Universidad de las Ciencias Informáticas
Facultad 6

Título: Aplicación para la publicación bajo demanda de grabaciones.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Daniel Angel Barrios Hernández

Tutor: Ing. Rayner Pupo Gómez

[La Habana, Junio 2013, Año 55 de la Revolución]

"Pero la juventud tiene que crear. Una juventud que no crea es una anomalía realmente."

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Daniel Angel Barrios Hernández

_____
Firma del Autor

Ing. Rayner Pupo Gómez

_____
Firma del Tutor

DATOS DE CONTACTO

Datos del Tutor:
Nombre y apellidos: Ing. Rayner Pupo Gómez
Correo electrónico: rpgomez@uci.cu
Categoría docente: Recién Graduado en Adiestramiento.
Año de graduación: 2011.
Profesión: Ingeniero en Ciencias Informáticas.
Breve descripción: Graduado en la Universidad de las Ciencias Informáticas. Actualmente se desempeña como desarrollador del Módulo Visor en el proyecto Video Vigilancia perteneciente al Departamento de Señales Digitales del Centro de Desarrollo “Geoinformática y Señales Digitales” de la facultad 6.

Dedico este trabajo:

A mi abuela Fisca, como todos le deciamos cariñosamente y a mi papá, que aunque ya no están presentes en este mundo, siempre dieron todo de sí para que cada día yo sea una mejor persona.

En especial a mi mamá por convertirse en un padre y una madre a mis 14 años y por apoyarme siempre y guiarme por el mejor camino para mi vida y por ser lo más grande que existe en este mundo para mi.

También quiero dedicar este trabajo a mi novia por aguantarme en estos más de 8 años y apoyarme en todo este tiempo.

Agradecimientos:

Este trabajo de diploma marca el final de una larga etapa de estudios. Quisiera agradecer a todos aquellos que han tenido que ver en mi formación como profesional y como persona en todos estos años.

A la Revolución y a Fidel, por darme la oportunidad de convertirme en un profesional.

A la UCI y en especial a la FEV, porque estando en ella he crecido y aprendido a ver la vida de una forma diferente.

A todos mis profesores, por brindarme todo sus conocimientos y ayuda a cambio de nada.

A mi tutor, por ayudarme, dándome los cocotazos que me merecía para poder salir adelante.

A mis compañeros y amigos que de una forma u otra han contribuido en el desarrollo de este trabajo, ellos son Guillermo, Leosmel, Dieter, Odiel, Vilmavis entre otros más.

A mis amigos, por aguantar mis pesadeces y darme su apoyo en momentos difíciles, como Ariel, Luis Manuel, Fliober, Jorge Osiel, Eddy y otros más que no menciono, no porque no sean importantes, si no porque son muchos.

A todos los amigos de la FEV con los cuales tuve la oportunidad de compartir parte de mi vida en estos 5 años y con los cuales he aprendido mucho de lo que hoy se.

A mis compañeros de los diferentes Apartamentos (149-102, 146-101, 96-101 y 112-203) y Brigadas (9102, 9207, 6305, 6405 y 6505) por los que he pasado y a los profesores y compañeros de mi proyecto (Video Vigilancia), por poder contar con ellos.

A mis suegros y mi cuñadita por formar parte de mi familia y brindarme todo su apoyo.

A mi familia por darme su apoyo desinteresadamente, en especial a mi tías Cuca e Irdenia y a mis tíos Misael y Saul.

A mi Novia, por darme todo su amor y quererme sin ningún interés y por escucharme y brindarme su apoyo y confianza en los momentos en que creí que no podía seguir adelante.

A mis hermanos, Dania y Dariel por ser parte de mi vida.

Y sobre todas las cosas, le agradezco a mi mamá por hacerlo todo por mí desinteresadamente, por enseñarme a ser una buena persona, por impulsarme siempre a superarme, por los consejos y la confianza que ha depositado en mí, por todo su amor y porque, a pesar de las dificultades, ha sabido ser la mejor mamá del mundo.

RESUMEN

La evolución tecnológica ha permitido el surgimiento de métodos novedosos para garantizar la seguridad física de cualquier entorno, como es el caso de la video vigilancia. En la Facultad 6 de la Universidad de las Ciencias Informáticas (UCI) se lleva a cabo la construcción de un sistema de video vigilancia, para el control y monitorización utilizando cámaras IP.

Actualmente el sistema carece de una funcionalidad que le permita a los operadores del sistema la compartición de los videos almacenados en el servidor de grabaciones, siendo el objetivo de la presente investigación, la planificación y desarrollo de las actividades necesarias para la implementación y puesta en marcha de una herramienta capaz de realizar la compartición de las medias demandadas por los módulos del sistema de Video Vigilancia SURIA.

Se destaca que la herramienta desarrollada se encargará de facilitar la realización de los procesos vinculados con la compartición de los recursos almacenados en el servidor de grabaciones, los cuales se pueden realizar de forma manual interactuando con la herramienta y de forma remota a través de peticiones por la red. Lo que proporciona un acceso remoto a las medias grabadas por el sistema de Video Vigilancia SURIA. Esta herramienta, además de ser una parte fundamental para este tipo de sistemas, pretende facilitar el acceso remoto de una forma más fácil y segura a las medias.

PALABRAS CLAVE

Acceso remoto, Compartición, Medias, Video Vigilancia.

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE GENERAL

INTRODUCCIÓN

La seguridad siempre ha sido un factor de suma importancia para el hombre, con los adelantos tecnológicos, el incremento del volumen de los datos y la llegada de las Tecnologías de la Informática y las Comunicaciones (TIC) aparecieron los sistemas dedicados a la vigilancia de viviendas, negocios y edificios posibilitando de esta manera conocer o detallar dónde se encuentran objetos, personas, la identificación de los mismo, así como la labor que realizan en tiempo real.

El desarrollo de la informática ha permitido enriquecer las funcionalidades y los campos de aplicación para los sistemas de video vigilancia, por lo que cada día se hace más frecuente en todo el mundo el uso de estas aplicaciones. Estos sistemas fueron concebidos para garantizar un control más efectivo y la mejora de los métodos de protección planificados por cada entidad y para asistir al personal encargado de la vigilancia en situaciones específicas como la ubicación rápida de alguna actividad delictiva o de algún factor de interés en el ámbito de una empresa determinada. De esta forma, se convierten en una herramienta con un amplio campo de aplicación en cualquier actividad que se especialice en la protección de bienes o recursos materiales.

En Cuba, una institución destacada en el desarrollo de *software* es la Universidad de las Ciencias Informáticas (UCI), la misma cuenta con el Centro de desarrollo Geoinformática y Señales Digitales (GEYSED). Uno de los proyectos que se encuentra en dicho centro es Video Vigilancia el mismo está designado a desarrollar sistemas de video vigilancia que sean competitivos en el mercado actual, fácilmente personalizables de acuerdo con las exigencias de los clientes, que se puedan integrar con otras aplicaciones y con soporte para adicionar nuevas funcionalidades en dependencia de las oportunidades de negocio.

Actualmente ya se liberó la primera versión del producto Video Vigilancia *SURIA*, el cual es un sistema de video vigilancia de tercera generación capaz de interactuar con cámaras IP de diversos fabricantes. El sistema está compuesto por diferentes módulos que interactúan coordinadamente para lograr la monitorización exitosa de cualquier entorno. Este sistema ha sido desarrollado utilizando la tecnología .NET por medio del lenguaje de programación C#, pero actualmente se está migrando la aplicación a *software* libre, ya que ante la eminente realidad de ser un país bloqueado económicamente se impone la

necesidad de no usar software propietarios y enmarcarse en las políticas del país y de la universidad en específico de utilizar *software* libre.

Los módulos con los que cuenta el sistema son, el **Gestor:** es el orquestador del sistema ya que gestiona la integración entre los demás módulos, el de **Administración:** es el encargado de gestionar los usuarios, roles y permisos en el sistema así como la autenticación en los demás módulos, el **Visor:** permite visualizar los flujos de videos capturados por las cámaras, el **Cliente de Grabación:** encargado de gestionar el almacenamiento de los flujos de video obtenidos de las cámaras y el **Cliente de Análisis:** es el encargado de gestionar el procesamiento de los flujos de videos capturados por las cámaras o almacenados en el sistema a petición de un cliente o por configuración, los cuales tienen el objetivo de facilitar la realización de múltiples funciones enfocadas a mejorar la vigilancia en las instituciones.

La grabación y almacenamiento de los flujos de video es una de las principales funciones de los sistemas de video vigilancia. Esta permite tener constancia de los hechos que ocurrieron en una fecha y hora determinada, lo cual constituye una prueba documental ante la investigación de cualquier delito que se cometa. Por tal motivo se debe de tener una mayor seguridad de las medias (videos grabados por el sistema) que se encuentran guardadas en el servidor de grabaciones, aunque es necesario resaltar que el sistema presenta problemas de seguridad y de accesibilidad a la hora de revisar una grabación determinada.

Actualmente en el sistema de Video Vigilancia *SURIA* para poder revisar una grabación de las que se encuentran en el servidor de grabaciones, los operadores del sistema tienen que compartir el recurso donde se encuentran las medias manualmente o acceder directamente al servidor de grabaciones. Esto dificulta el trabajo para los operadores del sistema ya que deben de realizar varios pasos para poder revisar una grabación determinada. También se evidencia un problema de seguridad, ya que pueden existir determinadas medias que no se deban mostrar por las características que poseen.

A partir del análisis de la situación anteriormente descrita se identificó el siguiente **problema a resolver:** ¿Cómo facilitar el acceso a las medias demandadas por los módulos del sistema de Video Vigilancia de una forma segura?

Para dar solución al problema se define como **objetivo general**: Desarrollar una herramienta para la publicación bajo demanda de grabaciones de seguridad.

Partiendo del problema a resolver de la investigación se propone como **objeto de estudio**: Los procesos relacionados con la compartición bajo demanda de grabaciones de forma segura. Delimitando como **campo de acción:** Los procesos relacionados con la compartición bajo demanda de grabaciones de forma segura en el sistema Video Vigilancia *SURIA*. Se propone como **idea a defender** que si se desarrolla de una herramienta para la publicación personalizada o especializada de los videos almacenados por el sistema de Video Vigilancia *SURIA* se logrará el acceso a los mismos de un modo seguro.

Las **tareas de la investigación** que se delimitaron para lograr la culminación exitosa del proceso de construcción de la herramienta para la publicación bajo demanda de grabaciones fueron:

- Caracterizar los procesos relacionados con la compartición bajo demanda de grabaciones.
- Caracterizar las herramientas existentes para la compartición bajo demanda de grabaciones.
- Caracterizar las herramientas, tecnologías y metodología para la implementación de la herramienta para publicación bajo demanda de grabaciones.
- Realizar el diseño de clases de la herramienta para publicación bajo demanda de grabaciones.
- Implementar la herramienta para publicación bajo demanda de grabaciones.
- Diseñar e implementar las pruebas para la herramienta para la publicación bajo demanda de grabaciones.

Para dar cumplimiento al grupo de tareas planteadas servirán de ayuda los métodos teóricos siguientes:

Métodos teóricos:

1. Analítico-Sintético: este método permite estudiar y llegar a una conclusión o un tipo de resumen de lo estudiado, se utilizara para estudiar los servidores de archivos que existen actualmente.
2. Modelación: mediante este método se realizarán los modelos que especifican como se va a desarrollar la herramienta y que permitirán un mejor entendimiento a los desarrolladores para una posterior implementación.

3. Inductivo-deductivo: se utiliza para llegar a un grupo de conclusiones particulares sobre lo que se quiere lograr gracias a la obtención de un conocimiento general acerca de las principales técnicas para la compartición de archivos.
4. Hipotético deductivo: para el análisis y la definición de la hipótesis de la investigación que será verificada o probada en función del estudio de elementos particulares o de menor nivel de generalidad.

Este trabajo de diploma está estructurado en 4 capítulos, los cuales se describen brevemente a continuación:

Capítulo 1: Fundamentación teórica de los procesos para la compartición archivos y tecnologías, herramientas y metodología para el desarrollo de la aplicación: En este capítulo se exponen los conceptos asociados al problema planteado y se describe el objeto de estudio. Se presenta un análisis según las necesidades de la investigación de las características de las soluciones informáticas existentes que abordan el tema de compartición de archivos. Se especifican además las herramientas, tecnologías y metodología de desarrollo de *software* que se emplearán en el desarrollo de la herramienta para la publicación bajo demanda de grabaciones.

Capítulo 2: Características de la herramienta para la publicación bajo demanda de grabaciones: En este capítulo se abordan los temas relacionados con el modelo de dominio, el levantamiento de los requisitos tanto funcionales como no funcionales que debe cumplir la herramienta y el diagrama de casos de uso así como la descripción de los mismos.

Capítulo 3: Diseño de la herramienta para la publicación bajo demanda de grabaciones: En este capítulo se realiza la descripción del estilo arquitectónico usado en el desarrollo de la herramienta y los patrones de diseño. También se realizará el diagrama de clases del diseño y los diagramas de secuencias para cada caso de uso.

Capítulo 4: Implementación y validación de la solución propuesta: En este capítulo se exponen el diagrama de despliegue y el de componentes. Además se presentan las pruebas realizadas a la aplicación para validar la solución propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LOS PROCESOS PARA LA COMPARTICIÓN DE ARCHIVOS Y TECNOLOGÍAS, HERRAMIENTAS Y METODOLOGÍA PARA EL DESARROLLO DE LA APLICACIÓN

Introducción

En el presente capítulo se abordan los principales conceptos asociados a los procesos relacionados con la compartición de archivos, y se realiza una descripción del objeto de estudio de la investigación. Se presenta un análisis según las necesidades de la investigación de las características de las soluciones informáticas existentes que abordan el tema de la compartición de archivos. Por último se especifica la metodología de desarrollo de *software*, las herramientas y tecnologías a utilizar para el desarrollo de la herramienta informática.

1.1. Conceptos asociados al dominio del problema

Para lograr un mayor entendimiento del problema planteado a continuación se reflejan los conceptos asociados al dominio del problema:

Protocolo: Es un conjunto de reglas predefinidas que rigen la forma en que dos o más procesos se comunican e interactúan para intercambiar datos. Los procesos pueden estar en la misma máquina o en máquinas diferentes. Por ejemplo, un programa de la capa de transporte en una máquina utiliza un protocolo para comunicarse con su contraparte en otra máquina. Los protocolos están generalmente asociados a determinados servicios o tareas, tales como el empaquetamiento de datos o el enrutamiento de paquetes. Un protocolo especifica las reglas para la creación, realización, y terminación de una conexión de comunicaciones, y también especifica el formato que los paquetes de información deben tener cuando se viaja a través de esta conexión (Rodriguez Alvarez, 2012).

IP (Protocolo de Internet): Es el protocolo de capa de red con más amplio apoyo para la Internet. Es uno de los protocolos en la suite TCP/IP. Este protocolo define y envía datagramas a través de Internet y ofrece un servicio de transporte no orientado a conexión. El protocolo IP utiliza la conmutación de paquetes y hace el mejor esfuerzo para entregar sus paquetes (Ariganello, 2005).

Acceso remoto: Es poder acceder desde una computadora a un recurso ubicado físicamente en otra computadora que se encuentra geográficamente en otro lugar, a través de una red local o externa como Internet (Ramirez, 2010).

Servidor de archivos: Tipo de servidor en una red de ordenadores cuya función es permitir el acceso remoto a archivos almacenados en él o directamente accesibles por este (Ospina, 2011).
Compartir archivos: Consiste en poner a disponibilidad el contenido de uno o más directorios a través de la red (Kioskea ES, 2013).
Bajo demanda: En el presente trabajo, es compartir un recurso en tiempo real dada una determinada petición.

1.2. Descripción del Objeto de Estudio

Los procesos asociados con la compartición bajo demanda de grabaciones de forma segura están relacionados con los servidores de archivos, para un mejor entendimiento se realizó un estudio sobre estos servidores llegando a la conclusión de que en principio un servidor de archivos es un servidor que se encargará de servir archivos a clientes que lo soliciten, cualquier ordenador conectado a una red con un software apropiado, puede funcionar como servidor. Desde el punto de vista del cliente de un servidor de archivos, la localización de los recursos compartidos es transparente; por lo tanto, normalmente no hay diferencias perceptibles si un recurso está almacenado en un servidor remoto o en el disco de la propia máquina. Su uso es de gran importancia ya que facilita el trabajo para los usuarios a la hora de revisar una información, solo tiene que acceder a él a través de una red remotamente.

