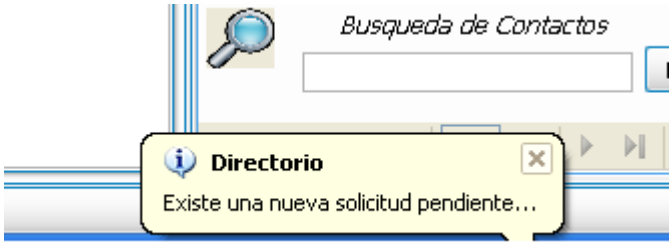
  

  Cuando el usuario se conecta o en caso que el contacto al que se acaba de invitar está en línea, aparece la siguiente notificación:

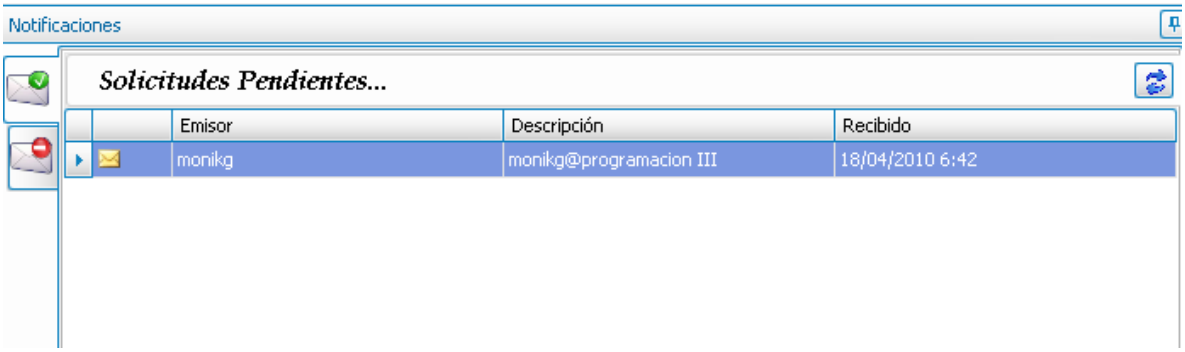
  

  Al contacto remoto le aparece el siguiente mensaje:

Si en esta ventana no se selecciona una de las opciones que aparecen en un transcurso de 5 seg, la invitación se guardara en la sección de Solicitudes Pendientes. Primero se mandar una notificación que diga:

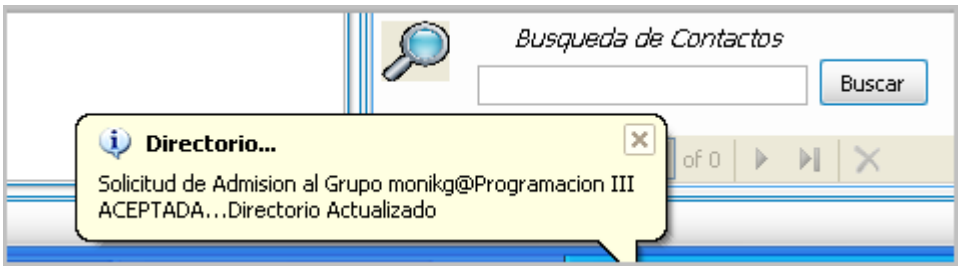


Y en la sección de Solicitudes Pendientes aparecerá así:



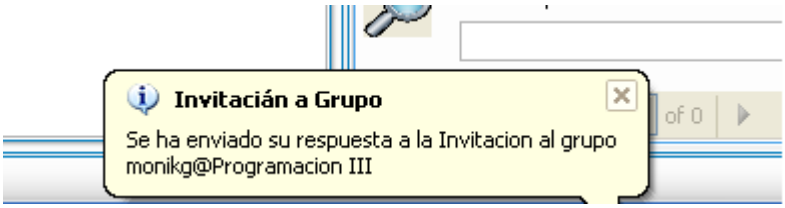
Dando doble click en el sobre, esta ventana muestra las opciones para poder *Aceptar, Denegar o Bloquear a este contacto.*

Al aceptar la invitación, se muestra al contacto que acepto la siguiente notificación:

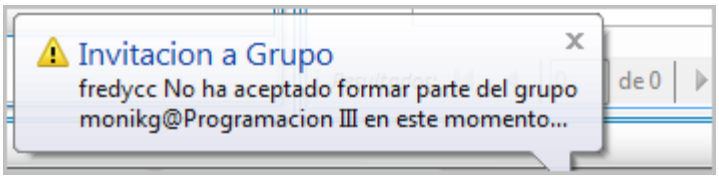


Y al usuario (anfitrión) quien hizo la invitación, este:

Si La invitación se Deniega, al usuario que no acepto la invitación le parece:

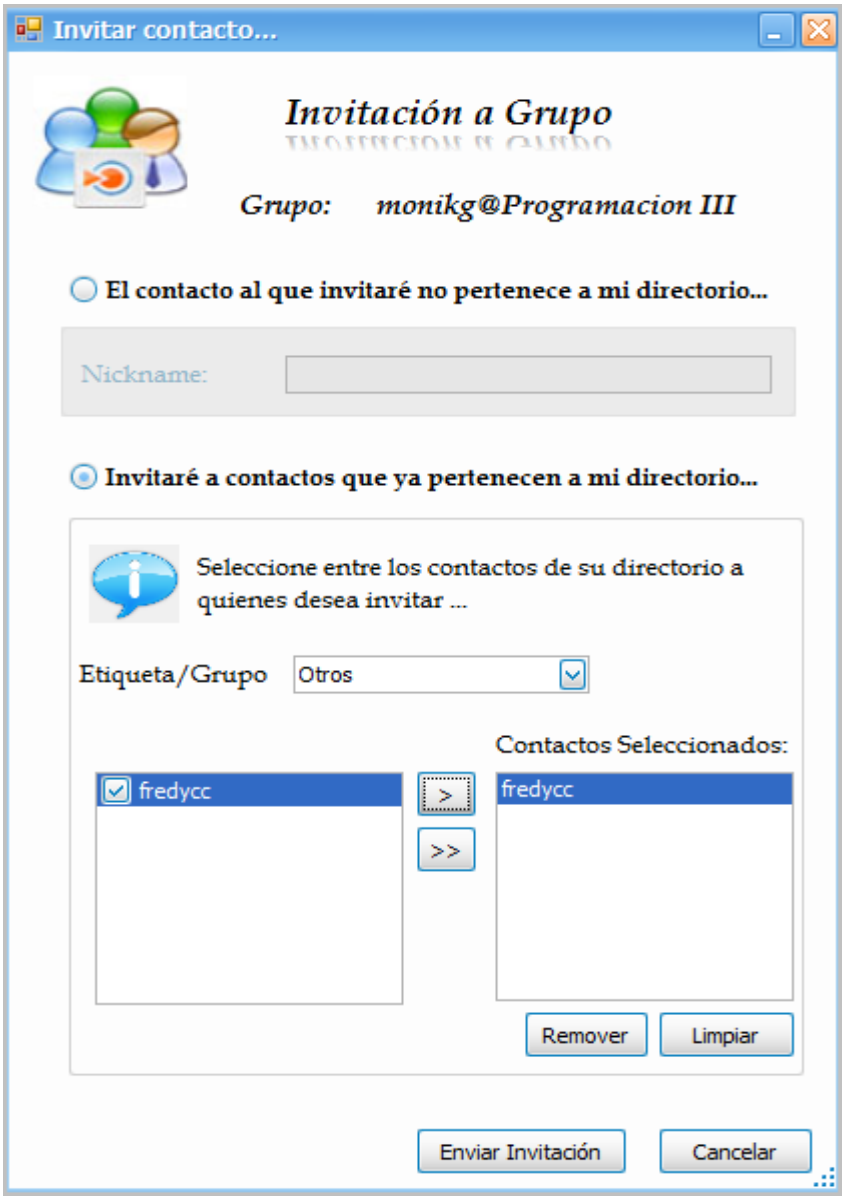


Y al anfitrión:

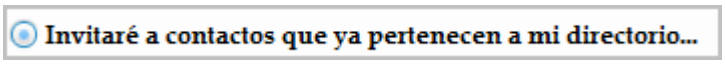


Si la invitación se Bloquea, esta no muestra ninguna acción ni mensaje. Es una acción de ignorar la invitación.

