Asistente de configuración para la instalación de las plataformas televisivas PTARTV y Primicia

---

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Rafael Sánchez Martín

Tutor:

Ing. Dayami Chávez Ayala

La Habana, Junio de 2011.

"Año 53 de la Revolución"

"La Ciencia es una tentativa en el sentido de lograr que la caótica diversidad de nuestras experiencias sensoriales corresponda a un sistema de pensamiento lógicamente ordenado."

Albert Einstein

Declaración de autoría

Declaro que soy el único autor de este trabajo:

Asistente para la instalación y configuración de plataformas de televisión en un entorno libre.

Autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

_____	_____
Rafael Sánchez Martín	Ing. Dayami Chávez Ayala

Datos de contacto

Tutor(a):

Nombre y apellidos: Dayami Chávez Ayala.

Sexo: __M _x_ F

Institución: Universidad de las Ciencias Informáticas.

Dirección de la institución: Carretera a San Antonio de los Baños, Km. 2 1⁄2, Reparto: Torrens, Municipio: Boyeros, Provincia: Ciudad de La Habana.

Correo electrónico: dchavez@uci.cu.

Teléfono del trabajo: -

Teléfono particular: -

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas.

Año de graduación: 2010.

Institución donde se graduó: Universidad de las Ciencias Informáticas.

Agradecimientos

A mis padres Odalys y Rafael, por darme la vida, el amor, la confianza y las fuerzas para seguir mi camino. A mis abuelos y a mis hermanos: Amanda, Jaidier y Reinier.

A mi primo Miguel Ángel, por ser siempre mi ídolo a seguir, a mis primos Michel, Katia, Agnelito, Yamuska, Rosa María y Adrián, a todos mis tíos y en general a toda mi familia por haber depositado tanta confianza en mí.

A mi novia Lisett por haberme dado tanto amor, apoyo y por estar siempre a mi lado en gran parte de mi carrera. A toda su familia por tratarme como un miembro más de la misma.

A mi mejor amigo Félix por estar siempre apoyando durante muchos años de mi vida.

A mis amigos de la universidad con los que he compartido y aprendido en las buenas y las malas: Dariel, Iván, Jorge, Rayner, Fernando.

A los amigos de mi zona con los cuales he compartido durante mi infancia y juventud: Michel, Ricardo, Javier, Eric y a todos los que han compartido conmigo de una forma u otra.

A mis compañeros y profesores del proyecto los cuales me han ayudado en la tesis en las tareas que he realizado: Dennis, Jean, Eduardo, Alex, Pedro, Reinaldo, Albrecht, Yandy, Dunier, Abel, Ángel, Frank, Magalys, Ana Lizandra, Yelen, María Luisa.

A mi tutora Dayami que siempre está atrás de mí para que no me entretenga y trabaje y por la ayuda que me ha dado durante la realización de mi tesis.

A mi tribunal y oponente por sus críticas constructivas.

A todos los que me hay apoyado y confiado en mi… gracias.

Dedicatoria

A mis padres y abuelos por darme siempre la confianza y la fuerza necesaria para seguir adelante.

A mis hermanos a los cuales quiero con la vida y a mi familia en general por estar siempre orgullosos de mí.

A mi novia por darme amor, cariño y el apoyo que necesitaba.

A todos mis amigos por estar siempre compartiendo conmigo.

Resumen

La actividad de configurar e instalar un software de dimensión elevada en sistemas operativos GNU/Linux resulta complicada. Las librerías, binarios y configuraciones de los software están almacenados en directorios independientes que a la vez son comunes para todos los programas. Esto disminuye el espacio en disco necesario para su instalación, pero complejiza el proceso de configuración por el gran número de directorios implicados en la instalación del programa.

En el Departamento de Señales Digitales del centro (GEYSED) de la Universidad de las Ciencias Informáticas (UCI) se desarrollan la Plataforma de Televisión Informativa Primicia y la Plataforma de Transmisión Abierta para Radio y Televisión (PTARTV), desarrolladas en sistemas operativos GNU/Linux. El despliegue o prueba de estas plataformas se realiza de forma manual, lo que conlleva a un gran atraso su realización.

La presente investigación surge por la necesidad de un sistema que automatice el proceso de configuración e instalación de estas plataformas televisivas para realizar su despliegue de forma rápida y efectiva. Este documento recoge los resultados obtenidos y las conclusiones de la investigación realizada. En el mismo se evidencian las tendencias y tecnologías actuales, las técnicas de programación y herramientas utilizadas para el desarrollo de la aplicación.

Se obtuvieron varios resultados como: documentación de sistemas similares y el modelado ingenieril de la solución. El resultado final es un sistema con fácil maniobrabilidad capaz de automatizar el proceso de instalación y configuración de las plataformas televisivas PTARTV y Primicia.

Palabras clave: configuración, despliegue, instalación, plataformas televisivas

Abstract

The activity of configure and install a large-scale software in the operating systems GNU/Linux can be complicated. The libraries, binaries and software configurations are stored in separate directories at the same time are common to all programs. This reduces the disk space required for installation, but it complicates the setup process for the large number of directories involved in the installation of the program.

In the Department of Digital Signals of the Center(GEYSED) at the University of Informatics Sciences(UCI) are developing the Plataforma de Televisión Informativa Primicia and the Plataforma de Televisión Abierta para Radio y Televisión(PTARTV), developed in operating systems GNU/Linux. The deployment or testing of these platforms is done manually, which involves great delay in their realization.

This research arises from the need for a system that automates the setup and installation of these television platforms for deployment quickly and effectively. This document contains the results and conclusions of the research. The same is evidenced by current trends and technologies, programming techniques and tools used for application development.

Several results were obtained as documentation of similar systems engineering and modeling of the solution. The end result is a system with easy maneuverability able to automate the process of installing and configuring television platforms PTARTV and Primicia.

Keywords: configuration, deployment, installation, TV platforms

Índice de Figuras

Índice de Tablas

Índice de contenido

Introducción

En la actualidad, la actividad de configurar e instalar un software de dimensión elevada en un sistema operativo puede resultar complicada y más cuando este software se instalará en sistemas operativos GNU/Linux. Para un usuario acostumbrado a trabajar en el sistema operativo Windows este proceso resulta mucho más difícil porque en GNU/Linux se utiliza una filosofía de instalación diferente.

Las librerías, binarios y configuraciones de los software están almacenados en directorios independientes que a la vez son comunes para todos los programas. Esta forma de organizar los programas posee una serie de ventajas pues permite que se utilicen las librerías de forma común y no teniendo una copia para cada uno de ellos. Esto disminuye el espacio en disco necesario para su instalación, pero a su vez complejiza el proceso de configuración por el gran número de directorios implicados en la instalación del programa.

Una de las variantes al instalar un programa computacional es realizarlo mediante un único archivo ejecutable, esta técnica es muy utilizada en sistemas operativos Windows. En sistemas GNU/Linux no es tan común esta alternativa, mayormente se instalan los programas desde un repositorio. Un ejecutable es un programa que ha sido traducido a código máquina[1] en un formato que puede cargarse en la memoria y ejecutarse (1).Un instalador ejecutable muestra al usuario un asistente que lo guía durante el proceso de instalación.

Un repositorio en GNU/Linux consiste en un servidor donde se almacenan programas para su posterior descarga e instalación. La instalación de un programa desde el repositorio se realiza mediante un gestor de instalación. Éste permite gestionar los programas de forma centralizada pero limitando su configuración porque los programas se instalan con la configuración predeterminada y para modificarla es necesario editar el archivo manualmente.

Actualmente un universo de posibilidades abren las comunicaciones satelitales, la telefonía inalámbrica, Internet, la televisión digital y la computación a un país como Cuba,

[1] Conjunto de instrucciones entendibles directamente por el ordenador, puesto que se componen de unos y ceros. Generalmente, el programador utiliza un lenguaje de programación basado en el lenguaje natural, y éste es traducido a código máquina posteriormente.

a pesar del bloqueo económico, comercial y financiero con el que Estados Unidos, le impide a la isla el acceso a su mercado y la obliga a invertir varias veces más recursos al tener que recurrir a mercados muy distantes. A pesar de eso, basándose sobre todo en sus recursos humanos y optimizando los recursos materiales y financieros, Cuba avanza en su informatización, priorizando el uso social y colectivo de las Tecnologías de la Información y las Comunicaciones (TICs) lo que da al país una connotación diferenciada al resto de los países del mundo en cuanto a Tecnologías de la Información se trata (2).