Una de las mayores ventajas de tener un servidor de archivos, es que toda la información importante queda centralizada en un solo lugar, lo cual facilita la administración y la seguridad de la información; de esta manera no quedan recursos importantes aislados en otros lugares y se tiene la posibilidad de acceder a los mismos remotamente desde cualquier computadora conectada a una red. Esto brinda una mayor seguridad para los archivos que se encuentran en él, ya que no van a estar en distintas computadoras. Los mismos se van a encontrar en un único servidor para que puedan acceder los usuarios o el personal autorizado.

En la actualidad existen varios servidores de archivos que permiten publicar y acceder a archivos en la red, para esto son utilizadas las implementaciones de los protocolos. Los protocolos pueden ser implementados por *hardware*, *software*, o una combinación de ambos. A su más bajo nivel, este define el comportamiento de una conexión de *hardware* (Rodriguez Alvarez, 2012).

Los servidores de archivos que permiten acceder o compartir archivos en una red determinada brindan un conjunto de ventajas. Una de estas ventajas es que la información va a estar mejor distribuida, la misma se va a encontrar en un disco de almacenamiento para ser accedida por todos los usuarios que la necesiten, otra de las ventajas es que se mejora la obtención de la información. También se reduce la duplicidad de trabajos ya que los mismos se van a encontrar accesibles por los usuarios en un único servidor, por lo que se mejora la productividad de trabajos y la seguridad de los mismos (Caceres, 2008).

1.3. Soluciones informáticas existentes asociadas a la compartición de archivos

1.3.1. System-Config-Samba

Esta solución es una herramienta con interfaz gráfica de usuario que pueden usar los administradores de Samba. System-config-samba se puede instalar en varias de las distribuciones de Linux y es fácil de usar. Con esta herramienta, se puede realizar la compartición o la eliminación de alguna compartición de los directorios, así como la configuración del servidor Samba. Esta herramienta se encuentra desarrollada con el lenguaje de programación python (Wallen, 2012). En la siguiente ilustración se muestra una imagen de esta herramienta.

Ilustración 1: Herramienta System-config-samba.

La herramienta System-config-samba cumple con los objetivos de compartir recursos en una red determinada y de ser *software* libre. La misma tiene como limitante de que solo se puede realizar la compartición a través de un usuario que interactúa con la herramienta y no a través de peticiones por la red, es decir no se puede realizar bajo demanda.

1.3.2. Gadmin-Samba

Otra de las soluciones existentes es la herramienta Gadmin-Samba la cual es una aplicación con una interfaz gráfica de usuario para administrar recursos compartidos con Samba y el servidor de impresión. La herramienta Gadmin-Samba permite la creación y manipulación de usuarios, incluyendo la creación de nombres de usuario y contraseñas, garantizando con esto la seguridad en el acceso a los recursos. También cuenta con tres niveles de estrategias de gestión de dominios. Esta herramienta está desarrollada en C y soporta varias distribuciones del sistema operativo Linux (Freecode, 2013). En la siguiente ilustración se muestra una interfaz de esta herramienta.

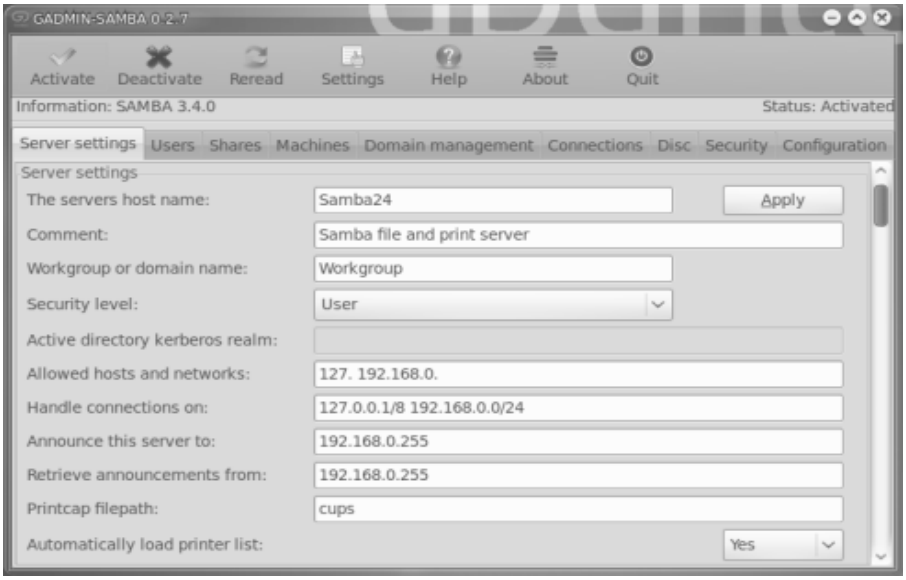


Ilustración 2: Herramienta Gadmin-Samba.

Al igual que en la herramienta descrita en el epígrafe 1.4.1 cumple con los objetivos para la compartición de archivos, pero también tiene como limitante de que necesita de la interacción directa del usuario para su manipulación.

1.4. Tecnologías, herramientas y metodología de desarrollo de *software* a utilizar para el desarrollo de la solución del servicio.

A continuación se exponen las características de la metodología de desarrollo, lenguaje de modelado, herramienta *CASE* (*Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Computadora), lenguaje de programación, marco de trabajo, entorno de desarrollo integrado (IDE) y las

tecnologías para la comunicación y la compartición de recursos, seleccionados para el desarrollo del presente trabajo.

1.4.1. Metodología de desarrollo de software

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo *software* (Cataldi, y otros, 2012).

Como metodología de desarrollo de *software* se selecciona *RUP* (*Rational Unified Process,* Proceso Unificado de *Rational*) por ser una metodología que se adapta a las características propias del *software* que se está desarrollando, a través de la cual se pueden eliminar los riesgos que podrían presentarse durante el desarrollo del *software*. Permite enfocarse en trabajar de forma organizada, donde se controla y documenta todo lo relacionado con el proyecto que se esté realizando. Esta metodología ayuda a construir un *software* de alta calidad, que sea desarrollado en el tiempo planificado, además satisface la necesidad de ser elaborado de una forma rápida.
RUP es un proceso de desarrollo de *software* que junto con el lenguaje *UML* constituye la metodología estándar de mayor utilidad para el análisis, implementación y documentación de sistemas orientados a objetos. También es una plataforma flexible de procesos de desarrollo de *software* que provee ayuda con guías consistentes y personalizadas de procesos para todo el equipo del proyecto. A diferencia de otras metodologías *RUP* es capaz de hacer que el proceso sea práctico con bases de conocimientos y guías para ayudar el despegue de la planificación del proyecto, integrar rápidamente a los miembros del equipo y poner en marcha el proceso personalizado.

Una de las mejores prácticas de *RUP* es la posibilidad de desarrollar iterativamente ya que organiza los proyectos en términos de disciplina y fases, consistiendo cada una de ellas en una o más iteraciones. La aproximación iterativa ayuda a mitigar los riesgos de forma temprana y continua, con un proceso demostrable y constantes *releases* ejecutables. Estas fases son (Cataldi, y otros, 2012):

- **Concepción:** se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos.
- **Elaboración:** se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.

- **Construcción:** se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.
- **Transición:** se implementa el producto en el cliente y se entrenan a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Para cada una de estas fases se definen los siguientes flujos de trabajo:

1. Modelado del negocio.
2. Requisito.
3. Análisis y diseño.
4. Implementación.
5. Prueba.
6. Despliegue.
7. Configuración y manejo de cambios.
8. Administración del proyecto.
9. Gestión del entorno.

A continuación se muestra una ilustración donde se puede observar la descripción de la composición RUP:

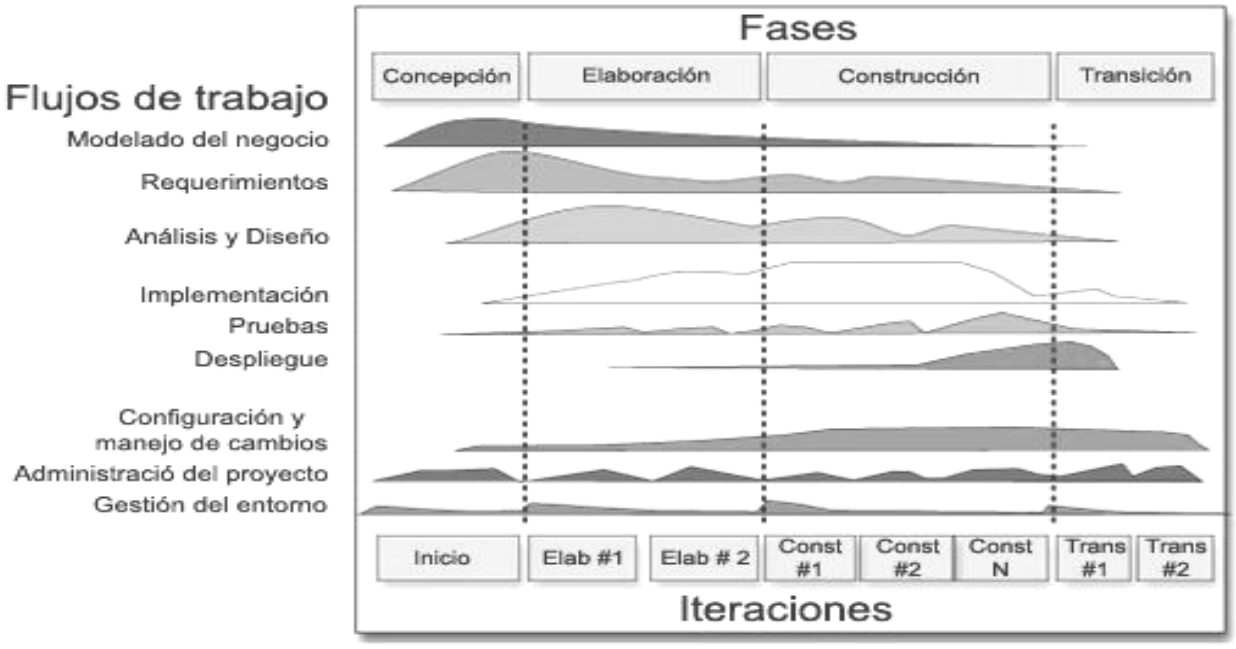


Ilustración 3: Descripción de la composición de RUP.

1.4.2. Herramientas a usar para el desarrollo de la aplicación

Lenguaje de modelado:

Para el modelado de la herramienta se utilizará el Lenguaje Unificado de Modelado (*UML*) en su versión 2.0, ya que es un lenguaje de modelado de propósito general que pueden usar todas las personas de la rama de la informática. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática, también se debe destacar que es el lenguaje propuesto por la metodología RUP la cual fue seleccionada para el desarrollo de la herramienta.

UML es empleado para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Da apoyo a la mayoría de los procesos de desarrollo orientados a objetos (Rumabugh, 1998).

El lenguaje de modelado *UML* capta la información sobre la estructura estática y el comportamiento dinámico de un sistema de *software*. La modelación de un sistema desde diferentes puntos de vista, separados por relaciones, permite entender los sistemas para diferentes propósitos.

Herramienta CASE:

Las herramientas *CASE* (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste del mismo en términos de tiempo y dinero (Lazalde Miranda, 2010).

Las herramientas *CASE* proporcionan al ingeniero la posibilidad de automatizar actividades manuales y de mejorar su visión general de la ingeniería. Al igual que las herramientas de ingeniería y de diseño asistidos por computadora que utilizan los ingenieros de otras disciplinas, las herramientas *CASE* ayudan a garantizar que la calidad se diseñe antes de llegar a construir el producto (Pressman, 2005).

El uso de una herramienta *CASE* brinda una serie de ventajas (Pressman, 2005):

- Facilita la verificación y mantenimiento de la consistencia de la información del proyecto.
- Facilita el establecimiento de estándares en el proceso de desarrollo.
- Facilita el mantenimiento del sistema y las actualizaciones de la documentación.

- Facilita la aplicación de las técnicas de una metodología.
- Disponibilidad de funciones automatizadas tales como: obtención de prototipos, generación de código, generación de pantallas e informes, generación de diseños físicos de bases de datos, verificadores automáticos de consistencia.
- Facilita la aplicación de técnicas de reutilización y reingeniería.
- Facilita la planificación y gestión del proyecto informático.

Como herramienta *CASE* para el modelado de los diagramas se empleará la herramienta *Visual Paradigm* en su versión 8.0, la misma está diseñada para ayudar al desarrollo de *software* y soporta los principales estándares de la industria tales como *UML*.

Visual Paradigm ofrece un conjunto completo de herramientas a los equipos de desarrollo de software, necesario para la captura de requisitos, *software* de planificación, la planificación de controles y la clase de modelado (Viñola Sosa, y otros, 2012).

Soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Proporciona abundantes tutoriales de *UML*, demostraciones interactivas de *UML* y proyectos *UML*. Presenta licencia gratuita y comercial y es compatible entre ediciones (Free Download Manager, 2007).

Características principales (Free Download Manager, 2007):

1. Soporte de *UML*
2. Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
3. Ingeniería inversa - Código a modelo, Código a diagrama.
4. Editor de figuras.

1.4.3. Lenguaje de Programación

Un lenguaje de programación es un conjunto de instrucciones, órdenes, comandos y reglas que permite la creación de programas (Glosario.NET, 2007).

Para llevar a cabo la implementación de la aplicación se utilizará como lenguaje de programación C++, el cual es un lenguaje de programación muy difundido y popular que puede ser usado para resolver diferentes tipos de problemas, desde los más simples hasta los más complejos, y cuyo código puede ser compilado en diversas plataformas.

C++ posee las características siguientes (Pozo, 2009):

1. Difusión: posee un gran número de usuarios y existe una cantidad de libros, cursos y páginas web dedicadas a él, ya que es uno de los lenguajes más empleados en la actualidad.
2. Portabilidad: un mismo código fuente se puede compilar en diversas plataformas.
3. Eficiencia: C++ es uno de los lenguajes más rápidos en cuanto a ejecución.
4. Herramientas: existe una gran cantidad de compiladores, depuradores y bibliotecas como Qt.

1.4.4. Marco de trabajo

Como marco de trabajo para la implementación de la herramienta para la publicación bajo demanda de grabaciones se selecciona Qt 4.8, el cual es multiplataforma, desarrollado bajo código abierto que se utiliza generalmente para desarrollar aplicaciones de software haciendo uso de una interfaz gráfica, es una intuitiva biblioteca de C++ que garantiza la portabilidad entre sistemas operativos embebidos y de escritorio. Este marco de trabajo permite la utilización de hilos independientemente del sistema operativo, contiene un módulo para el trabajo con protocolos de red y posee soporte para aplicaciones orientadas a componentes. Estas características hacen de Qt un marco de trabajo altamente aplicable a aplicaciones de escritorio sin tener que utilizar bibliotecas externas (ZonaQt, 2010).

1.4.5. Entorno de desarrollo integrado (IDE)

Un entorno de desarrollo integrado (IDE) es una herramienta informática que ofrece servicios integrales a los programadores para el desarrollo de *software*. Un IDE normalmente cuenta con un editor de código fuente, un depurador, un compilador y / o intérprete y un constructor de Interfaz gráfica de usuario (LinuxLinks, 2012).

Los IDE se deben seleccionar en dependencia de los lenguajes en que se piensa desarrollar la aplicación, debido a que no todos los IDE soportan los lenguajes existentes. Como se ha seleccionado C++ como lenguaje de programación para la implementación de la herramienta se decidió usar como IDE *Qt Creator*

en su versión 2.4 debido a las múltiples ventajas que ofrece el mismo para la programación con C++, además de las facilidades que brinda para el trabajo con el *framework* Qt.
Dentro de las principales características de *Qt Creator* se encuentran:

- Editor de código sofisticado: Avanzado editor de código. *Qt Creator* ofrece soporte para la edición de C++, ayuda sensible al contexto, completado de código y navegación.

- Control de versiones: *Qt Creator* se integra con la mayoría de los sistemas de control de versiones populares, incluyendo *Git, Subversion, Bazaar, Perforce, CVS y Mercurial*.

- Los diseñadores de interfaz de usuario integrada: *Qt Creator* proporciona dos editores integrados visuales: *Qt Designer* para la creación de interfaces de usuario de *widgets* Qt y *Qt Designer* para el desarrollo de interfaces de usuario rápido de animación con el lenguaje QML.

- Construcción y gestión proyecto: Si se importa un proyecto existente o crear uno desde cero, *Qt Creator* genera todos los archivos necesarios (NOKIA CORPORATION, 2008).

1.4.6. Tecnología para la comunicación

Como tecnología de comunicación se seleccionó XMLRPC en específico libqxmlrpc en su versión 1.0, la misma ofrece la ventaja de estar implementada en el *framework* de Qt, integrado al IDE seleccionado para el desarrollo de la aplicación. Esta tecnología tiene como ventaja que en cuestiones de licencia no existen limitantes al ser una LGPL, una licencia para software libre aplicable a las políticas de la universidad.