- *Para agregar a un contacto desde otra etiqueta o grupo*



Para invitar a un usuario a formar parte del grupo seleccionado. Se da click en el chequesito:



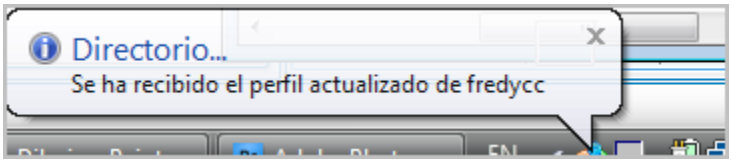
En la lista desplegable que aparece en Etiqueta/Grupo, se selecciona desde donde se quiere agregar al nuevo usuario del grupo, en este caso Programación III. Al seleccionarla aparecerán en el cuadro de la izquierda todos los contactos que pertenecen a la Etiqueta/Grupo seleccionados de la lista desplegable. Desde

ahí, se puede seleccionar uno por uno los contactos y dar click en el botón  para que los seleccionados aparezcan en el cuadro de la derecha o si se desee que todos los contactos de la Etiqueta/Grupo formen parte del grupo, se da click en el botón  , una vez aparezcan todos los contactos en el cuadro de la derecha, ya se está listo para hacer la invitación, dando click en el botón *Enviar Invitacion …*

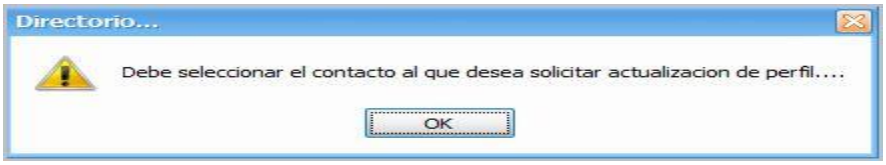
Una vez enviada la invitación a formar parte del grupo la secuencia de mensajes son las mismas que al haber invitado a un contacto sin tenerlo agregado en otra Etiqueta/Grupo (Ver Seccion **"Procesos que se dan al Invitar a un contacto"**

2.2.5. Actualizar Perfil.

Esta opción permite al usuario actualizar el perfil de los contactos que tiene agregados en su lista de grupos. Para hacerlo se selecciona el contacto que desea actualizar y se da click en el botón  , en ese momento el sistema mostrara una notificación que el perfil se ha actualizado, así:

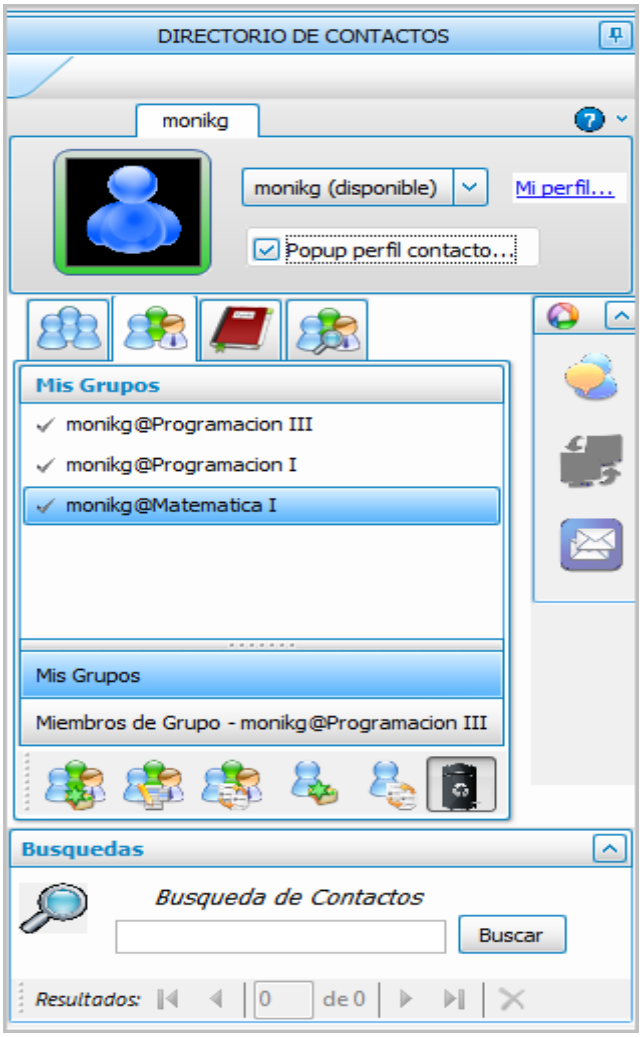


Si el contacto que se quiere actualizar el perfil no fue seleccionado previamente, entonces el sistema mostrara:

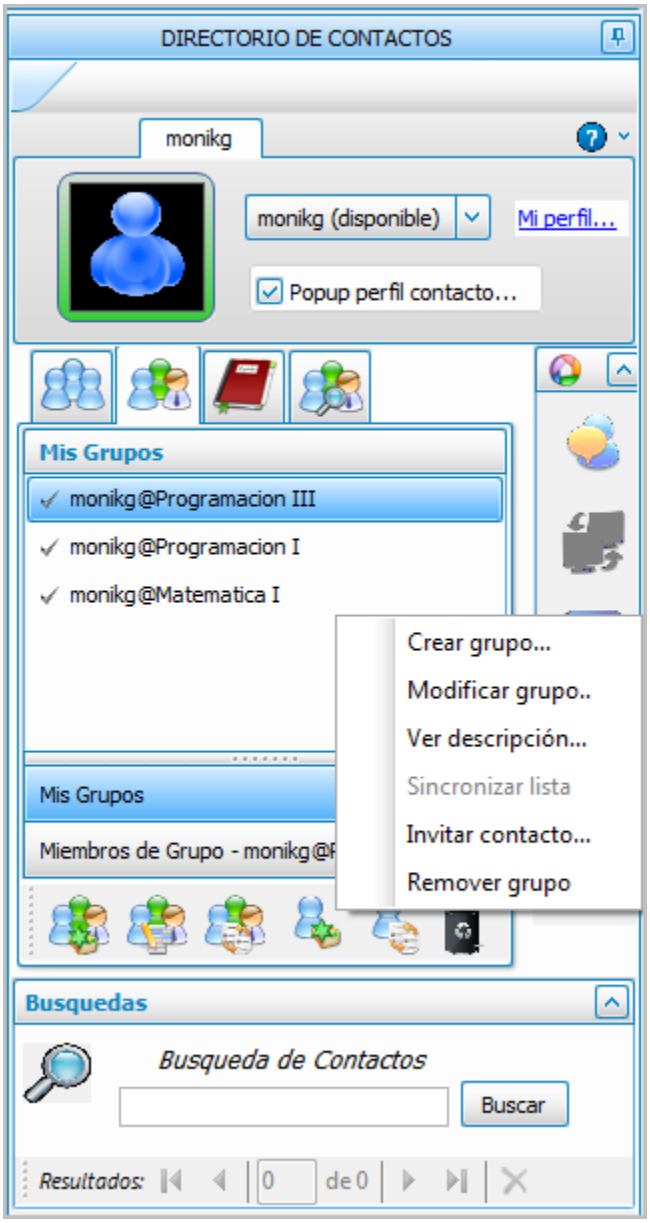


2.2.6. Mis Grupos

Esta opción muestra al usuario la lista de todos los grupos que ha creado.



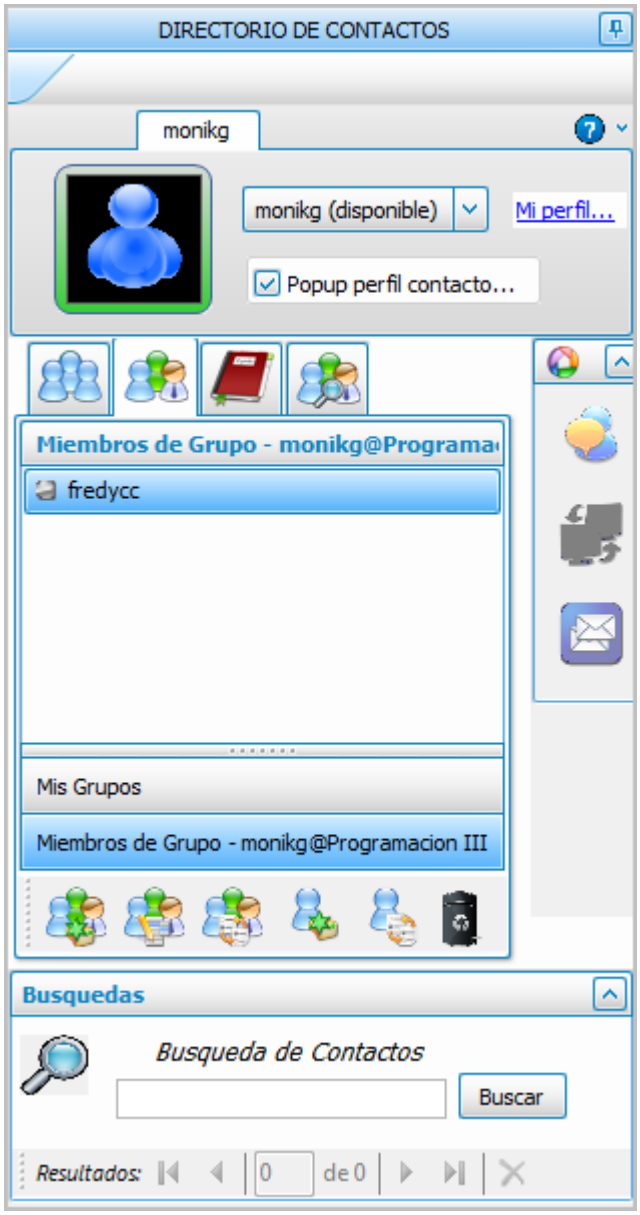
Al dar click derecho sobre el área donde aparece el listado de grupos, aparece un menú que permite hacer todas las acciones que se hacen desde el menú inferior, y aparece así:



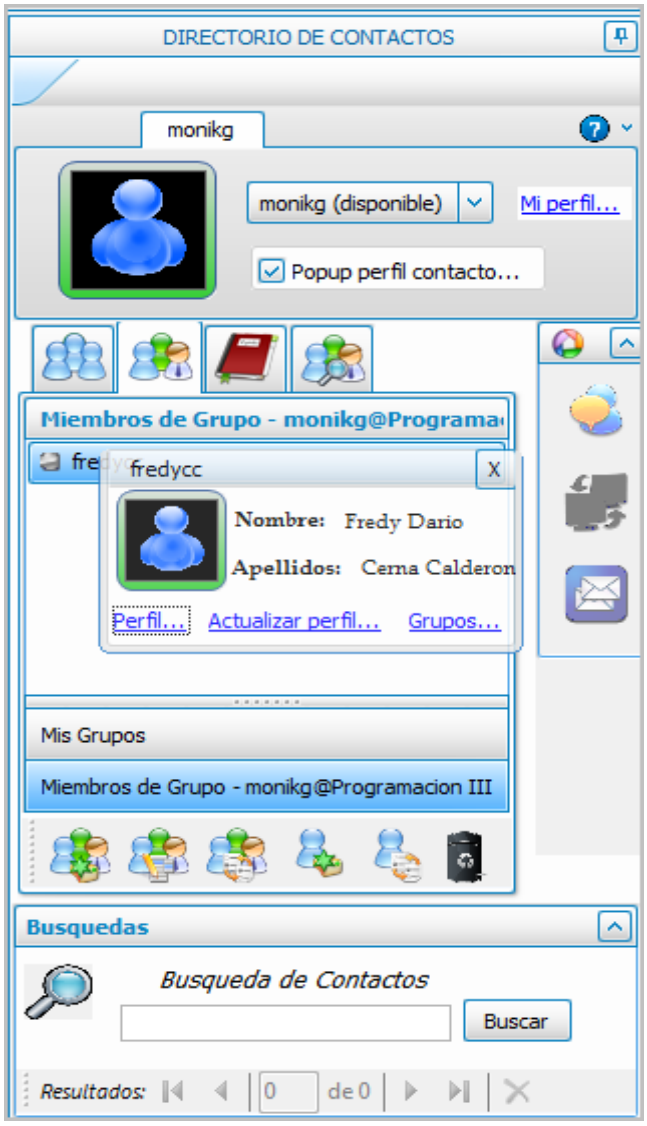
2.2.7. Miembros de Grupo

Esta opción del SCIP2P permite al usuario mostrarle todos los contactos que pertenecen al grupo seleccionado.

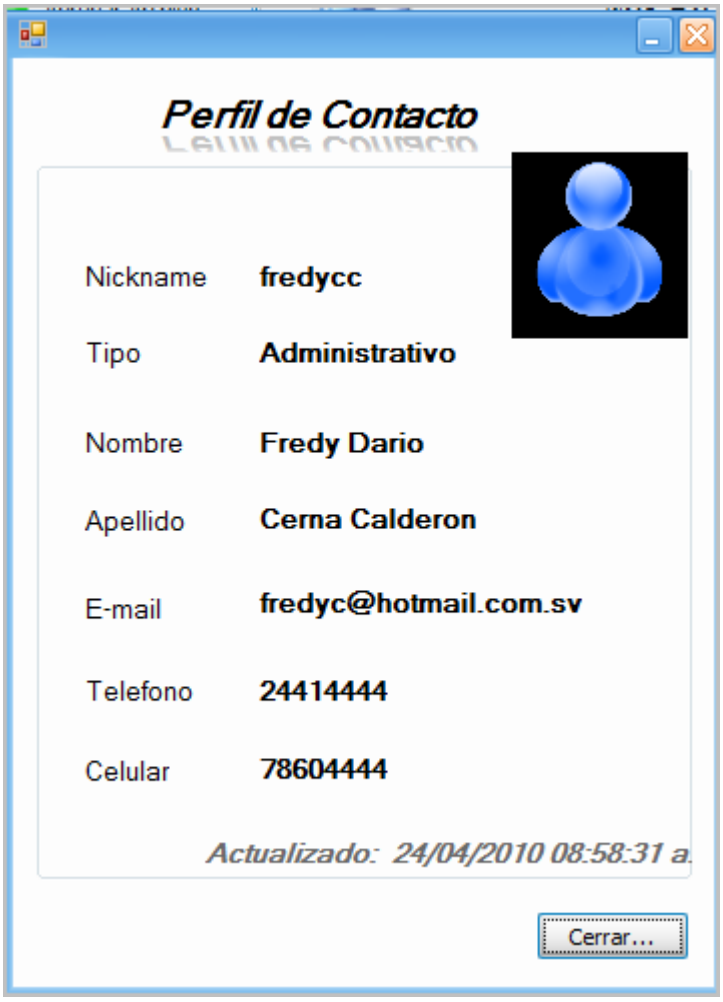
Para hacer uso de esta opción se selecciona un grupo público y se da click en el botón Miembros de Grupo



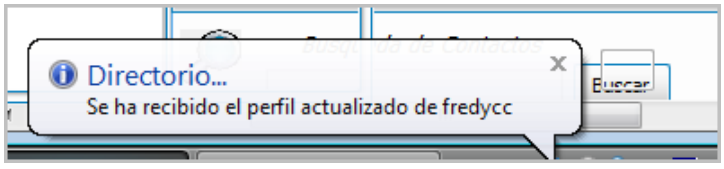
Si se tiene seleccionada la opción de Popup y se pasa el mouse sobre un contacto, se muestran así:



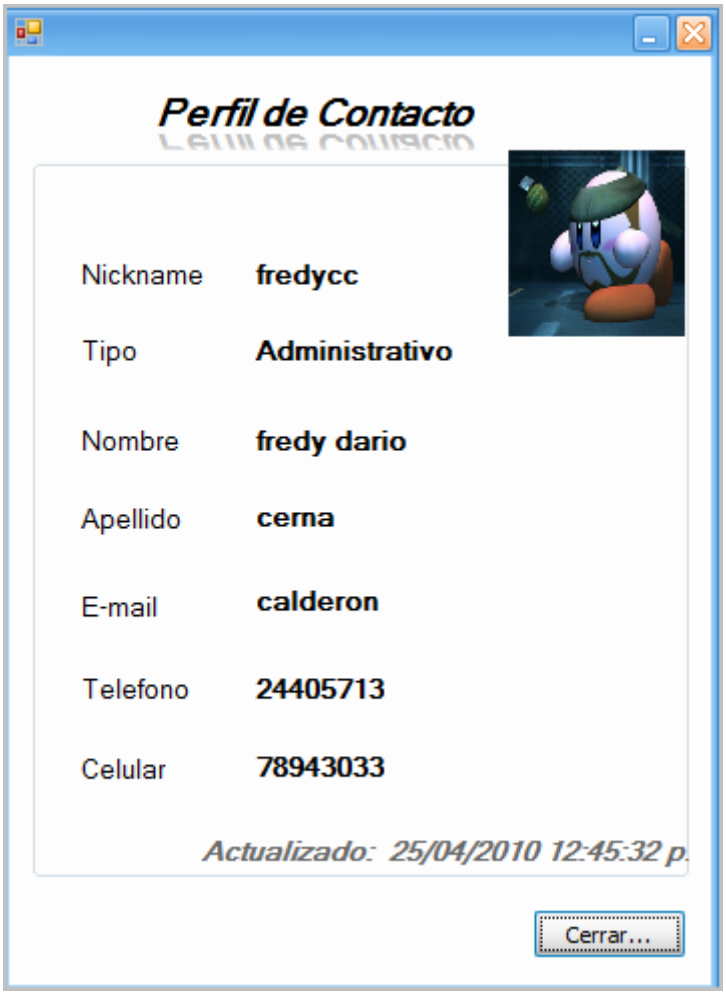
En estas opciones se permiten ver el perfil que se tiene actualmente del contacto seleccionado.



Si el contacto a hecho alguna modificación en su perfil aparece habilitada la opción *Actualizar perfil…* Al dar click en dicha opción se muestra una notificación avisando que se actualizo el perfil del contacto seleccionado.

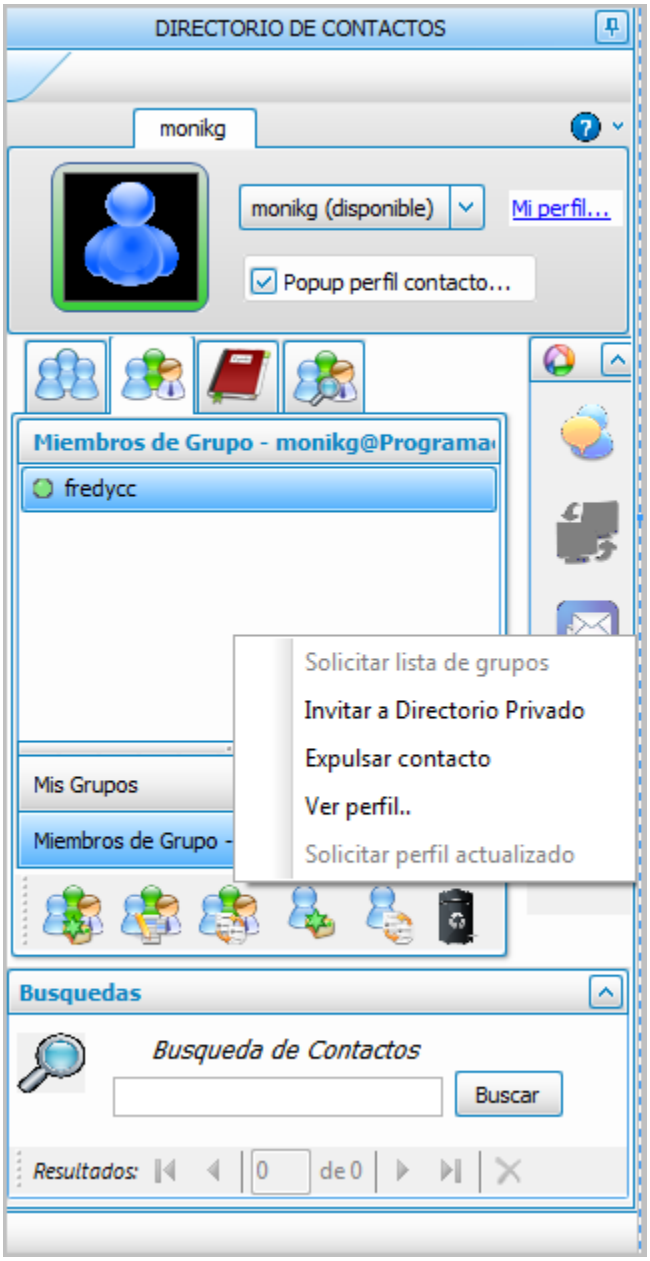


Y ya se muestra la actualización que hizo el contacto:



También, desde dicho Popup se puede solicitar la lista de grupos públicos del contacto seleccionado.

