# Universidad de las Ciencias Informáticas

Facultad 9

# Título: Gestor de ficheros para el Entorno de Escritorio Quántico

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

**Autor:** Reynier Pupo Gómez

**Tutor:** Ing. Jorge L. Rodríguez Carpio

Ciudad de la Habana, Cuba
Año del 51 aniversario de la Revolución

## 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 _____.

Autor

_____

Reynier Pupo Gómez

Tutor

_____

Ing. Jorge L. Rodríguez Carpio

*"Si no existe la organización, las ideas, después del primer momento de impulso, van perdiendo eficacia."*

Ernesto Che Guevara

# Observaciones introductorias

Se recomienda consultar el glosario de términos antes de leer el texto principal para evitar malentendidos o confusiones.

# Datos de contacto

**Ing. Jorge** Luis Rodríguez Carpio .
**Contacto:** jlrodriguez@uci.cu

# Agradecimientos

A todos los que han contribuido a mi formación profesional y personal, especialmente a:

Mis padres Dania Gómez Benítez y Raúl Pupo Martínez por constancia y dedicación incondicional.

A mi hermano Rayner por impulsarme a la constante preparación y por su apoyo en todo momento.

A mis abuelos Edilia y Pedro por ser los mejores del mundo.

A Frank por poder contar siempre con él.

A mi novia Yanisley por brindarme tanto apoyo y confianza.

A todos mis amigos en especial a Yuniesky por su ayuda incondicional.

A mi tutor Carpio.

A todos los que han hecho posible que hoy exprese estas líneas de agradecimiento en el trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

# Dedicatoria

Dedico este trabajo a todos los que han hecho posible que esté ahora realizándolo, a todas las personas que han contribuido a mi formación profesional, a los que desde temprano me enseñaron a pensar y a las que hacen mi vida productiva.

# Resumen

Como parte de una estrategia para ganar en rendimiento al trabajar con el sistema operativo de las FAR se creó un proyecto denominado Quántico, el cual consiste en un entorno de escritorio ligero, que precisamente se encuentra dirigido a ese objetivo. Hasta el momento este solo contiene funcionalidades básicas de representación de un escritorio, manejo de ventanas(Window Manager), reconocimiento de tipos y algunas más por lo que es muy poco utilizable. Actualmente es imposible dentro de Quántico explorar carpetas y archivos, siendo esta una de las funcionalidades más importantes de todo entorno de escritorio dándole posibilidad a los usuarios de trabajar con sus documentos, información personal y todo tipo de datos. El objetivo de esta tesis es tanto el diseño como la implementación de una herramienta que solucione la falta de funcionalidades en cuanto al manejo de los recursos almacenados dentro de las computadoras que cumpla con los estándares de desarrollo impuestos para proyecto.

**Palabras claves:** Quántico, Entorno de Escritorio, Gestor de Ficheros

# Índice general

# Introducción

Como parte de la estrategia de migración a Software Libre dentro del país y específicamente en las FAR se han venido sentando las bases para un uso extensivo de Sistemas Operativos GNU/Linux, entre los cuáles se encuentra una personalización de la distribución Nova creada en nuestra Universidad. A pesar de ser las FAR una de las instituciones nacionales con mayor apoyo tecnológico, todavía se encuentran en explotación computadoras con pocos recursos de hardware e incluso es llevada a cabo una idea de realizar el trabajo de varios centros utilizando la tecnología de Clientes Ligeros y máquinas sin disco. Por tanto se hace necesario un aprovechamiento al máximo de los recursos computacionales actuales, por lo cual surge un proyecto que va dirigido a la creación de un Entorno de Escritorio ligero: Quántico, el cual persigue ese objetivo. Este como principales características tiene que depende solamente de las bibliotecas del marco de trabajo de Qt y desarrollado en el lenguaje de programación C++, por lo que se decidió que cada una de sus aplicaciones siguieran sus mismas características en aras de lograr en uniformidad, baja dependencia y ligereza.

Una de las funcionalidades de todo entorno de escritorio debe ser la capacidad de navegar por el Sistema de Ficheros de la computadora, así como proveer utilidades para el trabajo con cada uno de los archivos y carpetas, como son copiar, cortar, pegar, crear, eliminar y renombrar, todo esto a través de una interfaz gráfica que facilite el trabajo. Quántico actualmente no cuenta con estas funcionalidades. Por lo que surge el siguiente problema a resolver: **¿Cómo gestionar los archivos y carpetas en el Entorno de Escritorio** Quántico**?**

Mediante la problemática planteada anteriormente podemos establecer el Objeto de Estudio de la presente investigación, el cual se centra en herramientas de trabajo con ficheros, enmarcándose en el Campo de Acción de las aplicaciones que gestionan ficheros en el sistema operativo GNU/Linux. Defendiendo por lo tanto la idea de que al desarrollar un gestor de ficheros basado completamente en Qt se podrán manejar los ficheros en Quántico, manteniendo los principios del mismo. Planteada la idea a defender del presente trabajo, se pretende como Objetivo General desarrollar la herramienta de explorar ficheros para el entorno de escritorio Quántico .

Para llevar a cabo el objetivo general se requieren los siguientes Objetivos Específicos:

1. Definir herramientas, tecnologías y software a utilizar, para dar soporte a los procesos de modelado e implementación.

   *a*) Tarea 1: Realizar estudio del estado del arte de Entornos de Desarrollo Integrados que se ajusten a las características del proyecto.

   *b*) Realizar estudio del estado del arte de herramientas de modelado que se ajusten al proceso de desarrollo del centro UCID.

   *c*) Realizar estudio del estado del arte de lenguajes y marcos de trabajo con características favorables para

el desarrollo del proyecto.

2. Realizar el diseño de la arquitectura de un manejador de ficheros orientado a componentes.
3. Implementar y probar el módulo cargador de componentes.
    *a*) Estudiar y caracterizar las librerías del Framework Qt para el trabajo con plugin.
4. Implementar y probar los componentes esenciales para el trabajo con el sistema de archivos.
    *a*) Estudiar y caracterizar las librerías del Framework Qt para el trabajo con el sistema de archivos.
5. Integrar el cargador de componentes con el conjunto básico de componentes.
6. Realizar pruebas al producto terminado.

Entre los métodos científicos a utilizar se encuentran:

- Teóricos:
    - Analítico - Sintético: Utilizado al estudiar de la documentación especializada permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio permitiendo arribar a las conclusiones de la investigación, así como determinar las características de los requisitos a implementar.
    - Modelación: Para observar la organización e interacción de los elementos del análisis y diseño de la solución propuesta.
- Empíricos:
    - Entrevista: Utilizado en dos etapas, la primera para la captura de requisitos y la segunda para el análisis de la aceptación del producto por parte de los usuarios.

Este trabajo tendrá la siguiente estructura:

- En el Capítulo 1 "Fundamento teórico", se hará una breve revisión de los Gestores de Archivos, especialmente los de la plataforma GNU/Linux. Se expondrán características de Quántico. Se abordará sobre el marco de trabajo Qt. Serán clasificadas las posibles herramientas de desarrollo así como especificaciones del proceso de desarrollo.
- En el Capítulo 2 "Características del sistema", se realizará el modelado del dominio con el objetivo de comprender el contexto del sistema; se hará una propuesta, describiendo cómo debe funcionar y destacando sus características distintivas; se especificarán sus Requisitos Funcionales y No Funcionales.

- En el Capítulo 3 "Diseño del Sistema", se presentarán: Diagramas de Secuencia, evidenciando el orden de las acciones en los procesos del negocio cuando son invocados, de sus Clases del Diseño se representará el diagrama y una descripción detallada del mismo. Además, se mostrarán los patrones utilizados para su Diseño y las técnicas que facilitarán documentarlo para su uso y tratar posibles errores en tiempo de ejecución.
- En el Capítulo 4 "Implementación y pruebas", se plasmarán los artefactos resultantes de la implementación así como los diseños de casos de prueba y los resultados de su aplicación.

Finalmente se presentan las Conclusiones, Recomendaciones, Referencias, Bibliografía, Anexos y el Glosario de Términos.

Capítulo 1

# Fundamento teórico.

En este capítulo se expresan los conceptos fundamentales de los temas entre los cuales se van a estar tratando a lo largo de este trabajo así como características y estados actuales de desarrollo. Luego se realizarán las propuestas de herramientas y tecnologías a utilizar para soportar el proceso de modelación e implementación.

## 1.1. Generalidades de un Entorno de Escritorio.

Un entorno de escritorio no es más que una interfaz GUI que está diseñada con la finalidad de ofrecer una ayuda al usuario para el trabajo con el sistema operativo, ya sea en tareas de configuración, modificación o manejo de ventanas. Dicho sistema operativo es a su vez un conjunto de programas que se encuentran ejecutándose dentro de un ordenador con tareas de controlar el hardware como dispositivos de almacenamiento o de entrada/salida y manejar los procesos que se ejecutan en el microprocesador. El esquema que se representa en la figura 1.1 denota la relación de estos componentes.

Figura 1.1: Esquema relación computadora-sistema operativo-entorno de escritorio.

### 1.1.1. Entorno de Escritorio Antico .

La primera impresión que se lleva un usuario de determinado sistema operativo depende en gran parte de las características de usabilidad, rendimiento, apariencia y facilidades de configuración. Con el objetivo de lograr un entorno de escritorio que cumpliera con las expectativas de los usuarios fue creado Antico. Iniciado como un proyecto

independiente la mejor característica fue la de ser ligero al desligarse de las tradicionales kde-libs[1] y no utilizar o depender de un gran número de librerías externas.

### 1.1.2. Entorno de Escritorio Quántico.

Quántico es el término que se ha elegido para una personalización o rama de Antico, el cual fue abandonado por el equipo de desarrollo original. Como principales características cuenta con gran rapidez, muy poco consumo de memoria física, toda la configuración se realiza mediante ficheros de configuración y al utilizar solo del marco de trabajo Qt y sus dependencias lo hace bastante fácil en tiempo de desarrollo. Debido al poco tiempo de vida del proyecto hay una gran carencia de utilidades aún y el desarrollo se basa más bien en la corrección de errores dándole más peso que a la implementación de nuevas funcionalidades.

## 1.2. Generalidades de un manejador de ficheros.

Un manejador de ficheros en un programa que provee una interfaz para el trabajo con sistemas de ficheros, entre las acciones más comunes se encuentra crear nuevo fichero o archivo, copiar, pegar, eliminar, cortar, renombrar y otros. Además algunos cuentan la con la capacidad de trabajar con ficheros remotos utilizando los distintos protocolos de red.

### 1.2.1. Ejemplo de gestores de ficheros para GNU/Linux

#### 1.2.1.1. Basados en modo texto.

- Midnight Commander: Es un gestor de ficheros visual, porta la licencia GPL por lo cual es considerado software libre. Es una aplicación de pantalla completa a modo texto que permite entre otras cosas copiar, mover, eliminar ficheros o un árbol completo de directorios, buscar ficheros y ejecutar aplicaciones. Incluye además un visor y editor interno. Midnight Commander está basado en las interfaces de modo texto, como es Ncurses[2] o S-Lang, los cuales permiten trabajar directamente en una consola, dentro de una terminal X Window[3], sobre una conexión SSH[4] o sobre cualquier tipo de consola remota.[5]

- Vifm: Es un gestor de ficheros basado en Ncurses con el estilo de accesos de teclado de vi. Si se ha usado vi, Vifm proporciona un acceso completo a los ficheros mediante el teclado sin tener que aprender nuevos comandos. [6]

[1] Librerías del entorno de escritorio KDE, en su mayor parte componentes llamados widgets basados en Qt.
[2] Ncurses es una biblioteca de programación que provee una API que permite al programador escribir interfaces basadas en texto.
[3] El sistema X Window es un método gráfico y distribuido para trabajar.
[4] Es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente.[1]

#### 1.2.1.2. Con interfaz gráfica no basados en Qt.

- X File Explorer (Xfe): Es un explorador de ficheros estilo Commander para el sistema X. Está basado en el popular pero descontinuado proyecto X Win Commander. Como principal característica tiene que está desarrollado con el marco de trabajo Fox lo cual le ayuda en una gran rapidez y baja carga de los recursos del sistema. No cuenta con facilidades de conexiones a sistemas de directorios remotos.

- Gentoo: Es un gestor de ficheros basado en GTK[5], está escrito en el lenguaje de programación C, es configurable directamente con la interfaz, reconoce a los ficheros por su tipo, permite realizar cualquier operación de archivos y carpetas, entre otras funcionalidades.

- Gnome Commander: Es un gestor de ficheros de dos paneles que sigue la tradición del Midnight Commander, escrito utilizando el marco de trabajo GTK y GnomeVFS. Además cuenta con el reconocimiento de los MIME-Types[6] de Gnome, acceso a conexión Samba y ftp, auto-montaje de particiones y buscador.

- Krusader: Es un gestor de ficheros integrado a KDE con grandes potencialidades. Presenta una interfaz basada en dos paneles. Entre sus características están las conexiones que soportan samba y ftp[7], emulador de consola, manejador de montaje, sincronización de directorios, consola de java script y muchos más.

#### 1.2.1.3. Con interfaz gráfica basados en Qt.

- OrganizasyonizM: Es un gestor de ficheros dirigido particularmente a usuarios que desean organizar su música pues cuenta con un conjunto de funcionalidades como etiquetador id3[8], organización de las carpetas con mp3 y reproductor de audio incluido.

- PyDingo: Es una aplicación multipropósito creada con PyQt4. Presenta un sistema de plugins o agregados que manejan URLs permitiendo crear componentes para reproducir música, editar documentos, buscar ficheros y cualquier otro que se desee agregar.

- QtCommander: Gestor de ficheros de dos paneles con funcionalidades de acceso ftp, reconocimiento de MIME-TYPES, buscador interno y otras.

### 1.2.2. Resumen de Gestores Analizados.

Dentro de los ejemplos analizados se encuentran un conjunto de características o funcionalidades que deben estar presentes en todo gestor de ficheros pero las debilidades de unos son las fortalezas de otros, por ejemplo es el caso

[5] Graphics Tool Kit, biblioteca para crear interfaces gráficas de usuario.
[6] Tipo de un archivo dependiendo de su función.
[7] Protocolo de Transferencia de Ficheros.
[8] Estándar para incluir metadatos en archivos de medias.

de Krusader cuya debilidad es que depende de las librerías de KDE(recordar que este trabajo se encuentra dirigido al marco de trabajo Qt),sin embargo uno con menos funcionalidades como el QtCommander solo depende de Qt. Luego hay casos que dependen de Qt solamente pero están escritos en Python y eso es una debilidad mirado desde el punto de vista de rendimiento pues una aplicación creada en ese lenguaje consume más recursos que una escrita en C++, lenguaje de desarrollo de Quántico.

## 1.3. Marco de trabajo Qt.

Qt es marco de trabajo multiplataforma para el desarrollo de aplicaciones e interfaces de usuario. Incluye librerías multiplataforma, herramientas de desarrollo integradas y un IDE multiplataforma. Presenta un gran número características entre las que se destacan las siguientes:

- Es una intuitiva librería de C++,
- Garantiza la portabilidad entre sistemas operativos embebidos y de escritorio
- Presenta herramientas de desarrollo integrado con IDE multiplataforma
- Alto desempeño en dispositivos embebidos.