El poderoso auge de las TIC ha cambiado los paradigmas y estrategias reconocidas y establecidas por muchos años como válidas. Dentro de las TIC, la industria del software alcanza una posición relevante. En el mundo existen varias empresas y organismos dedicados a la producción de software y servicios teniendo como ejemplo: ADCO Development, IQual Ingenieros, Omnisciens y muchas más. En Cuba con el desarrollo de la informática también se trabaja en este sentido por lo cual existen ya varias empresas dedicadas a este sector, teniendo como ejemplo de estas empresas: Citmatel, Avante, Desoft, SoftCal, Albet. El origen y desarrollo de esta última empresa está vinculado estrechamente a la Universidad de Ciencias Informáticas (UCI). Albet posee los derechos comerciales de todos los productos y servicios que desarrolla la UCI y mediante la alianza con otras prestigiosas entidades ofrece soluciones integrales en la esfera de las tecnologías de la información y las comunicaciones. La UCI es un modelo de universidad productiva que agrupa más de doce mil profesionales, técnicos y estudiantes (3). Esta universidad fue creada el 22 de septiembre de 2002, siendo actualmente uno de los pilares de mayor importancia para la economía del país en la industria del software. En la UCI se desarrollan diversos proyectos de innovación, investigación y desarrollo que utilizan software libre, con el objetivo de alcanzar la soberanía tecnológica del país.

Enmarcado en la importancia de las nuevas tecnologías, desde un principio fueron creados en la universidad varios canales de televisión con fines educativos, culturales, informativos así como de entretenimiento. Todos estos canales de televisión son controlados por la Dirección de Televisión Universitaria (DTU) apoyada por el Departamento de Señales Digitales del centro Geoinformática y Señales Digitales (GEYSED) de la Facultad 6. En este departamento se desarrollan la Plataforma de Transmisión Abierta para Radio y Televisión (PTARTV) y la Plataforma de Televisión Informativa Primicia, desarrolladas sobre arquitecturas similares. Estas plataformas no

cuentan con un instalador y configurador automático de sus proyectos que permita realizar su despliegue de forma rápida y efectiva.

Configurar e instalar en un entorno de despliegue un proyecto desarrollado por el Departamento de Señales Digitales es complejo pues es necesario modificar gran número de ficheros ubicados en diferentes carpetas e instalar y configurar programas de los que depende. Este proceso hay que repetirlo en cada una de las computadoras que requiera esta instalación. Actualmente esta instalación y configuración se efectúa de forma manual lo que conlleva a un mayor atraso al realizar el despliegue o prueba de cualquiera de los proyectos en general. Además existe la posibilidad de que ocurra algún error durante este proceso perdiendo todas las configuraciones realizadas lo que implica una demora adicional.

De la anterior situación se identifica como **problema a resolver:** Los procedimientos actuales para la instalación de plataformas con arquitectura común, desarrolladas en el Departamento Señales Digitales, provocan dificultades en el despliegue de las mismas.

La investigación posee como **objeto de estudio:** Los procesos para la configuración e instalación de aplicaciones informáticas en entornos libres y tiene como **campo de acción:** Los procesos de configuración e instalación de las plataformas radiales y televisivas desarrolladas en el Departamento de Señales Digitales que posean arquitecturas similares.

Se propone como **objetivo general** de la investigación: Desarrollar una herramienta para la configuración e instalación de las plataformas desarrolladas en el Departamento de Señales Digitales que posean arquitecturas similares.

Por lo planteado anteriormente se asume como **idea a defender** que: El desarrollo de una herramienta para la configuración e instalación de las plataformas desarrolladas con arquitecturas similares en el Departamento de Señales Digitales facilitará el despliegue de dichas plataformas.

Para dar cumplimiento al objetivo propuesto se definen las siguientes tareas de la investigación:

1. Caracterizar las técnicas y procedimientos de instalación y configuración de aplicaciones o sistemas en entornos libres.

2. Caracterizar los procesos de instalación y configuración de plataformas desarrolladas en el departamento de Señales Digitales.
3. Determinar las tendencias y tecnologías actuales más factibles para el desarrollo de la aplicación.
4. Desarrollar artefactos y documentación según la metodología de desarrollo seleccionada.
5. Implementar las funcionalidades del sistema.
6. Validar los resultados obtenidos en la investigación desarrollada.

Para la realización de estas tareas se emplearon diferentes métodos de la investigación científica.

Métodos Teóricos:

Histórico – Lógico: Permitió determinar las características de las principales tendencias y etapas de los instaladores en el sistema operativo GNU\Linux. También posibilitó realizar el estudio de la trayectoria real, el desarrollo y evolución de los elementos que sirvieron como guía para la construcción de un asistente de instalación y configuración de plataformas de televisión en entornos libres.

Analítico-Sintético: Permitió valorar y definir las herramientas necesarias en los procesos de instalación y configuración de diferentes aplicaciones en sistemas operativos basados en GNU/Linux.

Modelación: En el desarrollo de la investigación se utilizó este método para hacer modelos que ofrecieron la posibilidad de crear abstracciones para explicar la realidad. Se hizo visible en el trabajo al crear modelos como el de implementación.

Métodos Empíricos:

Observación: Facilitó conocer el panorama real de una situación mediante la percepción directa, permitió caracterizar y analizar detalladamente cómo se realiza actualmente el proceso de instalación en los sistemas GNU\Linux.

Entrevista: Se realizaron entrevistas con los integrantes de los proyectos PTARTV y Primicia ambos pertenecientes al Departamento de Señales Digitales, con el objetivo de conocer las principales tendencias de la instalación y configuración de los software necesarios para el funcionamiento de estas aplicaciones. Para esto se tomó como población a los trabajadores y estudiantes de cada uno de esos proyectos, constituyendo una población de 62 miembros. De ellos como muestra se eligieron 16 compañeros para entrevistar lo que representa el 25,8% de la población. Se realizó una técnica de muestreo no probabilístico, donde el investigador elige los casos que más le interesan para una información más rica, específicamente muestreo intencional. Con la entrevista realizada se logró conocer los softwares y configuraciones necesarias para el correcto funcionamiento de las plataformas televisivas PTARTV y Primicia.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se aborda sobre el estado actual de los procesos para la creación de instaladores de software. Se elabora una pequeña descripción y comparación de las principales características de herramientas que brindan soporte para la creación de estos instaladores. También se describe el estado actual de la instalación de plataformas en entornos tanto web como escritorio.

1.2 Conceptos asociados

1.2.1 Asistente

Asistente en informática es el programa que guía al usuario, paso a paso en la realización de un proceso.

1.2.2 Configuración

La configuración es la adaptación de una aplicación software o un elemento hardware al resto de los elementos del entorno y a las necesidades específicas del usuario.

1.2.3 Sistema

Del latín systema, un sistema es módulo ordenado de elementos que se encuentran interrelacionados y que interactúan entre sí. (4)

1.2.3.1 Sistema de información

Un sistema de información es un conjunto organizado de elementos, que pueden ser personas, datos, actividades o recursos materiales en general. Estos elementos interactúan entre sí para procesar información y distribuirla de manera adecuada en función de los objetivos de una organización. (4)

Se define como sistema al conjunto de elementos que interactúan entre sí con el objetivo de llegar a un resultado común.

1.3 Objeto de estudio

1.3.1 Descripción General

1.3.1.1 Proceso de Despliegue del Software

El despliegue de un sistema de software involucra la transferencia o copia de sus componentes desde el lado del proveedor hacia el lado del cliente, este constituye su principal objetivo. Una vez desplegado, dicho sistema estará disponible para su uso en el lado del cliente.

En el Departamento de Señales Digitales actualmente se desarrollan 8 proyectos de innovación, investigación y desarrollo los cuales se desarrollan con la metodología RUP (Proceso Unificado de Software). Esta metodología es muy utilizada para la documentación y realización de software orientados a objetos. RUP propone varias iteraciones en el proceso de desarrollo de software lo que posibilita la corrección de errores y mejoras del software a medida que avanza el proyecto. El desarrollador cuenta con experiencia en el uso de esta metodología y además existe una abundante documentación de la misma lo que permite rapidez en el desarrollo del proyecto. Por estas características mencionadas se selecciona como metodología a utilizar RUP.