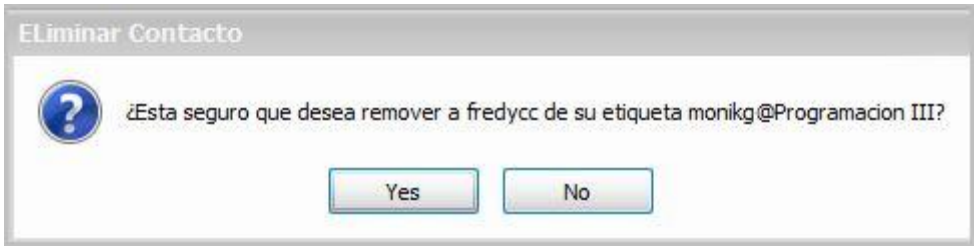
Al dar click derecho en el área donde aparece el listado de contactos aparecen las siguientes opciones:



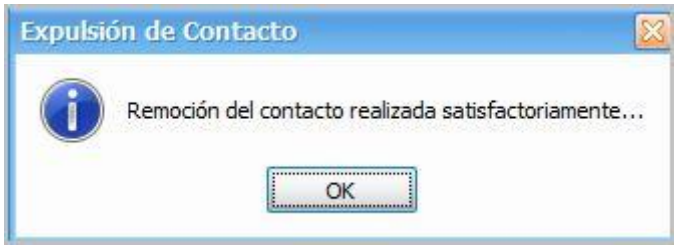
2.2.8. Eliminar

Esta opción permite tanto eliminar a un contacto de un grupo como a un grupo que ya no tenga contactos.

- *Para eliminar un contacto del grupo,* Se selecciona el contacto que se quiere eliminar y se da click en el botón  , al hacerlo apacere una ventana en la que pregunta si el usuario esta seguro de expulsar al contacto seleccionado:



Al aceptarlo aparece la siguiente ventana:

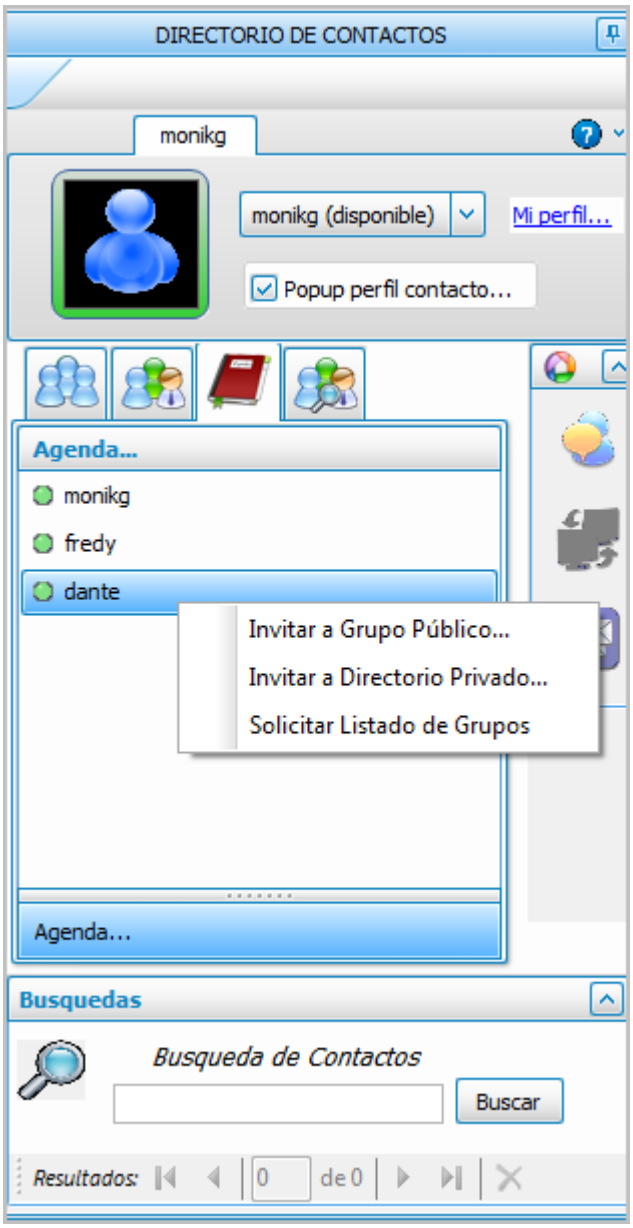


En este momento al usuario que se ha expulsado le aparece el siguiente mensaje:

Avisándole que el anfitrión del grupo a que pertenecía lo ha eliminado.

2.3. Agenda

En esta parte del SCIP2P se muestran todos los usuarios que no pertenecen a alguna etiqueta o grupo y estan conectados, desde aqui se pueden hacer una serie de acciones con los contactos, esto dando click derecho sobre uno de ellos:



Ver Secciones "**Agregar Contacto a Grupo Público", "Agregar un Contacto" y "Solicitar Listado de Grupos".**

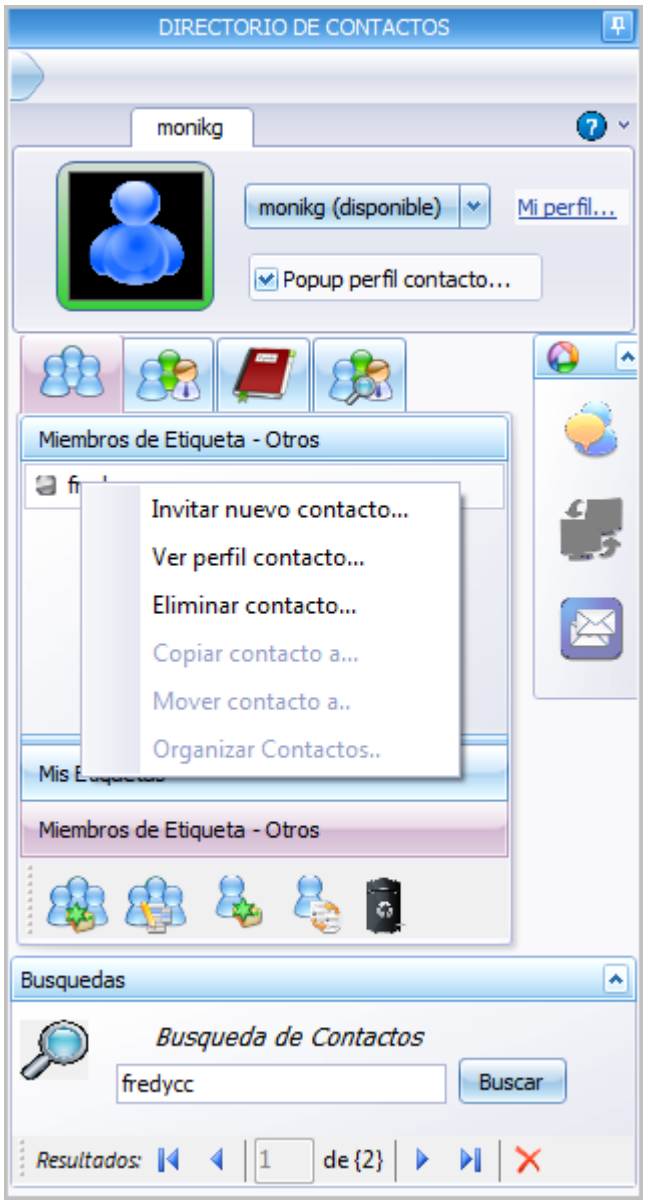
2.4. Buscar Grupos

Esta parte permite al usuario escribir el nombre del grupo que esta buscando y en el area de listado de grupos, mostrara si se encuntra alguna concordancia.

Al encontrar alguna concordancia con respecto al grupo que se esta buscando, se puede Ver la descripcion del grupo (Ver Seccion **"Detalle de Grupo"**) y Solicitar admision a Grupo (Ver Seccion **"Agregar Contacto a Grupo Público"**).

2.5. Buscar Contactos

Esta parte permite al usuario buscar un contacto especifico, si encuentra una concordancia se puede, ver el perfil del contacto, invitarlo a grupo privado o público y solicitarle la lista de grupos.