XMLRPC es un protocolo de llamada remota a procedimientos que funciona sobre Internet, mediante el intercambio de mensajes entre cliente y servidor, utilizando el protocolo HTTP para el transporte de los mensajes. XMLRPC envía los mensajes en formato XML, señalando el procedimiento que se va a ejecutar en el servidor y los valores correspondientes. Luego de realizada la operación el servidor devuelve el resultado en formato XML. XMLRPC está diseñado para ser sencillo, la curva de aprendizaje de XMLRPC es muy suave, por lo que un programador novato en este campo, puede conseguir resultados satisfactorios con poco esfuerzo (XMLRPC.COM, 2011).

1.4.7. Tecnología para la compartición de archivo

Existen diferentes tecnologías para la compartición de archivos, por lo que en este epígrafe se realiza un estudio de las más importantes para posteriormente poder realizar una selección adecuada en dependencia del problema planteado.

1.4.7.1. Gene6 FTP Server

Dentro de los distintos servidores de archivos que son usados para acceder o compartir información se encuentra Gene6 FTP server, es un servidor para la transferencia de archivos entre sistemas conectados a una red TCP, está basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos. Un problema básico del FTP (*File Transfer Protocol*) es ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el nombre de usuario y contraseña en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor y apropiarse de los archivos transferidos (Kioskea.net, 2012). En la siguiente ilustración se puede ver la comunicación del protocolo FTP:

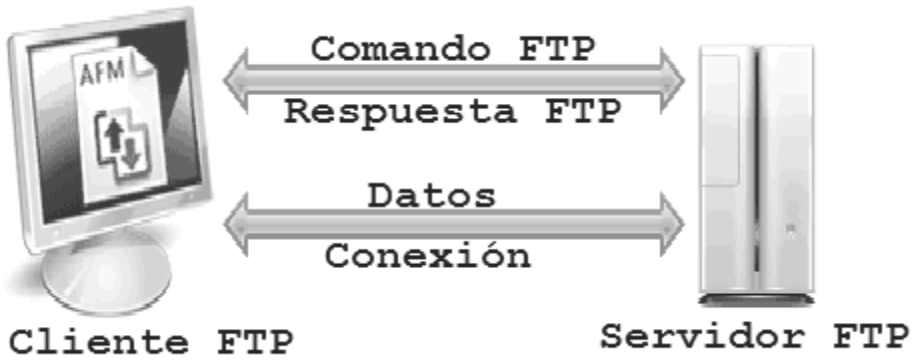


Ilustración 4: Comunicación FTP.

El Gene6 FTP Server es un servidor FTP que se utiliza para *Windows* el cual es caracterizado por su velocidad en conexión y personalización. Sus principales característica son la administración remota de los recursos y la facilidad de uso. Su rápido rendimiento le permite ejecutar servidores de archivos con una fuerte carga de información. (Free Download Manager, 2007).

Otras de las características de Gene6 FTP Server es que se puede destacar que crea directorios virtuales para que los archivos puedan ser compartidos con otros servidores. Una de las ventajas que brinda este servidor es que los archivos pueden ser transferidos sin problema, ya que estos pueden ser comprimidos para una transferencia más rápida (PHPNuke, 2013).

1.4.7.2. HTTP File Server

Otro de los servidores de archivos usados a nivel mundial es el servidor HTTP File Server**,** también conocido como HFS, es un servidor web específicamente diseñado para publicar y compartir archivos con otros usuarios a través de una red. HFS es el *software* que permite enviar y recibir archivos, también permite limitar el intercambio de archivos a usuarios específicos, o estar abierto a todo el mundo.

HFS es un servidor web que utiliza la tecnología web para ser más compatible con la actual Internet, puesto que es en realidad un servidor web, los usuarios pueden descargar los archivos utilizando un navegador web como si estuvieran descargándolos de un sitio web, como Internet Explorer o Firefox. La mayoría de los servidores web son utilizados para publicar sitio web, pero HFS no está diseñado para hacer esto, este te permite compartir tus archivos y descargarlos, vale destacar que HFS ha tenido varios problemas de seguridad que todavía están presentes. Uno de estos problemas es que algunas versiones de HFS permite a un atacante remoto crear archivos arbitrarios en el disco duro donde se encuentra el servidor (Snapfiles, 2011).

1.4.7.3. Samba

Samba es un servidor de archivos que permite compartir o acceder a archivos en la red, es un proyecto de *software* libre que sirve como servidor de impresión y administrador de redes para sistemas de tipo Unix, entre los cuales se encuentra a GNU/Linux, BSD o Mac OS X, que incluye una colección de programas implementando el protocolo SMB (*Server Message Block*), permitiendo servir archivos e impresoras a sistemas Windows, NT, OS/2 y clientes DOS.

Es un conjunto de aplicaciones UNIX que entienden el protocolo SMB. Muchos sistemas operativos, entre ellos Windows y OS/2, utilizan SMB para operaciones de red cliente-servidor. Mediante el soporte de este protocolo, Samba permite a los servidores UNIX comunicarse con el mismo protocolo de red que Windows. De esta manera, una máquina UNIX con Samba puede enmascararse como servidor en una red Microsoft y ofrecer los siguientes servicios.

- Compartir uno o más sistemas de archivos.
- Compartir impresoras, instaladas tanto en el servidor como en los clientes.
- Autenticar clientes contra un dominio Windows.
- Proporcionar un servidor con resolución de nombre WINS.

El creador de Samba fue Andrew Tridgell, en un comienzo Tridgell creó un programa servidor de archivos para LAN (Local Área *Network*) que soportaba el protocolo DEC (*Digital Equipement Corporation*) de la empresa Digital *Pathworks*. Tiempo después este protocolo se convirtió en SMB, y ahí nació realmente el proyecto. En un principio simplemente era conocido como servidor SMB, pero luego no pudo mantener ese nombre y su creador tuvo que cambiarlo. De esta manera, nació el nombre de Samba, el paquete completo se distribuye bajo la GNU GPL (Muñoz Casals, 2008).

Puesto que Samba es una implementación para Unix del protocolo SMB, la mejor forma de entender Samba es comenzar por describir SMB con un poco más de detalle. SMB es un protocolo de los denominados petición-respuesta, indicando que las comunicaciones se inician siempre desde el cliente como una petición SMB de servicio al servidor, que la procesa y retorna una respuesta ha dicho cliente. La respuesta del servidor puede ser positiva con el resultado de procesar la petición del cliente o negativa con un mensaje de error, en función del tipo de petición, la disponibilidad del recurso y el nivel de acceso o permisos que posea el cliente. En la siguiente ilustración se observa la comunicación de un servidor de archivos Samba:

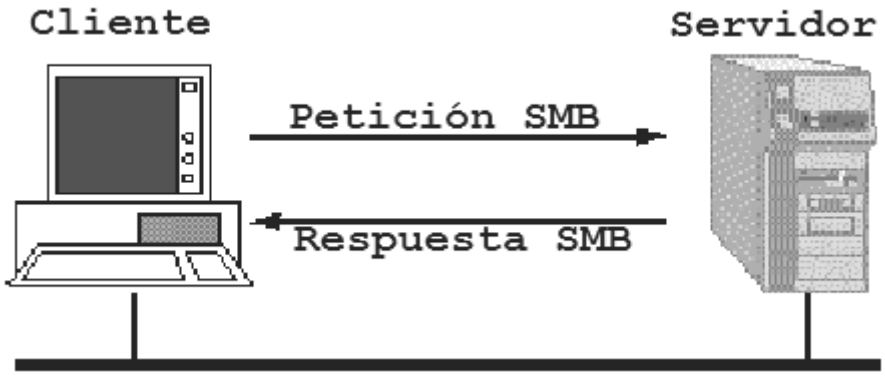


Ilustración 5: Comunicación por el protocolo SMB.

SMB soporta dos modos de autentificación alternativos, denominados *share* y *user* para controlar el acceso de los usuarios a los recursos compartidos. Los mismos son (Rallyexpresso, 2012):

- Cuando se comparte un recurso en modo *share*, la protección de dicho recurso recae en una contraseña que se asocia al mismo, de forma que cualquier usuario que conozca la contraseña podrá acceder sin restricciones al recurso (este es el mecanismo de autentificación por defecto en las implementaciones de SMB para *Windows* 9X, por ejemplo).
- En modo *user*, el servidor recibe inicialmente del sistema cliente unas credenciales de usuario (nombre, dominio y contraseña), que debe autentificar para autorizar el acceso al recurso. Concretamente, si el dominio de las credenciales es conocido, la autentificación se delega a algún controlador de dicho dominio; y en caso contrario, el usuario y la contraseña se autentifican contra la base de datos local del equipo servidor.

  En cualquier caso, en modo *user*, el control de acceso sobre el recurso se realiza en función de qué permisos posee sobre dicho recurso el usuario cuyas credenciales se han enviado desde el cliente. En otras palabras, una vez el sistema servidor ha identificado y autentificado al usuario que desea conectarse al recurso, este sistema dispone ya de un SID (Identificador de Seguridad) válido con el que puede contrastar los permisos que dicho SID posee sobre el recurso. Es conveniente recordar en este punto que si el recurso en cuestión es una carpeta compartida, se tienen en cuenta tanto los permisos del recurso. El modo *user* es el mecanismo de autentificación por defecto en las versiones de SMB de sistemas Windows NT y posteriores.

Selección: Luego de realizado un estudio de las características de los servidores de archivos descritas anteriormente se decidió seleccionar Samba en su versión 2.3 como tecnología para la compartición de recursos para el desarrollo de la herramienta para la publicación bajo demanda de grabaciones. El uso de esta tecnología se selecciona debido a sus características y porque garantiza la seguridad de los recursos mediante los modos de autentificación que posee, lo que permite cumplir con el objetivo trazado para el desarrollo de la investigación en relación con la seguridad.

1.5. Conclusiones del capítulo

La descripción de conceptos asociados al dominio del problema posibilitó reflejar con claridad los elementos que forman parte del entorno donde se desarrolla la aplicación. La profundización acerca del objeto de estudio garantizó precisar aspectos importantes respecto a los procesos relacionados con la compartición de archivos. Con el análisis de las soluciones informáticas existentes a partir de las necesidades de la investigación, se pudo verificar que éstas no brindan una solución completa a la investigación. Atendiendo a lo planteado con anterioridad surge la necesidad de desarrollar una herramienta para la publicación de bajo demanda de grabaciones.

Al finalizar las actividades se concluye que la caracterización de tecnologías y herramientas posibilitó realizar una selección de estas teniendo en cuenta los principios de soberanía tecnológica que defiende el país. La inclusión de la metodología RUP asegura una guía para regir todo el proceso de desarrollo del componente y propone para cada flujo de trabajo las actividades a realizar para generar un conjunto de artefactos ingenieriles. Además es válido agregar que la calidad del software dependerá en gran medida del cumplimiento de estas tareas y de la combinación de herramientas a utilizar. Partiendo de la caracterización de los procesos de compartición de archivos descritos en el capítulo y las tecnologías, herramientas y metodología a utilizar, se orienta el trabajo al desarrollo de la herramienta.

CAPÍTULO 2: CARACTERÍSTICAS DE LA HERRAMIENTA PARA LA PUBLICACIÓN BAJO DEMANDA DE GRABACIONES

Introducción

En el presente capítulo se presenta el modelo del dominio de la herramienta informática, además se realiza la especificación de los requisitos funcionales y no funcionales del sistema. Se presenta el diagrama de casos de uso del sistema, y la descripción de los actores y los casos de uso de este artefacto.

2.1. Modelo de Dominio

El Modelo de Dominio o Modelo Conceptual, permite de manera visual mostrar los principales conceptos que se manejan en el dominio del sistema en desarrollo. Un Modelo de Dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. Este modelo no se trata de un conjunto de diagramas que describen clases de *software* u objetos de *software* con responsabilidades, sino que puede considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio (Larman, 2003). Aprovechando las bondades de los diagramas UML para representar conceptos, el Modelo de Dominio se presenta en forma de diagrama de clases donde figuran los principales conceptos y roles de la herramienta en cuestión.

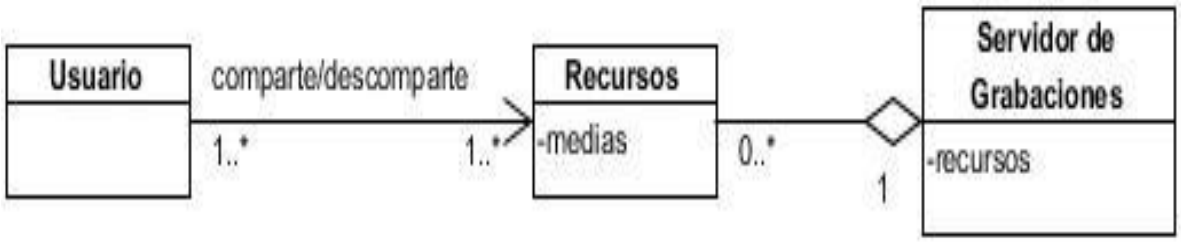


Ilustración 6: Modelo de Dominio.

Descripción del Modelo de Dominio:

Usuario: El usuario, puede ser un usuario del sistema o un usuario remoto, es el encargado de realizar todas las acciones sobre la herramienta.

Servidor de Grabaciones: Servidor donde se encuentran todas las medias grabadas por el sistema de video vigilancia.

Recursos: Las carpetas que contienen los videos almacenados que han sido grabados por el sistema de video vigilancia.

En la ilustración 6 se muestra como el usuario realiza las acciones necesarias para satisfacer las demandas de los operadores del sistema de video vigilancia, en dependencia a dicha demanda se realiza la compartición o eliminación de alguna compartición de los recursos que contienen las medias que se encuentran almacenadas en el servidor de grabaciones.

2.2. Modelado del sistema

2.3.1. Requisitos funcionales (RF)

Los requisitos funcionales (RF) de una aplicación son las capacidades o condiciones que el sistema debe cumplir para satisfacer las necesidades primarias del cliente (Jacobson, y otros, 2000). A continuación se definen los requisitos funcionales seleccionados para dar cumplimiento al desarrollo de la aplicación.

RF1 Administrar recursos de forma manual: esta funcionalidad debe permitir a un usuario compartir o dejar de compartir un recurso determinado que se encuentra en el servidor de grabaciones.

RF1.1 Compartir recursos de forma manual: esta funcionalidad debe permitir a un usuario compartir un recurso determinado que se encuentra en el servidor de grabaciones.

RF1.2 Dejar de compartir recursos de forma manual: esta funcionalidad debe permitir a un usuario dejar de compartir un recurso determinado que se encuentra en el servidor de grabaciones.

RF2 Mostrar los recursos que se encuentran compartidos: esta funcionalidad debe permitir mostrar los datos de los recursos que se encuentran compartidos en el servidor de grabaciones.

RF3 Editar los recursos que se encuentren compartidos: esta funcionalidad debe permitir editar los recursos que se encuentran compartidos en el servidor de grabaciones.

RF4 Administrar recursos de forma remota: esta funcionalidad debe permitir al gestor compartir o dejar de compartir un recurso determinado que se encuentra en el servidor de grabaciones de forma remota.

RF4.1 Compartir recursos de forma remota: esta funcionalidad debe permitir al gestor compartir un recurso determinado que se encuentra en el servidor de grabaciones de forma remota.

RF4.2 Dejar de compartir recursos de forma remota: esta funcionalidad debe permitir al gestor dejar de compartir un recurso determinado que se encuentra en el servidor de grabaciones de forma remota.

2.3.2. Requisitos no funcionales (RN)

Los requisitos no funcionales son las propiedades o cualidades que el producto debe presentar. Estos requisitos definirán las características del producto final y muchas veces están implicados en el éxito final que se pueda alcanzar (Jacobson, y otros, 2000).

- **Requisitos de usabilidad (RNU)**

RNU 1 El sistema debe tener una interfaz gráfica, visualmente atractiva para el usuario.

RNU 2 El sistema debe mostrar mensajes al usuario, que le ayuden a llevar a cabo la tarea que realiza.

RNU 3 Se debe hacer uso de botones, con imágenes que indiquen de modo intuitivo la función que realizan.

RNU 4 Los controles visuales deben mostrar mensajes que indiquen su función.

- **Requisitos de fiabilidad (RNF)**

RNF 5 El sistema debe estar disponible de forma permanente.

- **Requisitos de diseño e implementación (RNDI)**

RNDI 6 La herramienta debe ser implementada con el lenguaje C++ ajustándose a las funcionalidades del *framework* de desarrollo QT.

- **Requisitos de interfaz de usuario (RNIU)**

RNIU 7 Se requiere que la herramienta tenga una interfaz gráfica que permita la interacción con el usuario.

RNIU 8 Se requiere una interfaz ligera, que permita de manera sencilla la compartición de los recursos.