1.3.1.2 Proceso de Despliegue del Software en RUP

Este proceso de despliegue está contenido dentro de los nueve flujos de trabajo básicos en el Proceso Unificado de Software (RUP) formando parte de las fases de Elaboración, Construcción y Transición de esta metodología de desarrollo.

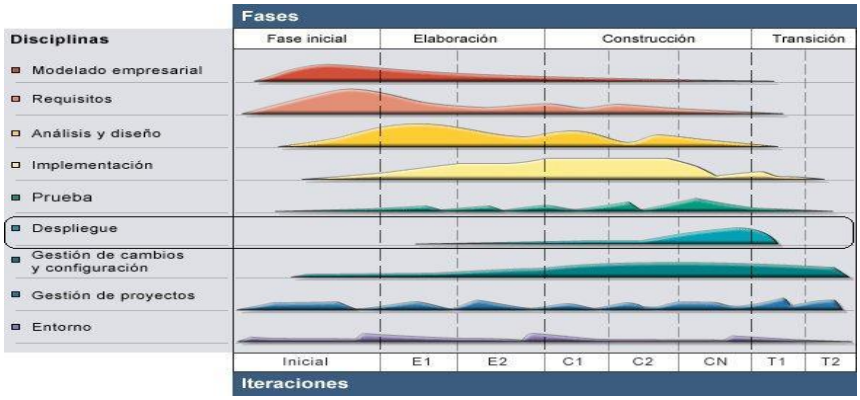


Figura 1. El Despliegue como flujo de trabajo en RUP (5)

Este flujo de trabajo para RUP tiene como objetivo producir y entregar con éxito versiones del producto a los usuarios finales.

Durante el flujo de trabajo de despliegue se realizan una amplia gama de actividades, de ellas el sistema tiene más incidencia en las siguientes:

- Producción de versiones externas del software.
- Empaquetado del software.
- Distribución del software.
- Instalación del software.

Aunque las actividades de despliegue son en su mayoría centradas en la fase de transición, muchas de ellas necesitan ser incluidas en fases anteriores para preparar el despliegue a finales de la fase de construcción.

1.3.1.3 Actividades que se llevan a cabo durante el flujo de trabajo de despliegue

El proceso de despliegue es generalmente descrito como un ciclo de vida que involucra varios subprocesos o tareas interrelacionadas posteriores al desarrollo.

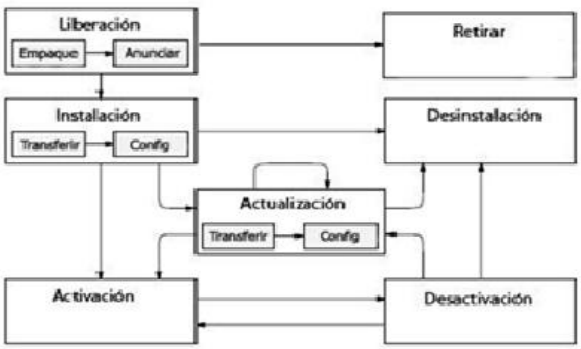


Figura 2. Actividades del flujo de trabajo de despliegue de software. (6)

Actividades del flujo de trabajo de despliegue:

Liberación: Constituye la interfaz entre el proceso de desarrollo y el proceso de despliegue. Abarca todas las operaciones necesarias para preparar un producto y que pueda ser correctamente implantado en el sitio del consumo. Así, la liberación debe

determinar todos los recursos necesarios por un sistema de software para operar correctamente en el sitio de consumo. También debe recoger toda la información necesaria para la realización de actividades posteriores del proceso de despliegue. Esta información puede ser derivada de una variedad de fuentes, entre ellas el proceso de desarrollo y el conocimiento humano sobre la estructura y funcionamiento del sistema. La liberación incluye el empaquetado del producto para que pueda ser transportado. Este paquete debe contener los componentes y una descripción del sistema que incluye los requisitos y las dependencias de otros componentes externos, los procedimientos de instalación, y toda la información que sea relevante para la gestión de la aplicación. Otro paso en la liberación es la capacitación, es decir, el conjunto de operaciones que se necesitan para difundir la información pertinente a las partes interesadas en las características y la utilización del sistema que se va a desplegar (7).

La liberación de un software es el proceso que ocurre antes de su instalación. En este proceso se definen las características necesarias para que el sistema funcione correctamente. También se recoge toda la información necesaria para entender las tareas a realizar después del proceso de despliegue y para la capacitación al personal sobre la estructura y funcionamiento del sistema. Por último se debe dejar listo al software para su posterior instalación.

Instalación: Comprende la inserción inicial de un sistema en el lado del cliente. Por lo general, es la más compleja de las actividades de despliegue, ya que se ocupa de la correcta concentración de todos los recursos necesarios para utilizar el producto de software. Con el término Instalación se hace referencia a dos tareas principales. La primera de ellas es la transferencia y entrega del producto desde el proveedor hasta el cliente en el sitio a ser utilizado. La segunda consiste en todas las operaciones de configuración que son necesarias para que el sistema esté listo para la activación (7).

La instalación es el proceso por el cual el nuevo software es transferido y configurado en el lugar a ser utilizado por el cliente, dejándolo listo para su activación.

Activación: Se refiere a la ejecución de aquellos componentes de un sistema que son necesarios para el funcionamiento del producto que se va a desplegar. Para una herramienta simple, la activación involucra el establecimiento de alguna forma de mando (hacer clic en el icono gráfico) para la ejecución de componentes binarios de la

herramienta. Para un sistema complejo, podría ser necesario iniciar servidores demonios antes de que el sistema de software sea activado. Tener en cuenta que el proceso de instalación en sí mismo, puede requerir la invocación de otros instrumentos y posiblemente, la instalación de estos (7).

La activación es la ejecución de los componentes necesarios para el funcionamiento del producto que se está desplegando, esto constituye también su validación.

Desactivación: Constituye la actividad inversa de la activación, y se refiere al cierre de cualquier componente en ejecución de un sistema instalado. En general, la desactivación es necesaria antes de que otras actividades de despliegue se puedan llevar a cabo tales como la actualización y la desinstalación (7).

La desactivación como bien dice su concepto es la actividad inversa a la activación, esta actividad es necesaria para poder desinstalar o actualizar el software que se está desplegando.

Desinstalación: En algún momento, un sistema en su conjunto ya no es necesario en un determinado sitio de consumo y puede ser eliminado. Se supone que la desinstalación está precedida por la desactivación. La actividad posiblemente implique una modificación de las configuraciones de otros sistemas, así como la retirada de los archivos pertenecientes a la aplicación que ha de ser desinstalada. La desinstalación no es necesariamente un proceso trivial. Una de las razones es que este proceso podrá asumir ser el único usuario de algún recurso compartido, como por ejemplo diversos archivos de datos, por lo que pueden liberar esos recursos y causar que otros sistemas puedan fallar (7).

La desinstalación es la actividad que precede la desactivación, esta actividad implica la eliminación de todos los archivos pertenecientes a la aplicación que va a ser desinstalada.

Actualización: Esta actividad constituye un caso especial de la instalación. Es por lo general menos compleja debido a que muchos de los recursos necesarios ya se han obtenido durante el proceso de instalación. Normalmente, el ciclo de vida del despliegue incluye una secuencia repetida en el que un sistema es desactivado, se instala una nueva versión y este vuelve a ser activado. Para algunas aplicaciones, la desactivación puede no ser necesaria y la actualización se puede realizar al mismo tiempo donde una versión

anterior está aún activa. De manera similar a la instalación, la actualización incluye la transferencia de todos los componentes necesarios para completar la operación (7).

La actualización es la actividad donde se remplaza la versión instalada del software por una nueva versión, esta actividad al igual que la instalación implica la transferencia y configuración de todos los archivos necesarios para el funcionamiento del software en el lugar a ser utilizado por el cliente.

Estas actividades anteriormente descritas son parte de un mismo flujo de trabajo. En el despliegue de un producto informático es muy importante la integración de cada una de ellas porque en este punto se necesita un mecanismo que acople todo este proceso lógico de una forma automatizada, la herramienta que posibilita este acoplamiento es llamada instalador de software.