Entre las posibilidades que brinda este marco de trabajo están la del trabajo con hilos independientemente del sistema operativo, contiene un módulo para el trabajo con protocolos de red, posee soporte para aplicaciones orientadas a componentes, trabajo con los gestores de bases de datos más conocidos además de poder implementar soporte para otros y muchas posibilidades más que hacen de Qt altamente aplicable a cualquier aplicación de escritorio sin tener que utilizar librerías externas. Hasta el momento ha sido liberado bajo dos licencias, la LGPL y la Comercial[9], a continuación se brinda una tabla con las características de ambas:

| | LGPL | Comercial |
|---|---|---|
| Pago por licencias de desarrollo. | | x |
| Los cambios al código de Qt deben ser compartidos. | x | |
| Se puede crear aplicaciones propietarias. | x | x |
| Disponibilidad de soporte técnico. | x | x |
| Mantener las opciones de la licencia de la distribución abiertas. | x | |

Para este gestor de ficheros se ha elegido trabajar con la versión LGPL[10] de Qt puesto que provee un mayor intercambio con la comunidad de clientes y desarrolladores y además que de no ser así hubiera que pagar una

[9] En las primeras liberaciones de Qt solo existía la versión comercial, luego se libera una versión en licencia GPL lo que convierte a Qt en un proyecto de amplio crecimiento.

[10] Licencia Pública General Reducida, otorga ciertas libertades extras a la licencia GPL

licencia de desarrollo. Al Quántico ser un entorno de escritorio basado completamente en Qt se decidió por parte de el equipo de desarrollo del proyecto que todos los agregados o programas dependan solamente de Qt incrementando así el nivel de integración, la homogeneidad del código y disminución de riesgos.

### 1.3.1. Lenguajes de programación soportados por Qt.

Qt Jambi es una integración de Qt para JAVA en su versión 4.5.2 estable soportada por la compañía Nokia y por la comunidad de Qt. Junto con esta se adiciona un agregado al IDE Eclipse para su desarrollo.

Por otra parte PyQt es un binding de Qt para Python desarrollado por la compañía Riverbank, este se puede utilizar en todas las plataformas soportadas por Qt incluyendo también todos sus módulos.

Otro binding conocido es QtRuby, este aporta un gran número de posibilidades de Qt bajo código nativo en Ruby. Está mantenido por la comunidad de Ruby y dedicado mayormente al entorno de escritorio KDE.

Anteriormente han sido expuestos algunos de los bindings más utilizados de Qt, pero no son los únicos, también existe PHPQt para PHP, Qt# para C#, PySide para Python, QtAda para Ada, lqt para Lua y otros.

Debido a que es una necesidad del entorno de escritorio Quántico que sus aplicaciones sean lo más ligeras y con las menores dependencias posibles se ha elegido utilizar para la implementación del gestor de ficheros la versión nativa, o sea la que utiliza a C++ como lenguaje de programación puesto que este lenguaje cuenta con características de portabilidad, es multiplataforma, posee abundante documentación, mayor comunidad de desarrollo y además de que al ser este lenguaje bastante cercano al lenguaje nativo y poder trabajar incluso a bajo nivel permite un mayor rendimiento a cuanto a consumo de memoria como de aprovechamiento del procesador. Dentro de las características de C++ podemos distinguir:

- Alto nivel: Este tipo de lenguaje permite al programador abstraerse de la arquitectura y funcionamiento del hardware.
- Orientado a objetos: La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real. Además, permite la reutilización del código de una manera más lógica y productiva.
- Portabilidad: Un código escrito en C++ puede ser compilado en muchos sistemas operativos.
- Brevedad: El código escrito en C++ es muy corto en comparación con otros lenguajes, sobretodo porque en este lenguaje es preferible el uso de caracteres especiales que las palabras claves.
- Programación modular: Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos. Además, esta característica permite unir código en C++ con código producido en otros lenguajes de programación como Ensamblador o el propio C.

- Velocidad: El código resultante de una compilación en C++ es muy eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivela la vez y a la reducida medida del lenguaje.

### 1.3.2. Elección de Entorno de Desarrollo Integrado.

Para la elección de entorno de desarrollo integrado (IDE) se tuvo en cuenta que fuera una aplicación sobre software libre por las ventajas que conlleva. Entre estas se destacan:

- Evita la dependencia tecnológica de empresas foráneas.
- Ahorros por pagos de licencias de software.
- Posibilidad de revisar el código fuente.

Entre los IDE que existen para GNU/Linux para programar con Qt los más populares son:

- KDevelop: Surgió en 1998 con el fin de desarrollar un IDE fácil de usar para KDE (K Desktop Environment). Desde entonces está públicamente disponible bajo licencia GPL y soporta lenguajes de programación como: C, C++, Java, Ada, SQL, Python, Perl y Pascal. Solo existe en sistemas GNU/Linux y otros sistemas Unix. Su última versión es la 3.5.5 y salió el 5 de agosto del 2009. Tiene como limitantes principales que su entorno gráfico es algo pobre y sólo corre sobre plataforma GNU/Linux[3].
- Eclipse: Es un IDE multiplataforma desarrollado por IBM. En la actualidad lo mantiene la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos, capacidades y servicios complementarios. Pese a que esté escrito en su mayor parte en Java (salvo el núcleo), se ejecua sobre una máquina virtual de ésta y su uso más popular sea como un IDE para Java, Eclipse es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, etc. (Furmankiewicz, 2008). Sus mayores ventajas radican en su gran comunidad de desarrollo que lo ubican como el mejor IDE Java[4].
- QtCreator: Es un IDE creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas Qt, requiriendo su versión 4.x. Los sistemas operativos que soporta en forma oficial son: GNU/Linux 2.6.x, para versiones de 32 y 64 bits con Qt 4.x instalado. Además hay una versión para GNU/Linux con gcc 3.3. . Mac OS X 10.4 o superior, requiriendo Qt 4.x. Windows XP y Vista, requiriendo el compilador MinGW y Qt 4.4.3 para MinGW. Este IDE tiene la ventaja de ser soportado por la misma compañía que soporta a Qt, por lo que propicia una mayor integración y aumenta el número de funcionalidades.

Para la implementación del gestor de ficheros se ha elegido QtCreator porque cumple con muchas funcionalidades incluyendo soporte para sistemas de gestión de proyectos como Subversión y Git, depurador integrado, integración de ayuda sensible al contexto acerca de Qt, poderoso editor de código con múltiples funcionalidades, es mucho más

ligero que el Eclipse e integra todas las herramientas de Qt de forma que hace mucho más fácil la programación y la abstracción de los pasos de generación y compilación del código.

## 1.4. Proceso de desarrollo de software

Un proceso de desarrollo de software tiene como objetivo la producción eficiente de un producto de software que satisfaga los requisitos de un cliente con una planificación y una estimación de recursos predecibles. Los elementos de un proceso y sus relaciones deben responder Quién debe hacer Qué, Cuándo y Cómo. Esto se logra modelando las interacciones y relaciones que suceden entre las personas (roles), las actividades que estas desarrollan y los artefactos que se crean o actualizan durante el proceso.


Figura 1.2: Diagrama de proceso de desarrollo UCID.

### 1.4.1. Modelo de desarrollo de software.

El modelo de desarrollo de software propuesto describe la secuencia de actividades de alto nivel para la construcción y desarrollo de soluciones. Se logra con la combinación entre los modelos basado en Componentes, el Iterativo y el Incremental. Se emplearán las técnicas de prototipado, si son requeridas, para los requerimientos del usuario de los que no existe una visión clara por parte de estos, con el objetivo de desarrollar una definición mejorada de los requisitos del usuario para el sistema.

- Desarrollo iterativo e incremental: Es un enfoque en el que el ciclo de vida está compuesto por iteraciones, estas son pequeños procesos compuestos de varias actividades cuyo objetivo es entregar una parte del sistema parcialmente completo, probado, integrado y estable. Todo el software es integrado en cada entrega de cada

iteración hasta obtener el producto de software completo en la última iteración. En cada iteración se obtiene como resultado un incremento.

- Desarrollo basado en componentes: Nos lleva a alcanzar un mayor nivel de reutilización de software, aún en contextos distintos a aquellos para los que fue diseñado. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

*"Un componente es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio" [2]*

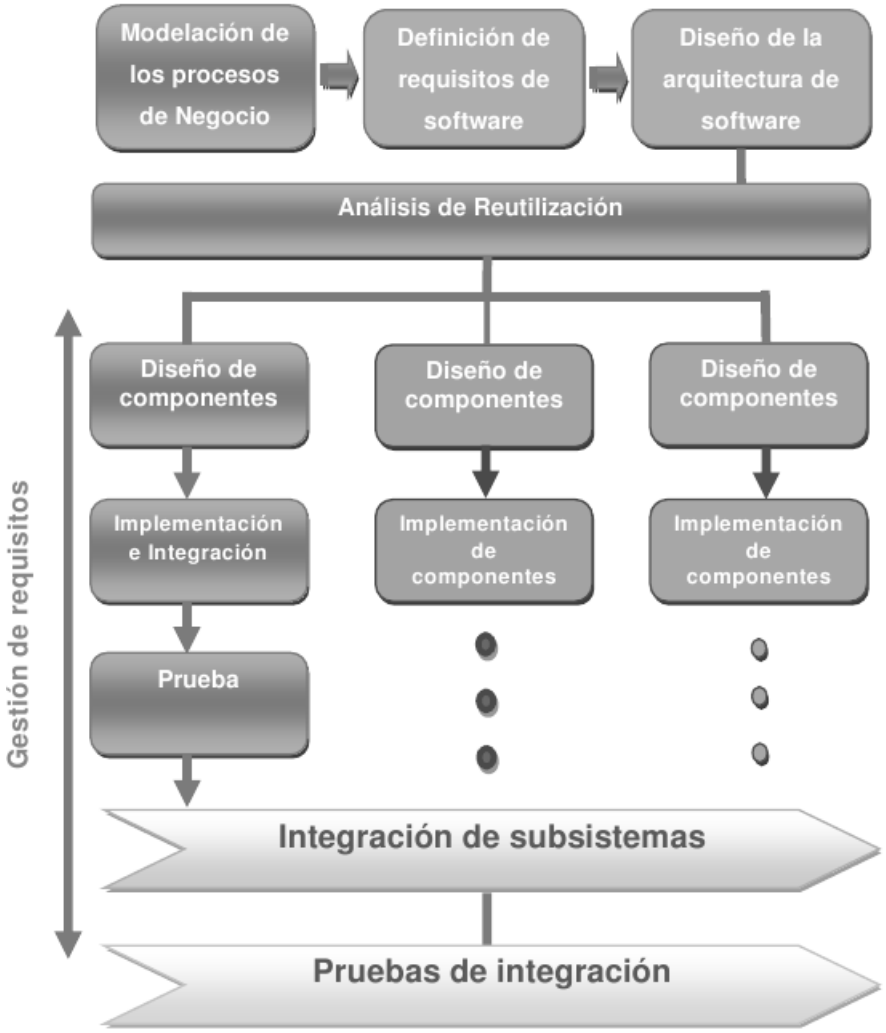


Figura 1.3: Modelo de Desarrollo de Software.

## 1.5. Lenguaje de modelado

El modelado es una parte central de todas las actividades que conducen a la producción de buen software, porque a través de él logramos comprender el sistema a desarrollar. El Lenguaje Unificado de Modelado (Unified Modeling Language, UML) es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

El Lenguaje Unificado de Modelado es el lenguaje que permite la modelación de sistemas de software con tecnología

orientada a objetos más conocido y utilizado en la actualidad. Es un lenguaje gráfico que cuenta con un grupo de diagramas, los cuales son utilizados para visualizar, especificar, construir y documentar un sistema de software en cada una de las etapas por las que tiene que pasar. UML indica que es lo que supuestamente hará el sistema, pero no como lo hará.

## 1.6. Herramienta CASE: Visual Paradigm

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son las aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas contribuyen de manera directa 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.

Para el modelado de los artefactos y diagramas generados a lo largo del ciclo de vida del proyecto se decidió emplear Visual Paradigm en su versión 3.4, pues su uso está muy estandarizado a nivel mundial y constituye una herramienta multiplataforma muy madura y acabada.

Visual Paradigm para UML es una herramienta profesional fácil de utilizar, que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una más rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar todos los tipos de diagramas de clases, permite la realización de ingeniería tanto directa como inversa, generar el código desde diagramas y generar la documentación automáticamente en varios formatos como Web o Formato de Documento Portable (pdf), permite el control de versiones. Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto. Proporciona también abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Además es robusta y portable.

Capítulo 2

# Características del sistema

Un entorno de escritorio sin un gestor de ficheros es muy poco funcional, debido a que no todos los usuarios dominan las formas de acceso a archivos y carpetas por líneas de comando mediante una consola o terminal, además un gestor de archivos permite realizar un conjunto de operaciones básicas en pro de ganar en tiempo y facilidad de uso.

En este capítulo se obtiene una visión más específica del sistema, donde se exponen conceptos específicos sobre los elementos que actuarán en el sistema, así como los requisitos funcionales y no funcionales que regirán el desarrollo de la solución al problema.

## 2.1. Análisis de la situación problemática.

Actualmente existen un gran número de aplicaciones que manejan ficheros, algunas cuentan con muy buenas características de rendimiento y otros de usabilidad. Sin embargo dada la premisa de que el Entorno de Escritorio Quántico como objetivo de desarrollo tiene el de crear todas sus aplicaciones utilizando el marco de trabajo Qt con el lenguaje de programación C++, se ve un poco reducido el espectro de los manejadores que serían útiles para este. También se encuentra el hecho de que algunas aplicaciones con las mejores características que están basadas en Qt y escritas con C++ también dependen de las bibliotecas de KDE, las cuales cargan demasiado los recursos de las computadoras.

## 2.2. Propuesta del Solución

Basado en las características propias de Quántico y las necesidades del mismo se propone el desarrollo de un gestor de archivos y carpetas con las características esenciales de acceso a los elementos del sistema de archivos que sea completamente dependiente de las librerías del propio entorno de escritorio. Además esta aplicación será basada en componentes lo que le permite la extensibilidad y el agregado de cualquier característica que se desee, esto será posible al gestor comportarse como un cargador de componentes completamente personalizable.

## 2.3. Conceptos del dominio del problema. Modelo Conceptual.

El propósito de esta actividad es de identificar y representar conceptos relacionados con el dominio del problema. Para ello se tiene como entrada el Glosario de Términos y se obtiene como salida el Modelo Conceptual. Este

Modelo Conceptual explica los conceptos significativos en el dominio del problema. Puede mostrarnos: conceptos, asociaciones entre conceptos y atributos de conceptos. Una cualidad esencial que debe ofrecer un Modelo Conceptual es que debe representar cosas del mundo real. En la figura 2.1 se muestra dicho mapa conceptual.