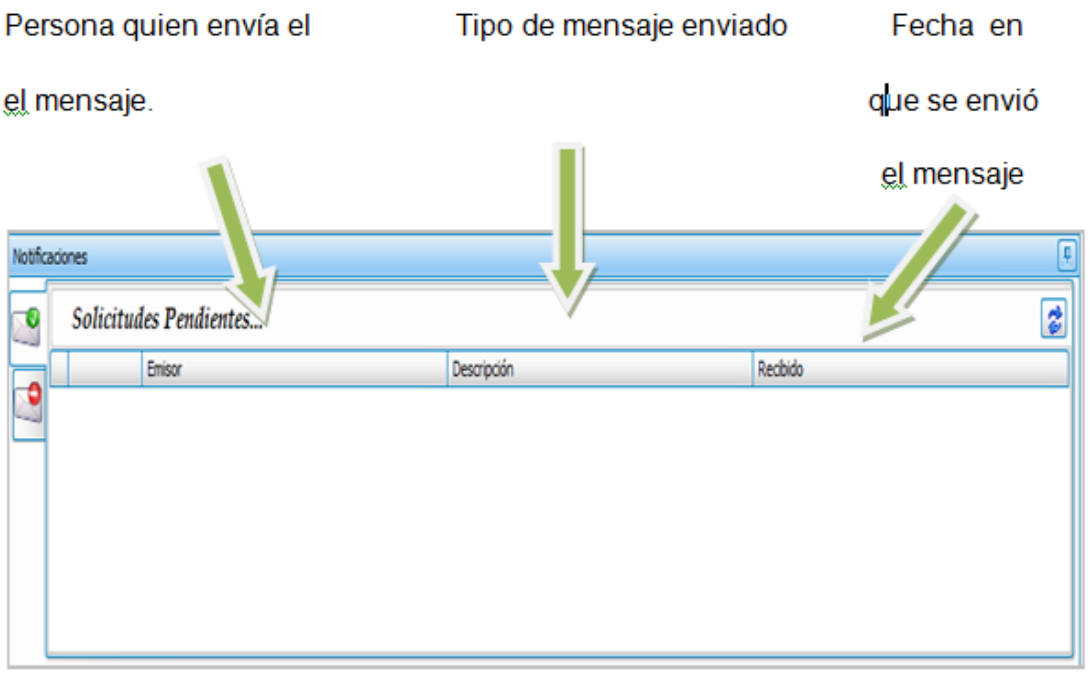


3. Notificaciones

Esta sección, sirve al usuario para guardar todos los mensajes remotos que el contacto a quien ha sido enviada no contesta en un tiempo de cinco segundos.

Estas pueden ser contestadas cuando el usuario así lo desea, ya que se tiene a disposición hasta luego de desconectarse y volver a iniciar sesión.

El área de trabajo donde se almacena, como se ha visto en todo el manual es la siguiente:

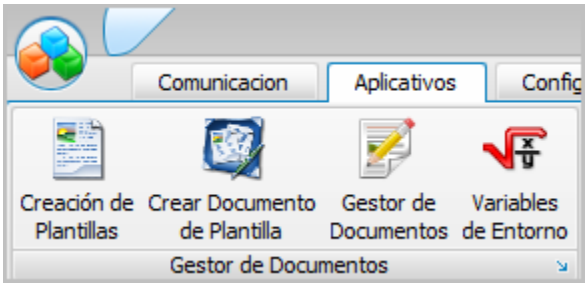


4. Aplicativos

4.1. Gestor de Documentos

Este aplicativo tiene ayuda a que los usuarios creen y por lo tanto tengan a su disposición formas, documentos o plantillas.

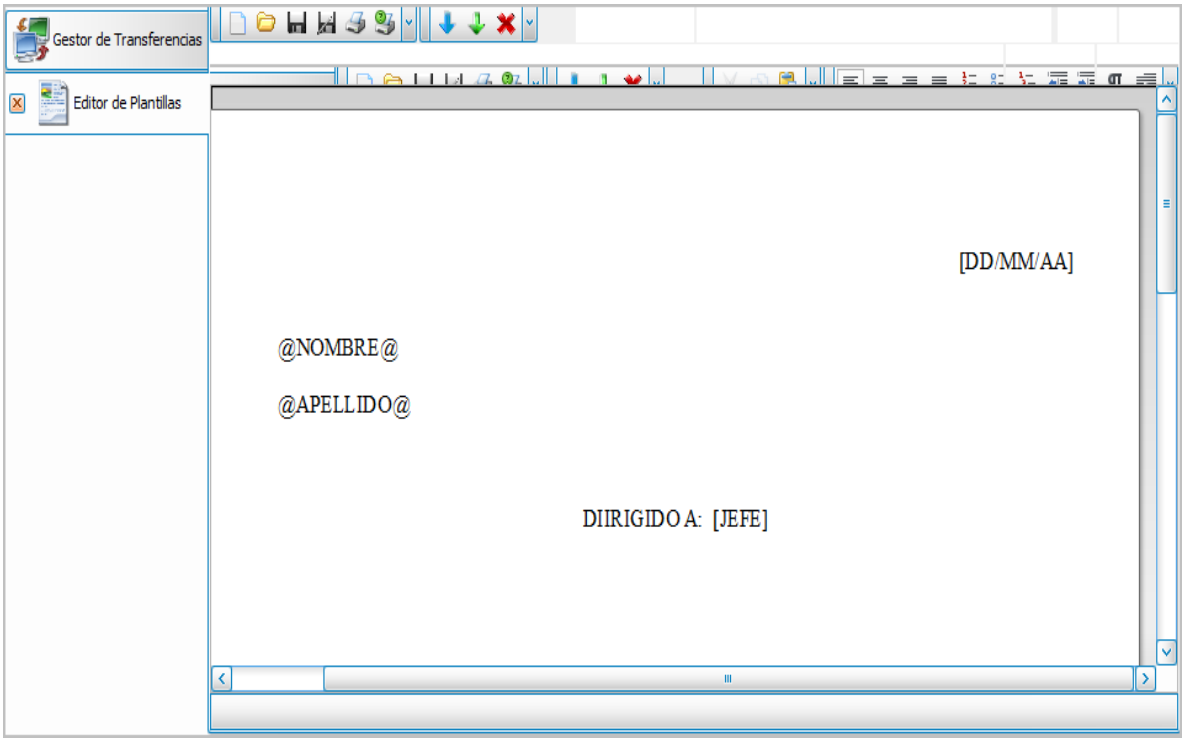
Para hacer uso de este, se selecciona la pestaña de aplicativos de la parte superior del área de trabajo y aparecerá un menú asi:



Desde el cual se tienen las funciones:

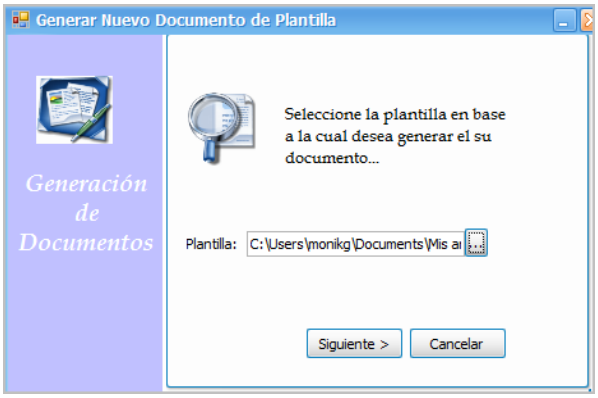
- Creación de Plantillas

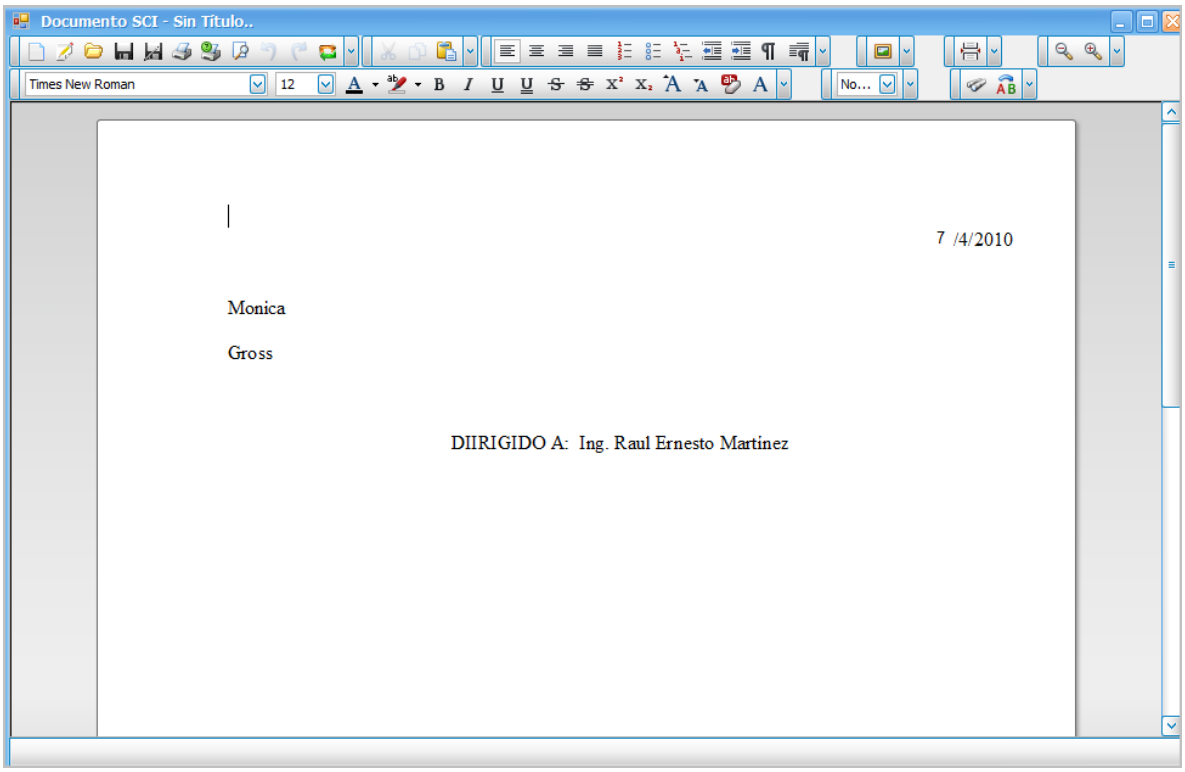
Esta parte permite al usuario crear las platillas, y darle el formato que él desee y almacenarla.