- **Requisitos de hardware (RNHW)**

RNHW 9 Memoria RAM 1 Gb o superior.

RNHW 10 Procesadores Pentium 4 o superior.

RNHW 11 Disco duro con 200 Mb de espacio o superior.

- **Requisitos de software (RNSW)**

RNSW 12 Se debe tener instalado el sistema operativo GNU/Linux.

RNSW 13 Se debe de tener instalado SAMBA.

RNSW 14 Se debe de tener instalado la biblioteca Libqxmlrpc.

2.3.3. Definición de los Casos de Usos del Sistema

Un diagrama de casos de uso documenta el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar. Estos diagramas sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema (Tello, 2013).

2.3.3.1. Actores del sistema

Los actores representan terceros fuera del sistema que interactúan con este. El actor es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso. Por lo regular estimula el sistema con eventos de entrada o recibe algo de él. Conviene escribir su nombre con mayúscula para facilitar la identificación (Hensgen, 2003).

Acorde a la situación en la que se desarrollará la herramienta se encontraron los siguientes actores:

Tabla 1: Definición de los actores del sistema.

Actor	Descripción
Usuario	Es el actor del sistema que representa una persona que puede compartir, dejar de compartir, mostrar y editar los recursos de las medias.
Gestor	Es un actor remoto, que representa una persona, encargado de enviar las peticiones a la herramienta para compartir o dejar de compartir los recursos de las medias.

2.3.3.2. Casos de uso del sistema

Los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. Un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores (Universidad Carlos III de Madrid, 2013).

Los casos de uso tienen la peculiaridad de ser tan globales o específicos como se quiera, por lo que en un caso de uso se pueden encapsular varias acciones realizadas por el sistema. Un caso de uso es un proceso que da un resultado de valor para un actor determinado.

1. **CU** Administrar Recursos Remoto
2. **CU** Administrar Recursos Manual
3. **CU** Mostrar Recursos Compartidos

4. **CU** Editar Recursos Compartidos

2.3.3.3. Diagrama de Casos de Uso del Sistema

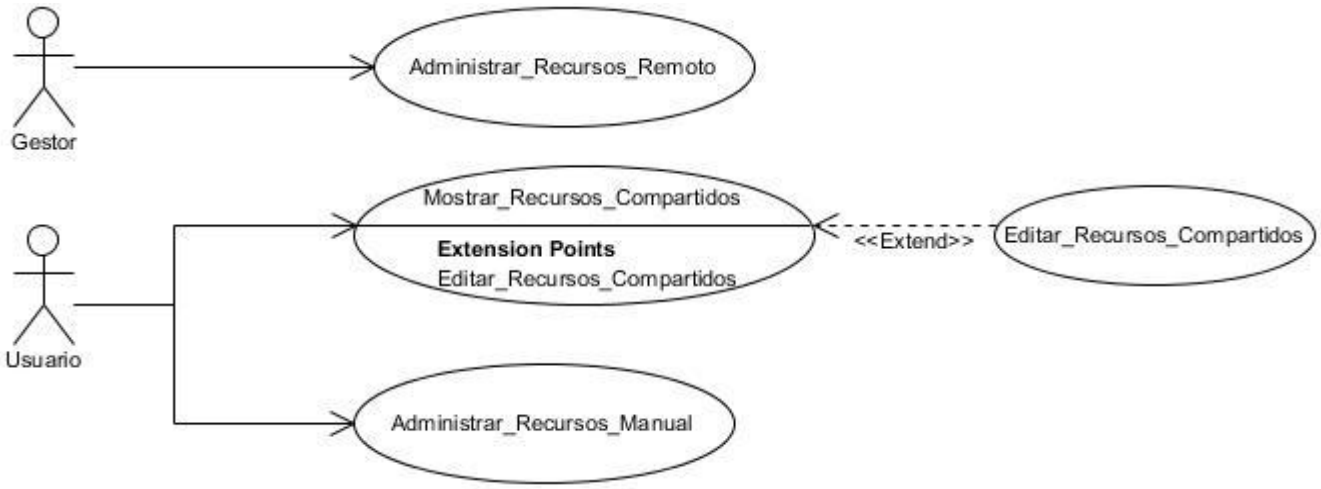


Ilustración 7: Diagrama de Casos de Usos de Sistema.

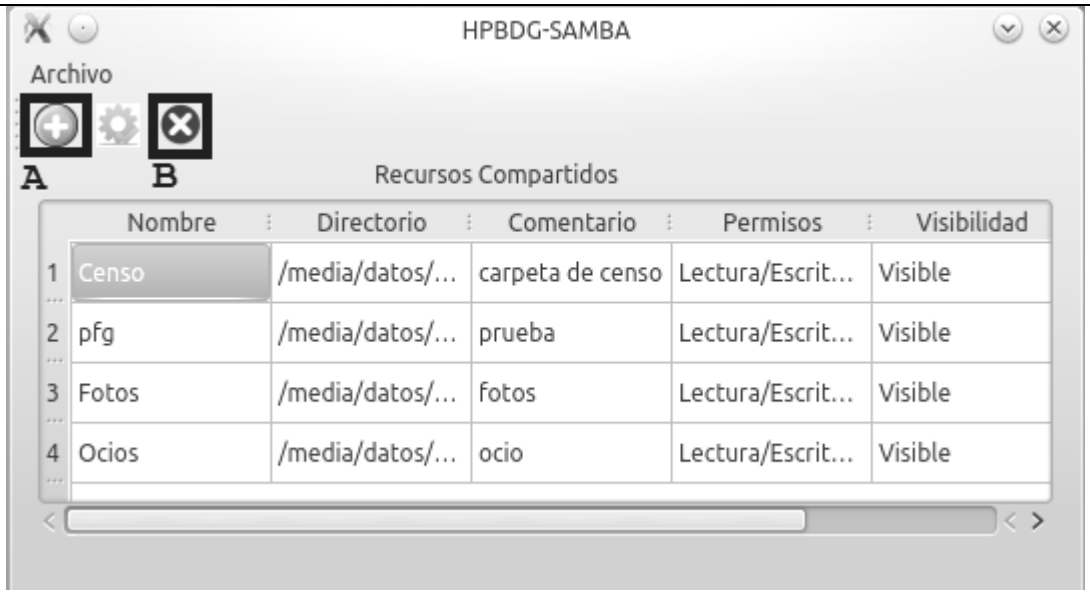
2.3.4. Descripción de los Casos de Usos del Sistema

Tabla 2: Descripción del Caso de Uso Administrar Recursos Manual.

Objetivo	Administrar Recursos Manual.	
Actores	Usuario (Inicia).	
Resumen	El caso de uso inicia cuando el usuario selecciona la opción Adicionar recurso compartido o borrar recurso compartido y termina cuando se confirma la realización de la operación.	
Complejidad	Media.	
Prioridad	Crítico.	
Precondiciones	Para dejar de compartir un recurso debe existir al menos un recurso compartido.	
Postcondiciones	Se mostró de forma satisfactoria la operación realizada.	
Flujo de eventos		
Flujo básico: Administrar Recursos Manual.		
	Actor	**Sistema**

1.	Selecciona la opción Adicionar Recurso Compartido (A) ver sección 1 Compartir Recursos o selecciona un recurso y luego la opción Borrar Recurso Compartico (B) ver sección 2 Dejar de Compartir Recursos.	

Prototipo de Interfaz



Sección 1: Compartir Recursos

Flujo Básico: Compartir Recursos**.**

	Actor	**Sistema**
1.		Muestra un formulario donde el usuario debe introducir los siguientes datos: -Directorio del recurso. -Nombre. -Comentario. -Usuario válido. -Permisos de Escritura/Lectura. -Visibilidad.

2.	Introduce los datos y selecciona la opción Aceptar. (C)	
3.		Verifica que los datos son correctos.
4.		Comparte el recurso solicitado.
5.		Muestra un mensaje informando de que la acción se ha realizado con éxito y finaliza el caso de uso.

Prototipo de Interfaz

Flujo Alterno

3 Evento. Los datos introducidos son incorrectos.

	Actor	**Sistema**
3.1		Si los datos no son correctos se lanza un mensaje especificando que existe un error.
3.1.1	Corrige el error y continúa a partir del paso 3 del flujo básico.	
2.1	Selecciona la opción Cancelar. (D)	

2.1.1		Cierra el formulario y finaliza el caso de uso.
Sección 2: Dejar de Compartir Recursos		
Flujo Básico: Dejar de Compartir Recursos.		
	Actor	**Sistema**
1.		Muestra un mensaje de confirmación.
2.	Selecciona la opción SI.	
3.		Deja de compartir el recuso seleccionado y se muestra un mensaje informando que la operación se ha realizado con éxito y finaliza el caso de uso.
Flujo Alterno		
1 Evento. Mensaje de confirmación.		
	Actor	**Sistema**
1.1	Selecciona la opción NO.	
1.1.1		Cancela la operación y finaliza el caso de uso.

Relaciones	**CU Incluidos**	Ninguno.
	CU Extendidos	Ninguno.
Requisitos no funcionales	Ninguno.	
Asuntos pendientes	No procede.	

Tabla 3: Descripción del Caso de Uso Administrar Recursos Remotos.

Objetivo	Administrar Recursos Remoto.
Actores	Gestor (Inicia).

Resumen	El caso de uso inicia cuando el Gestor envía una petición de compartir o dejar de compartir un recurso y termina cuando se envía un mensaje.	
Complejidad	Alta.	
Prioridad	Crítico.	
Precondiciones	Para dejar de compartir un recurso debe existir al menos un recurso compartido.	
Postcondiciones	Se mostró de forma satisfactoria la operación realizada.	
Flujo de eventos		
Flujo básico: Administrar Recursos Remoto.		
	Actor	**Sistema**
1.	Envía la información a la herramienta para compartir o dejar de compartir un recurso.	
2.		Comprueba cual es la solicitud del gestor. Si la solicitud es Compartir ver la Sección 1 Compartir Recursos y si es Descompartir ver la Sección 2 Dejar de Compartir Recursos.
Sección 1: Compartir Recursos		
Flujo Básico: Compartir Recursos**.**		
	Actor	**Sistema**
1.		Verifican que los datos sean correctos.
2.		Comparte el recurso solicitado por el gestor.
3.		Envía un mensaje de que la acción se ha realizado con éxito y finaliza el caso de uso.
Flujo Alterno		
1 Evento. Los datos introducidos son incorrectos.		

	Actor	**Sistema**
1.1		Si los datos no son correctos se envía un mensaje especificando que existe un error.
1.1.1	Corrige el error, envía los datos y continúa a partir del paso 1 del flujo básico.	
Sección 2: Dejar de Compartir Recursos		
Flujo Básico: Dejar de Compartir Recursos.		
	Actor	**Sistema**
1.		Verifica si el recurso existe.
2.		Deja de compartir el recurso, se envía un mensaje al gestor y finaliza el caso de uso.
Flujo Alterno		
1 Evento. El recurso no existe.		
1.1		Si el recurso no existe se envía un mensaje de error al gestor.
1.1.1	Corrige el error y envía los datos y continúa a partir del paso 1 del flujo básico.	
Relaciones	**CU Incluidos**	Ninguno.
	CU Extendidos	Ninguno.
Requisitos no funcionales	Ninguno.	
Asuntos pendientes	No procede.	

Las descripciones de los demás casos de uso se encuentran en el Anexo 1.

2.3. Conclusiones del capítulo

A partir de la realización y cumplimiento de las actividades que pertenecen al flujo de trabajo Requisitos se hizo posible definir los requisitos funcionales del sistema y las restricciones correspondientes para su desarrollo, donde se identificaron cuatro requisitos funcionales y 14 no funcionales. Se logró un mayor entendimiento del entorno donde se desarrolla el sistema mediante la realización del modelo de dominio y el sistema propuesto a través del diagrama de casos de uso y se presenta la descripción asociada a sus actores y casos de uso.

Los artefactos elaborados en este capítulo permitieron construir las bases para la realización del diseño de la herramienta para la publicación bajo demanda de grabaciones.

CAPÍTULO 3: DISEÑO DE LA HERRAMIENTA PARA LA PUBLICACIÓN BAJO DEMANDA DE GRABACIONES

Introducción

En este capítulo se presentan los artefactos ingenieriles asociados al diseño de la herramienta, los cuales son: el diagrama de clases del diseño y los diagramas de secuencia. Se explica el tipo de arquitectura que sustentará el desarrollo de la herramienta, así como los patrones de diseño a tener en cuenta para la construcción del diagrama de clase del diseño.

3.1. Arquitectura de *Software*

Sucede con mucha frecuencia que no se le presta especial atención a la asignación de responsabilidades. Una decisión errónea en estos aspectos puede desencadenar la realización de sistemas poco perdurables y sujetos a muchos cambios, perdiendo así tiempo y en muchos casos dinero con el cliente.

La arquitectura de un software define como va a estar estructurada la herramienta. Especifica el sistema de un software en términos de componentes computacionales y las interacciones entre los mismos (Pressman, 2005).

3.2. Estilos y patrones arquitectónicos

La decisión por un estilo o patrón arquitectónico no define el resultado final de una aplicación. Es necesario que el escogido sea aplicado correctamente para el desarrollo del software. Un estilo arquitectónico expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como las reglas para su construcción.

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. Definen la estructura de un sistema de software, los cuales a su vez se componen de subsistemas con sus responsabilidades. Se define como una regla que consta de tres partes, la cual expresa una relación entre un contexto, un problema y una solución (Camacho, y otros, 2004).

Contexto: Es una situación de diseño en la que aparece un problema de diseño.

Problema: Es un conjunto de fuerzas que aparecen repetidamente en el contexto.

Solución: Es una configuración que equilibra estas fuerzas. Ésta abarca:

- Estructura con componentes y relaciones
- Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes, la comunicación entre ellos (Camacho, y otros, 2004).

3.3.1. Estilo Arquitectónico llamada y retorno

El estilo arquitectónico llamada y retorno, el cual se utiliza en el desarrollo de la herramienta, refleja la estructura del lenguaje de programación. El mismo permite al diseñador del software construir una estructura de programa fácil de modificar y ajustar a escala. Se basan en la conocida abstracción de procedimientos, funciones y métodos. Los miembros de esta familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas (Pressman, 2005).

3.3.2. Patrón Arquitectónico 3-capas

Para el diseño de la herramienta se seleccionó una arquitectura en capas, esta arquitectura es una de las técnicas comunes que los arquitectos de software utilizan para dividir los sistemas. En un sistema en capas, cada capa descansa sobre la inferior. En este esquema la capa más alta utiliza los servicios definidos por la inferior, pero la inferior no conoce lo que se realiza en la capa superior. Además cada capa oculta las capas inferiores de las superiores a esta y la interacción entre las capas está limitada solo a las que son adyacentes (Reynoso, y otros, 2004).

Los beneficios de trabajar un sistema en capas son:

- Se puede entender una capa como un todo, sin considerar las otras.
- Las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos.
- Se minimizan dependencias entre capas.
- Las capas posibilitan la estandarización de servicios.
- Luego de tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel.

La variante utilizada en el desarrollo de la herramienta es 3-capas, la cual son técnicas que se utilizan para dividir los sistemas. Esta se seleccionó porque permite tener acopladas las clases de la aplicación y conocer el flujo de datos generado por las mismas. Las capas identificadas son:

Capa de presentación: se encuentran los formularios con los cuales el usuario podrá interactuar. El uso de la capa de presentación permite la captura de los datos ingresados por el usuario y visualiza la información solicitada. Además la misma se encarga de mostrar las notificaciones de alerta ante la ocurrencia de errores, asegurando la validación de los mismos durante la utilización de la herramienta.

Capa de lógica del negocio: en esta capa se establecen todas las reglas que deben cumplirse manteniendo comunicación con la capa de presentación, para recibir las solicitudes y presentar los resultados. Contiene las clases donde se realizan los procesos para compartir y dejar de compartir un recurso determinado. Se encuentra una clase para gestionar las peticiones enviadas por el gestor, una para interpretar la información que se encuentra en el fichero de configuración de Samba y otra clase que guardar los datos de una compartición determinada.

Capa de accesos a datos: se encuentra la clase con las funcionalidades para leer y escribir en el fichero de configuración de Samba.

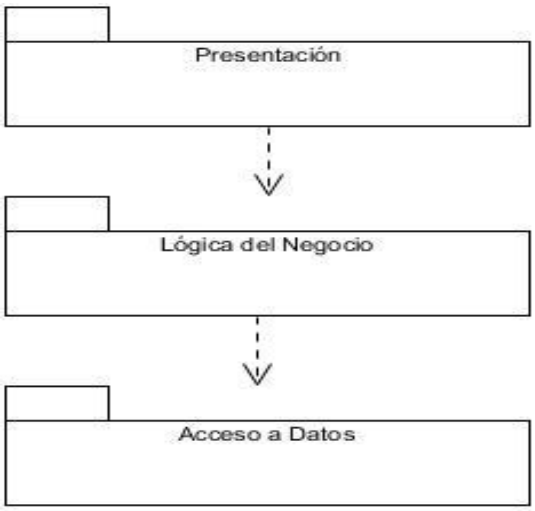


Ilustración 8: Patrón Arquitectónico 3-capas.