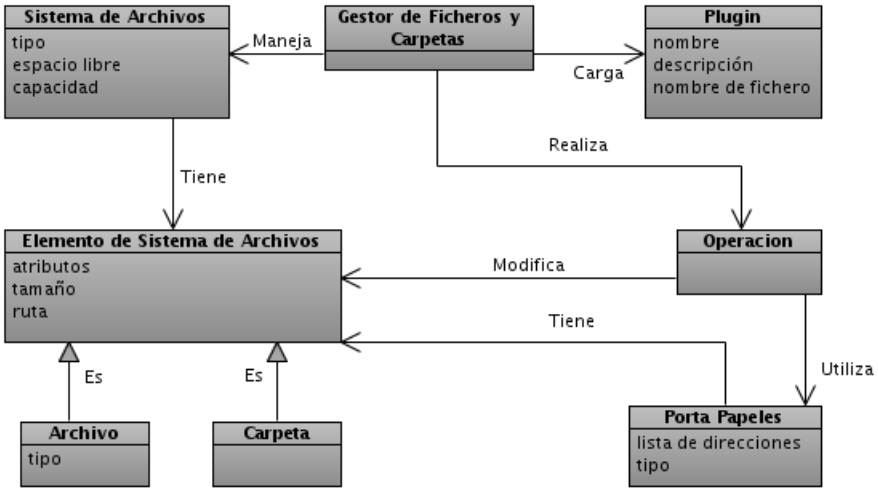


Figura 2.1: Modelo Conceptual.

### 2.3.1. Descripción de elementos del modelo conceptual

**Entidad Sistema de Archivos.**

| | |
|---|---|
| **Nombre de la entidad** | Sistema de Archivos |
| **Descripción de la entidad** | Es la forma en que se estructura o se guarda la información dentro de un dispositivo de almacenamiento determinado. |

| | | | | | **Restricciones** | |
|---|---|---|---|---|---|---|
| **Nombre del Atributo** | **Descripción** | **Tipo** | **¿Puede ser Nulo?** | **¿Es único?** | **Clases Válidas** | **Clases no Válidas** |
| tipo | Se refiere al tipo de sistema de archivos, por ejemplo EXT3, EXT4, NTFS u otros. | string | No | No | EXT3, EXT4, REISERFS, NTFS, FAT32 | |
| espacio libre | Dimensión en bytes que expresa la cantidad de espacio libre de determinado sistema de archivos. Es también expresable en forma de porciento. | int | No | No | Números Naturales | |
| capacidad | Se refiere a la cantidad total de información que puede ser almacenada en un sistema de archivos. | int | No | No | Números Naturales | |

**Entidad Plugin.**

| **Nombre de la entidad** | Plugin |
|---|---|
| **Descripción de la entidad** | No es más que un componente que se adhiere como funcionalidad extra a una aplicación. |

| | | | | | **Restricciones** | |
|---|---|---|---|---|---|---|
| **Nombre del Atributo** | **Descripción** | **Tipo** | **¿Puede ser Nulo?** | **¿Es único?** | **Clases Válidas** | **Clases no Válidas** |
| nombre | Se refiere al nombre por el que será conocido a nivel de usuario. | string | No | No | texto | |
| descripción | Define a forma de resumen la funcionalidad del plugin o algún dato extra. | string | Si | No | texto | |
| nombre del fichero | Nombre del fichero que proporciona el plugin. | string | No | Si | notación de librería *.so | |

**Entidad Elemento de Sistema de Archivos.**

| **Nombre de la entidad** | Elemento de Sistema de Archivos |
|---|---|
| **Descripción de la entidad** | Es una unidad dentro del sistema de archivos a la que le corresponde una entrada en la tabla de particiones. Puede comportarse como una carpeta o como un archivo. |

| | | | | | **Restricciones** | |
|---|---|---|---|---|---|---|
| **Nombre del Atributo** | **Descripción** | **Tipo** | **¿Puede ser Nulo?** | **¿Es único?** | **Clases Válidas** | **Clases no Válidas** |
| nombre | El identificador por el cual se distingue dentro de la carpeta contenedora. | string | No | No | Letras, Números, _, . | Caracteres especiales. |
| atributos | Determinadas propiedades que se le atribuyen a un elemento que determinan permisos, tiempo de acceso y tiempo de modificación. | Estructura | No | No | Clases de atributos | |
| tamaño | Es la capacidad que ocupa un elemento dentro del sistema de archivos. | int | No | No | Números Naturales | |
| ruta | Es la dirección que se le asigna al elemento dentro del sistema de archivos, su acceso depende del mismo y del sistema operativo. | string | No | No | Localizaciones o rutas. | |

**Entidad Archivo.**

| | |
|---|---|
| **Nombre de la entidad** | Archivo |
| **Descripción de la entidad** | Es un grupo de bits dentro de un sistema de archivos organizados de forma tal que tienen determinado significado en un conjunto. |

| | | | | | **Restricciones** | |
|---|---|---|---|---|---|---|
| **Nombre del Atributo** | **Descripción** | **Tipo** | **¿Puede ser Nulo?** | **¿Es único?** | **Clases Válidas** | **Clases no Válidas** |
| tipo | Este tipo se refiere a un rol en especial dentro de un sistema operativo que juegue esa clase de archivo, ya sea un archivo de audio, un binario ejecutable, un archivo de texto, etc. | string | Si | No | Tipos de Archivos Posibles | |

**Entidad Carpeta.**

| | |
|---|---|
| **Nombre de la entidad** | Carpeta |
| **Descripción de la entidad** | Es un contenedor de archivos que organiza de forma jerárquica la información dentro de un sistema de archivos. |

**Entidad Porta Papeles.**

| **Nombre de la entidad** | Porta Papeles |
|---|---|
| **Descripción de la entidad** | Se refiere a el almacenamiento de direcciones para una futura acción de pegar. Este es llenado por las operaciones de cortar y copiar. |

| | | | | | **Restricciones** | |
|---|---|---|---|---|---|---|
| **Nombre del Atributo** | **Descripción** | **Tipo** | **¿Puede ser Nulo?** | **¿Es único?** | **Clases Válidas** | **Clases no Válidas** |
| lista de direcciones | Incluye todas las direcciones relativas de archivos y carpetas a pegar. | lista de string | Si | No | lista de string con direcciones o rutas | |
| tipo | Indica si la operación de pegado fue generada por una operación de copiar o si fue por una de cortar. | Enumerativo | Si | No | Copia o Corte | |

## 2.4. Requerimientos de la Solución

El propósito de la definición de requisitos es especificar las condiciones o capacidades que el sistema debe cumplir y las restricciones bajo las cuales debe operar, logrando un entendimiento entre el equipo de desarrollo y el cliente, especificando las necesidades reales de forma que satisfaga sus expectativas.

En esta actividad se derivan los requerimientos del sistema a través de la observación de sistemas existentes y entrevistas con usuarios, de forma tal que se defina cómo el sistema o producto se ajusta mejor a las necesidades del negocio y cómo va a ser utilizado éste por los usuarios. Se deben definir además los requerimientos no funcionales, determinando como se va a comportar el sistema y que cualidades debe tener. Esta actividad debe ser cooperativa e iterativa, logrando que exista de ambas partes comprensión y formalidad.

### 2.4.1. Requisitos Funcionales

Los requisitos funcionales constituyen las capacidades o condiciones, o sea, las utilidades expuestas por la solución para los usuarios. A continuación se listan estas funcionalidades:

**Navegar por el Sistema de Archivos.**

| **Conceptos Tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Sistema de Archivos | tipo |
| | Elemento de Sistema de Archivos | ruta |
| | Carpeta | |
| | Archivo | tipo |
| | Operación | |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Para poder navegar entre carpetas es necesario utilizar direcciones válidas. | No procede |
| **Descripción** | Brindar la posibilidad de cambiar el directorio de trabajo de una manera intuitiva y organizada.<br>1. Presentar características de navegación como “ir Atrás”, “ir Delante” e “ir Arriba”. Mantener actualizada la interfaz en todo momento indicándole al usuario cuando es posible o no realizar estas acciones independientemente de qué navegador se encuentre seleccionado.<br>2. Acceso rápido de carpetas predeterminadas. Se contará con un panel de lugares predeterminados que indican una forma rápida de acceder a los lugares de interés del usuario. Cada pulsación sobre un determinado lugar desencadenará una actualización en el navegador seleccionado. Este panel puede ser configurado.<br>3. Se brindará una barra de dirección por cada navegador que indica el directorio actual de trabajo. Esta barra tendrá características de auto completamiento de direcciones. Un cambio de dirección válido desencadenará una actualización del directorio de trabajo. | |

*Continúa...*

| | |
|---|---|
| | 4. Al pulsar doble click o la tecla Enter sobre una carpeta, automáticamente será actualizado el directorio de trabajo para esa carpeta en caso de tener permisos, en caso de no tenerlo deberá mostrar un mensaje de error.<br>5. Al pulsar doble click o la tecla Enter sobre un archivo este deberá abrirse con la aplicación determinada por el entorno de escritorio en funcionamiento. En caso de no poder abrirse se debe mostrar un mensaje de error. |
| **Validaciones** | No Procede |
| **Post-condiciones** | Ha sido cambiado el directorio de trabajo para la nueva dirección. |
| **Post-requisitos** | No Procede |

**Eliminar Archivos o Carpetas.**

| | | |
|---|---|---|
| **Conceptos Tratados** | **Conceptos** | **Atributos** |
| | Sistema de Archivos | - |
| | Elemento de Sistema de Archivos | ruta, atributos |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Para poder Eliminar archivos y carpetas en necesario tener suficientes permisos de usuario. | No procede |
| **Descripción** | Brindar la posibilidad de eliminar archivos y carpetas.<br>1. Pedir al usuario confirmación de eliminación.<br>2. Mostrar mensajes de error en caso de una posible falla con una descripción detallada, ejemplo falta de permisos suficientes.<br>3. El usuario puede cancelar la operación de Eliminar si desea. | |
| **Validaciones** | No Procede | |
| **Post-condiciones** | Han sido eliminados los archivos y carpetas seleccionados. | |
| **Post-requisitos** | No Procede | |

**Crear Archivos y carpetas**

| | | |
|---|---|---|
| **Conceptos Tratados** | **Conceptos** | **Atributos** |
| | Sistema de Archivos | espacio libre |
| | Elemento de Sistema de Archivos | nombre, ruta, atributos, tamaño |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Para poder crear archivos y carpetas debe haber espacio disponible en el sistema de archivos y no debe existir un archivo o carpeta relativamente con el mismo nombre dentro de la ruta donde se va a crear. | No procede |
| **Descripción** | Brindar la posibilidad de crear archivos y carpetas.<br>1. Pedirle al usuario el nombre del nuevo elemento a crear, así como su tipo.<br>2. En caso de existir un elemento con el mismo nombre que el proporcionado o que el nombre contenga caracteres inválidos volver a pedirlo.<br>3. Mostrar mensajes de error en caso de no tener permiso de escritura en el directorio actual.<br>4. El usuario puede cancelar la operación de crear si desea. | |
| **Validaciones** | No Procede | |
| **Post-condiciones** | Ha sido creado un archivo o carpeta dentro de la ruta indicada. | |
| **Post-requisitos** | No Procede | |

**Copiar Archivos y Carpetas**

| | | |
|---|---|---|
| **Conceptos Tratados** | **Conceptos** | **Atributos** |
| | Sistema de Archivos | - |
| | Elemento de Sistema de Archivos | ruta, atributos |
| | Porta Papeles | lista de direcciones, tipo |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Para poder copiar archivos y carpetas deben haber sido seleccionados antes dentro de determinado directorio. | No procede |

*continúa...*

| | |
|---|---|
| **Descripción** | Brindar la posibilidad de copiar archivos y carpetas.<br>1. En caso de el usuario pulsar la combinación de teclas Control+C o hacer click en el menú "Copiar" el contenido de la selección dentro del navegador seleccionado pasará al porta papeles.<br>2. En caso de Arrastrar elementos válidos para un navegador con la tecla Control presionada, si se tiene permisos de escritura debe mostrar una ventana de progreso de la copia, al final de ésta debe mostrar confirmación como finalizada.<br>3. En caso de no tener permisos de escritura o arrastrar un elemento inválido debe notificar el error al usuario.<br>4. El usuario puede cancelar la operación de copiar si desea. |
| **Validaciones** | No Procede |
| **Post-condiciones** | Ha sido actualizado el porta papeles con las nuevas direcciones a copiar y se ha puesto su tipo en "copia" |
| **Post-requisitos** | Pegar Archivos y Carpetas |

**Cortar Archivos y Carpetas**

| | | |
|---|---|---|
| **Conceptos Tratados** | **Conceptos** | **Atributos** |
| | Sistema de Archivos | - |
| | Elemento de Sistema de Archivos | ruta, atributos |
| | Porta Papeles | lista de direcciones, tipo |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Para poder cortar archivos y carpetas deben haber sido seleccionados antes dentro de determinado directorio y además tener los privilegios suficientes de modificación para dichos archivos | No procede |

*Continúa...*

| | |
|---|---|
| **Descripción** | Brindar la posibilidad de cortar archivos y carpetas.<br>1. En caso de el usuario pulsar la combinación de teclas Control+X o hacer click en el menú "Cortar" el contenido de la selección dentro del navegador seleccionado pasará al porta papeles.<br>2. En caso de Arrastrar elementos válidos para un navegador, si se tiene permisos de escritura debe mostrar una ventana de progreso del corte, al final de esta debe mostrar confirmación como finalizada.<br>3. En caso de no tener permisos de escritura o arrastrar un elemento inválido debe notificar el error al usuario.<br>4. El usuario puede cancelar la operación de cortar si desea. |
| **Validaciones** | No Procede |
| **Post-condiciones** | Ha sido actualizado el porta papeles con las nuevas direcciones a cortar y se ha puesto su tipo en "corte" |
| **Post-requisitos** | Pegar Archivos y Carpetas |

**Pegar Archivos y Carpetas**

| | | |
|---|---|---|
| **Conceptos Tratados** | **Conceptos** | **Atributos** |
| | Sistema de Archivos | espacio libre |
| | Elemento de Sistema de Archivos | ruta, atributos |
| | Porta Papeles | lista de direcciones, tipo |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Para poder pegar archivos y carpetas debe haber en el porta papeles una lista de elementos a pegar y tener privilegios de pegar dentro del directorio seleccionado. | Copiar Archivos y Carpetas<br>Cortar Archivos y Carpetas |

*Continúa...*

| | |
|---|---|
| **Descripción** | Brindar la posibilidad de pegar archivos y carpetas.<br>1. Al presionar la combinación de teclas Control+V o hacer click en el menú "Pegar" se debe pegar el contenido del porta papeles en el navegador seleccionado.<br>2. En caso de no tener permisos de escritura en el directorio actual se debe notificar al usuario.<br>3. En caso de ser una operación de corte indicada por el porta papeles se debe indicar al usuario cuando no sea posible mover el contenido de la carpeta fuente por problemas de permisos.<br>4. Notificar al usuario cuando la operación termine correctamente.<br>5. El usuario puede cancelar la operación de pegar si desea. |
| **Validaciones** | No Procede |
| **Post-condiciones** | Ha sido pegado el contenido del porta papeles a las ruta de destino y si el tipo de el porta papeles es corte es eliminado el contenido de la carpeta fuente. |
| **Post-requisitos** | No Procede. |