- **Crear Documento de Plantilla**

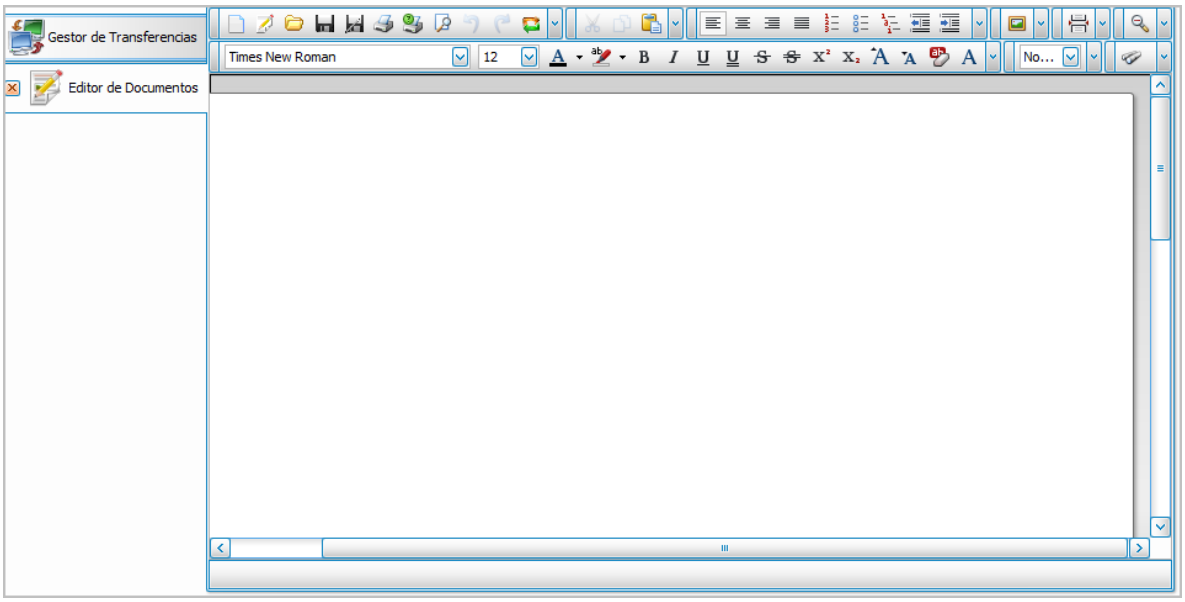
Esta función permite elegir una plantilla que se guardo anteriormente y generarla como documento. Si al momento de crear la plantilla el usuario le ingreso campos el sistema los pregunta previamente a mostrar el documento, si el usuario ingreso variables el sistema los agarra de este, para luego mostrar el documento ya ordenado.





- Gestor de Documentos

Esta parte sirve al usuario para crear documentos que el SCIP2P puede leer propiamente, ya que al guardo este se crea con la extensión .pln .

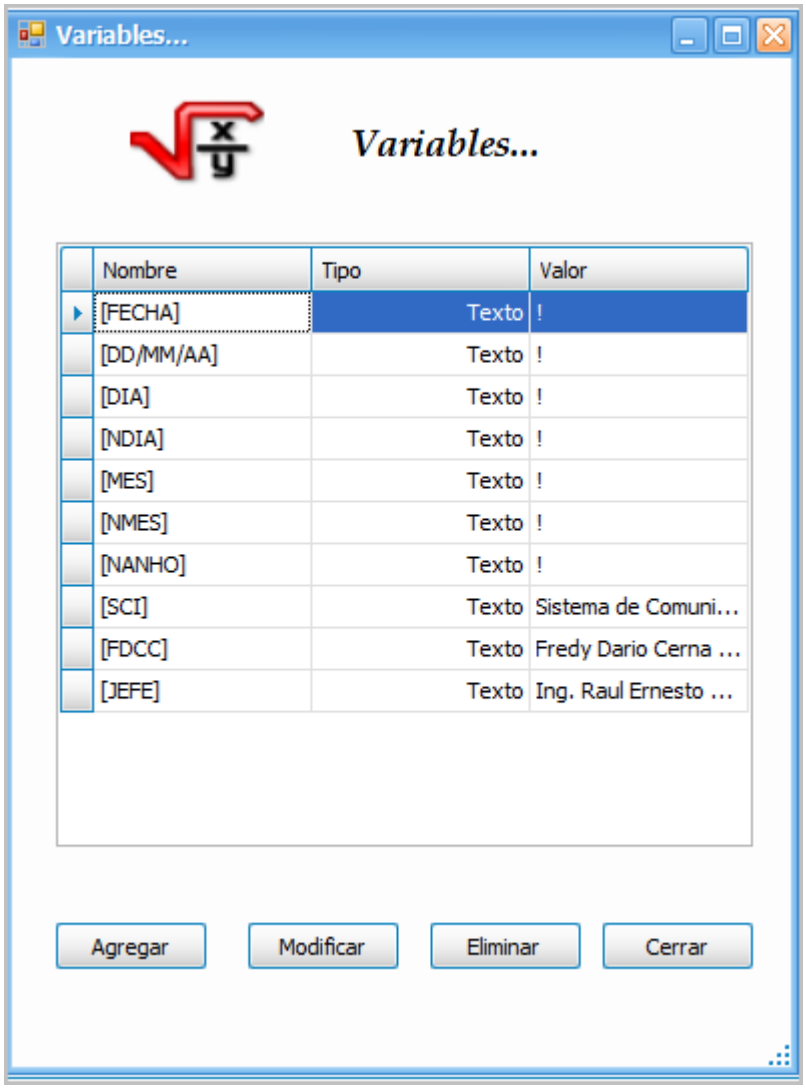


- Variable de Entorno

El sistema permite agregar las variables que el usuario quiera guardar, para que de esta forma puede tenerlo a disposición en plantillas futuras.

Para esto se da click en el botón 

y parece la siguiente ventana:



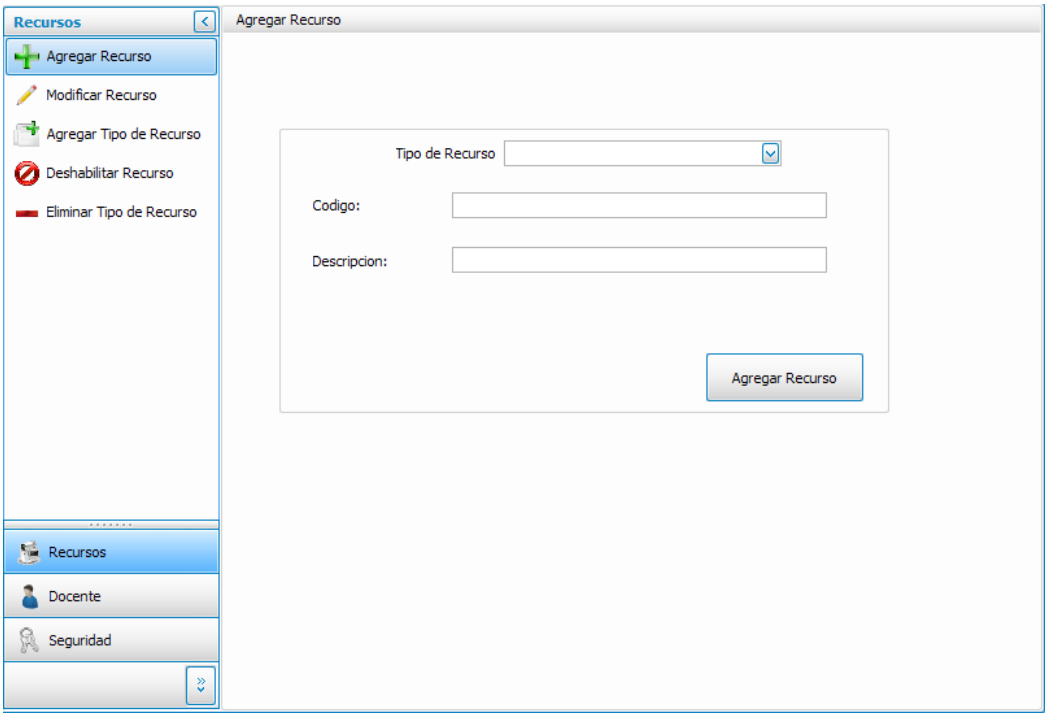
Donde se pueden agregar, modificar y eliminar variables.