3.3.3. Patrones de diseño

Un patrón es una descripción de un problema y una solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos. Muchos patrones proporcionan guías sobre el modo en que

deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema (Larman, 2003).

Los patrones de diseño ayuda a como se debe estructurar las clases y objetos para guiar todo el procesos de creación del *software*. Componen soluciones concretas a problemas que se presentan durante el diseño de una aplicación. Entre los patrones de diseño más utilizados se encuentran los patrones GRASP (*General Responsibility Assignment Software Patterns*) y los patrones GoF (*Gang of Four*).

En la construcción del diagrama de clases para la solución se emplearon los siguientes patrones GRASP:

- **Experto:** Asigna una responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Un ejemplo de esto es la clase SambaFile la cual es la encargada de la comunicación con el fichero de configuración de Samba.
- **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos. Se aplica en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. Un ejemplo donde se utiliza este patrón es en las clases MainWindows y NewShare, pues contienen objetos de las clases con las que se comunican.
- **Bajo Acoplamiento:** Asigna una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Un ejemplo de esto se ve en las clases SambaMount y SambaUmount.
- **Alta Cohesión:** Asigna una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.

Los patrones GoF utilizados fueron:

- **Observador:** Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos. El uso de este patrón se ve reflejado en las clases donde se utilizan señales e slot, por ejemplo en las clases SambaServer y MainWindow.

3.4. Modelo de diseño

Para la transición de los requisitos al diseño normalmente se utiliza el modelo de análisis porque es una forma de suavizar dicha transición. Este modelo tiene dos propósitos fundamentales: refinar los casos de

uso con más detalle y establecer la asignación inicial de funcionalidades del sistema a un conjunto de objetos que proporcionan el comportamiento. Es una especificación detallada de los requisitos y funciona como primera aproximación del modelo de diseño.

El análisis tiene como misión el estudio de los requisitos, así como el estructurarlos y darle el refinamiento pertinente a los mismos, con el objetivo de conseguir una comprensión más precisa de ellos. Por su parte, el diseño es una representación abstracta de lo que se va a construir, contribuye a formar una arquitectura sólida y ayuda a crear un plano para la implementación. Mediante él se modela el sistema para que soporte todos los requisitos incluyendo los no funcionales (Pressman, 2005).

El análisis es una etapa de la metodología RUP que debe de evidenciarse en la construcción del software. De esta se puede prescindir de acuerdo a la manera en que se haya decidido implementar el sistema, pasando a analizar los requisitos como parte del diseño o como parte integrada de la captura de los requisitos, lo que deja en evidencia que la manera de emplear el análisis es consecuente con la construcción del sistema (Jacobson, y otros, 2000).

Se decide pasar del flujo de requisitos directamente al diseño, sin la implementación del análisis debido a que:

- Es posible obtener un mayor formalismo en el modelo de casos de uso pues es fácil de comprender los resultados que estos pueden proporcionar.
- Los requisitos son simples, bien conocidos y se cuenta con cierta comprensión de los mismos.
- No se conservaría la estructura generada en el análisis dado que el diseño debe considerar las tecnologías a utilizar en la construcción del sistema.
- Se logra evitar el costo en tiempo y recursos de mantener este flujo.

Teniendo en cuenta que se decidió no realizar el Modelo de Análisis, se puede pasar directamente al Modelo de Diseño. Los modelos de diseño muestran los objetos o clases en un sistema y donde sea apropiado los diferentes tipos de relaciones entre estas entidades. Son el puente entre los requisitos y la implementación del sistema. El modelo de diseño tiene que ser abstracto con el fin de que el detalle innecesario no oculte las relaciones entre ellos y los requisitos del sistema. Tiene que incluir suficiente detalle para que los programadores tomen las decisiones para la implementación (Pressman, 2005).

3.4.1. Diagrama de clases del diseño

Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro (Pressman, 2005). Seguidamente se muestra una representación del diagrama de clases del diseño de la herramienta para la publicación bajo demanda de grabaciones. El mismo cuenta con tres capas, la capa de presentación, lógica de negocio y la de acceso a datos. En la presentación se encuentran las clases interfaz. En la capa lógica de negocio se encuentran las clases de las cuales hacen instancias las de la presentación y la clase que poseen la comunicación con el gestor y en la capa de acceso a datos se encuentra la clase encargada de comunicarse con el archivo de configuración de Samba. El diagrama íntegro se encuentra en el Anexo 2.

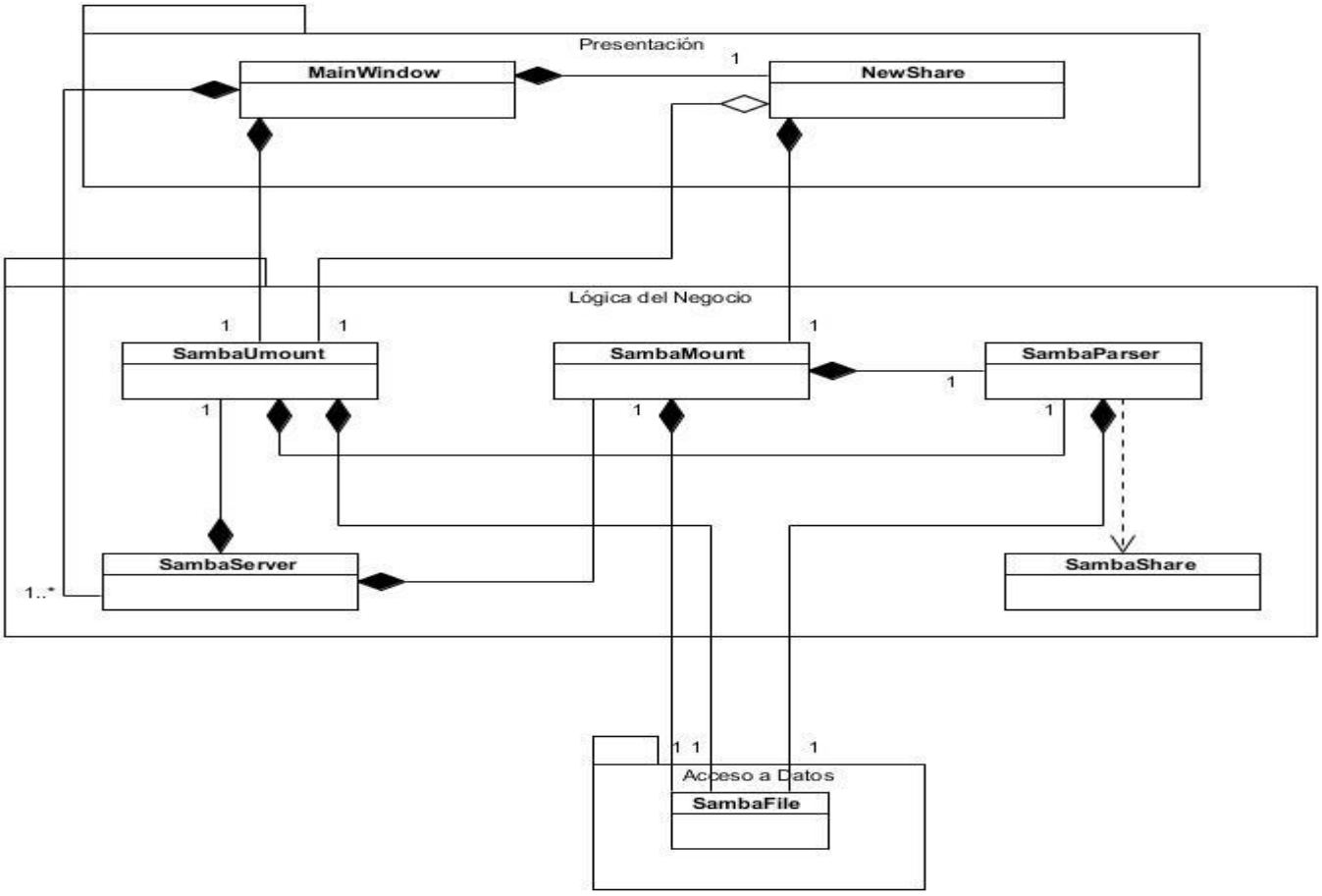


Ilustración 9: Diagrama de diseño simplificado.

Descripción del diagrama de clases:

Capa Presentación: residen los formularios, los cuales son los encargados de brindarles la información a los usuarios en forma de interfaces para viabilizar la interacción de los mismos con la herramienta. Como formulario e interfaz principal se encuentra MainWindow, encargado de recibir todos los eventos y acciones generados por el usuario sobre la herramienta. Mediante el formulario NewShare el usuario podrá compartir un nuevo recuso insertando los datos necesarios.

Capa Lógica de Negocio: se encuentran las clases que implementan todas las funcionalidades que garantizarán el cumplimiento de todos los eventos generados desde la capa de Presentación, logrando así el correcto funcionamiento de la herramienta, estas clases son SambaMount y SambaUmount encargadas de compartir y dejar de compartir un recurso respectivamente, SambaParser es la clase encargada de interpretar la información que se encuentra en el fichero de configuración de Samba, SambaShare guarda los datos de una compartición determinada y la clase SambaServer es la encargada de recibir y procesar las peticiones enviadas por el gestor.

Capa de Acceso a Datos: contiene la clase de acceso al fichero de configuración de Samba, utilizándose la clase SambaFile la cual cuanta con las funcionalidades necesarias para leer y escribir en el fichero de configuración de Samba.

3.4.2. Diagramas de secuencia

En esta sección se describe cómo ocurre la interacción entre los objetos que participan en la implementación de cada uno de los casos de uso presentados en el capítulo 2. Para esto se emplean diagramas de secuencia, que describen la distribución de tareas entre componentes. De esta forma, se puede ver cómo ocurren los procedimientos en la herramienta.

- Diagrama de secuencia para el caso de uso Administrar Recursos Manual:

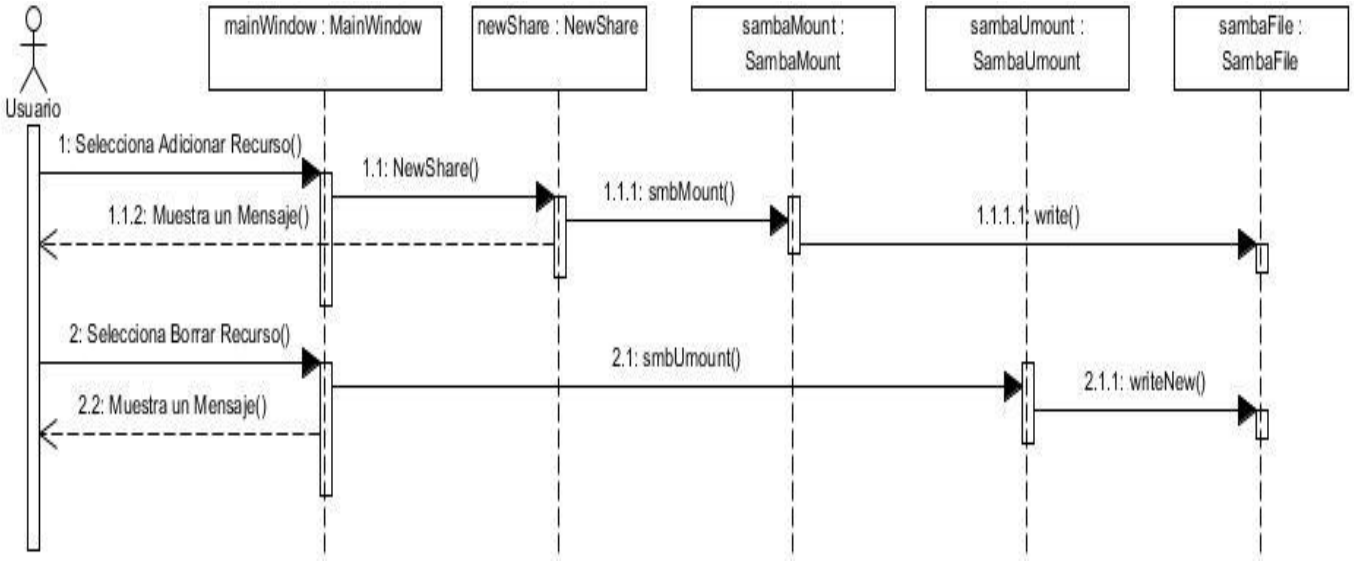


Ilustración 10: Diagrama Secuencia para el Flujo Normal del CU Administrar Recursos Manual.

- Diagrama de secuencia para el caso de uso Administrar Recursos Remoto:

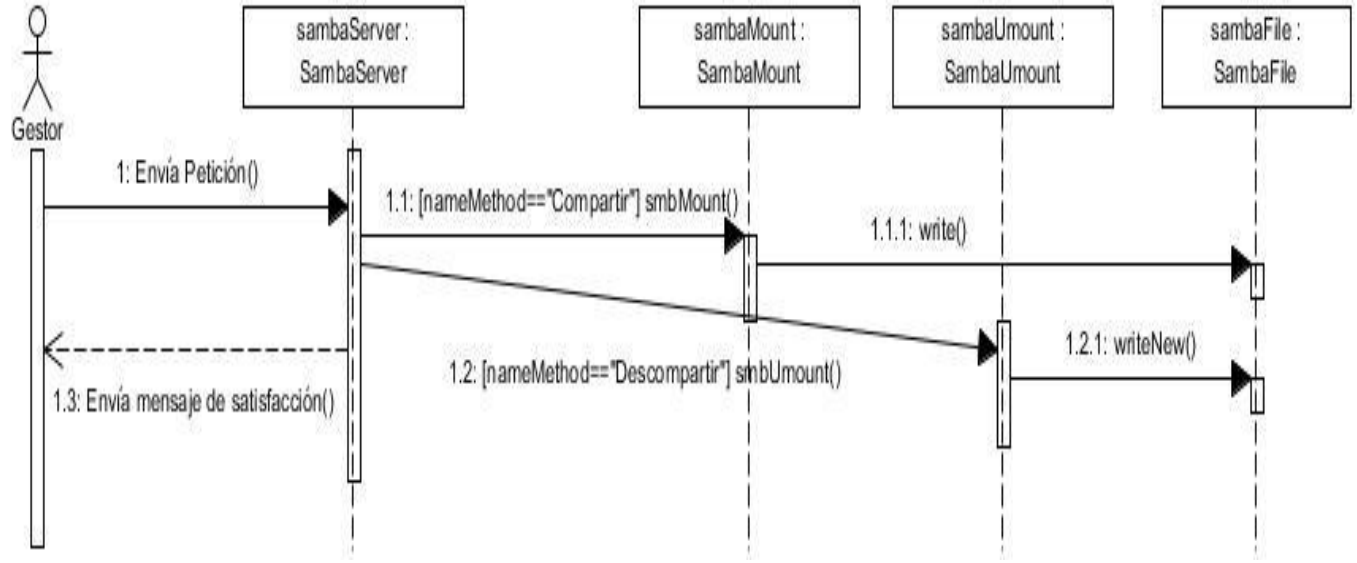


Ilustración 11: Diagrama Secuencia para el Flujo Normal del CU Administrar Recursos Remoto.

Los diagramas de secuencias de los casos de uso restantes se pueden ver en el Anexo 3.

3.5. Conclusiones del capítulo

En la elaboración del flujo de trabajo del Diseño se seleccionó una arquitectura en capas que garantizó definir una estructura lógica para el sistema, asegurando una mejor organización e interrelación para sus componentes. La aplicación de patrones de diseño GRASP y GoF ayudó a complementar la arquitectura del sistema evidenciando el uso de soluciones demostradas como parte de las buenas prácticas de la programación en la realización del diagrama de clases del diseño. Esta fase posibilitó modelar la estructura de la herramienta siguiendo el estilo arquitectónico definido, así como, representar el flujo de eventos durante la interacción de los usuarios con el sistema.

Con la elaboración de los artefactos ingenieriles asociados al Diseño se posee el fundamento básico para realizar la implementación de la herramienta.

CAPÍTULO 4: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

En este capítulo se presentan el diagrama de componente y el diagrama de despliegue asociado al flujo de trabajo de implementación. Se especifica el tipo de prueba a utilizar para verificar el nivel de cumplimiento de las funcionalidades definidas en los requisitos funcionales, así como los resultados que estas arrojaron.