**Renombrar Archivos y Carpetas**

| **Conceptos Tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Elemento de Sistema de Archivos | ruta, atributos |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Para poder renombrar un archivo o carpeta debe haber sido seleccionado antes dentro de determinado directorio. El nuevo nombre no debe entrar en conflicto con un nombre de archivo o carpeta existente dentro de determinada ruta. | No procede |

*continúa...*

| | |
|---|---|
| **Descripción** | Brindar la posibilidad de renombrar archivos y carpetas.<br>1. Pedir al usuario el nuevo nombre.<br>2. En caso de que ya exista un elemento con ese mismo nombre dentro de ese directorio o que contenga caracteres inválidos, volver a pedir el nombre.<br>3. El usuario puede cancelar la operación de renombrar si desea. |
| **Validaciones** | No Procede |
| **Post-condiciones** | Ha sido renombrado un archivo o carpeta en una determinada ruta. |
| **Post-requisitos** | No Procede |

**Cargar y Configurar Componentes**

| | | |
|---|---|---|
| **Conceptos Tratados** | **Conceptos** | **Atributos** |
| | Plugin | nombre, descripción, nombre de fichero |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Para cargar los componentes primero se debe haber seleccionado el directorio que contiene los plugins. Además este directorio debe contener algún plugin. | No procede |
| **Descripción** | Brindar la posibilidad de Cargar y Configurar Plugins en la aplicación.<br>1. Al iniciar la aplicación automáticamente se cargarán los plugin previamente seleccionados por el usuario en la ventana de configuración.<br>2. En caso de que existan plugin en la carpeta contenedora que no sean válidos estos no se cargarán.<br>3. En tiempo de ejecución se podrá cargar o descargar plugins encontrados que sean válidos.<br>4. Se podrá cambiar por el usuario la dirección de la carpeta contenedora de plugins. | |
| **Validaciones** | No Procede | |
| **Post-condiciones** | No Procede | |
| **Post-requisitos** | No Procede | |

### 2.4.2. Requisitos No Funcionales

Los requisitos no funcionales especifican las características o propiedades que debe tener la solución a desarrollar. A continuación se listan estas características:

#### 2.4.2.1. Requerimientos de Usabilidad

- Podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.
- Tendrá siempre visible la opción de Ayuda, lo que posibilitará un mejor aprovechamiento por parte de los usuarios de sus funcionalidades.
- Presentará una correcta disposición de elementos gráficos que sea lo más asequible posible.

#### 2.4.2.2. Requerimientos de Confiabilidad

- Cada operación deberá tener una recuperación en cuanto se detecte un fallo del sistema que implique mal funcionamiento del gestor.

#### 2.4.2.3. Requerimientos de Rendimiento

- Garantizar un alto desempeño en cuanto a la navegación por el sistema de ficheros así como la respuesta a eventos del usuario.

#### 2.4.2.4. Requerimientos de Soporte

Se debe brindar soporte técnico por el mismo desarrollador quien garantiza la corrección de errores en la aplicación así como recuperación en caso de fallas.

#### 2.4.2.5. Restricciones de diseño

- Presentar interfaces sencillas de fácil uso y con rápida respuesta del sistema.
- Presentar interfaces uniformes y con los mismos colores y diseños.
- Presentar imágenes claras y con la correcta visualización de su contenido.
- Presentar mensajes y notificaciones sin ambigüedades.

#### 2.4.2.6. Requerimientos de Implementación

- El sistema debe de ser desarrollado en el lenguaje de programación C++.
- La aplicación debe ser desarrollada utilizando el marco de trabajo Qt.
- El sistema de se orientado a componentes utilizando el soporte que brinda Qt para ello.

#### 2.4.2.7. Requerimientos de ayuda y documentación

- Cada una de las etapas del proceso de desarrollo deberá contar con la documentación según el proceso de desarrollo de centro UCID.
- Los textos de los han de ser claros, precisos y en lenguaje de fácil comprensión.
- Cada proceso tiene que disponer de una Ayuda en línea que dé una explicación adecuada de su funcionamiento.
- Se debe confeccionar un manual de usuario, dicho manual tendrá una explicación detallada y de fácil comprensión de cada opción y proceso, haciéndose hincapié en cómo operarlos y en las acciones a tomar ante las diferentes alternativas que se presenten.

#### 2.4.2.8. Requerimientos de Interfaces

- Se debe garantizar que los colores de la interfaz de la aplicación sean claros. La interfaz debe ser de fácil comprensión e intuitiva en su funcionamiento permitiendo la utilización del sistema sin mucho entrenamiento.

#### 2.4.2.9. Requerimientos de licencias y patentes

- La aplicación será liberada con licencia LGPL.

#### 2.4.2.10. Requerimientos de Portabilidad

- El sistema debe funcionar en sistemas de la familia GNU/Linux y Windows, es decir multiplataforma.

#### 2.4.2.11. Requerimientos de Seguridad

- Debido a que cada sistema operativo propone un sistema de seguridad y protección del contenido del sistema de archivos así como un nivel de seguridad basado en usuarios no es nuestro objetivo proveer de mecanismos propios de seguridad.

#### 2.4.2.12. Requerimientos de Software

- Debido a que la aplicación será desarrollada utilizando el marco de trabajo Qt entonces es necesario tener instaladas las librerías del mismo donde se despliegue la aplicación.
- Es necesario ejecutar la aplicación en el sistema operativo GNU/Linux o Microsoft Windows.

#### 2.4.2.13. Requerimientos de Hardware

- Será posible utilizar la aplicación un una computadora de pocos recursos de hardware, recomendando un mínimo de 128 MB de memoria teniendo en cuenta que también se estarán ejecutando otras aplicaciones y el sistema operativo y 30 MB de espacio libre en disco.

### 2.4.3. Prototipo de Interfaz de Usuario

Se cuenta con una barra de menú con todo el conjunto de operaciones posibles del gestor así como temas de ayuda un configuración. Además existe una barra de herramientas que se puede personalizar y brindando acciones de navegación básicas. También existe un panel de utilidades que puede ser desacoplado de la aplicación o acoplado en cualquier momento brindando varias utilidades como búsqueda, acceso a lugares personalizados, previsualizaciones y otras. Luego en el área central se encuentra la vista de carpetas donde existen paneles de navegación independientes, cada uno de los cuales contiene su propia barra de dirección.

Interfaz de usuario principal.

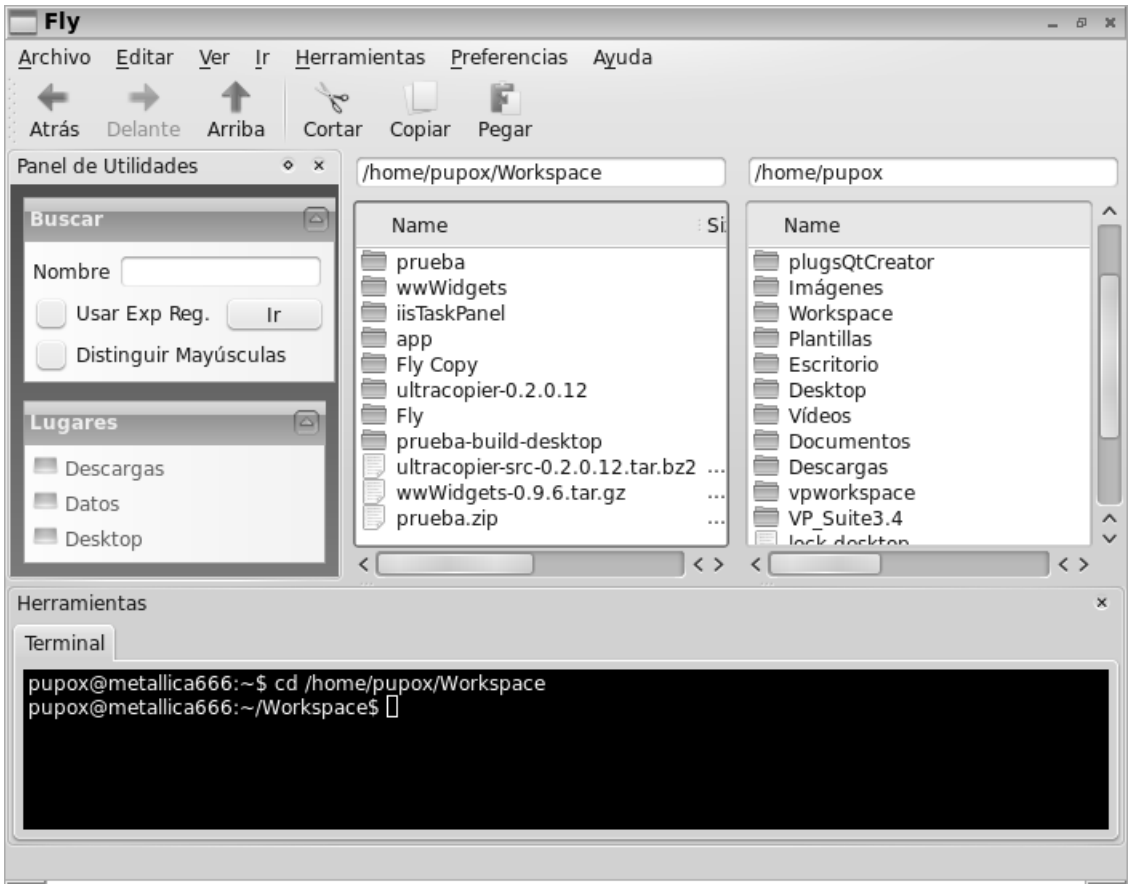


Figura 2.2: Prototipo de Interfaz de Usuario.

## 2.5. Conclusiones

En este capítulo han sido definidas todas las cuestiones que dan una visión del sistema en general así como esquemas de organización que brindan una guía para el futuro diseño del sistema.

Capítulo 3

# Diseño del sistema

La arquitectura de sistema representa una proyección simétrica de alto nivel de los procesos de negocio que se trabajan, expresada en elementos, conectores, configuraciones y restricciones. Primeramente se abordarán los niveles de empaquetamiento del diseño arquitectónico, estos tienen un alto impacto en el diseño de la solución. Una adecuada selección de los mismos suele implicar mejor organización del equipo de desarrollo y paralelización de las tareas de implementación, más facilidad en los procesos de Gestión de Configuración de Software y una granulación adecuada de las partes del software con implicación positiva en la gestión de la integración continua y el propio mantenimiento del sistema.

El profesor de Ciencias de la Computación David Garlan establece que la Arquitectura de Software constituye un puente entre el requerimiento y el código, ocupando el lugar que en los modelos antiguos se reservaba para el diseño. Por otra parte en el documento de IEEE Std 1471-2000, adoptada también por Microsoft, se define la arquitectura como: *... la organización fundamental de un sistema encarnado en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución.*

La Arquitectura enfoca el diseño del software desde varias perspectivas, de la calidad de estos procesos dependerá el éxito del diseño detallado, implementación e integración de la aplicación.

## 3.1. Diagramas de Interacción

Los diagramas de interacción tienen como objetivo la modelación de la interacción de objetos en un sistema. Específicamente un diagrama de secuencia es un diagrama de interacción que muestra a los objetos como una línea de vida a lo largo de un proceso.

A continuación se presentan los diagramas de secuencia.

### 3.1.1. Diagrama de Secuencia Crear Archivo o Carpeta.

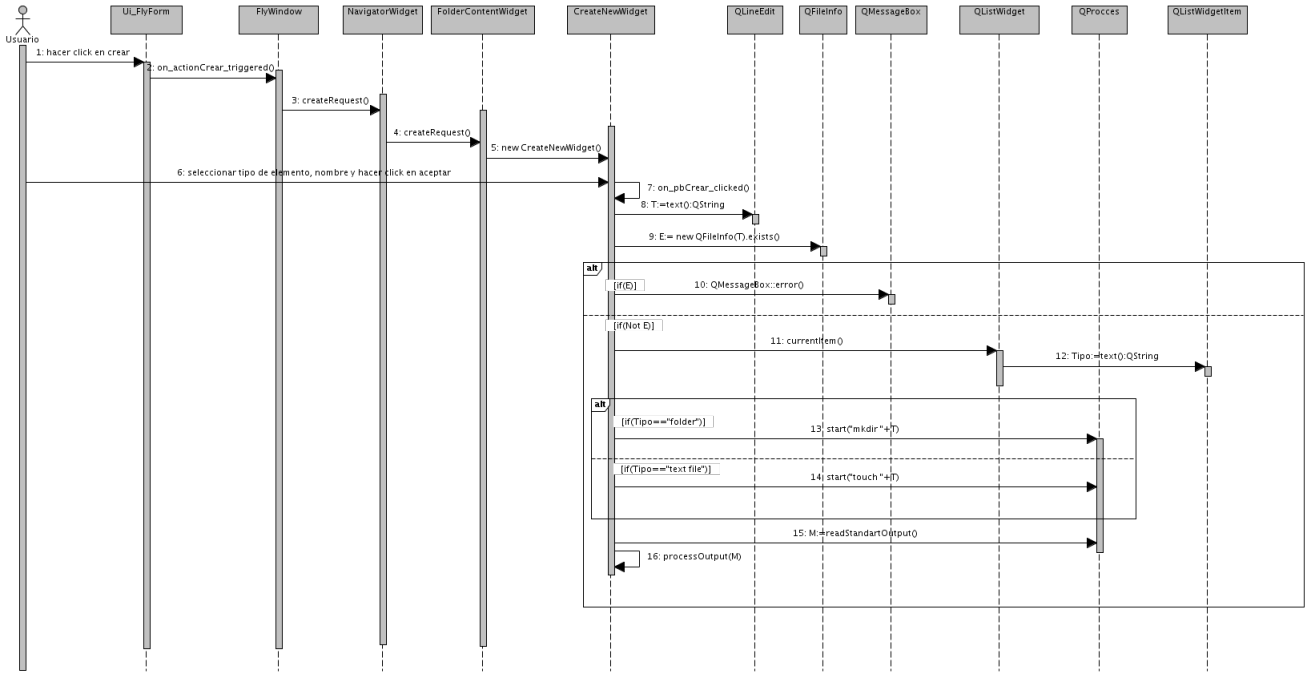


Figura 3.1: Diagrama de Secuencia Crear Archivo o Carpeta.