1.3.2 Descripción actual del dominio del problema

En la UCI se encuentra el centro GEYSED con el objetivo de desarrollar productos, servicios y soluciones informáticas en el campo del procesamiento de Señales Digitales y en el mundo de la Geociencia, permitiendo un posicionamiento en el mercado nacional y latinoamericano. Este centro cuenta con el Departamento de Señales Digitales el cual es un colectivo formado por profesores y estudiantes de diferentes años de la UCI, que se encargan del desarrollo de productos, sistemas, servicios y soluciones informáticas en el campo del procesamiento digital de imágenes y señales. Uno de sus objetivos lo constituye el desarrollo de componentes de alto valor agregado relacionados con el tema de procesamiento digital de imágenes y señales para ser comercializados con productos de otros centros. Además promueve la implantación de aplicaciones especializadas en diversos sectores bajo plataforma de software libre.

El departamento orienta las investigaciones en tres líneas fundamentales que tributan a la formación de pregrado y postgrado, además de guiar el desarrollo de los productos:

- Sistemas de Transmisión de Señales.
- Recuperación, Gestión y Procesamiento de Medias.
- Procesamiento Digital de Imágenes y Señales.

Dentro de los proyectos que se desarrollan en este departamento se encuentran la Plataforma de Transmisión Abierta para Radio y Televisión y la Plataforma de Televisión Informativa Primicia. Ambas plataformas son abordadas en la investigación en curso. Estas plataformas en su etapa de despliegue realizan la instalación y configuración de sus softwares y dependencias de forma manual, la realización de este proceso conlleva a una gran demora.

1.3.3 Situación Problemática

1.3.3.1 Plataformas de transmisión

Se define como plataforma de transmisión, al conjunto de elementos interconectados a partir de la base de intercambios y de conectividad, gestionado por ciertas reglas, ciertas capacidades y ciertas limitaciones en un ambiente determinado, permitiendo la realización de numerosas tareas referentes a la acción de transmitir contenidos por un medio; integra la posibilidad de controlar el proceso de transmisión.

Plataforma de Transmisión Abierta para Radio y Televisión (PTARTV)

PTARTV constituye una solución tecnológica integral que automatiza la transmisión y administración de los canales de radio y televisión. A su vez provee funcionalidades para gestionar y controlar todos los procesos de gestión y procesamiento de material multimedia. Además brinda la posibilidad de utilizar la web como medio de difusión de materiales audiovisuales.

El sistema informático PTARTV está estructurado en doce subsistemas, estos son:

Subsistemas	Tipo de aplicación
Administración de la transmisión	Escritorio
Transmisión	Escritorio
Monitoreo	Escritorio
Radial	Escritorio
Gestión de medias	Escritorio
Programación	Web

Seguridad	Web
Transferencia	Web
Web	Web
Reporte	Web
Equipamiento	Web
Producción	Web

Tabla 1. Subsistemas de PTARTV

Plataforma de Televisión Informativa Primicia

Primicia es un producto informático que se ha estado desarrollando desde el año 2005 por la UCI, constituyendo una solución integral capaz de proveer un canal de televisión para la transmisión automática y constante de informaciones en distintos formatos. El sistema informático ha sido desarrollado y es soportado completamente en software libre.

El sistema informático Primicia se encuentra estructurado en dos subsistemas, estos son:

Subsistemas	Tipo de aplicación
Transmisión	Escritorio
Administración	Web

Tabla 2. Subsistemas de Primicia

Ambas plataformas están desarrolladas en dos entornos al mismo tiempo, web y escritorio, por lo que realizar el despliegue de las mismas es muy complejo. Actualmente el mismo se realiza de forma manual instalando cada subsistema independiente del otro. Este proceso es realizado por un especialista de cada plataforma por la complejidad de los requerimientos de cada una. Para agilizar este proceso es necesario la creación de un instalador que permita la instalación y configuración de los subsistemas de estas plataformas sin importar el entorno en el que se desplieguen.

1.4 Análisis de otras soluciones existentes

1.4.1 Instaladores de Software

En la actualidad es muy amplio el número de instaladores de software, siendo poco común ver una aplicación sin su propio instalador. Un instalador de software es una

herramienta que guía al usuario mientras la aplicación se despliega, desvinculando al usuario de configuraciones engorrosas. Estas aplicaciones permiten que cualquier usuario pueda instalar un determinado producto sin tener conocimientos avanzados sobre ellos.

1.4.2 Tipos de instaladores de Software

Debido a la gran variedad de aplicaciones que existen actualmente en el mundo de la informática se dividen en la investigación en dos grupos:

- Instaladores estándares.
- Sistema de gestión de paquetes.

Los instaladores estándares son herramientas específicas con el objetivo de desplegar solamente los archivos que ellas contienen.

Un paquete de software es un conjunto de uno o más archivos que son necesarios para la ejecución de un programa de computación o añadir a las características de un programa ya instalado en uno o más ordenadores (8).

En los sistemas GNU/Linux una instalación de una aplicación puede abarcar una gran cantidad de paquetes, los sistemas de gestión de paquetes son herramientas encargadas de automatizar este proceso de instalación, actualización, configuración y eliminación de estos paquetes. Un paquete está compuesto por el nombre completo del software, su descripción, número de versión, proveedores y una lista completa de las dependencias para su funcionamiento.

Aunque hay una gran variedad de aplicaciones, se encuentran pocas herramientas que automaticen el proceso completo de instalación de una aplicación desarrollada en entorno web y escritorio. En los pocos casos que existen estas herramientas requieren configuraciones manuales complejas por lo que debe hacerlas un especialista en la aplicación que se quiere instalar.

Actualmente la mayoría de los instaladores que existen son para aplicaciones escritorio solamente, debido a la amplia cantidad de herramientas que existen para su confección haciendo más fácil su desarrollo.

1.4.3 Herramientas para la creación de instaladores de software

Actualmente hay una infinidad de instaladores los cuales pueden ser encontrados bajo licencia comercial de software propietario o sobre código abierto, libre o gratuito.

A continuación varios ejemplos de instaladores de software y una breve descripción de los mismos:

Herramientas bajo licencia comercial de software propietario

Microsoft Windows Installer: es un servicio de instalación y configuración proporcionado por Windows. El servicio de instalación permite a los clientes realizar el despliegue en las empresas y proporciona un formato estándar para la gestión de los componentes (9).

InstallShield: es una herramienta para crear instaladores o paquetes de software. Esta herramienta es usada sobre todo para instalar software del escritorio y las plataformas de servidor de Microsoft Windows, pero también se puede usar para administrar aplicaciones y paquetes de software en una amplia gama de dispositivos móviles y portátiles (10).

SetupBuilder: es un rápido desarrollador de instaladores y una herramienta de gestión de la configuración para sistemas operativos Windows, tales como: Windows Server 2008 ("Longhorn"), Windows Vista, Windows 2003, Windows XP, Windows 2000, Windows NT4 y Windows Me/98/95. Ofrece el conjunto completo de las características que se necesitan para construir instaladores y actualizadores robustos para este sistema operativo. Se trata de una potente, flexible, rápida e intuitiva herramienta basada en scripts utilizada en la gestión de configuración.

Herramientas sobre código abierto, libre o gratuito

NSIS (*NullsoftInstall scripts System*): es un sistema profesional de código abierto para crear instaladores sobre Windows. Está diseñado para construir instaladores pequeños y flexibles tanto como sea posible y, por lo tanto, es muy apropiado para la distribución por Internet. NSIS soporta diferentes idiomas, está basado en scripts y permite generar la lógica para manejar incluso las más complejas tareas de instalación.

ClickteamInstallCreator: es una herramienta que permite crear sistemas de instalación llevando todo el proceso paso a paso a través de un asistente. Estos pasos comprenden la selección de ficheros que incluye el programa, la elección del directorio donde se

instalarán, elección de textos y logos que aparecen, creación de iconos de acceso directo, entre otros, hasta llegar a la compilación de la instalación dejándola lista para su uso.

IzPack: es una herramienta para la creación de instaladores basado en tecnología Java. Es un proyecto cuyo diseño modular permite definir cómo debe verse el instalador, además es muy configurable mediante un API. Dentro de sus características fundamentales se encuentra el soporte multiplataforma que le permite ejecutarse en cualquier sistema operativo que tenga instalada la máquina virtual de Java.