4.1. Modelo de Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, se implementan en términos de componentes, ficheros de código fuente y ejecutables. Describe cómo se organizan los componentes de acuerdo a los mecanismos de estructuración disponibles en el entorno de implementación y lenguaje o lenguajes de implementación empleados, y cómo dependen los componentes unos de otros (Jacobson, y otros, 2000).

4.2.1. Diagrama de despliegue

Los diagramas de despliegue muestran los nodos procesadores, la distribución de los procesos y de los componentes. Son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema. Describen la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP, JDBC o RMI (Larman, 2003).

Ilustración 12: Diagrama de Despliegue.

Descripción de los nodos:

Gestor: Es el encargado de enviar las peticiones necesaria al Servidor de Grabaciones donde se encuentra la herramienta para compartir o dejar de compartir un recurso determinado.

Servidor de Grabaciones: Servidor donde se encuentran todas las medias grabadas por el sistema de video vigilancia.

4.2.2. Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de *software***:** código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del *software*, la reutilización, y las limitaciones imputadas por los lenguajes de programación y las herramientas utilizadas en el desarrollo (Larman, 2003). En la ilustración siguiente se muestra el diagrama de componentes.

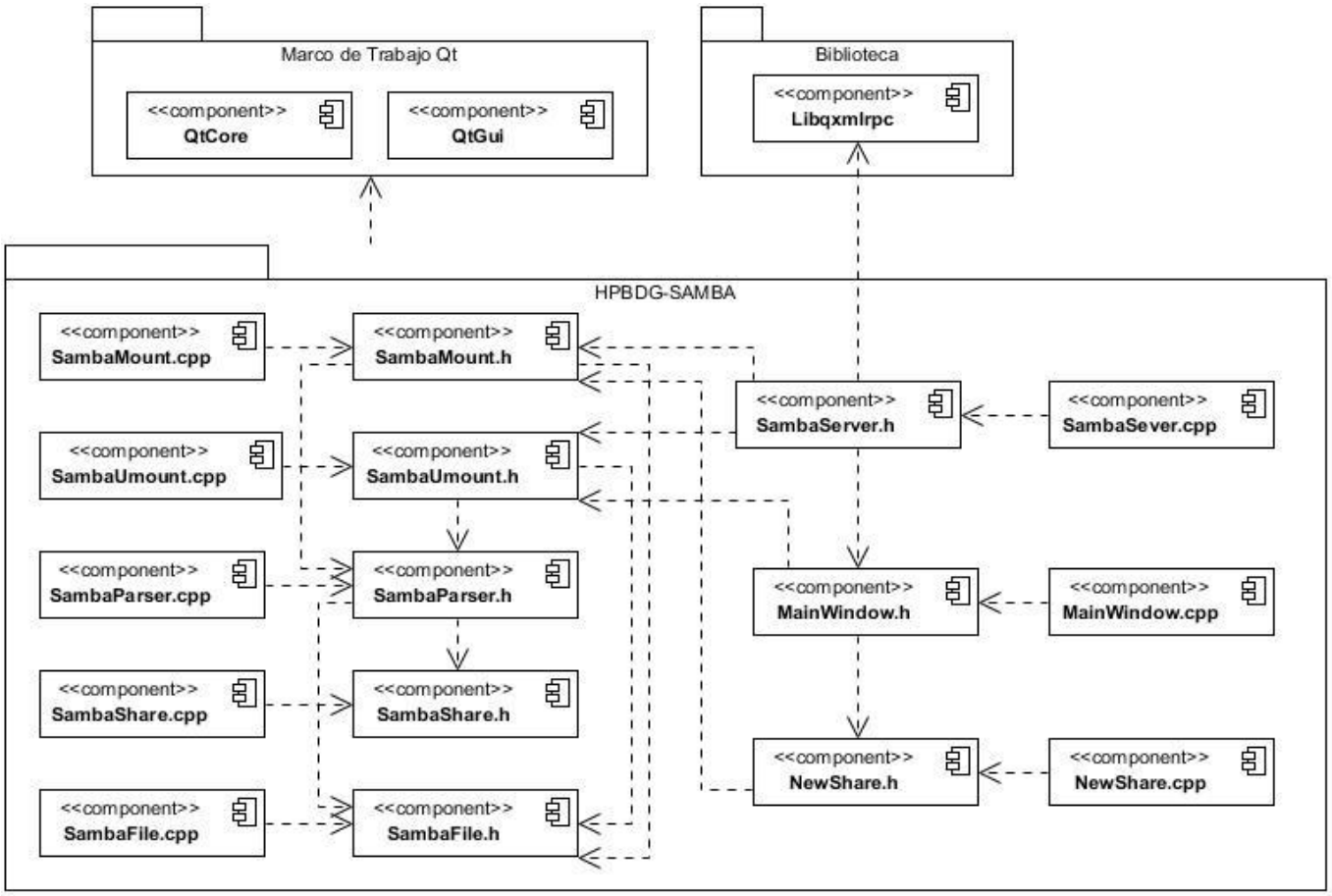


Ilustración 13: Diagrama de Componente.

Descripción del diagrama de componente:

El diagrama de componentes que se presenta en la ilustración 13 se encuentra dividido en tres paquetes:

1. En el paquete del Marco de Trabajo Qt se encuentran los módulos utilizados para el desarrollo de la herramienta los cuales son QtCore y QtGui:
   QtCore: Es el modulo que contiene las funcionalidades principales Qt.
   QtGui: Módulo que provee las funcionalidades para el trabajo con la interfaz gráfica de usuario.
2. En el paquete Biblioteca se encuentra libqxmlrpc es una biblioteca implementada para Qt que permite la comunicación del gestor con la herramienta desarrollada.
3. En el paquete HPBDG-SAMBA (Herramienta para la Publicación Bajo Demanda de Grabaciones SAMBA) se encuentran todos los ficheros utilizados para el desarrollo de la aplicación para la publicación bajo demanda de grabaciones.

4.3. Pruebas al Sistema

Las pruebas se le realizan a un *software* con la finalidad de descubrir errores, tanto en la lógica interna como en funciones externas del mismo. Las pruebas son un elemento crítico para garantizar la calidad del *software*, estas no mejoran ni aseguran la calidad del *software*, pero sí lo hace indirectamente previendo un grupo de debilidades asociadas a la organización y sus riesgos. Existen varios tipos de prueba, los dos métodos más utilizados son las pruebas de Caja Blanca (CB), y las de Caja Negra (CN) (Pressman, 2005).

4.3.1. Diseño de Prueba de Caja Negra

Las pruebas de caja negra permitirán identificar las posibles fallas en el funcionamiento de la herramienta. Estas pruebas se centran principalmente en las características de la herramienta independientemente del código. Para llevar a cabo estas pruebas es necesario tener conocimiento sobre los escenarios de prueba y secciones que serán probadas, con estos escenarios es muy fácil comprender el funcionamiento del requerimiento en cuestión (Pressman, 2005).

En los casos de pruebas se combinan los posibles juegos de datos válidos y no válidos que son necesarios para verificar el correcto funcionamiento de la herramienta. Estas pruebas se realizan a nivel de sistema, el tipo de prueba es funcional, donde se emplea el enfoque estructural o de caja negra, específicamente dando uso a la técnica de particiones equivalentes. Esta técnica es muy efectiva al

realizar pruebas sobre la interfaz de la aplicación, estas prueban la validez de cada entrada de datos o información al sistema. Pretenden demostrar que las funciones de la herramienta son operativas.

4.3.1.1. Diseño de caso de prueba para el CU Administrar Recurso Manual

Tabla 4: Secciones a probar en el CU Administrar Recursos Manual.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo central**
SC 1: Compartir Recursos Manual	EC 1.1: Compartir Recursos Manual con éxito.	El usuario selecciona la opción Adicionar Recurso Compartido e inserta los datos en el formulario que muestra el sistema y da clic en el botón Aceptar y el sistema muestra un mensaje de que la operación se ha realizado con éxito.	Opción Adicionar Recurso Compartido /botón “Aceptar”.
	EC 1.2: Compartir Recursos Manual con error.	El usuario selecciona la opción Adicionar Recurso Compartido e inserta los datos en el formulario que muestra el sistema y da clic en el botón Aceptar y el sistema muestra un mensaje de error que informa que los datos son incorrecto o existen campos vacíos.	Opción Adicionar Recurso Compartido /botón “Aceptar”.
	EC 1.3: Compartir Recursos Manual Cancelar.	El usuario selecciona la opción Adicionar Recurso Compartido se muestra el formulario y da clic en el botón Cancelar y el sistema cierra el formulario.	Opción Adicionar Recurso Compartido /botón “Cancelar”.
SC 2: Dejar de Compartir	EC 2.1: Dejar de Compartir Recursos	El Usuario selecciona el recurso que desea dejar de compartir en la tabla,	Seleccionar recurso de la

Recursos Manual	Manual con éxito.	se activa la opción de Borrar Recurso Compartido y da clic en ella y se muestra un cuadro de dialogo ¨Esta seguro que desea dejar de compartir el recurso¨ el usuario selecciona la opción ¨SI¨ y el sistema muestra un mensaje de que la operación se ha realizado con éxito.	tabla/ Opción Borrar Recurso Compartido / botón “SI”.
	EC 2.2: Dejar de Compartir Recursos Manual. Cancelar.	El Usuario selecciona el recurso que desea dejar de compartir en la tabla, se activa la opción de Borrar Recurso Compartido y da clic en ella y se muestra un cuadro de dialogo ¨Esta seguro que desea dejar de compartir el recurso¨ el usuario selecciona la opción ¨NO¨ y el sistema cancela la operación y vuelve a la interfaz anterior.	Seleccionar recurso de la tabla/ Opción Borrar Recurso Compartido / botón “NO”.

Tabla 5: Descripción de variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Directorio del Recurso	Campo de texto	No	Directorio del recurso que se desea compartir.
2	Nombre	Campo de texto	No	Nombre que va a tener el recurso compartido.
3	Comentario	Campo de texto	Si	Un comentario que puede tener o no el recurso.
4	Usuario Válido	Campo de texto	No	Es el usuario que va a tener permiso para acceder al recurso

				compartido.
5	Permisos	Campo chequeable	Si	Si el recurso va a tener permisos de lectura y escritura o solo de lectura.
6	Visibilidad	Campo chequeable	Si	Si el recurso va a estar visible o no.

Matriz de Datos.

Tabla 6: SC 1 Compartir Recursos Manual.

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Compartir Recursos Manual con éxito.	El sistema muestra un mensaje de que la operación a ha realizado con éxito.	Satisfactorio
EC 1.2	Compartir Recursos Manual con error.	El sistema muestra un mensaje de error que alerta que existen campos vacíos o con datos incorrectos.	Satisfactorio
EC 1.3	Compartir Recursos Manual Cancelar	El sistema cierra el formulario y regresa al formulario anterior.	Satisfactorio

ID del escenario	Directorio del recurso	Nombre	Comentario	Usuario Válido	Permisos	Visibilidad
EC 1.1	V/ “/media/datos”	V/ “datos”	V/ “Carpeta”	V/ “Administrador”	V/ “chequeado”	V/ “chequeado”

EC 1.2	I/ “/no/existe”	V/ “prueba”	V/ “”	V/ “Administrador”	V/ “no chequeado”	V/ “chequeado”
	V/ “/media/datos”	I/ “campo vacío”	V/ “Datos”	V/ “Administrador”	V/ “chequeado”	V/ “chequeado”
	V/ “/media/datos”	V/ “prueba”	V/ “Datos”	I/ “campo vacío”	V/ “chequeado”	V/ “chequeado”
	V/ “/media/datos”	I/ “campo vacío”	V/ “Datos”	V/ “Administrador”	V/ “no chequeado”	V/ “no chequeado”
	I/ “campo vacío”	V/ “prueba”	V/ “Datos”	V/ “Administrador”	V/ “chequeado”	V/ “no chequeado”
	I/ “campo vacío”	I/ “campo vacío”	V/ “”	I/ “campo vacío”	V/ “chequeado”	V/ “no chequeado”

Tabla 7: SC 2 Dejar de Compartir Recursos Manual.

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Dejar de Compartir Recursos Manual con éxito.	El sistema muestra un mensaje de que la operación a ha realizado con éxito.	Satisfactorio
EC 1.2	Dejar de Compartir Recursos Manuel. Cancelar.	El sistema cancela la operación y vuelve a la interfaz anterior.	Satisfactorio

Las restantes secciones a probar de los demás CU se pueden ver en el Anexo 4.

4.3.2. Resultado de las Prueba de Caja Negra

En la primera iteración de pruebas al sistema se encontraron 15 No Conformidades que fueron corregidas, luego se procedió a realizar una segunda iteración, la cual arrojó resultados satisfactorios, por lo que no se necesitó la realización de una tercera iteración. En el siguiente gráfico se muestran los resultados arrojados por las pruebas después de realizarse las dos iteraciones a la herramienta.

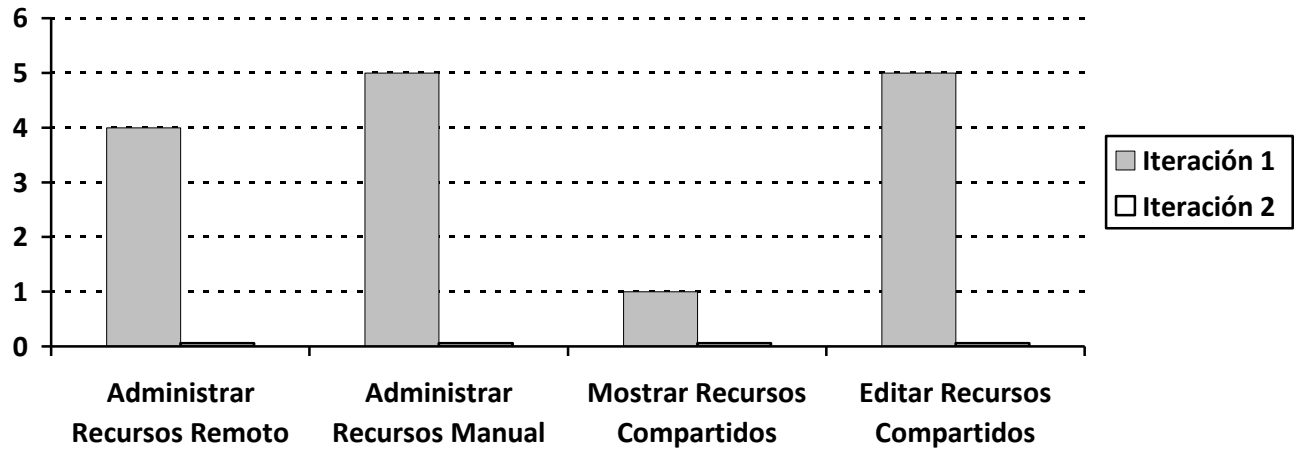


Ilustración 14: Resultado de las pruebas.

4.4. Conclusiones del capítulo

Al finalizar las actividades correspondientes a este capítulo se apreció la realización del flujo de Implementación y Prueba. Siguiendo el proceso de desarrollo en donde aparece la implementación, el uso del IDE Qt Creator posibilitó desarrollar una interfaz gráfica sencilla e intuitiva partiendo del uso de widgets como componentes gráficos. A partir de la utilización de la biblioteca qxmlrpc se garantizaron los procesos de comunicación con el gestor. Luego de ser generado el código fuente del componente se aplicaron casos de prueba mediante la técnica de caja negra lo que facilitó determinar las no conformidades al culminar la primera iteración de desarrollo. En la segunda iteración de pruebas se alcanzaron resultados satisfactorios debido a que no fue identificada ninguna no conformidad. A partir de la realización de pruebas se validó que los requisitos funcionales definidos para la herramienta sean totalmente operativos elevando la calidad del software.

CONCLUSIONES

El cumplimiento de las actividades propuestas durante la investigación hizo posible que el objetivo trazado se lograra con éxito. Lo que conllevó arribar a las siguientes conclusiones:

- La selección de la metodología de desarrollo, el lenguaje de modelado y la herramienta CASE a utilizar propició la correcta representación del sistema a través de los diferentes modelos.
- La elaboración de todos los artefactos asociados a la metodología RUP así como la utilización del lenguaje de C++, el IDE QT Creator, el framework QT, el protocolo de comunicación XMLRPC y SAMBA permitieron desarrollar una herramienta para la publicación bajo demanda de grabaciones de seguridad.
- La herramienta que se obtuvo como resultado de la investigación permitió la publicación bajo demanda de grabaciones de seguridad, por lo que, en comparación con las demás soluciones informáticas existentes semejantes a esta se evidencia una ventaja funcional.
- Se insistió en todo momento en la utilización de *software* y herramientas libres lo que contribuye con la soberanía tecnológica que defiende el país.