### 3.1.2. Diagrama de Secuencia Eliminar

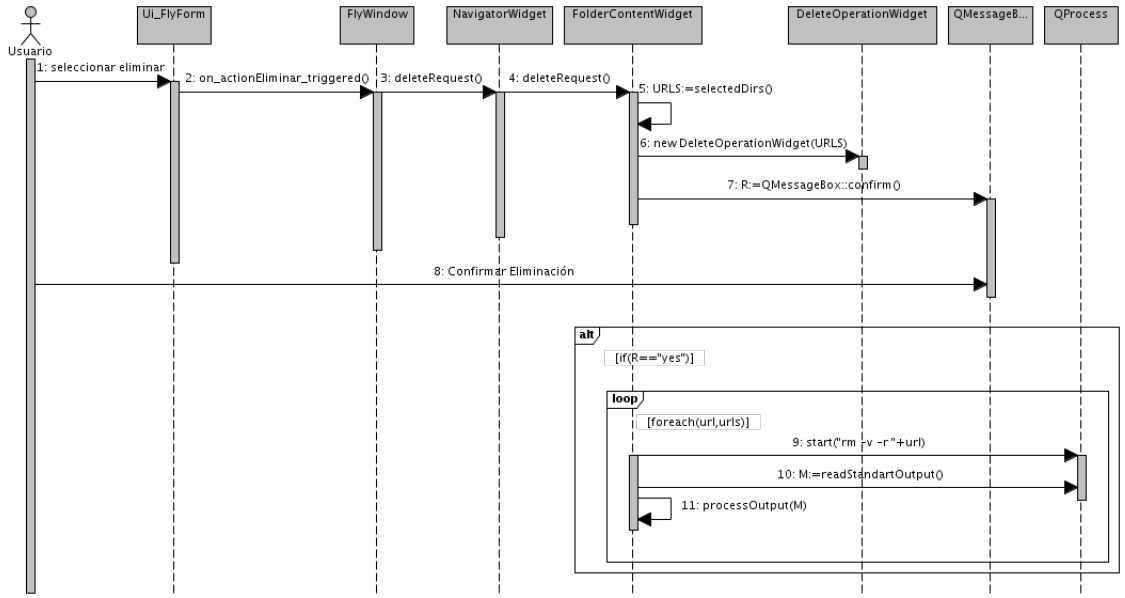


Figura 3.2: Diagrama de Secuencia Eliminar.

### 3.1.3. Diagrama de Secuencia Operación Copiar

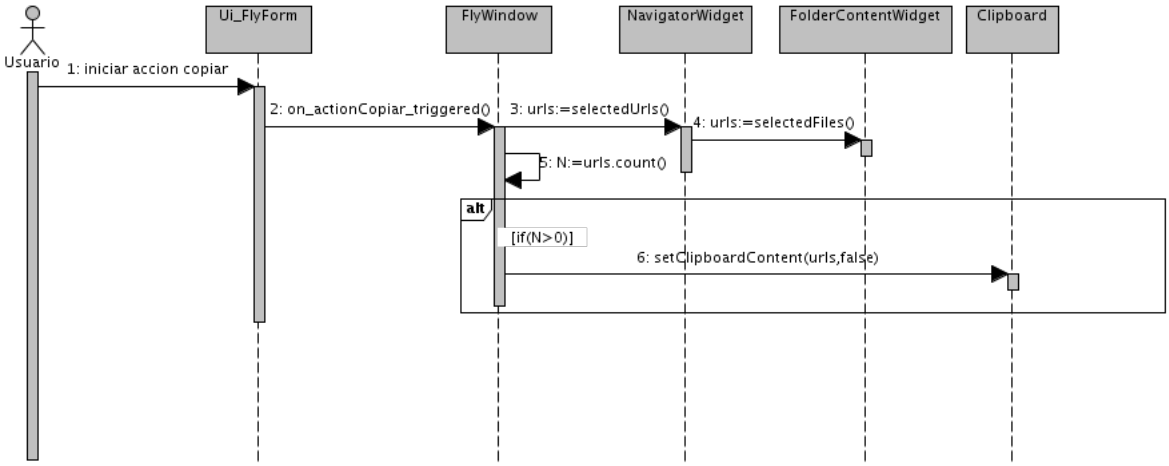


Figura 3.3: Diagrama de Secuencia Operación Copiar.

### 3.1.4. Diagrama de Secuencia Operación Cortar

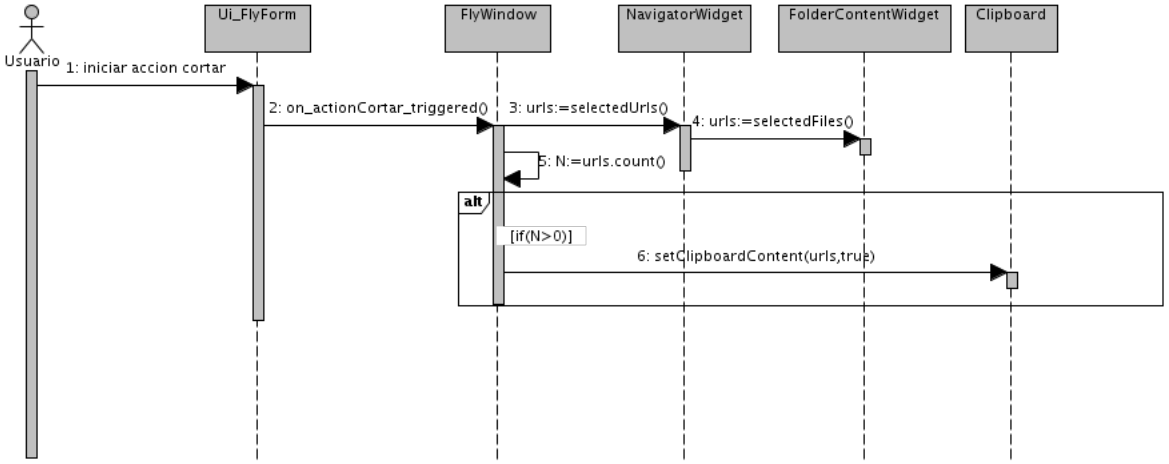


Figura 3.4: Diagrama de Secuencia Operación Cortar.

### 3.1.5. Diagrama de Secuencia Operación Pegar

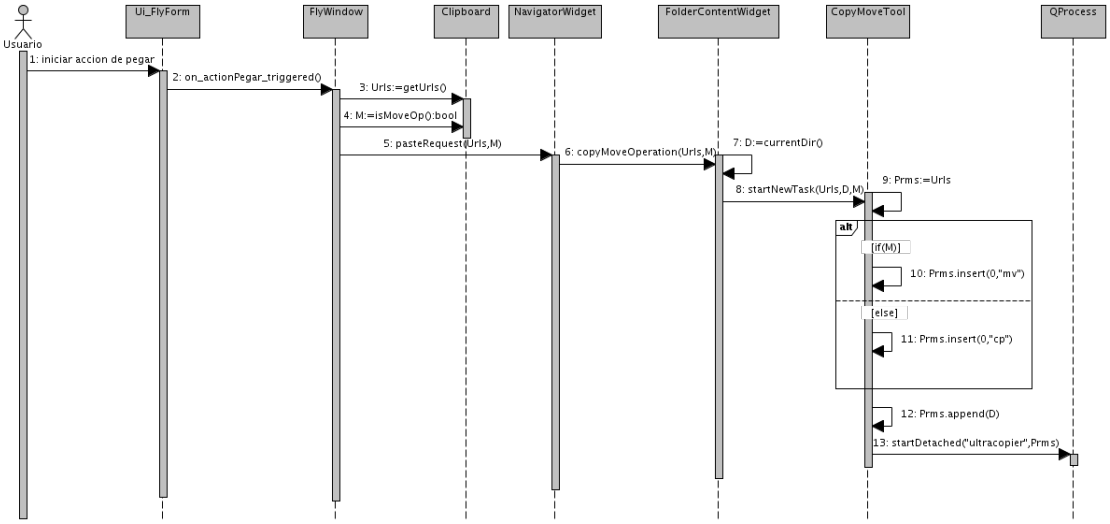


Figura 3.5: Diagrama de Secuencia Operación Pegar.

### 3.1.6. Diagrama de Secuencia Operación Renombrar

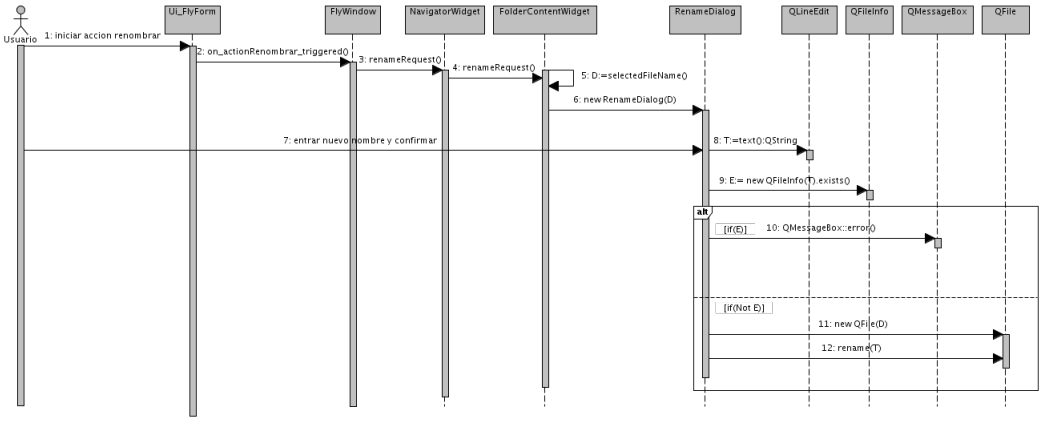


Figura 3.6: Diagrama de Secuencia Operación Renombrar.

### 3.1.7. Diagrama de Secuencia Ir Atrás

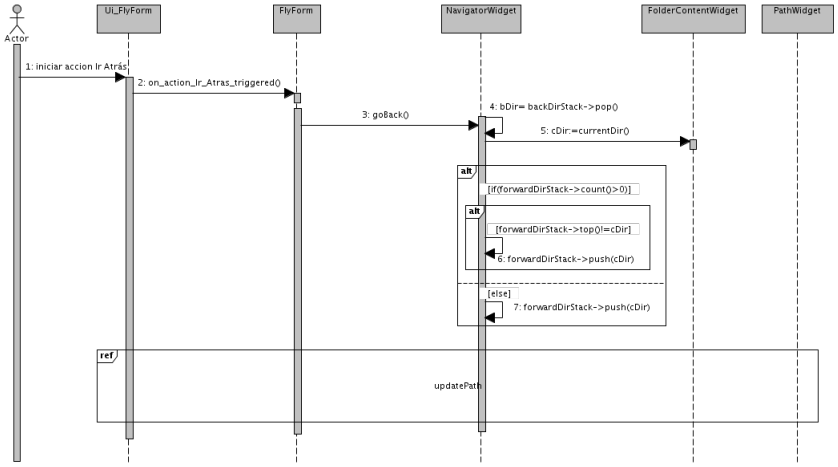


Figura 3.7: Diagrama de Secuencia Ir Atrás.

### 3.1.8. Diagrama de Secuencia Ir Arriba

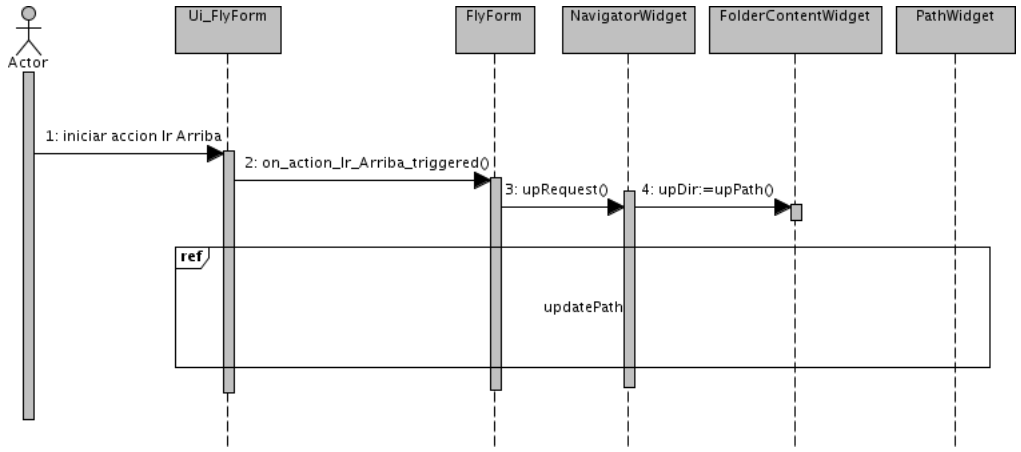


Figura 3.8: Diagrama de Secuencia Ir Arriba.

### 3.1.9. Diagrama de Secuencia Ir Delante

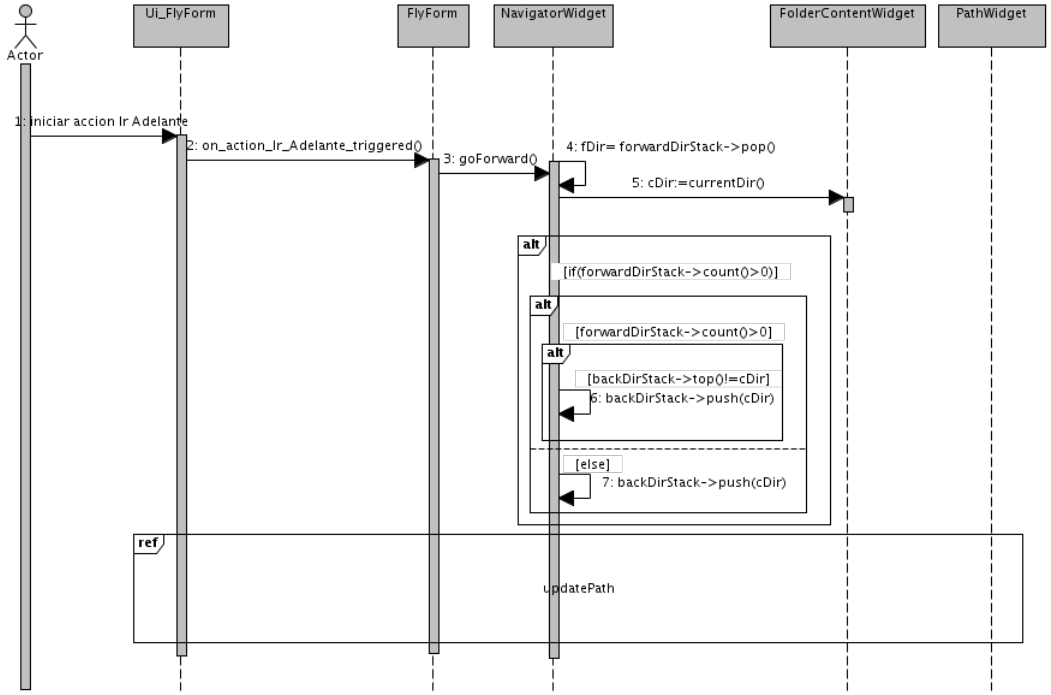


Figura 3.9: Diagrama de Secuencia Ir Delante.