1.4.4 Selección de la herramienta a utilizar

Dada las herramientas anteriormente descritas se muestra una comparación de acuerdo a las principales características que poseen. Los criterios a medir en la siguiente comparación son: entorno de instalación, entorno de desarrollo, licencia, presencia de componentes gráficos (GUI) y tipo de archivo que manejan. Se seleccionaron estos criterios para la comparación porque el producto final debe estar desarrollado sobre Linux con licencia libre, también debe ser desarrollado con un entorno gráfico para el manejo fácil de los usuarios que utilicen la aplicación final. Para comprobar que estos requerimientos se cumplan hay que tener en cuenta estas características de la comparación realizada:

Herramienta	**Entorno de Instalación**		**Entorno de Desarrollo**		**Licencia**	**Modos de Instalación**		**Tipo de Archivo**
	Linux	**Windows**	**Linux**	**Windows**		**Texto**	**Gráfico**	
Microsoft Windows Installer		x		x	Comercial		x	Script
InstallShield	x	x	x	x	Comercial	x	x	Script
SetupBuilder		x		x	Comercial		x	Script
NSIS		x	x	x	CPL	x	x	Script
ClickteamInst		x		x	Libre		x	Script

allCreator								
IzPack	x	x	x	x	GPL		x	XML

Tabla 3. Comparación entre herramientas para la creación de instaladores

Las herramientas descritas precedentemente no cumplen con los requerimientos que se necesitan para dar cumplimiento a la problemática de la investigación, pues estas solo se especializan en un entorno en particular, web o escritorio. Para dar solución a este problema es necesaria la creación de un instalador que vincule las instalaciones y configuraciones en los dos entornos, sin necesidad de tener dos programas diferentes para realizar el despliegue de alguna de las plataformas mencionadas en la investigación en curso.

1.5 Conclusiones parciales

En el presente capítulo se expusieron los principales conceptos asociados a la investigación en curso para una mejor comprensión de la misma. Se realizó una descripción general del proceso de despliegue de software describiendo detalladamente cada una de sus actividades en la metodología RUP.

Se describieron de forma general las plataformas abordadas en la presente investigación y el departamento en el cual estas se desarrollan. También se ha hecho una breve descripción de herramientas de instalación y configuración de softwares informáticos a las cuales se les realizó una breve comparación de sus principales características.

Estas herramientas descritas no le dan solución a la problemática de la investigación en curso. Se puede concluir que en la actualidad no se cuenta con una herramienta capaz de realizar la instalación y configuración completa de plataformas desarrolladas en entornos web y escritorio al mismo tiempo. Siendo así se propone la creación de un instalador que permita configurar e instalar plataformas radiales y televisivas en software libre, desarrolladas en entornos web y escritorio que posean una arquitectura común.

Capítulo 2: Tendencias y tecnologías actuales a desarrollar.

2.1 Introducción

En este capítulo se realiza un análisis de las tendencias y tecnologías actuales a considerar para ser empleadas en la construcción del software que dará solución al problema de investigación. Se analizan la metodología de desarrollo de software a utilizar, el Lenguaje Unificado de Modelado (UML) con su herramienta CASE. También se realiza una descripción del lenguaje de programación seleccionado así como el Entorno de Desarrollo Integrado (IDE) donde se va a poner en práctica este lenguaje para el desarrollo del software propuesto.

2.2 Metodología de desarrollo

Una pregunta fundamental en el desarrollo de cualquier software es ¿qué metodología utilizar? pues todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, no se obtiene el resultado deseado y el cliente no queda satisfecho. Estas en su mayoría establecen cómo trabajar eficientemente evitando los errores que conllevan al fracaso de los proyectos. Una buena metodología aumenta la calidad del software que se produce en todas y cada una de sus fases de desarrollo, por medio de una mayor transparencia y control sobre el proceso.

Para el desarrollo de esta investigación se determinó con anterioridad que la metodología idónea a utilizar es el Proceso Unificado de Software (RUP). A continuación se describe detalladamente las características que la distinguen de otras metodologías existentes.

2.2.1 Rational Unified Process (RUP)

RUP es una metodología estándar utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Como metodología, RUP es un proceso para el desarrollo de un proyecto de un software que define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto.

Descripción general

- El RUP es un proceso de ingeniería de software.
- Utiliza el paradigma de orientación a objetos para su descripción.

- Es un marco de proceso configurable para satisfacer necesidades específicas.
- Implementa las mejores prácticas de desarrollo de software.

Las tres características esenciales que presenta esta metodología son: (11)

- Dirigido por los Casos de Uso: orientan el proyecto a la importancia para el usuario y lo que este quiere.
- Está centrado en la arquitectura: relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden.
- Es iterativo e incremental: donde divide el proyecto en mini-proyectos donde los Casos de Uso y la arquitectura cumplen sus objetivos de manera más depurada.

RUP está basado en modelos y utiliza un lenguaje bien definido para tal fin, Lenguaje Unificado de Modelado, por lo que se hace necesario detallar algunas características de este.

2.3 Lenguaje Unificado de Modelado (UML)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. (12)

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. Es importante insistir que UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema. (13)

La representación en UML de un software está formada por las 4+1 vistas o modelos parciales separados, relacionados entre sí, estas vistas son:

- Vista de Casos de Uso.
- Vista lógica.
- Vista de procesos.

- Vista de implementación.
- Vista de despliegue.

Numerosos autores definen que UML no es, en sí, un lenguaje de descripción arquitectónica pues su forma de expresar ciertas características, sobre todo dinámicas de las estructuras no es suficiente para los arquitectos. (14)

Para realizar la modelación con este lenguaje se utilizan herramientas que se han desarrollado propiamente para ello, estas reciben el nombre de herramientas CASE.

2.4 Herramientas CASE

Las herramientas (Computer Aided Software Engineering o Ingeniería de Software Asistida por Ordenador) CASE son numerosas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Seleccionar una herramienta CASE no es una tarea simple. No existe una 'mejor' herramienta respecto de otra. Hay numerosas historias respecto al uso de CASE y las fallas que pueden producirse. Las fallas o las respuestas satisfactorias están en relación con las expectativas. Si el proceso de evaluación y selección de las herramientas CASE falla, entonces la herramienta no cumplirá con las especificaciones o expectativas del negocio. Esto puede ocurrir durante el proceso de implementación o ejecución del producto. (15)

Estas herramientas ayudan 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.

2.4.1 Visual Paradigm

Visual Paradigm es una herramienta colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto, constituye una herramienta profesional para el modelado UML que soporta el ciclo de vida completo del desarrollo de software.

Visual Paradigm es una herramienta que se utiliza para realizar modelado UML siguiendo el estándar UML 2.1. Esta herramienta tiene características gráficas muy cómodas que facilitan la realización de los siguientes diagramas de modelado: Diagramas de Clase, Casos de Uso, Comunicación, Secuencia, Estado, Actividad, Componentes, etc.

Entre las características más importantes que presenta esta herramienta se encuentran: (16)

-Integración con diversas IDE's como son:

- NetBeans (de Sun)
- JDeveloper (de Oracle)
- Eclipse (de IBM)
- JBuilder (de Borland)

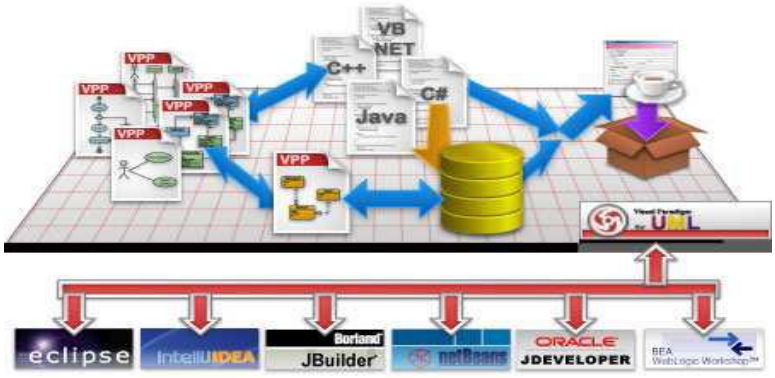


Figura 3. Compatibilidad con Entornos de Desarrollo y Lenguajes. (16)

-Ingeniería Inversa para:

- JAVA
- .NET
- XML
- Hibernate

-Exportación de imágenes jpg, png y svg (w3g estándar)

-Es Multiplataforma (incluye GNU\Linux).