RECOMENDACIONES

Al concluir el presente trabajo, se recomienda:

- Realizar la compartición de los recursos utilizando otros protocolos.
- Permitir la comunicación con la herramienta haciendo uso de otro midleware.

REFERENCIAS BIBLIOGRÁFICAS

- **Ariganello, Ernesto. 2005.** Aprende Redes.com. 2005.
- **Caceres, Cristian. 2008.** Tipos de Servidores. [En línea] 2008. [Citado el: 6 de Abril de 2013.] http://www.slideshare.net/crinodj/tipos-de-servidores.
- **CAMACHO, E, CARDESO, F y NUÑEZ, G. 2004.** Arquitecturas de Software. [En línea] 2004. [Citado el: 6 de Febrero de 2013.] http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia_Arquitectura.pdf.
- **Cataldi, Z, y otros. 2012.** Ingenieria de Software Educativo. [En línea] 2012. [Citado el: 5 de Diciembre de 2012.] http://www.iidia.com.ar/rgm/comunicaciones/c-icie99-ingenieriasoftwareeducativo.pdf.
- **Free Download Manager. 2007.** Sitio de descargas de software. [En línea] 2007. [Citado el: 5 de Febrero de 2013.] http://www.freedownloadmanager.org/es/downloads/servidor_de_ftp_de_gene6_gratis/.
- **Freecode. 2013.** Free (code). [En línea] 2013. [Citado el: 25 de Abril de 2013.] http://freecode.com/projects/gadmin-samba.
- **Glosario.NET. 2007.** Glosario.NET. HispaNetwork Publicidad y Servicios, S.L. [En línea] 2007. [Citado el: 4 de Febrero de 2013.] http://tecnologia.glosario.net/terminos-viricos/lenguaje-de-programaci%F3n-9768.html.
- **Hensgen, Paul. 2003.** Manual de Umbrello UML Modeller. [En línea] 2003. [Citado el: 6 de Febrero de 2013.] http://docs.kde.org/stable/es/kdesdk/umbrello/umbrello.pdf.
- **Jacobson, Ivar y Rumbaugh, James. 2000.** El Proceso Unificado de Desarrollo de Software. Madrid : Pearson Educacion, 2000.
- **Kioskea ES. 2013.** Kioskea. [En línea] 2013. [Citado el: 6 de Abril de 2013.] http://es.kioskea.net/contents/104-como-compartir-archivos-en-windows-xp.
- **Kioskea.net. 2012.** File Transfer Protocolo. [En línea] 2012. [Citado el: 25 de Noviembre de 2012.] http://es.kioskea.net/contents/internet/ftp.php3.
- **Larman, Craig. 2003.** UML y Patrones. 2003.
- **Lazalde Miranda, Cristina y Miranda Valles, Mayra Yanin. 2010.** EcuRed. [En línea] 2010. [Citado el: 4 de Fefrero de 2013.] http://www.ecured.cu/index.php/Herramienta_CASE.

- **LinuxLinks. 2012.** LinuxLinks.com. 9 of the Best Free Linux Integrated Development Environments (IDEs). [En línea] 2012. [Citado el: 4 de Febrero de 2013.] http://www.linuxlinks.com/article/20090620114618990/IDE.html.
- **Muñoz Casals, Ing. Velmour. 2008.** Proyecto de Software Libre “Samba”. 2008.
- **Nava, Maily. 2013.** Scribd. Arquitectura de Softwarw. [En línea] 2013. [Citado el: 6 de Mayo de 2013.] http://es.scribd.com/doc/23161581/Estilos-Arquitectonico.
- **NOKIA CORPORATION. 2008.** NOKIA CORPORATION. QT. [En línea] 2008. [Citado el: 6 de Febrero de 2013.] http://qt.nokia.com/products/developer-tools/.
- **Ospina, Jenny. 2011.** Servicios del servidor. [En línea] 2011. [Citado el: 6 de Abril de 2013.] http://www.slideshare.net/jennypao2011/servicios-del-servidor.
- **PHPNuke. 2013.** PHPNuke. [En línea] 2013. [Citado el: 5 de Febrero de 2013.] http://downloads.phpnuke.org/es/download-item-view-m-g-m-a-l/Gene6_FTP_Server.htm.
- **Pozo, Salvador. 2009.** Curso de C++. [En línea] 2009. [Citado el: 28 de Diciembre de 2012.] http://c.conclase.net/curso/.
- **Pressman, Roger S. 2005.** Ingeniería del Software.Un enfoque práctico. 2005. ISBN:9701054733.
- **Rallyexpresso. 2012.** El protocolo SMB. Configuración básica Samba. [En línea] 2012. [Citado el: 28 de Noviembre de 2012.] http://www.rallyexpresso.com/.
- **Ramirez, Herrera, Ing. Jhon Mauricio. 2010.** @yudas Web 2.0 Para Docentes y un poco más... [En línea] 2010. [Citado el: 25 de Noviembre de 2012.] http://ayudasparadocentes.blogspot.com/2012/01/que-es-un-acceso-remoto.html.
- **Reynoso, Carlos y Kicillof, Nicolás. 2004.** Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. 2004.
- **Rodriguez Alvarez, Ingrid. 2012.** Protocolo (informática). [En línea] 2012. [Citado el: 25 de Noviembre de 2012.] http://es.scribd.com/doc/33720640/Protocolo-informatica.
- **Rumabugh, James, Jacobson, Ivar y Booch, Grady. 1998.** El Leguaje Unificado de Modelado. 1998.
- **Snapfiles. 2011.** HTTP File Server. [En línea] 2011. [Citado el: 26 de Noviembre de 2012.] http://www.snapfiles.com/get/hfs.htm.
- **Tello, Jesús Cáseres. 2013.** Universidad de Alcalá. [En línea] 2013. [Citado el: 6 de Febrero de 2013.] http://www2.uah.es/jcaceres/capsulas/DiagramaCasosDeUso.pdf.

- **Universidad Carlos III de Madrid. 2013.** Modelado Básico con Casos de Uso. [En línea] 2013. [Citado el: 6 de Febrero de 2013.] http://ocw.uc3m.es/ingenieria-informatica/diseno-de-software-avanzado/material-de-clase-1/04-Modelado_Basico_con_Casos_de_Uso.pdf.
- **Viñola Sosa, Raidel Raul y Roquero Figueroa, Alexander. 2012.** Sistema Gestor de Proceso de Media v2. 2012. No: 7, Vol: 5.
- **Wallen, Jack. 2012.** Five tools for configuring Samba. [En línea] 2012. [Citado el: 25 de Abril de 2013.] http://www.techrepublic.com/blog/five-apps/five-tools-for-configuring-samba/1398.
- **Webnova. 2011.** Web Services - XML-RPC, SOAP, sobre PHP, Perl, y otros conceptos. [En línea] 2011. [Citado el: 6 de Febrero de 2013.] http://webnova.com.ar/articulo.php?recurso=426.
- **XMLRPC.COM. 2011.** XMLRPC.COM. [En línea] 2011. [Citado el: 6 de Febrero de 2013.] http://xmlrpc.scripting.com/.
- **ZonaQt. 2010.** ZonaQt. [En línea] 2010. [Citado el: 12 de Diciembre de 2012.] http://www.zonaqt.com/tutoriales/tutorial-b%C3%A1sico-de-qt-4.

BIBLIOGRAFÍA CONSULTADA

- **Álvarez, Sara. 2012.** Desarrollo Web. [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://www.desarrolloweb.com/articulos/protocolo-http-ftp.html.
- **Anonimo. 2010.** Aprenda Qt dede hoy mismo. 2010.
- **Arregocés, Benyi Carrere. 2007.** Consumer. [En línea] 2007. [Citado el: 26 de Noviembre de 2012.] http://www.consumer.es/web/es/tecnologia/internet/2007/08/30/166291.php.
- **Campo, Gustavo Damian. 2009.** Patrones de Diseño, Refactorización y Antipatrones. 2009.
- **Cataldi, Z, y otros. 2012.** Ingenieria de Software Educativo. [En línea] 2012. [Citado el: 5 de Diciembre de 2012.] http://www.iidia.com.ar/rgm/comunicaciones/c-icie99-ingenieriasoftwareeducativo.pdf.
- **gobiernodecanarias. 2012.** Gobierno de Canarias. Protocolos de Red. [En línea] 2012. [Citado el: 25 de Noviembre de 2012.] http://www.gobiernodecanarias.org/educacion/conocernos_mejor/paginas/protocol1.htm.
- **Jacobson, Ivar y Rumbaugh, James. 2000.** El Proceso Unificado de Desarrollo de Software. Madrid : Pearson Educacion, 2000.
- **Larman, Craig. 2003.** UML y Patrones. 2003.

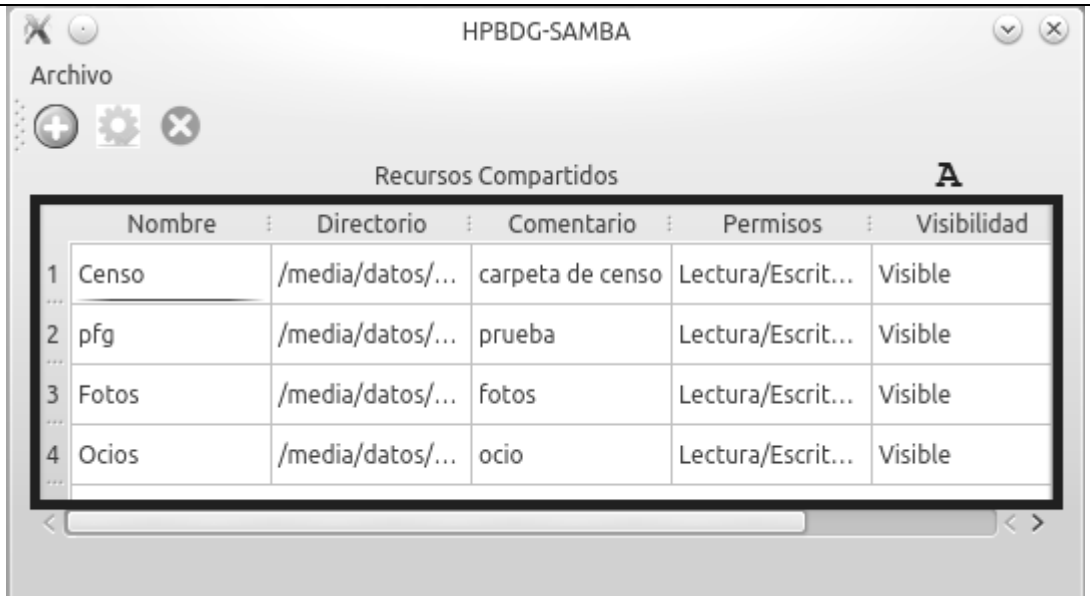
- **mit. 2012.** Red Hat Enterprise Linux 4: Manual de referencia. [En línea] 2012. [Citado el: 26 de Noviembre de 2012.] http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ftp.html.
- **Nava, Maily. 2013.** Scribd. Arquitectura de Softwarw. [En línea] 2013. [Citado el: 6 de Mayo de 2013.] http://es.scribd.com/doc/23161581/Estilos-Arquitectonico.
- **Reynoso, Carlos y Kicillof, Nicolás. 2004.** Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. 2004.
- **Pressman, Roger S. 2005.** Ingeniería del Software.Un enfoque práctico. 2005. ISBN:9701054733.
- **Sande, Martín. 2004.** Programacion en C++ con Qt bajo Entorno GNU/Linux. Argentina : s.n., 2004. Linux User #281622.
- **Sharpe, Richard. 2002.** Just what is SMB? [En línea] 2002. [Citado el: 28 de Noviembre de 2012.] http://www.samba.org/cifs/docs/what-is-smb.html.
- **Sierra, Daniel. 2012.** Slide Share. Visual Paradigm For Uml. [En línea] 2012. [Citado el: 4 de Deciembre de 2012.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.
- **Visual Paradingm. 2007.** Visual Paradingm. [En línea] 2007. [Citado el: 2 de Diciembre de 2012.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.
- **Webnova. 2011.** Web Services - XML-RPC, SOAP, sobre PHP, Perl, y otros conceptos. [En línea] 2011. [Citado el: 6 de Febrero de 2013.] http://webnova.com.ar/articulo.php?recurso=426.
- **zonaqt. 2012.** Zona Qt. [En línea] 2012. [Citado el: 8 de Febrero de 2013.] www.zonaqt.com/.

ANEXOS

Anexo 1 Descripción de los Casos de Uso

Tabla 8: Descripción del Caso de Uso Mostrar medias compartidas.

Objetivo	Mostar Recursos Compartidos.	
Actores	Usuario (Inicia).	
Resumen	El caso de uso inicia cuando el Usuario ejecuta la herramienta y termina cuando se visualizan en la tabla los recursos compartidos.	
Complejidad	Media.	
Prioridad	Baja.	
Precondiciones	Para mostrar los recursos compartidos debe de existir al menos un recurso compartido.	
Postcondiciones	Se mostró de forma satisfactoria la operación realizada.	
Flujo de eventos		
Flujo básico: Mostar Recursos Compartidos.		
	Actor	**Sistema**
1.	El caso de uso se inicializa cuando se ejecuta el sistema.	
2.		Busca los recursos que se encuentran compartidos.
3.		Muestra en una tabla (A) los recursos compartidos con los siguientes datos: nombre del recurso, directorio, comentario, permisos y visibilidad, y finaliza el CU.
Prototipo de Interfaz		



Relaciones	**CU Incluidos**	Ninguno.
	CU Extendidos	Editar los recursos que se muestran en la tabla en el CU. Editar Recursos Compartidos.
Requisitos no funcionales	Ninguno.	
Asuntos pendientes	No procede.	

Tabla 9: Descripción del Caso de Uso Editar Recursos Compartidos.

Objetivo	Editar Recursos Compartidos.
Actores	Usuario (Inicia).
Resumen	El caso de uso inicia cuando el Usuario selecciona de la tabla un recurso y selecciona la opción Editar Recursos Compartido y termina cuando se muestra un mensaje.
Complejidad	Media.
Prioridad	Baja.
Precondiciones	Para editar un recurso compartido debe de existir al menos un recurso compartido.

Postcondiciones	Se mostró de forma satisfactoria la operación realizada.	
Flujo de eventos		
Flujo básico: Editar Recursos Compartidos.		
	Actor	**Sistema**
1.	Selecciona el recurso que desea editar de la tabla.	
2.	Selecciona la opción Editar Recurso Compartido. (A)	
3.		Muestra un formulario con los datos para editar: -Directorio del recurso. -Nombre. -Comentario. -Usuario válido. -Escritura/Lectura. -Visibilidad.
4.	Modifica los datos que desee y selecciona la opción Aceptar. (B)	
5.		Valida que los datos sean correctos.
6.		Modifica los datos y se envía un mensaje de que la acción se ha realizado con éxito y finaliza el caso de uso.
Prototipo de Interfaz		

Flujo Alterno

5 Evento. Los datos introducidos son incorrectos.

	Actor	**Sistema**
2.1	Seleccionar la opción Cancelar. (C)	
2.1.1		Cierra el formulario y finaliza el caso de uso.

5.1		Si los datos no son correctos se lanza un mensaje especificando que existe un error.
5.1.1	Corrige el error y continúa a partir del paso 4 del flujo básico.	

Relaciones	**CU Incluidos**	Ninguno.
	CU Extendidos	Ninguno.
Requisitos no funcionales	Ninguno.	
Asuntos pendientes	No procede.	

Anexo 2: Diagrama de clases del Diseño.