### 3.1.10. Diagrama de Secuencia Cambiar Directorio Escrito

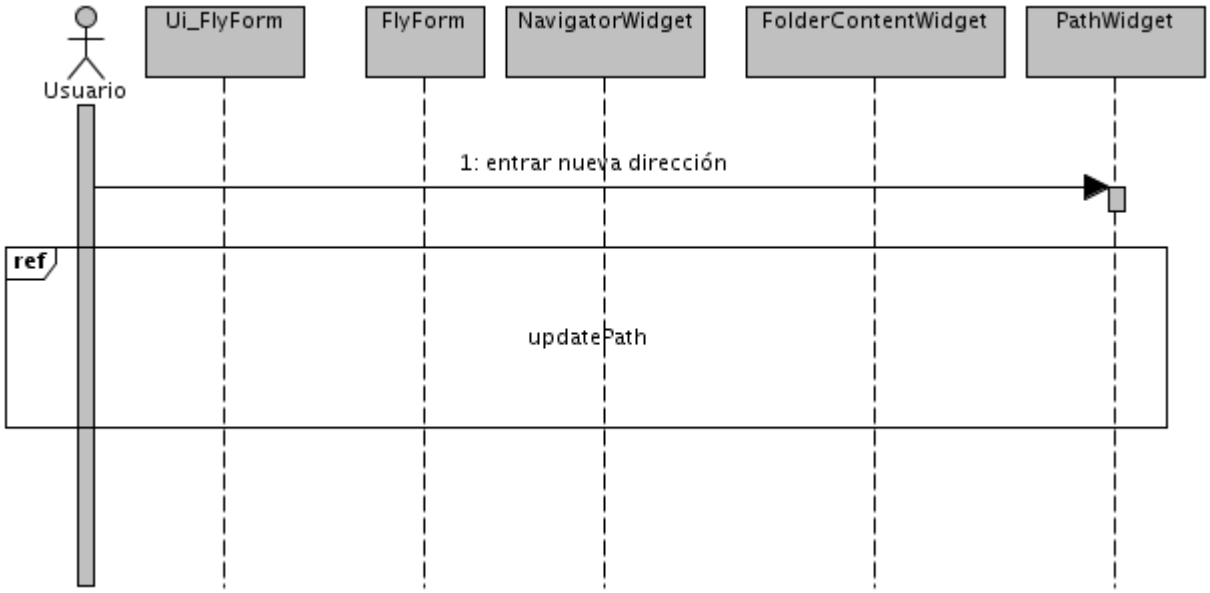


Figura 3.10: Diagrama de Secuencia Cambiar Directorio Escrito.

### 3.1.11. Diagrama de Secuencia Cambiar desde Contenido

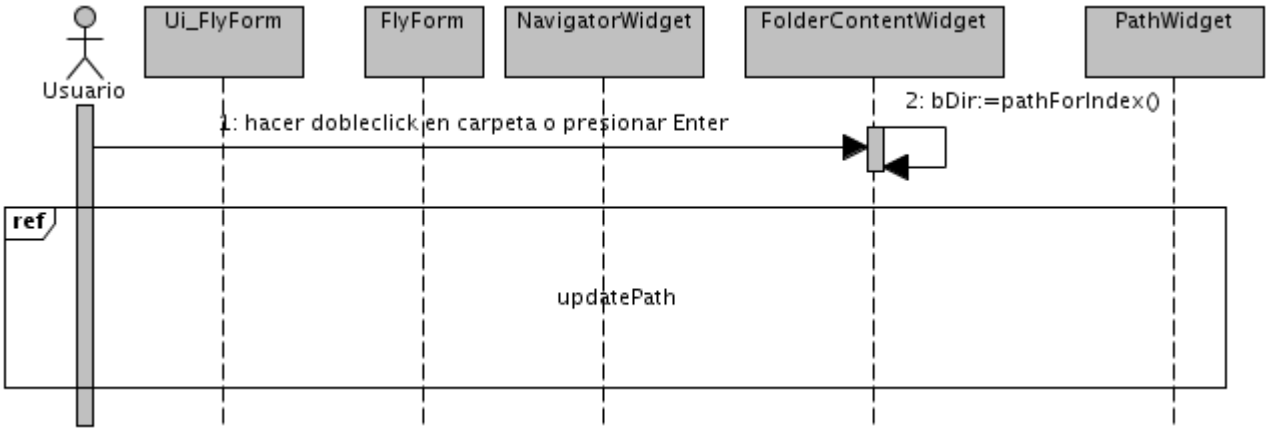


Figura 3.11: Diagrama de Secuencia Cambiar Directorio desde Contenido.

### 3.1.12. Diagrama de Secuencia Actualizar Ruta

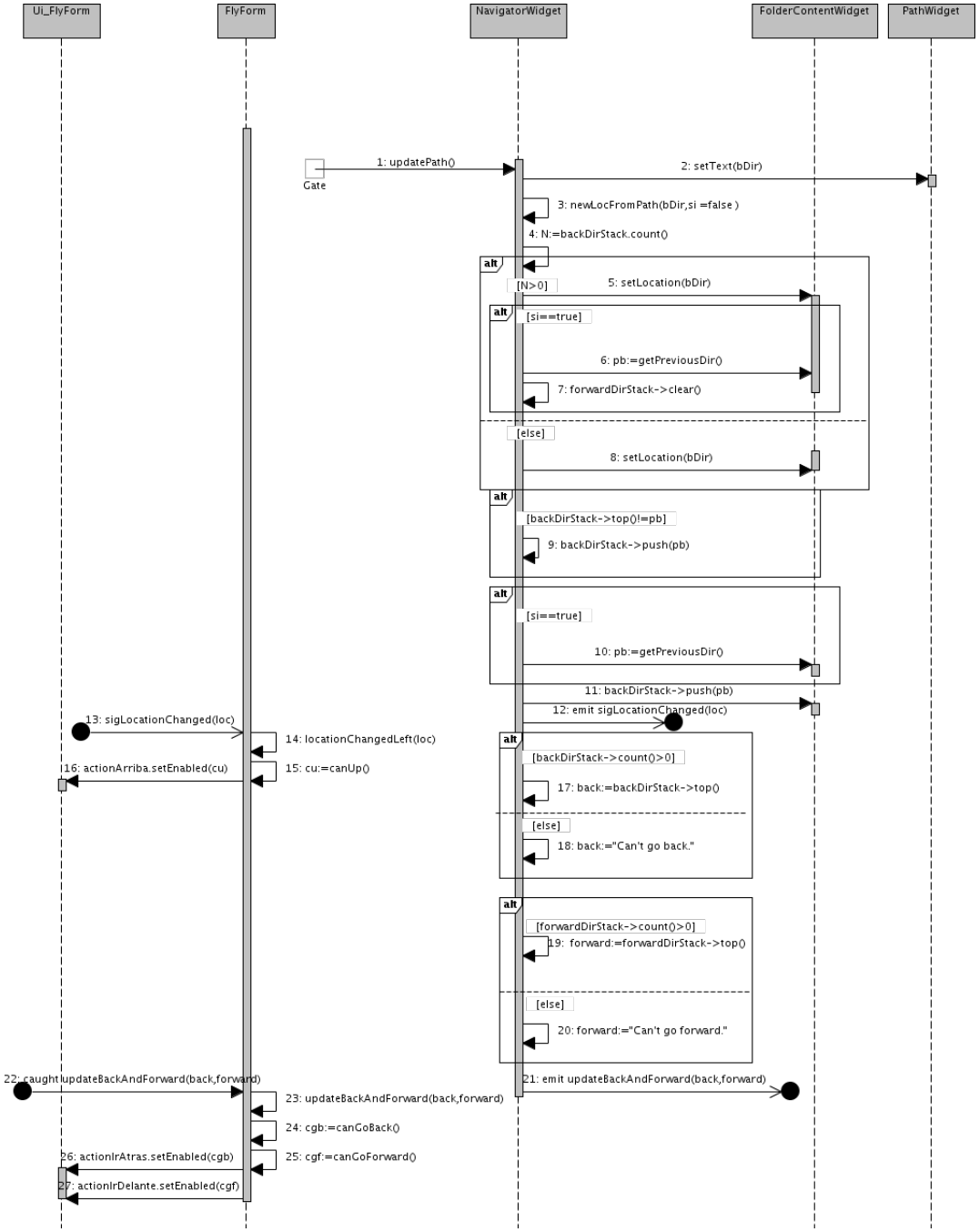


Figura 3.12: Diagrama de Secuencia Actualizar Ruta.

## 3.2. Diagrama de Clases

Los diagramas de clases de diseño tienen como premisa el análisis del sistema y su objetivo es dejar la estructura de la solución planteada en un lenguaje de programación específico.

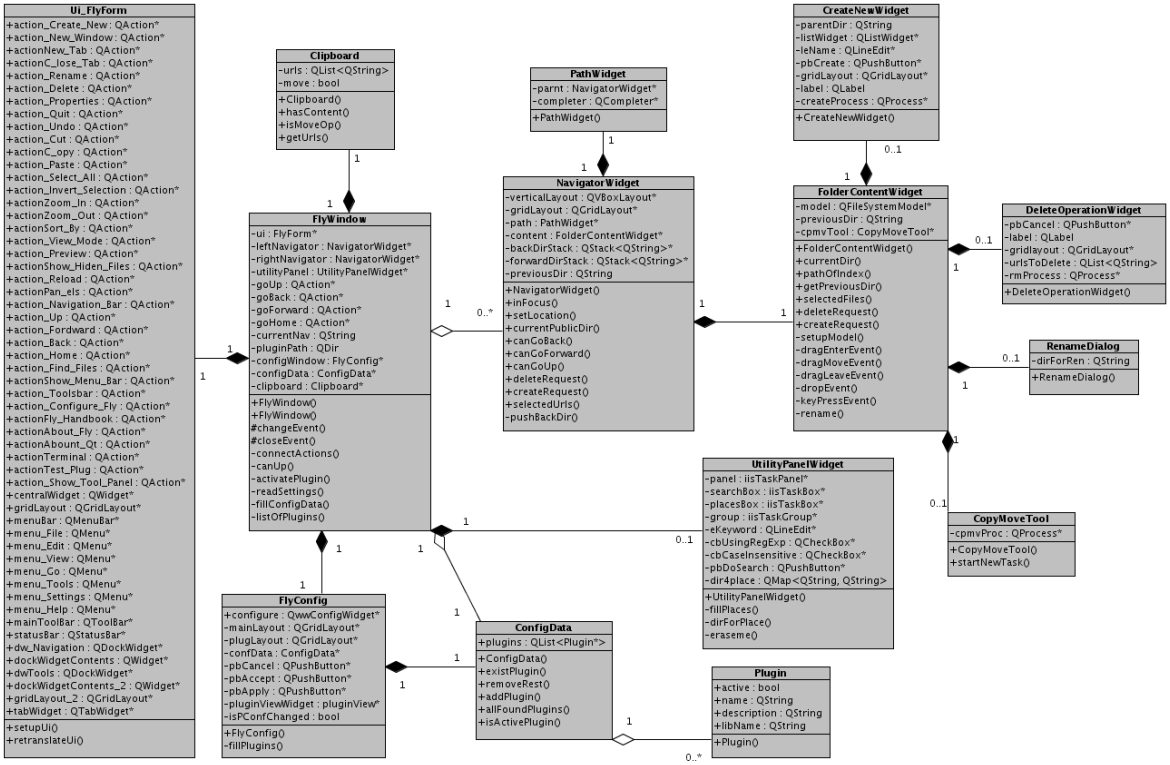


Figura 3.13: Diagrama de Clases.

A continuación se brinda una vista más general del diagrama de clases agrupado en paquetes.

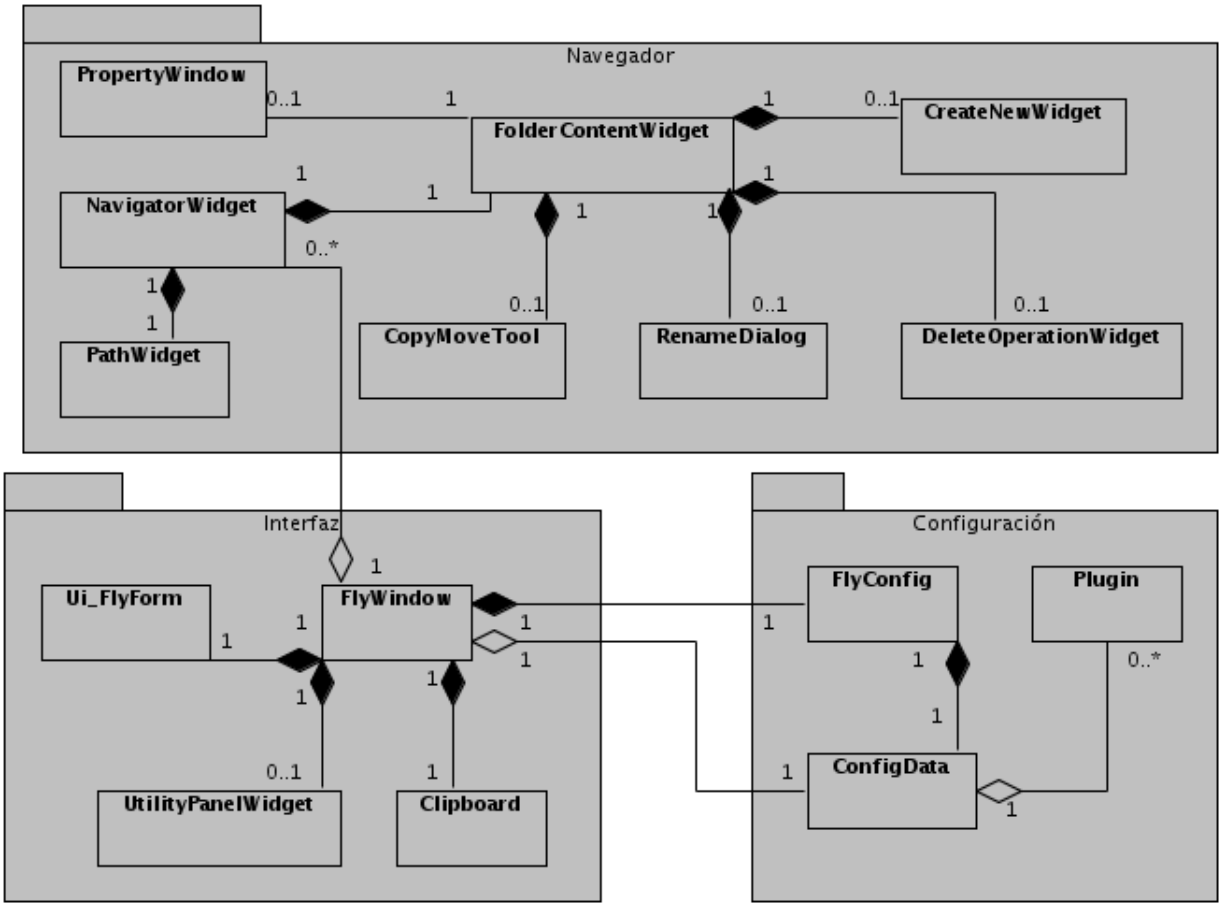


Figura 3.14: Diagrama de Clases en paquetes.