Visual Paradigm es la herramienta seleccionada pues emplea UML para el modelado, es la herramienta por excelencia para ser utilizada en software libre. Permite crear de forma visual diferentes tipos de diagramas. Es muy sencillo de usar, instalar y actualizar.

2.5 Lenguaje de programación

Los lenguajes de programación son un conjunto de símbolos, sintaxis y reglas semánticas que permiten la comunicación entre una persona y un ordenador. Constituyen una técnica estándar de comunicación para entregarle instrucciones a la computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático.

Un lenguaje de programación permite a un programador especificar de manera precisa sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. (17)

Entre los principales criterios asumidos para seleccionar un lenguaje de programación están: rendimiento (velocidad de ejecución, estabilidad), flexibilidad (existencia de bibliotecas de funciones, clases), disposición de documentación viable, unido a la facilidad de uso y aprendizaje del mismo.

2.5.1 Lenguaje C++

El lenguaje de programación C++ es uno de los más empleados en la actualidad. Se puede decir que C++ es un lenguaje híbrido, ya que permite programar tanto en estilo procedimental (como si fuese C), como en estilo orientado a objetos, como en ambos a la vez. Además, también se puede emplear mediante programación basada en eventos para crear programas que usen interfaz gráfico de usuario.

El nacimiento de C++ se sitúa en el año 1980, cuando Bjarne Stroustrup, de los laboratorios Bell, desarrolló una extensión de C llamada “C with Classes" que permitía aplicar los conceptos de la programación orientada a objetos con el lenguaje C. Stroustrup se basó en las características de orientación a objetos del lenguaje de programación Simula, aunque también tomó ideas de otros lenguajes importantes de la época como ALGOL68 o ADA. Durante los siguientes años, Stroustrup continuó el desarrollo del nuevo lenguaje y en 1983 se acuñó el término C++. (18)

Las principales ventajas que presenta el lenguaje C++ son:

- **Difusión:** al ser uno de los lenguajes más empleados en la actualidad, posee un gran número de usuarios y existe una gran cantidad de libros, cursos y páginas web dedicadas a él.
- **Versatilidad:** C++ es un lenguaje de propósito general, por lo que se puede emplear para resolver cualquier tipo de problema.
- **Portabilidad:** el lenguaje está estandarizado y un mismo código fuente se puede compilar en diversas plataformas.
- **Eficiencia:** C++ es uno de los lenguajes más rápidos en cuanto ejecución.
- **Herramientas:** existe una gran cantidad de compiladores, depuradores, librerías, entre otros.

Este lenguaje cuenta con la ventaja, a excepción del ensamblador, de generar los programas más compactos y rápido. El código es portable y podrá ejecutarse en cualquier máquina y bajo cualquier sistema operativo. Y si es necesario, proporcionan un acceso a bajo nivel de hardware sólo igualado por el ensamblador. Por estas características y las anteriormente señaladas se selecciona para la implementación de la solución propuesta el lenguaje de programación C++.

C++ es el lenguaje nativo que utiliza “QT Creator” el cual es IDE a utilizar para la implementación de la solución propuesta.

2.6 Entorno de Desarrollo Integrado (IDE)

Un (Integrated Development Environment) IDE es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse exclusivamente a un solo lenguaje de programación o bien, poder utilizarse para varios. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica de usuario. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

2.6.1 QT Creator

Qt Creator es una nueva plataforma de entorno de desarrollo integrado disponible junto con las bibliotecas Qt, bajo licencia LGPL. Qt Creator está diseñado para el trabajo con bibliotecas Qt para sus versiones 4.x, en este caso se define utilizar la versión 4.7. Estas bibliotecas de software son desarrolladas por Nokia para la creación de interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores.

Entre sus principales características se destacan: (19)

- Avanzado editor de código C++.
- Soporta los lenguajes C#, .NET, Python, Ada, Pascal, Perl, PHP y Ruby
- Trae un GUI integrado y un diseñador de formularios.
- Herramientas para la administración de proyectos.
- Ayuda sensible al contexto.
- Depurador visual
- Resaltado y autocompletado de código

Además de las características antes mencionadas se debe resaltar que las bibliotecas Qt son multiplataforma y las aplicaciones que se apoyan en ellas tienen buena respuesta y un consumo de recursos aceptable.

2.7 Conclusiones parciales

En el presente capítulo se realizó un estudio y análisis de las principales tendencias y tecnologías actuales para ser empleadas en la construcción del software que dará solución al problema de la investigación. Llegando a la conclusión de que la metodología de software a utilizar para obtener el resultado deseado y evitar errores que conlleven al fracaso del proyecto es RUP, desarrollando una descripción general de la misma así como del Lenguaje Unificado de Modelado. También se describió la herramienta CASE para la realización de la modelación de este lenguaje, llegando a la conclusión de que la

herramienta más factible a utilizar es el Visual Paradigm por una serie de características que presenta, descritas en el transcurso del capítulo.

Para la construcción del software que dará solución al problema de la investigación se decidió utilizar como lenguaje de programación C++, realizando una detallada descripción de sus características más importantes. Este es el lenguaje nativo utilizado por el IDE de desarrollo QT Creator, el cual es seleccionado para la implementación de la solución propuesta porque brinda gran facilidad al trabajar con él, además de que cuenta con amplia documentación sobre el trabajo con el mismo.

Capítulo 3: Presentación de la solución propuesta

3.1 Introducción

En este capítulo se analiza el modelo de dominio donde se describen los conceptos y principales eventos del entorno de la aplicación, también se presenta el Diagrama de Clases del Modelo de Dominio así como su glosario de términos. Se describen detalladamente los requisitos tanto funcionales como no funcionales y por último se realiza la descripción de la solución propuesta presentando el Diagrama de Casos de Uso del Sistema así como la descripción detallada de los Casos de Uso que se implementarán en la propuesta de solución.

3.2 Modelo de dominio

Un Modelo de Dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física. Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema (20).

El modelo de dominio tiene como objetivo la realización de un detallado análisis de los objetos como paso previo al diseño del sistema. Este modelo es usado por los analistas del sistema como un medio para el entendimiento del negocio a informatizar por el sistema a desarrollar.

3.2.1 Conceptos y principales eventos del entorno

La Plataforma de Transmisión Abierta para Radio y Televisión es un producto informático capaz de difundir informaciones mediante un sitio web o una aplicación de escritorio, haciendo uso de las nuevas tecnologías, logrando que se pueda acceder al contenido audiovisual que pudiera emitirse por estos medios en todo momento.

La Plataforma de Televisión Informativa Primicia es un producto informático capaz de proveer un canal de televisión para la transmisión automática y constante de informaciones en distintos formatos.

La instalación y configuración de estas plataformas o de alguno de sus subsistemas en específico es realizada de forma manual por un especialista con amplios conocimientos de las mismas. Este proceso conlleva la copia de varios archivos y carpetas, así como la instalación de gran número de software. Después tiene lugar un arduo proceso de configuración de varios de los softwares anteriormente instalados.

A continuación se muestra el diagrama de clases del modelo de dominio para una mejor compresión de lo descrito anteriormente.

3.2.2 Diagrama de Clases del Modelo de Dominio

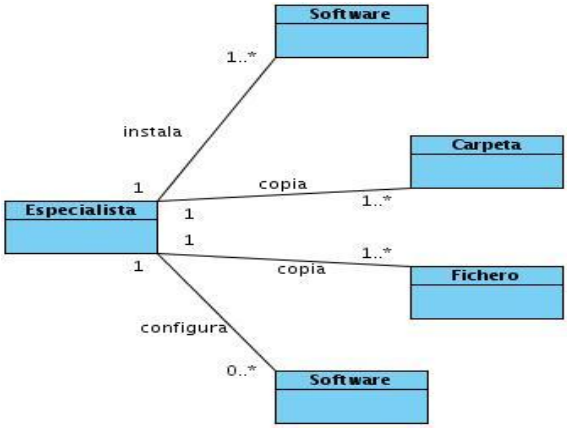


Figura 4. Diagrama de Clases del Modelo de Dominio.

3.2.3 Glosario de Términos del Dominio

Especialista: Todo aquel personal que tenga un conocimiento avanzado de la instalación y configuración de los subsistemas de las plataformas PTARTV y Primicia.