4.2. Sistema de Recursos Audio Visuales

Este sistema ayuda a la jefatura y secretaría del departamento en la administración y préstamo de dicho equipo.

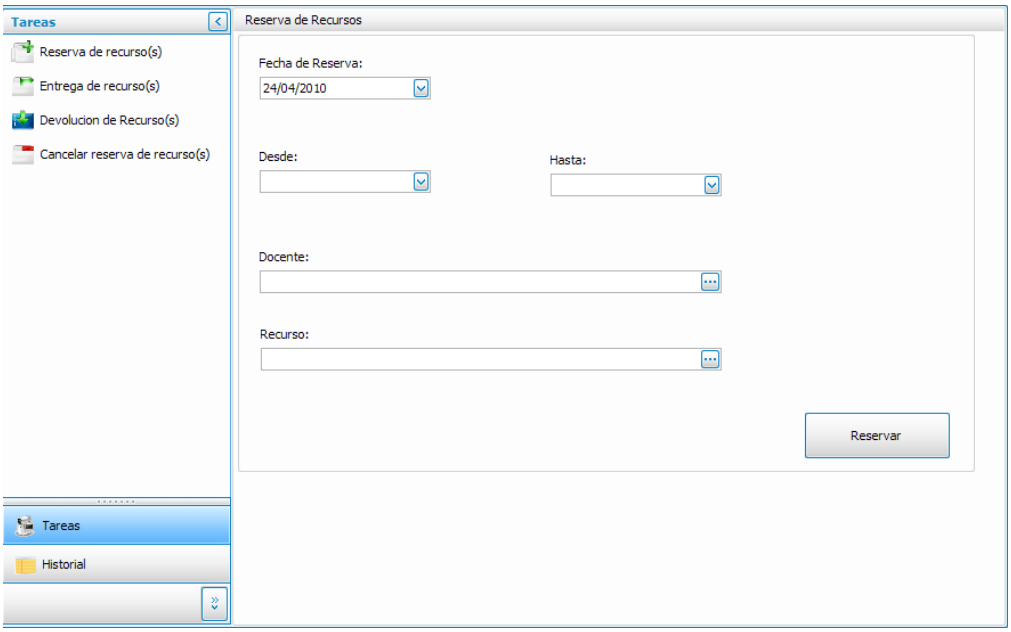
Para hacer uso de esta se da click en el botón 

que se encuentra ubicado en el menú de la pestaña “Aplicativos”. Se muestra la pantalla:



En la que se permite agregar recursos, modificar, deshabilitar en caso el recurso se arruine o eliminarlo.

Al seleccionar la parte de Tareas, el sistema permite realizar reservas eligiendo el recurso que se desea. Se ingresa la fecha y la hora. El sistema informara al usuario si hay en disponibilidad o no. Si hay disponibles el sistema lo reserva.
Una vez reservado, el sistema estará listo para dar de alta la orden de entrega y finalmente la devolución.

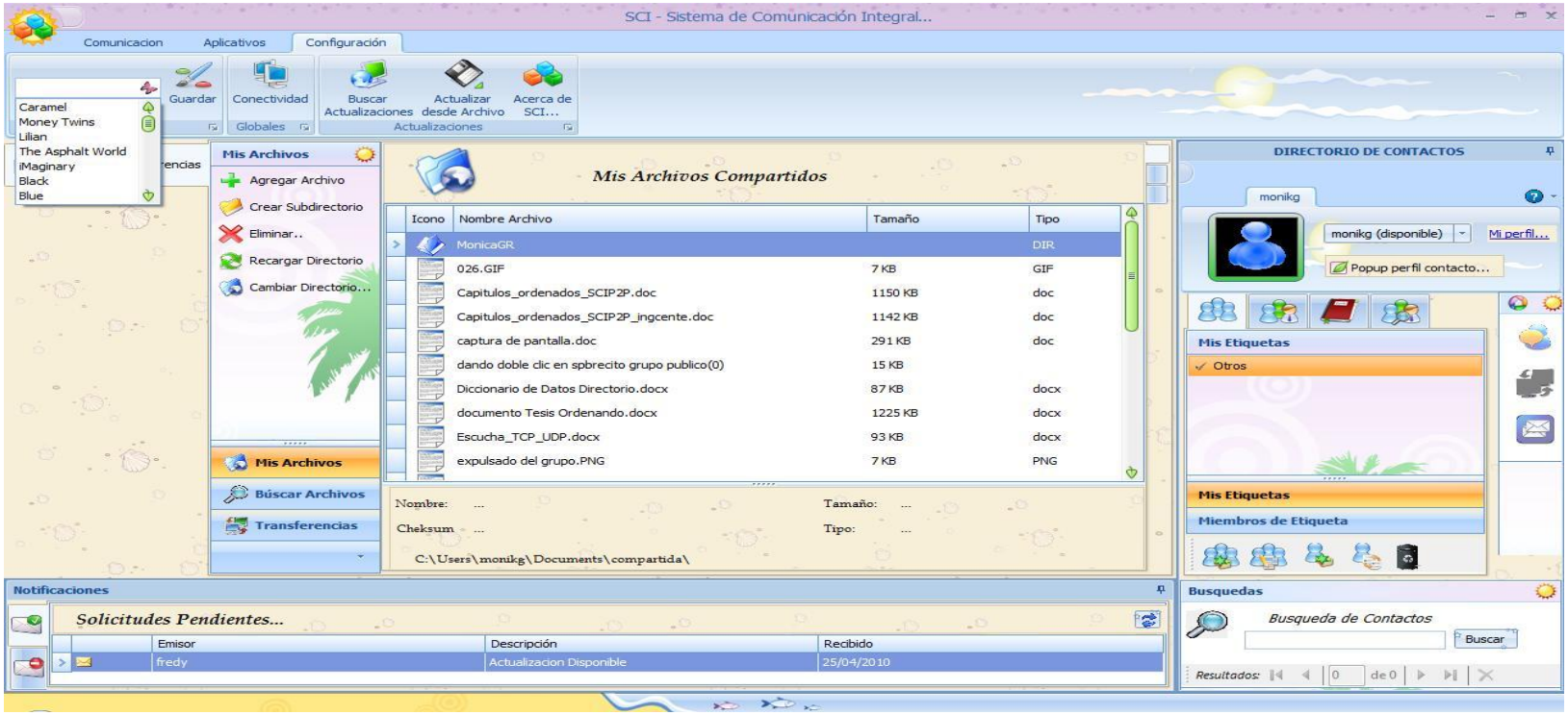

Tareas
Reserva de recurso(s)
Entrega de recurso(s)
Devolucion de Recurso(s)
Cancelar reserva de recurso(s)
Tareas
Historial
Reserva de Recursos
Fecha de Reserva:
24/04/2010
Desde:
Hasta:
Docente:
Recurso:
Reservar

- **Configuración del Sistema de Comunicación Integral Punto a Punto (SCIP2P).**

5. Configuración

Una vez Instalado el SCIP2P, e iniciada la Sesión (Ver Sección **"Iniciar SCIP2P"**) este tiene una serie de opciones que pueden ser configuradas por el usuario. Esto desde la Pantalla Principal en la pestaña de configuración.

- Skin

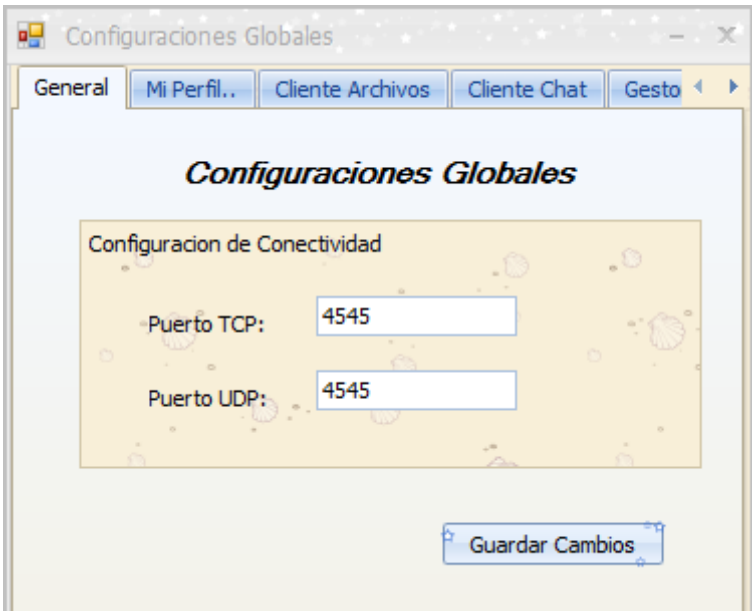


El usuario podrá dar el estilo que desee, desde la parte de skin. En el ejemplo se muestra el skin summer 2008.

- Globales

Esta parte permite hacer las configuraciones de todas las funciones que el SCIP2P da.