### 3.2.1. Descripción de las Clases

Para cada clase se especifica su descripción

#### 3.2.1.1. UI_FlyForm

| Interfaz | |
|---|---|
| **Atributos** | **Tipo** |
| action_Create_New | QAction * |
| action_New_Window | QAction * |

*continúa...*

| | |
|---|---|
| action_Rename | QAction * |
| action_Delete | QAction * |
| action_Properties | QAction * |
| action_Quit | QAction * |
| action_Undo | QAction * |
| action_Cut | QAction * |
| actionC_opy | QAction * |
| action_Paste | QAction * |
| action_Select_All | QAction * |
| action_Invert_Selection | QAction * |
| actionShow_Hiden_Files | QAction * |
| action_Reload | QAction * |
| action_Up | QAction * |
| action_Fordward | QAction * |
| action_Back | QAction * |
| action_Home | QAction * |
| action_Find_Files | QAction * |
| action_Configure_Fly | QAction * |
| actionFly_Handbook | QAction * |
| centralWidget | QWidget * |
| menuBar | QMenuBar * |
| mainToolBar | QToolBar * |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| setupUi() | Inicializa los elementos de la interfaz y establece la jerarquía de cada componente. |
| retranslateUi() | Traduce al idioma seleccionado los elementos de la interfaz. |

*continúa...*

#### 3.2.1.2. FlyWindow

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| ui | UI_FlyForm * |
| leftNavigator | NavigatorWidget * |
| rightNavigator | NavigatorWidget * |
| utilityPanel; | UtilityPanelWidget * |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| upRequest() | Ranura que se ejecuta en una acción de Ir Arriba |
| backRequest() | Ranura que se ejecuta en una acción de Ir Atrás |
| forwardRequest(); | Ranura que se ejecuta en una acción de Ir Adelante |
| locationChanged(QString) | Ranura que se ejecuta tras el cambio de directorio en alguno de los paneles de navegación de directorios. |
| updateBackAndForward(QString,QString) | Ranura que se ejecuta en una acción de cambio de directorio con el objetivo de actualizar el estado de las acciones de Ir Atrás e Ir Adelante. |
| newLocFromUtility(QString) | Ranura que se ejecuta cuando una dirección es seleccionada en panel de utilidades. |
| configRequest() | Ranura que se ejecuta cuando se requiere de la ventana de configuración de la aplicación. |
| writeSettings() | Ranura que se ejecuta en la salida de la aplicación con el objetivo de guardar preferencias y detalles de configuración. |
| updatePlugins() | Se ejecuta como comprobación de la existencia de plugins dentro de la carpeta contenedora de plugin. |
| reLoadPlugins() | Recarga los plugins dependiendo de la configuración del programa. |

*continúa...*

| | |
|---|---|
| deactivatePlugin(QObject) | Desactiva el plugin pasado por parámetro. |
| copyOperation() | Ranura que se activa tras una acción de copiar. |
| cutOperation() | Ranura que se activa tras una acción de cortar. |
| pasteOperation() | Ranura que se activa tras una acción de pegar. |
| deleteOperation() | Ranura que se activa tras una acción de eliminar. |
| createOperation() | Ranura que se activa tras una acción de crear. |
| focusedNav() | Devuelve el navegador activo o un objeto con valor 0 en caso de no encontrarse ninguno. |
| activatePlugin(QObject) | Activa el plugin pasado por parámetro |
| readSettings() | Lee los ajustes de la aplicación así como detalles de configuración. |

#### 3.2.1.3. NavigatorWidget

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| path | PathWidget * |
| content | FolderContentWidget * |
| backDirStack | QStack<QString> * |
| forwardDirStack | QStack<QString> * |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| inFocus() | Retorna verdadero si se encuentra enfocado el componente, si no devuelve falso. |
| setLocation(QString) | Establece una nueva dirección. |
| currentPublicDir() | Devuelve la dirección del directorio donde se encuentra el navegador. |
| canGoBack() | Retorna verdadero si es posible ir atrás en la navegación, si no devuelve falso. |

*continúa...*

| | |
|---|---|
| canGoForward() | Retorna verdadero si es posible ir adelante en la navegación, si no devuelve falso. |
| canGoUp() | Retorna verdadero si es posible ir arriba en la navegación, si no devuelve falso. |
| deleteRequest() | Atiende una petición de eliminar elementos del sistema de archivos. |
| createRequest() | Atiende una petición de crear elementos en el sistema de archivos. |
| selectedUrls() | Retorna las direcciones seleccionadas en el navegador. |
| updatePath(QString) | Actualiza la dirección a la que apunta la barra de dirección del navegador. |
| upRequest() | Atiende una petición de ir arriba en la navegación. |
| newLocFromPath(QString) | Actualiza la dirección del navegador introducida desde la barra de navegación. |
| newLoc(QString) | Actualiza la dirección del navegador. |
| newLocFromContent(QString) | Actualiza la dirección del navegador introducida desde la vista de carpeta. |
| newLocFromSource(QString) | Actualiza la dirección del navegador introducida desde una fuente externa al mismo. |
| goBack() | Atiende una petición de ir atrás en la navegación. |
| goFordward() | Atiende una petición de ir adelante en la navegación. |
| pasteRequest() | Atiende una petición de pegar en la carpeta actual el contenido del porta papeles. |

#### 3.2.1.4. FolderContentWidget

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| model | QFileSystemModel * |
| cpmvTool | CopyMoveTool * |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| currentDir(); | Retorna el directorio actual. |
| pathOfIndex(QModelIndex) | Retorna la ruta de un determinado índice del modelo que representa al sistema de ficheros. |
| selectedFiles() | Devuelve una lista con las direcciones de los ficheros seleccionados. |
| deleteRequest() | Atiende una petición de eliminación. |
| createRequest() | Atiende una petición de creación de archivos o carpetas. |
| setLocation(QString) | Establece una ruta nueva. |
| itemDoubleClick(QModelIndex) | Es invocado cuando ocurre un evento de doble click a un elemento en el sistema de archivos. |
| upPath() | Devuelve la dirección de la carpeta contenedora de la actual. |
| copyMoveOperation(QList<QString>,bool) | Atiende una petición de copia o movida. |
| void dragEnterEvent(QDragEnterEvent) | Es invocado cuando se inicia una operación de arrastre de un elemento hacia el navegador. |
| dropEvent(QDropEvent) | Es invocado cuando se termina una operación de arrastre de un elemento hacia el navegador. |

#### 3.2.1.5. PathWidget

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| completer | QCompleter * |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| void newDirSelected() | Ranura que se ejecuta al entrar una nueva dirección. |

#### 3.2.1.6. CreateNewWidget

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| parentDir | QString |
| listWidget | QListWidget * |
| leName | QLineEdit * |
| pbCreate | QPushButton * |
| createProcess | QProcess * |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| pbCreateClicked() | Ranura que se ejucatará al darle click al botón de crear. |

#### 3.2.1.7. DeleteOperationWidget

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| pbCancel | QPushButton * |
| urlsToDelete | QList<QString> |
| rmProcess | QProcess * |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| pbCancelClicked() | Ranura que se ejecutará al confirmar la eliminación. |

#### 3.2.1.8. RenameDialog

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| dirForRen | QString |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| rename() | Ranura que se ejecutará ante una acción de renombrado. |

#### 3.2.1.9. CopyMoveTool

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| cpmvProc | QProcess * |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| startNewTask(QList<QString> ,QString,bool) | Ranura que se ejecutará ante una nueva acción de cortar o mover. |

#### 3.2.1.10. UtilityPanelWidget

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| dirForplace | QMap<QString,QString> |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| fillPlaces() | Inicializa los lugares de acceso rápido. |
| dirForPlace(QString) | Retorna la dirección de determinado lugar de acceso rápido. |

#### 3.2.1.11. FlyConfig

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| configure | QwwConfigWidget * |
| pbCancel | QPushButton * |
| pbAccept | QPushButton * |
| pbApply | QPushButton * |
| pluginViewWidget | PluginViewWidget * |
| isPConfChanged | bool |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| fillPlugins() | Utilizado para cargar la configuración global de los plugins. |
| pConfModified() | Ranura que se ejecuta al variar una configuración. |
| confirm() | Ranura que se ejecuta al confirmar cambios en la configuración. |

#### 3.2.1.12. ConfigData

| Controladora | |
|---|---|
| **Atributos** | **Tipo** |
| _instance | ConfigData * |
| plugins | QList<Plugin*> |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| instance() | Método estático que retorna una instancia de la clase de configuración. |
| bool existPlugin(QString) | Retorna verdadero si existe el plugin con determinado nombre. |
| removeRest(QList<QString>) | Elimina los plugin que no se encuentren dentro de determinada lista. |
| addPlugin() | Añade un plugin. |
| allFoundPlugins() | Retorna los nombres de los plugins encontrados. |

#### 3.2.1.13. Plugin

| Entidad | |
|---|---|
| **Atributos** | **Tipo** |
| active | bool |
| name | QString |
| description | QString |
| libName | QString |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |

#### 3.2.1.14. Clipboard

| Entidad | |
|---|---|
| **Atributos** | **Tipo** |
| urls | QList<QString> |
| move | bool |
| **Responsabilidades** | |
| **Nombre** | **Descripción** |
| hasContent() | Retorna verdadero si existe contenido dentro del porta papeles, si no retorna falso. |
| setClipboardContent(QList<QString>,bool) | Establece el contenido del porta papeles así como el tipo de operación si es corte o copia. |

## 3.3. Patrones Arquitectónicos

Un patrón se encuentra descrito por cuatro elementos: nombre del patrón, problema, solución y consecuencias. Por tanto un patrón arquitectónico se encuentra enfocado al ámbito del diseño de la arquitectura de software. Para el desarrollo de la arquitectura se tendrán en cuenta un conjunto de patrones arquitectónicos que guiarán todo el diseño.

### 3.3.1. Estilos de Arquitectura

Primeramente se tendrá en cuenta el estilo arquitectónico de llamada y retorno el cual permite al diseñador de software construir una estructura de programa relativamente fácil de modificar y ajustar a escala, dentro de ella existen dos subestilos, utilizando la arquitectura de programa principal/subprograma.

También formará parte de la propuesta la arquituctura orientada a objetos la cual especifica que los componentes de un sistema encapsulan los datos y las operaciones que se deben realizar para manipular los datos

Por último se tendrá en cuenta la arquitectura estratificada, específicamente en la forma de Modelo Vista Controlador(MVC) que consiste en separar el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Maneja la visualización de la información.

Controlador: Controla el flujo entre la vista y el modelo (los datos).

El marco de trabajo Qt utiliza una variante de MVC que une la vista con el controlador, denominado model/view, para la representación de datos en Item Views que serán los componentes donde se representarán los elementos del sistema de ficheros alojados en un determinado modelo. Este modelo se comunica con una fuente de datos, proporcionando una interfaz para los otros componentes de la arquitectura. La naturaleza de la comunicación depende específicamente de cada fuente de datos y de la forma en que el modelo es implementado. La vista obtiene los índices del modelo; estos representan datos de cada elemento.

En las formas de visualizar un modelo, un delegado representa un elemento de datos, cuando este es editado, el delegado se comunica directamente con el modelo utilizando índices.

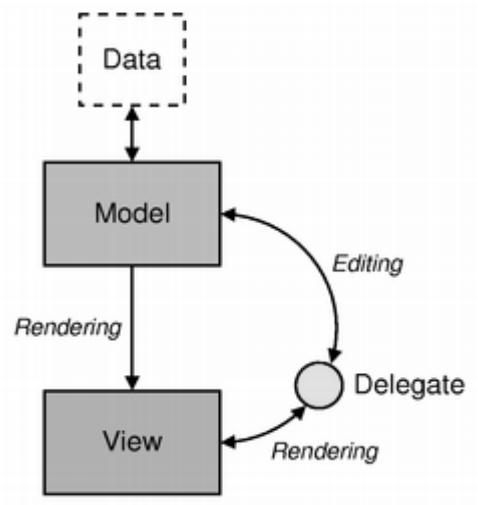


Figura 3.15: Arquitectura modelo/vista.

### 3.3.2. Patrones GRASP

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

Para la asignación de responsabilidades se utilizará un patrón Experto el cual indica que la clase que cuenta con la información necesaria para cumplir la responsabilidad es la responsable de manejar la información. Luego dadas las propiedades de todo sistema orientado a objetos se utilizará el patrón Creador aplicado en cualquier caso de agregación o composición de clases. Para fomentar la reutilización de código se tendrá en cuenta el patrón Bajo Acoplamiento que indica una menor dependencia entre clases.

### 3.3.3. Patrones GOF

Los patrones GOF(Gang Of Four) son dirigidos al desarrollo de sistemas orientados a objetos y se encuentran divididos en tres grupos, los de creación utilizados en la abstracción de como es creado un objeto, de estructura que indican como se encuentran compuestas las clases y comportamiento utilizado en la asignación de responsabilidades en las clases.

En el marco de trabajo Qt se utilizan un conjunto de patrones, entre ellos uno evidente es el patrón de estructura Composición, este representa objetos en forma de árbol jerárquico y es el caso del objeto QObject que contiene otro QObject que es su padre, además hay otros objetos como el QTreeWidgetItem que es un elemento dentro de un árbol que puede tener hijos o padre de su mismo tipo.

Otro patrón a utilizar y esta vez es uno de comportamiento es el Observador que define la dependencia muchos a muchos de forma tal que el cambio en uno es notificado a todas sus dependencias, es el caso del mecanismo de señales y eventos que propone Qt donde cada objeto que herede de la base QObject puede emitir señales y/o recibirlas, cada señal puede ser recibida por varios objetos mediante una ranura(SLOT) o incluso desencadenar otra señal[1].

Qt provee un mecanismo de plugins, agregados externos a la aplicación, que pueden fomentar el crecimiento de funcionalidades, estos plugins no son mas que objetos que implementan una determinada interfaz lo que posibilita que la aplicación que los use pueda tener conocimiento de su estructura, esto no es más que la utilización del patrón de estructura Fachada. Por otra parte será visible la utilización del patrón Sigleton, el cual asegura que un objeto tenga solo una instancia y provee un acceso a ella, este es utilizado nativamente por Qt en cualquier aplicación mediante la macro qApp que retorna una instancia global de la misma aplicación.

También este patrón será utilizado en objetos como los que guardan datos de configuración o almacenan datos temporales como el porta papeles debido a que serán utilizados en varias ocasiones y tendrán una única instancia global. Otro patrón de comportamiento utilizado es el de Orden que proporciona encapsular una petición en un objeto, este será utilizado en múltiples escenarios, es el caso de los QAction, objeto que pueden ser representados dentro de menús, barras de herramientas, botones y otros, la acción que desencadena es independiente de donde se encuentre.

Por último se utilizará en patrón Cadena de Responsabilidades con el objetivo de tener definida una estructura de atención a determinada funcionalidad, delegando responsabilidades de un componente a otro con el fin de llevar a cabo la tarea.

### 3.3.4. Otros Patrones

Qt implementa otros patrones internamente como es el caso del MetaObject que en algunos casos es denominado de reflejo, este indica la posibilidad de obtener información acerca de las propiedades y métodos de un QObject

[1]El mecanismo de señales y ranuras no es una creación de Qt, existe también una implementación de c++ para esto.

en tiempo de ejecución. Otro patrón es el Mono-estado que indica como múltiples instancias de un objeto pueden compartir el mismo estado dependiendo de sus atributos, es el caso del objeto QSettings utilizado en el acceso a preferencias para cada aplicación, dos objetos diferentes de este tipo pueden referenciar a lo mismo si el nombre de la empresa y el nombre de la aplicación es el mismo en ambos.