Instalar software: Es uno de los procesos que realiza el especialista donde instala todas las dependencias de software para el funcionamiento de un subsistema específico o una plataforma en general.

Copiar archivo: Es el proceso donde se copian todos los archivos subsistema específico o una plataforma en general.

Copiar carpetas: Es el proceso donde se copian todas las carpetas de un subsistema específico o una plataforma en general.

Configurar software: Es el proceso que se realiza después de haber instalado los software necesarios, donde se realiza la configuración que requieran los mismos debido a las funcionalidades de los subsistemas que se quieran instalar.

3.3 Requerimientos

Según el glosario de término de la IEEE, "Requisitos" son:

Condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo.

Condición o capacidad que debe exhibir o poseer un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta.

Los requerimientos puedes dividirse en requisitos funcionales y requisitos no funcionales los cuales se describen a continuación.

3.3.1 Requisitos Funcionales

Los requisitos funcionales son los que definen las funciones que el sistema será capaz de realizar.

RF1 Seleccionar plataforma

Este requisito le permite al usuario seleccionar la plataforma donde posteriormente se le creará el instalador al o los subsistemas de la misma.

RF2 Gestionar subsistema

Este requisito le permite al usuario adicionar o eliminar subsistemas a la plataforma seleccionada.

RF2.1 Gestionar dependencias

Este requisito permite adicionar software y librerías definidos por el usuario al subsistema seleccionado.

RF2.2 Gestionar ficheros

Este requisito permite adicionar carpetas definidas por el usuario al subsistema seleccionado.

RF2.3 Gestionar configuraciones

Este requisito permite adicionar nuevas configuraciones definidas por el usuario al subsistema seleccionado.

RF3 Crear instalador

Este requisito permite crear un instalador de los subsistemas anteriormente seleccionados en una ubicación específica.

RF3.1 Ubicar dirección del instalador

Este requisito permite seleccionar la dirección donde se encontrará el instalador después de creado.

3.3.2 Requisitos No Funcionales

Los requerimientos no funcionales son las características que pueden limitar el sistema de una forma u otra. Estos requerimientos se clasifican en diversas categorías, las cuales hacen que el producto sea confiable, seguro, rápido y usable.

Requisitos de Usabilidad: El sistema debe ser de fácil maniobrabilidad para que pueda ser utilizado por personas con conocimiento básico sobre el tema y el manejo de computadoras. El sistema deberá estar regido por la norma ISO 9241-11, la cual define los parámetros internacionales de usabilidad de cualquier software.

Requisitos de Ayudas y Documentación en línea: El sistema debe presentar un manual de usuario que guíe a los usuarios los pasos a seguir en caso de dudas.

Restricciones en el diseño y la implementación:

Para el modelado del sistema se utilizará la herramienta Visual Paradigm y el lenguaje UML. Se utiliza el lenguaje de programación C++ para la implementación del sistema utilizando QT Creator como framework de desarrollo. Respondiendo a un sistema arquitectónico en tres capas.

Requisitos de Portabilidad: El sistema se realizará para el funcionamiento en sistemas operativos GNU/Linux basados en Debian.

Requisitos de Apariencia o Interfaz externa: La aplicación deberá se sencilla con una interfaz amigable que permita facilidad en su manejo y buen desempeño en la ejecución de sus funciones.

Requisitos de Disponibilidad: El sistema debe estar disponible en todo momento.

Requisitos de Confiabilidad: La información manejada o generada por el sistema debe estar protegida del acceso no autorizado y de divulgación a terceras personas.

Requisitos de Rendimiento: El rendimiento del sistema debe mantenerse en un nivel medio.

Requisitos Legales, Derecho de Autor y otros: Sus derechos de autor y otros están establecidos por la empresa comercializadora del producto, ALBET S.A y la entidad desarrolladora de software de la UCI.

Requisitos de Software:

Sistema operativo: GNU\Linux específicamente Ubuntu 10.10

Requisitos de Hardware:

- Pentium IV 3.0 GHz o superior.
- HDD 80 GB o superior.
- Tarjeta de red Fast Ethernet 10/100Mbps o superior.
- 1 GB de memoria RAM o superior.

3.4 Descripción del Sistema Propuesto

Luego de realizar el modelo de dominio y el levantamiento de requisitos del sistema tanto funcionales como no funcionales se describen los Casos de Uso del sistema. Para ello es necesario realizar una descripción de los actores que interactúan con dicho sistema.

3.4.1 Descripción de los Actores

Los actores pueden ser personas, software o incluso máquinas que interactúan con el sistema de una forma u otra pero no forman parte de él. A continuación se muestran una descripción de todos los actores que interactúan en el sistema

Actor	Descripción
Especialista	Integrante de las plataformas PTARTV o Primicia, que cuente con los conocimientos avanzados de la instalación y configuración de las mismas.

Tabla 4. Descripción de actores del sistema.

3.4.2 Diagrama de Caso de Uso del Sistema

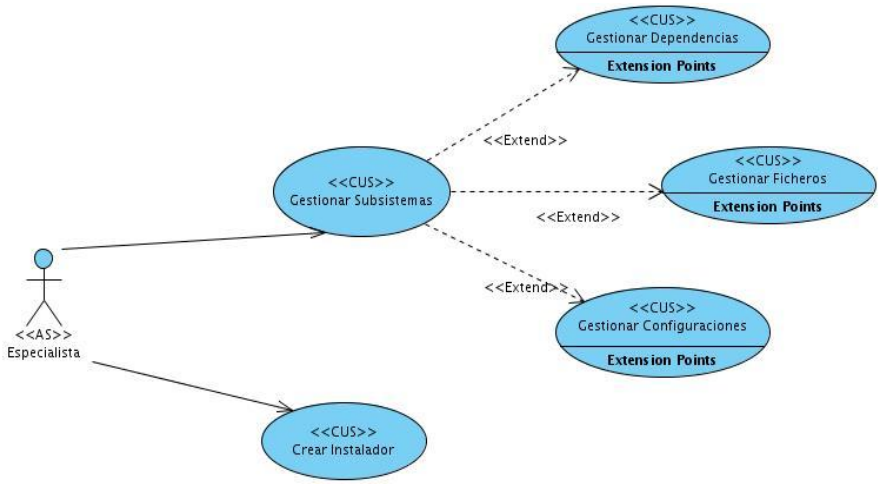


Figura 5. Diagrama de Caso de Uso del Sistema.

3.4.3 Descripción de los Casos de Uso del Sistema

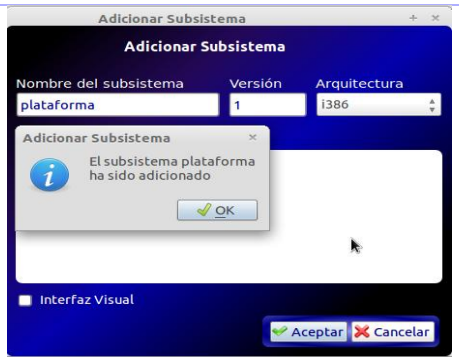
Los Casos de Uso describen una iteración típica entre un usuario y un sistema, mostrando información de cómo trabaja este sistema o como se desea que trabaje.