- ✓ General

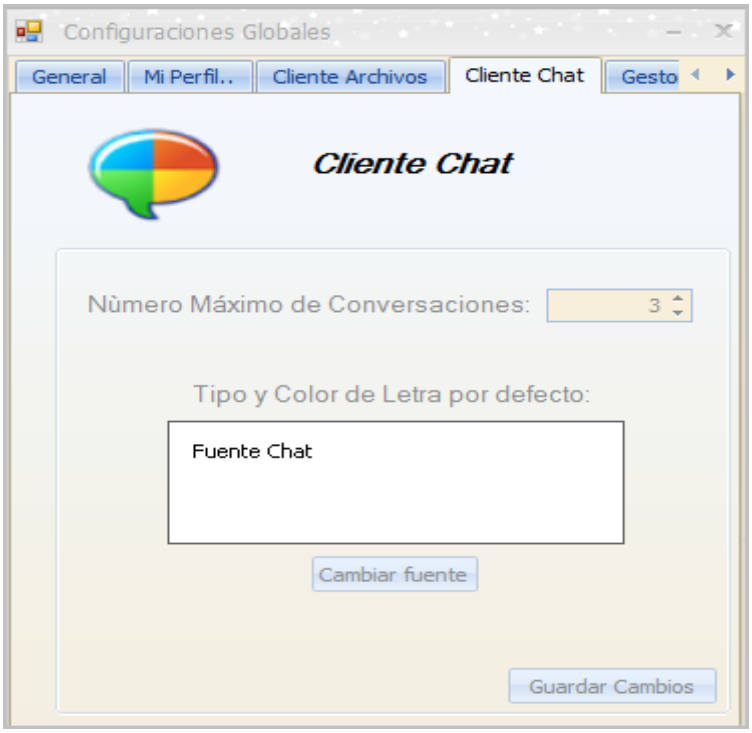


En esta parte se configura el puerto por el cual el aplicativo se estará comunicando en la red.

- ✓ Cliente Archivos

Permite configurar el numero max de descargas y de subidas , asi como también el tamaño de los segmentos que se transmitaran por la red, siendo estos 512 o 1024 kb.

- ✓ Cliente Chat

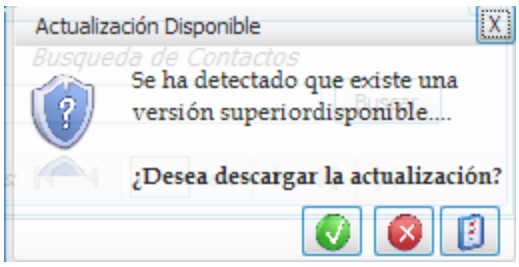


Esta parte permite al usuario configurar el número de chat máximo que puede tener el usuario, tipo y color de letra.

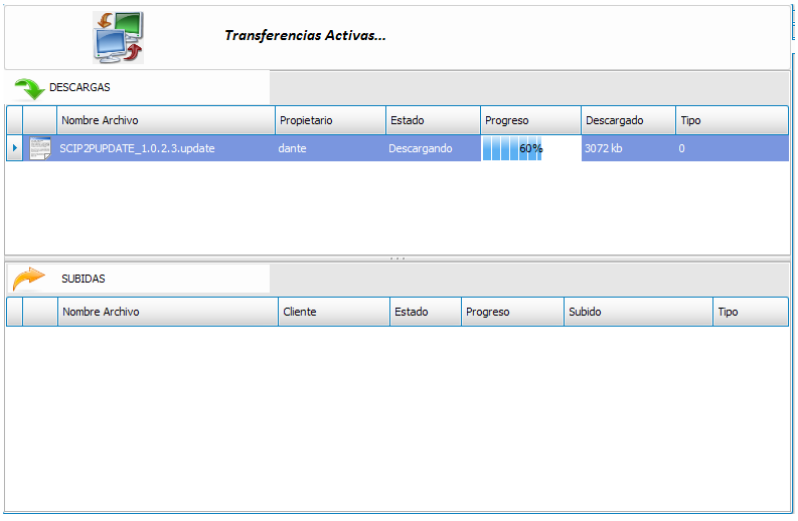
- Actualizaciones

Esta parte del SCIP2P sirve al usuario para que mantenga actualizado el sistema de todos los cambios de mejora que se le hagan. Para esto se puede hacer da dos maneras.

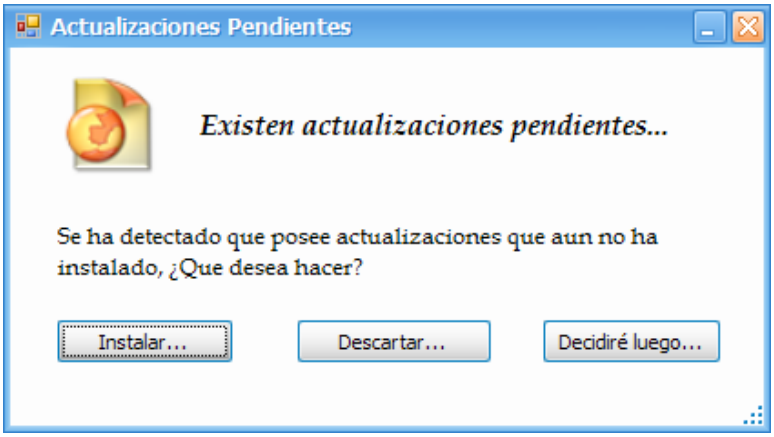
Al iniciar el SCIP2P internamente manda un mensaje de pregunta, si hay una versión nueva del sistema. Si es así, se muestra una notificación si se desea descargar.



Al dar click que si el botón  se comienza a descargar y se puede visualizar en la ventana de transferencias:

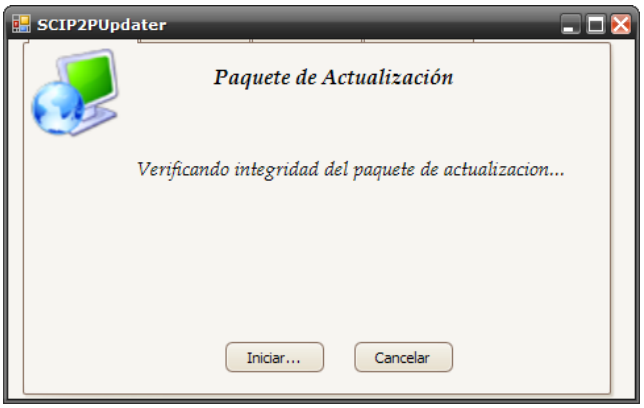


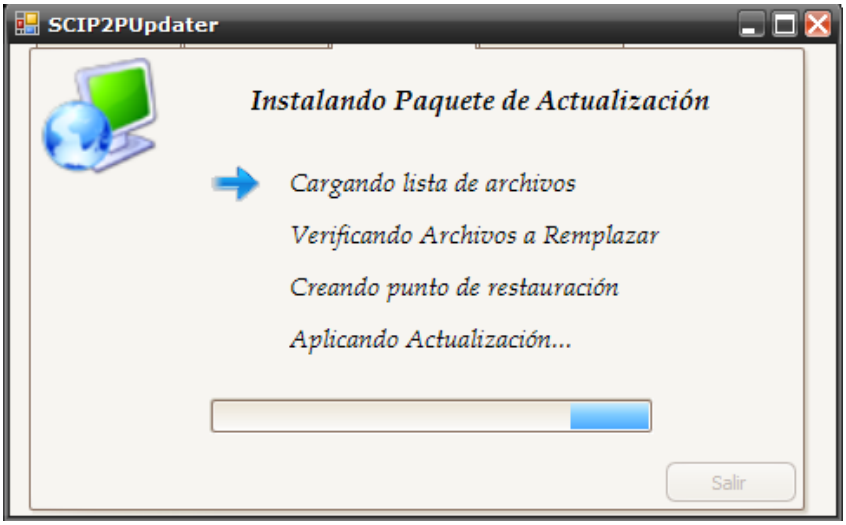
Al terminar de descargase el sistema informa al usuario que tiene una nueva versión para ser instalada:

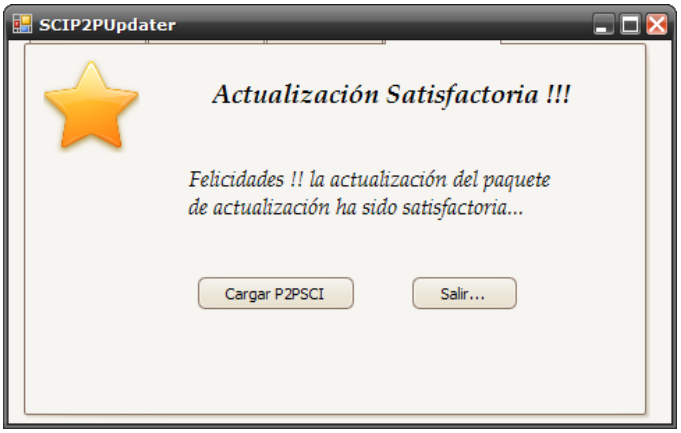


En esta parte se decide:

- ✓ ***Si quiere instalar en ese mismo momento***, es indispensale que se cierre el aplictivo. Luego, aparece la siguiente ventana:







Para instalar la nueva versión del aplicativo solamente es necesario ir dando click en el botón iniciar e instalar respectivamente. Al terminar de instalar el

aplicativo se muestra la ventana en la que informa al usuario que este fue instalado correctamente. Y se le pregunta al usuario si quiere iniciar nuevamente el SCIP2P.

- ✓ ***Si se decide Descartar,*** el sistema obviara el paquete de actualización, cuando se inicie sesión en ocasiones posteriores.
- ✓ ***Si se selecciona Decidiré Luego,*** cuando el usuario se vuelve a conectar el sistema de enviara una notificación que tiene una versión nueva sin ser instalada. Ahí se podrá decidir si se quiere instalar en ese momento (Ver Seccion **"Si se quiere instalar en ese momento"**).