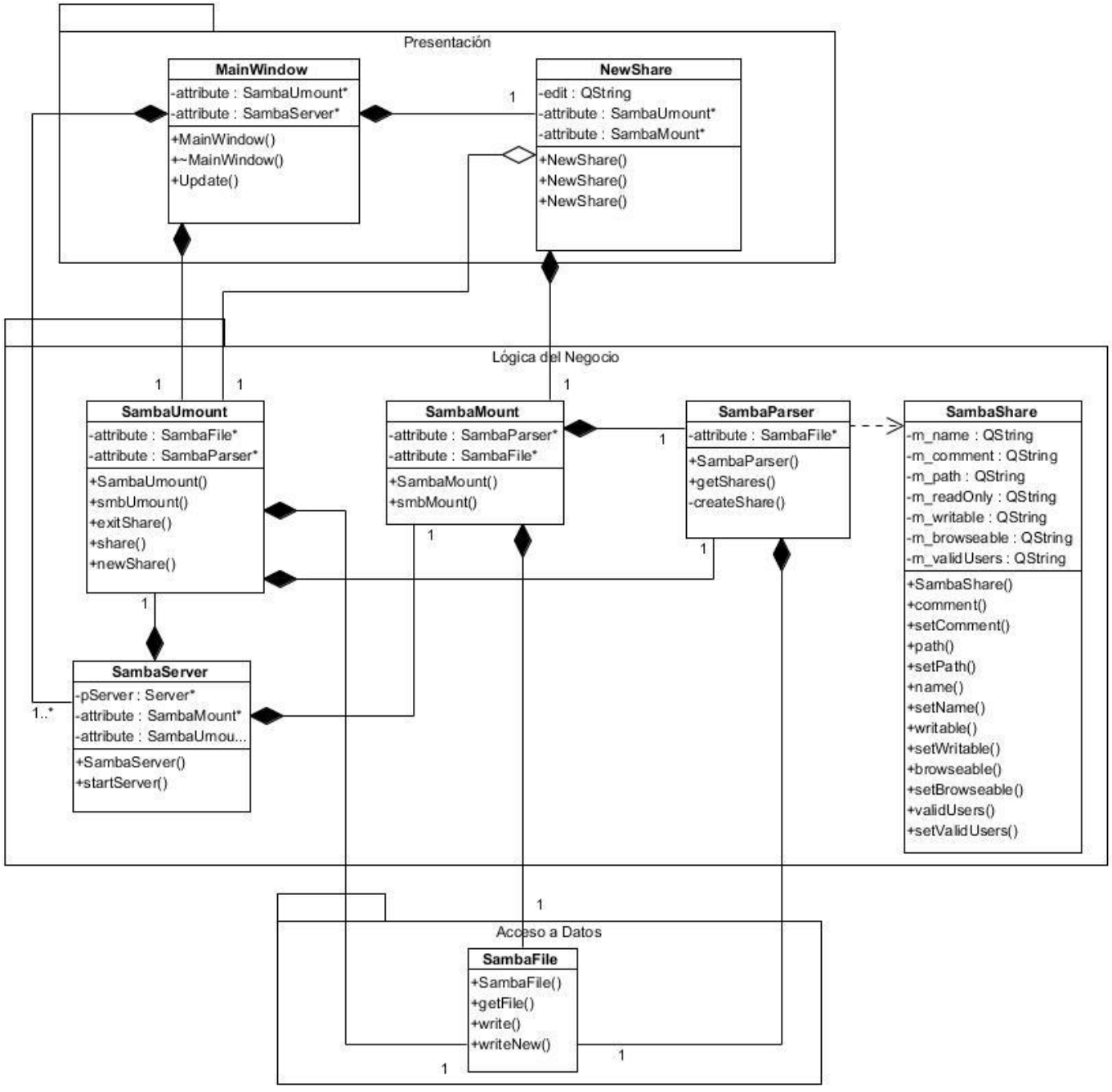


Ilustración 15: Diagrama de Clases del Diseño integro.

Anexo 3: Diagramas de secuencia

Diagrama de secuencia para el caso de uso Editar Recursos Remotos:

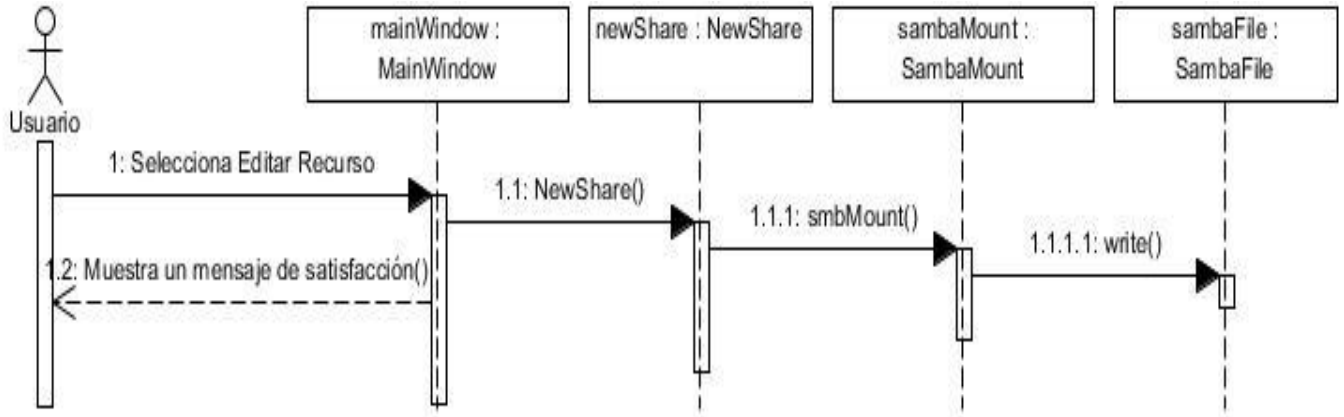


Ilustración 16: Diagrama de secuencia para el Flujo Normal del CU Editar Recursos Remotos.

Diagrama de secuencia para el caso de uso Mostrar Recursos Remotos:

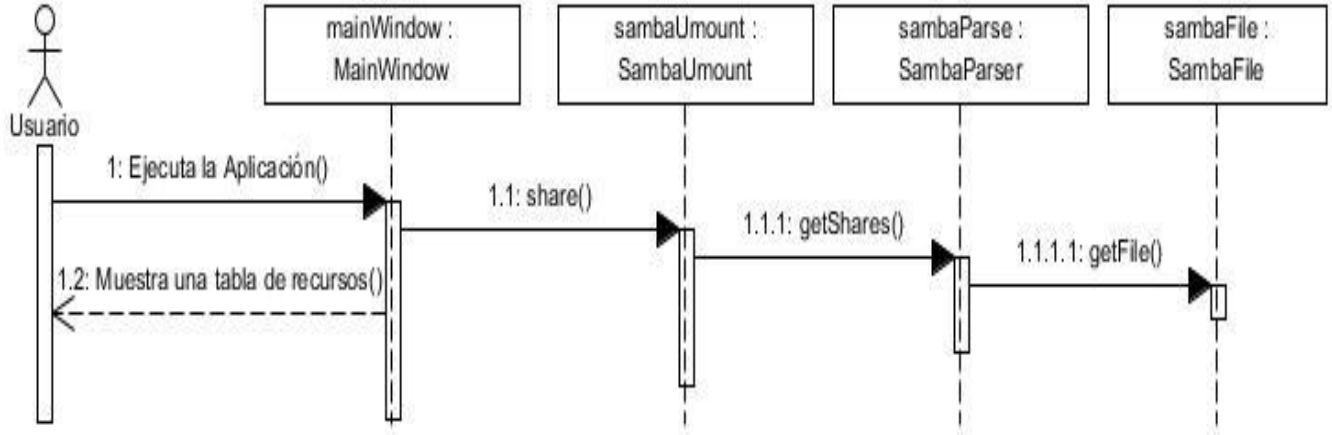


Ilustración 17: Diagrama de secuencia para el Flujo Normal del CU Mostrar Recursos Remotos.

Anexo 4: Descripción de los casos de prueba

- Caso de uso “Administrar Recursos Remoto”

Secciones a probar en el CU Administrar Recursos Remoto, para el desarrollo de estas pruebas y poder comprobar que el sistema cumple con las funcionalidades de compartir y dejar de compartir recursos de forma remota se desarrollo una aplicación cliente. Esta cuenta con un formulario con los datos necesarios para realizar la operación requerida y con dos botones uno llamado Compartir y el otro Descompartir que al dar clic en uno de ellos se envía la información a la herramienta. Para poder lograr la comunicación entre las dos herramienta se utilizó el protocolo de comunicación xmlrpc en especifico la biblioteca libqxmlrpc.

Tabla 10: Secciones a probar en el CU Administrar Recursos Remoto.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo central**
SC 1: Compartir Recursos Remoto	EC 1.1: Compartir Recursos Remoto con éxito.	El Gestor inserta los datos en el formulario y da clic en el botón Compartir y el sistema muestra un mensaje de que la operación se ha realizado con éxito.	botón “Compartir”
	EC 1.2: Compartir Recursos Remoto con error.	El Gestor inserta los datos en el formulario y da clic en el botón Compartir y el sistema muestra un mensaje de error que informa que los datos enviados son incorrectos.	botón “Compartir”.
SC 2: Dejar de Compartir Recursos Remoto	EC 1.1: Dejar de Compartir Recursos Remoto con éxito.	El Gestor inserta el nombre del recurso que desea dejar de compartir en el formulario y da clic en el botón Descompartir y el sistema muestra un mensaje de que la operación se ha realizado	botón “Descompartir”.

		con éxito.	
	EC 1.2: Dejar de Compartir Recursos Remoto con error.	El Gestor inserta el nombre del recurso que desea dejar de compartir en el formulario y da clic en el botón Descompartir y el sistema muestra un mensaje de error informando que el recurso no existe.	botón “Descompartir”.

Tabla 11: Descripción de variables.

No	**Nombre de campo**	**Clasificación**	**Valor Nulo**	**Descripción**
1	Directorio del Recurso	Campo de texto	No	Directorio del recurso que se desea compartir.
2	Nombre	Campo de texto	No	Nombre del recurso a compartir o des compartir.
3	Comentario	Campo de texto	Si	Un comentario que se le puedo poner o no al recurso.
4	Usuario Válido	Campo de texto	No	Es el usuario que va a tener permiso para acceder al recurso compartido.
5	Permisos	Campo chequeable	Si	Si el recurso va a tener permisos de lectura y escritura o solo de lectura.
6	Visibilidad	Campo chequeable	Si	Si el recurso va a estar visible o no.

Matriz de Datos.

Tabla 12: SC 1 Compartir Recursos Remoto.

ID del	**Escenario**	**Respuesta del sistema**	**Resultado de**

escenario			**la prueba**
EC 1.1	Compartir Recursos Remoto con éxito.	El sistema muestra un mensaje de que la operación a ha realizado con éxito.	Satisfactorio
EC 1.2	Compartir Recursos Remoto con error.	El sistema muestra un mensaje de error que alerta que existen campos vacíos o con datos incorrectos.	Satisfactorio

ID del escenario	**Directorio del recurso**	**Nombre**	**Comentario**	**Usuario Válido**	**Permisos**	**Visibilidad**
EC 1.1	V/ “/media/datos”	V/ “datos”	V/ “Carpeta”	V/ “Administrador”	V/ “chequeado”	V/ “chequeado”
EC 1.2	I/ “/no/existe”	V/ “prueba”	V/ “”	V/ “Administrador”	V/ “no chequeado”	V/ “chequeado”
	V/ “/media/datos”	I/ “campo vacío”	V/ “Datos”	V/ “Administrador”	V/ “chequeado”	V/ “chequeado”
	V/ “/media/datos”	V/ “prueba”	V/ “Datos”	I/ “campo vacío”	V/ “chequeado”	V/ “chequeado”
	V/ “/media/datos”	I/ “campo vacío”	V/ “Datos”	V/ “Administrador”	V/ “no chequeado”	V/ “no chequeado”
	I/ “campo vacío”	V/ “prueba”	V/ “Datos”	V/ “Administrador”	V/ “chequeado”	V/ “no chequeado”
	I/ “campo vacío”	I/ “campo vacio”	V/ “”	I/ “campo vacio”	V/ “chequeado”	V/ “no chequeado”

Tabla 13: SC 2 Dejar de Compartir Recursos Remoto.

ID del	**Escenario**	**Nombre**	**Respuesta del**	**Resultado de**

escenario			**sistema**	**la prueba**
EC 1.1	Dejar de Compartir Recursos Remoto con éxito.	V/ “Nombre”	El sistema muestra un mensaje de que la operación a ha realizado con éxito.	Satisfactorio
EC 1.2	Dejar de Compartir Recursos Remoto con error.	I/ “nombre incorrecto” I/ “campo vacío”	El sistema muestra un mensaje de error que alerta que existen campos vacíos o que el recurso no existe.	Satisfactorio

- Caso de uso “Editar Recursos Compartidos”

Tabla 14: Secciones a probar en el CU Editar Recursos Compartidos.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo central**
SC 1: Editar Recursos Compartidos	EC 1.1: Editar Recursos Compartidos con éxito.	El usuario selecciona el recurso que desea editar en la tabla y después la opción Editar Recurso Compartido, modifica los datos en el formulario que muestra el sistema y da clic en el botón Aceptar y el sistema muestra un mensaje de que la operación se ha realizado con éxito.	Seleccionar recurso de la tabla/ Opción Editar Recurso Compartido / botón “Aceptar”.
	EC 1.2: Editar Recursos Compartidos con	El usuario selecciona el recurso que desea editar en la tabla y después la opción Editar	Seleccionar recurso de la tabla/ Opción Editar Recurso Compartido / botón “Aceptar”.

	error.	Recurso Compartido, modifica los datos en el formulario que se muestra y da clic en el botón Aceptar y el sistema muestra un mensaje de error que informa que los datos son incorrecto o existen campos vacíos.	
	EC 1.3: Editar Recursos Compartidos Cancelar	El usuario selecciona el recurso que desea editar en la tabla y después la opción Editar Recurso Compartido y selecciona la opción Cancelar del formulario, el sistema cierra el formulario y regresa al anterior.	Seleccionar recurso de la tabla/ Opción Editar Recurso Compartido / botón "Cancelar".

Tabla 15: Descripción de variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Directorio del Recurso	Campo de texto	No	Directorio del recurso que se desea editar.
2	Nombre	Campo de texto	No	Nombre del recurso a editar.
3	Comentario	Campo de texto	Si	Un comentario que se le puedo poner o no al recurso.
4	Usuario Válido	Campo de texto	No	Es el usuario que va a tener permiso para acceder al recurso compartido.
5	Permisos	Campo chequeable	Si	Si el recurso va a tener permisos de lectura y escritura o solo de lectura.

6	Visibilidad	Campo chequeable	Si	Si el recurso va a estar visible o no.

Matriz de Datos.

Tabla 16: SC 1 Editar Recursos Compartido.

ID del escenario	**Escenario**	**Respuesta del sistema**	**Resultado de la prueba**
EC 1.1	Editar Recursos Compartido con éxito.	El sistema muestra un mensaje de que la operación a ha realizado con éxito.	Satisfactorio
EC 1.2	Editar Recursos Compartido con error.	El sistema muestra un mensaje de error que alerta que existen campos vacíos o con datos incorrectos.	Satisfactorio
EC 1.3	Editar Recursos Compartido Cancelar.	El sistema cierra el formulario y regresa al anterior.	Satisfactorio

ID del escenario	**Directorio del recurso**	**Nombre**	**Comentario**	**Usuario Válido**	**Permisos**	**Visibilidad**
EC 1.1	V/ “/media/datos”	V/ “datos”	V/ “Carpeta”	V/ “Administrador”	V/ “chequeado”	V/ “chequeado”
EC 1.2	I/ “/no/existe”	V/ “prueba”	V/ “”	V/ “Administrador”	V/ “no chequeado”	V/ “chequeado”
	V/ “/media/datos”	I/ “campo vacío”	V/ “Datos”	V/ “Administrador”	V/ “chequeado”	V/ “chequeado”
	V/	V/ “prueba”	V/ “Datos”	I/ “campo	V/ “chequeado”	V/

	“/media/datos”			vacío”		“chequeado”
	V/ “/media/datos”	I/ “campo vacío”	V/ “Datos”	V/ “Administrador”	V/ “no chequeado”	V/ “no chequeado”
	I/ “campo vacío”	V/ “prueba”	V/ “Datos”	V/ “Administrador”	V/ “chequeado”	V/ “no chequeado”
	I/ “campo vacío”	I/ “campo vacio”	V/ “”	I/ “campo vacio”	V/ “chequeado”	V/ “no chequeado”

- Caso de uso “Mostrar Recursos Compartidos”

Tabla 17: Secciones a probar en el CU Mostrar Recursos Compartidos.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo central**
SC 1: Mostrar Recursos Compartidos	EC 1.1: Mostrar Recursos Compartidos con éxito.	El usuario ejecuta la herramienta y la misma muestra una tabla con los recursos que se encuentran compartidos.	Ejecutar la herramienta.

Tabla 18: SC 1 Mostrar Recursos Compartido.

ID del escenario	**Escenario**	**Respuesta del sistema**	**Resultado de la prueba**
EC 1.1	Mostrar Recursos Compartido con éxito.	El sistema muestra una tabla con todos los recursos que se encuentran compartidos.	Satisfactorio

GLOSARIO

Términos	Definiciones
IP	Protocolo de Internet (Internet Protocol), es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados no fiable de mejor entrega posible sin garantías.
CASE	Ingeniería de Software Asistida por Ordenador (Computer Aided Software Engineering), Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.
UML	Lenguaje Unificado de Modelado (Unified Modeling Language), es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.
IDE	Entorno Integrado de Desarrollo (Integrated Develop Environment), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.
RUP	Proceso Unificado Racional (Rational Unified Process), proceso de desarrollo de software, es la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.
midleware	El middleware es un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.