### 3.3.5. Diseño de Interfaz de Usuario

El diseño de interfaz radica en describir como se comunica el Software consigo mismo, con los sistemas que operan junto con el y con los operadores y usuarios que lo emplean. Es importante dedicar tiempo y recursos a esta fase, pues en ella se definen los objetivos de usabilidad del programa, las tareas del usuario, los objetos y acciones de la interfaz, los iconos, vistas y representaciones visuales de los objetos, así como los menús de los objetos y ventanas.

La interfaz de usuario será lo más sencilla posible, esto posibilita que el usuario tenga el control en todo momento de lo que se encuentra haciendo, además de trabajar organizadamente. Muchas veces se incluye en la interfaz principal un número muy grande de elementos que tienden a la confusión.

Algunos de los elementos que muestran o manejan información serán desacoplables, lo que significa que se pueda cambiar la orientación o disposición el área de trabajo a gustos personales. Esto también es válido para las barras de herramientas, las que además deben podes ser ocultadas o mostrada a solicitud del usuario.

Se utilizaran las fuentes predefinidas del sistema debido a que se supone que cada usuario tiene configurado su sistema de acuerdo a sus preferencias por lo tanto no es necesario definir un mecanismo interno para el cambio de fuentes. Lo mismo sucede con los esquemas de color y el tema de escritorio.

Los iconos deberán ser lo más estándar posible, de hecho lo ideal es que dependan de el tema del entorno de escritorio dentro del cual se está ejecutando la aplicación, en caso de que el entorno de escritorio no posea un mecanismo de utilización de iconos que posibilite esta utilidad se cargarán iconos propios de la aplicación que vienen definidos por defecto.

Capítulo 4

# Implementación y pruebas

En este capítulo se tendrán en cuenta todos los aspectos del diseño del sistema a fin de llevar a cabo la construcción del software. Se presentará el diagrama de componentes. Luego se propone el diseño de las pruebas que validan una correcta implementación.

## 4.1. Implementación

### 4.1.1. Diagrama de Componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias lógicas entre un conjunto de componentes de software, sean éstos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia. En la figura 4.1 se muestra una representación gráfica del diagrama de componentes.

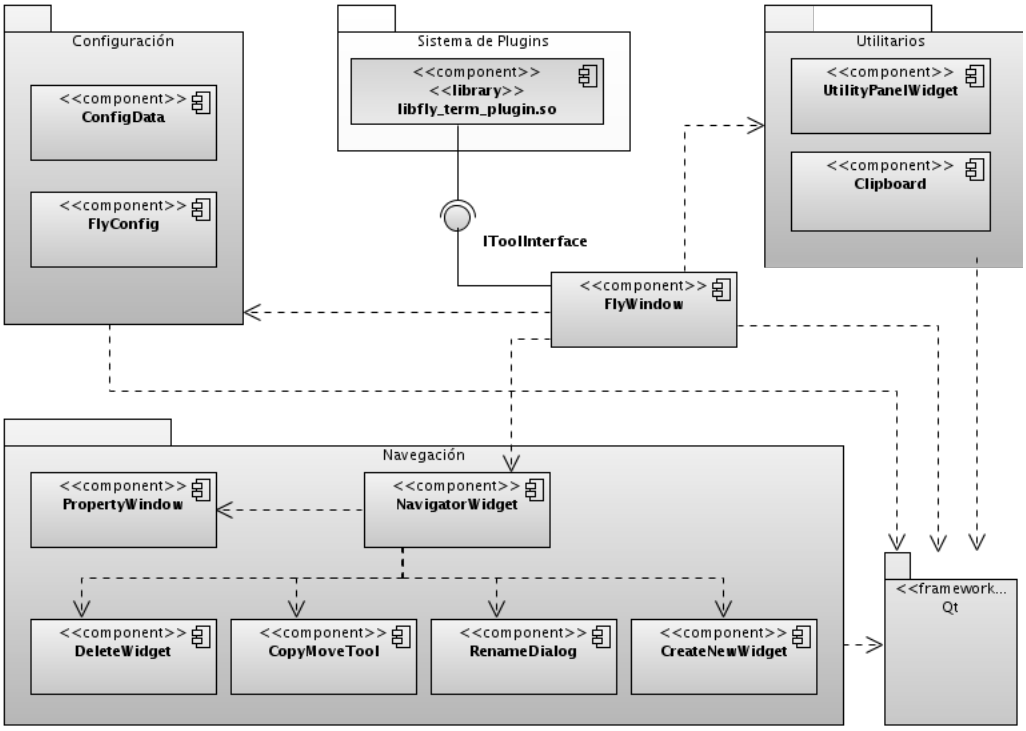


Figura 4.1: Diagrama de Componentes.

### 4.1.2. Matriz de Integración de Componentes

La siguiente matriz contiene todos los componentes definidos en el subsistema y especifica los servicios que consume el componente interno desarrollado de los componentes externos que se usaron.

| Componentes Externos | |
|---|---|
| **Componentes Internos** | **Terminal** |
| NavigatorWidget | Provee una forma fácil de ejecutar comandos en una terminal. Esta es actualizada independientemente por cada navegador. |

### 4.1.3. Diseño del sistema de agregados

La aplicación deberá ser capaz de trabajar con agregados externos independientemente de cómo hallan sido implementados. Para esto se brinda una interfaz (IToolInterface), esta contiene un conjunto de métodos que deberán ser implementados por el creador del agregado. Esta interfaz contiene las informaciones básicas y distintivas de cada plugin como su nombre, descripción, nombre y tipo de herramienta; así como la funcionalidad final para la que ha sido desarrollado.

Cada plugin será compilado y distribuido en un fichero binario de librería para Linux con extensión *.so.

El programa contiene un mecanismo de manejo de plugins el cual consiste en la búsqueda de agregados en determinado directorio que puede ser modificado por el usuario. Este directorio puede contener plugins que no sean válidos, en este caso no se tendrán en cuenta. Cada uno de estos podrán ser cargados y descargados en tiempo de ejecución mediante la ventana de configuración, su estado de actividad será recordado por la aplicación en caso de reabrirse.

## 4.2. Prueba

La realización de pruebas es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos específicos, los resultados son observados y registrados, luego se realiza una evaluación de algún aspecto del sistema o componente. En sí constituyen una etapa imprescindible durante el proceso de desarrollo de software, pues permiten detectar y corregir el máximo de errores posibles antes de la entrega al cliente del software desarrollado, por lo que el éxito de las mismas puede mejorar la percepción de calidad del usuario final.

El objetivo principal de las pruebas es asegurar que el software cumpla con las especificaciones requeridas y eliminar los posibles defectos que este pudiera tener. De forma más específica los propósitos de las pruebas son:

- Realizar la validación del software desarrollado, entendiendo como validación el proceso que determina si el software satisface los requisitos.
- Realizar la verificación del software desarrollado, entendiendo como verificación el proceso que determina si los productos de una fase satisfacen las condiciones de la misma.

Es importante considerar que las pruebas de software no garantizan que un sistema esté libre de errores, sino que se detecten la mayor cantidad de defectos posibles para su debida corrección.

### 4.2.1. Diseño de Casos de Prueba

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Los casos de pruebas tienen que ser escritos para generar las condiciones de salida deseadas.

En si un caso de prueba constituye un conjunto de condiciones o variables bajo las cuáles se determina si un requisito es parcial o completamente satisfactorio. Su propósito es especificar una forma de probar el sistema que incluya las entradas, los resultados esperados y las condiciones bajo las que ha de probarse. Los casos de prueba ayudan a validar que el sistema desarrollado realice las funciones para las que ha sido creado en base a los

requerimientos del usuario, por lo que resulta importante diseñar al menos un caso de prueba para cada requisito del sistema con el objetivo de asegurar que todas las funcionalidades sean comprobadas. Los casos de prueba se encuentran desarrollados en los Anexos al igual que cada definición de variables que se implican en la misma.

La validación de los casos de pruebas no son más que la aplicación de los casos de prueba ante un conjunto de situaciones específicas. Esta validación en parte se efectúa en la definición de un conjunto de juego de datos a probar en los escenarios de prueba que contengan las posibles entradas de los usuarios. En los Anexos se muestran las mismas.

### 4.2.2. Métodos

Para la realización de las pruebas se ha tenido en cuenta el método de caja negra, estas se realizan sin importar como han sido implementadas las funcionalidades o la estructura interna del programa y se basa en la interacción con la interfaz. Los casos de prueba pretenden:

1. Demostrar que las funciones del software son operativas.
2. Que las entradas se aceptan de la forma adecuada y que se produce el resultado correcto en la salida.
3. La información es manipulada de una forma correcta.

Estas pruebas permiten encontrar:

- Funciones incorrectas o no implementadas.
- Errores de interfaz de usuario.

### 4.2.3. Resultados

Fueron probados siete requisitos funcionales, los que representan el cien por ciento del total de requisitos. Para cada requisito se han tenido en cuenta varios escenarios que implican cada una de las posibles interacciones del usuario o combinaciones de situaciones posibles con el fin de evaluar el comportamiento del sistema ante cada funcionalidad.

El resultado de cada prueba coincide con la especificación de cada requisito funcional lo que demuestra la correcta implementación de los mismos.

Por otra parte se midió el consumo de memoria RAM de la aplicación al ser iniciada con un valor de 3.6 MB, luego de realizar un conjunto de operaciones el índice de consumo no excede a los 4 MB. También vale destacar una característica del modelo QFileSystemModel que mantiene una cache temporal de los directorios que se van visitando con el fin de disminuir drásticamente el tiempo de poblar el navegador en el caso de directorios con un número grande de elementos; haciéndolo ganar en rendimiento.

Estas características hacen valida y estratégica la implantación de la aplicación desarrollada en el Entorno de Escritorio Quántico.

# Conclusiones

Para la realización de esta tesis se han tenido en cuenta un conjunto de tareas de investigación que han guido el proceso de desarrollo. Fueron estudiados y clasificados los principales gestores de ficheros en vistas de buscar una solución al problema planteado, llegando a la conclusión de que ninguno satisfacía las necesidades de Quántico. Luego fueron definidas herramientas y tecnologías las cuales contribuyeron a un correcto proceso de modelación e implementación. Fue diseñada una arquitectura que uniendo elementos de distintos estilos arquitectónicos logró la implementación de una aplicación escalable y a la medida, teniendo el usuario en todo momento el control sobre la aplicación. Utilizando la arquitectura diseñada se logró la implementación de la solución dotando a esta de todos los requisitos detectados.

Se han logrado cada unos de los objetivos específicos propuestos lo que queda respaldado por los resultados en las pruebas. El producto finalizado se encuentra listo para su despliegue y utilización, incluyendo en otros entornos de escritorios aparte de Quántico. Solamente han sido utilizadas las librerías de desarrollo de Qt lo que cumple con los lineamientos de la arquitectura establecidos para el proyecto. La aplicación tiene la capacidad de mejorar ampliamente en funcionalidades mediante nuevos agregados que sean programados posteriormente.

# Recomendaciones

Se recomienda seguir profundizando en la implementación de componentes que doten de un mayor número de funcionalidades a la aplicación. Además se deben mejorar algunas funcionalidades básicas de la aplicación con respecto a la integración con el entorno de escritorio Quántico, el caso puntual es que este entorno no presenta una herramienta propia que vincule un tipo de fichero determinado con la aplicación que debe abrirlo lo que hace difícil implementar en el explorador de ficheros estas funcionalidades internamente.

También se recomienda crear particularmente la funcionalidad de búsqueda de carpetas y archivos dada su importancia y utilidad en cualquier entorno de escritorio.

# Referencias


[1] BURTCH, K. O., *Linux Shell Scripting with Bash*, Sams Publishing, 2004.

[2] CORCHUELO, R., Componentes: Test de conceptos, 2007.

[3] GRAY, T., www.kdevelop.org, 2010.

[4] www.eclipse.org, 2010.

[5] http://www.midnight-commander.org/, 2010.

[6] http://vifm.sourceforge.net/, 2009.

# Bibliografía


[7] AMBLER, S. W., *The Elements of UML 2.0 Style*, Cambridge, 2005.

[8] ELLEN SIEVER, AARON WEBER, S. F. R. L., *Linux in a Nutshell*, O'Reilly Media, 2005.

[9] ERICH GAMMA, RICHARD HELM, R. J. J. V., *Design Patterns*, Addison Wesley, 1998.

[10] GLASS, G., *Linux for Programmers and Users*, Prentice Hall, 2006.

[11] GRADY BOOCH, IVAR JACOBSON, J. R., *El Proceso Unificado de Desarrollo de Software*, Addison Wesley, 2000.

[12] JASMIN BLANCHETTE, M. S., *C++ GUI Programming with Qt4, 2/E*, Prentice Hall, 2008.

[13] KIM HAMILTON, R. M., *Learning UML 2.0*, O'Reilly, 2006.

[14] LARMAN, C., *UML y Patrones*, Prentice Hall, 1998.

[15] MOLKENTIN, D., *The Book of Qt 4*, No Starch Press, 2007.

[16] PETERSON, M. D., www.oreillynet.com **1** (2006).

[17] REMPT, B., www.oreillynet.com **2** (2009).

[18] SCHMULLER, J., *Aprendiendo UML en 24 Horas*, Prentice Hall, 2004.

[19] SOBELL, M. G., *Practical Guide to Linux Commands, Editors, and Shell Programming, A, 2/E*, Prentice Hall, 2009.

[20] STEWARD, S., www.oreillynet.com **1** (2006).

[21] TEAM, G., Gnu general public license, 1995.

[22] TEAM, Q. D., Qt documentation, 2009.

[23] Visual paradigm for uml - uml tool for software application development, 2010.

[24] UCID, *Proceso de Desarrollo y Gestion de Proyectos de Software*, 2009.

# Glosario de términos

**FAR** Fuerzas Armadas Revolucionarias.

**Gnome** Entorno de escritorio popular creado con las librerías de desarrollo GTK.

**KDE** Entorno de escritorio refinado creado con las librerías de desarrollo Qt.

**GNU** Sistema operativo cuyo nombre es un acrónimo recursivo que significa "GNU No es UNIX".

**Sistema de Ficheros** (File System) Es la forma o estructura en la que queda almacenada la información dentro de un dispositivo de almacenamiento.

**Nova** Distribución de Linux hecha por estudiantes y profesores de la Universidad de las Ciencias Informáticas basada en Gentoo.

**Entorno de Escritorio** Es un conjunto de aplicaciones que constituyen la interfaz gráfica de determinado sistema operativo.

**Quántico** Entorno de Escritorio ligero derivado de Antico desarrollado por el proyecto de Sistema Base de UCID.

**GPL** Licencia Pública General creada a finales de 1980 para servir de sostén legal a las aplicaciones surgidas bajo el término copyleft.

**GUI** Interfaz gráfica de usuario, es el artefacto tecnológico de un sistema inactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

**Linux** Es el núcleo (kernel) del sistema operativo libre GNU

**Qt** Biblioteca multiplataforma para el desarrollo de interfaces gráficas.

**UCID** Centro de Desarrollo de de Aplicaciones Informáticas de las FAR en la UCI.

**Antico** Entorno de Escritorio ligero basado completamente en las librerías del marco de trabajo Qt.

**IDE** Entorno de desarrollo integrado.

**UCID** Centro de Desarrollo de de Aplicaciones Informáticas de las FAR en la UCI.