3.4.3.1 Gestionar Subsistema

Caso de Uso:	Gestionar Subsistema.
Actores:	Especialista.
Resumen:	El Caso de Uso comienza cuando se selecciona el menú Edición, este permite las opciones de adicionar o eliminar un subsistema a la plataforma seleccionada. También permite gestionar dependencias, ficheros y configuraciones al subsistema seleccionado. Este Caso de Uso finaliza cuando el usuario haya realizado satisfactoriamente

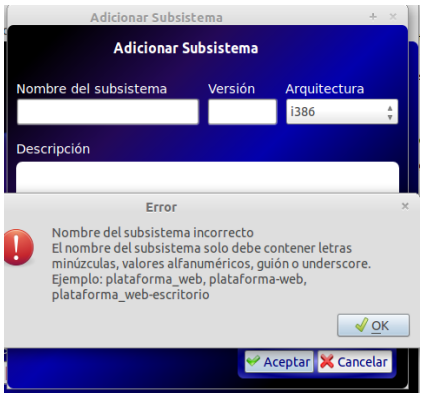
	o no alguna de las operaciones mencionadas anteriormente.
Precondiciones:	Para eliminar un subsistema debe haberlo seleccionado.
Referencias	RF2
Prioridad	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Caso de Uso se inicia cuando el usuario selecciona el menú Edición.	1.1. El sistema ejecuta alguna de las siguientes opciones:: • Adicionar subsistema, ir a la sección “Adicionar subsistema”. • Eliminar subsistema, ir a la sección “Eliminar subsistema”. • Gestionar dependencias, ir al Caso de Uso “Gestionar Dependencias”. • Gestionar ficheros, ir al Caso de Uso “Gestionar Ficheros”. • Gestionar configuraciones, ir al Caso de Uso “Gestionar Configuraciones”.
Sección “Adicionar subsistema”	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Adicionar subsistema”.	1.1. El sistema muestra un formulario donde el usuario llena los datos necesarios para crear un subsistema nuevo.
2. El usuario introduce los datos en el formulario.	2.1. El sistema verifica los datos introducidos. 2.2. El sistema muestra el mensaje: “El subsistema ha sido adicionado”.
Prototipo de Interfaz	



Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	2.1. Si introdujo datos no válidos el sistema muestra un mensaje de acuerdo al dato incorrecto. El sistema brinda la posibilidad de modificar los datos.

Prototipo de Interfaz



Poscondiciones	Se adiciona o no un nuevo subsistema.

Sección "Eliminar subsistema**"**	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Eliminar subsistema".	1.1. El sistema verifica que se haya seleccionado un subsistema. 1.2. El sistema elimina el/los subsistema/s seleccionado/s.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.1. Si el usuario no ha seleccionado un subsistema el sistema muestra el

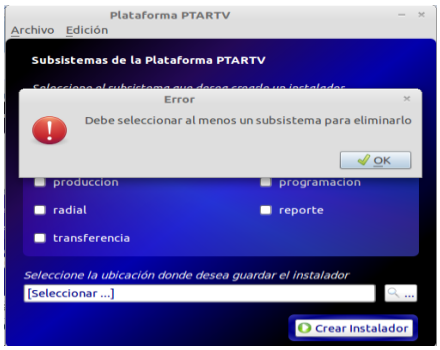
	mensaje “Debe seleccionar al menos un subsistema para eliminarlo”.
Prototipo de Interfaz 	
Poscondiciones	Se elimina o no uno o varios subsistemas.

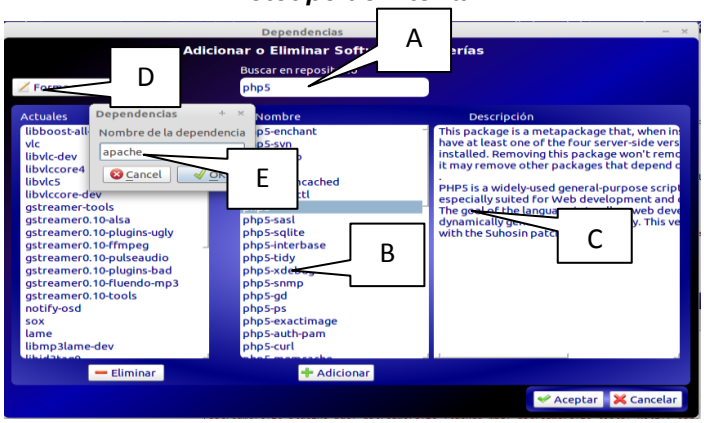
Tabla 5. Descripción detallada del Caso de Uso Gestionar Subsistema.

3.4.3.2 Gestionar Dependencias

Caso de Uso:	Gestionar Dependencias.
Actores:	Especialista.
Resumen:	El Caso de Uso comienza cuando se selecciona en el menú Edición / Editar subsistema la opción Dependencias, este permite adicionar o eliminar dependencias al subsistema seleccionado. El Caso de Uso finaliza cuando el usuario acepta o no los cambios realizados a las dependencias del subsistema seleccionado.
Precondiciones:	Debe haber un subsistema seleccionado.
Referencias	RF2.1
Prioridad	Crítico.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Caso de Uso comienza cuando se selecciona en el menú Edición / Editar subsistema la opción Dependencias.	1.1. El sistema muestra un formulario donde se encuentran las dependencias con que cuenta actualmente el subsistema, un campo de búsqueda y un botón para introducir el nombre de la librería manualmente.
2. El usuario introduce el nombre de la librería	2.1. El sistema va autocompletando con

que desea adicionar en el campo de búsqueda. (A) 2.2. El usuario selecciona una librería. 2.4. El usuario adiciona la librería seleccionada.	las librerías que coinciden con el nombre que está introduciendo el usuario. (B) 2.3. El sistema muestra una breve descripción de la librería seleccionada. (C) 2.5. El sistema verifica que la librería seleccionada no exista actualmente en el subsistema.
3. El usuario selecciona el botón Forma manual. (D) 3.2. El usuario introduce el nombre de la librería en el campo de texto.	3.1. El sistema muestra un campo de texto. (E) 3.3. El sistema verifica que la librería no exista actualmente en el subsistema.
4. El usuario elimina una librería existente.	
5. El Caso de Uso finaliza al seleccionar la opción aceptar.	5.1. El sistema guarda la configuración realizada.

Prototipo de Interfaz



Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	2.5. Si la librería a adicionar existe en el subsistema el sistema muestra el mensaje: “La librería ya existe”.
	3.3. Si la librería a adicionar existe en el subsistema el sistema muestra el mensaje: “La librería ya existe”.

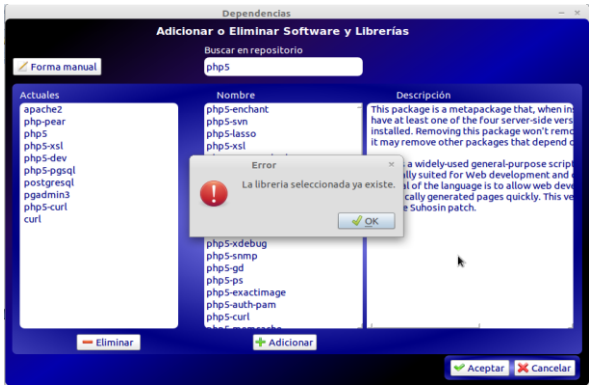
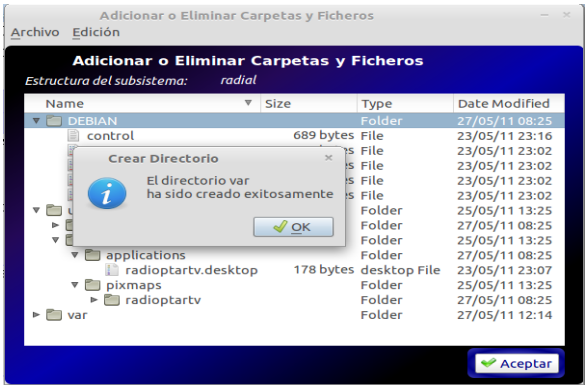
Prototipo de Interfaz 	
Poscondiciones	Se crea o no una nueva configuración del subsistema.

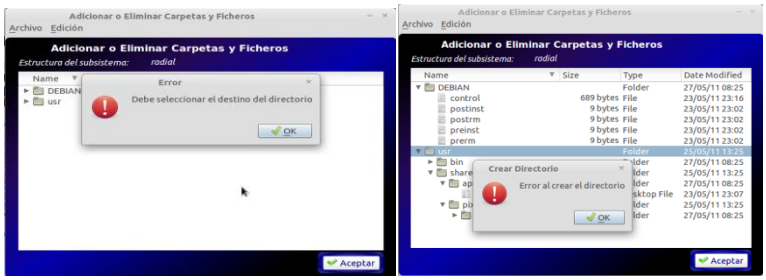
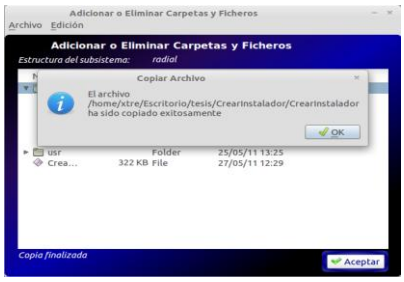
Tabla 6. Descripción detallada del Caso de Uso Editar Dependencias.

3.4.3.3 Gestionar Ficheros

Caso de Uso:	Gestionar Ficheros.
Actores:	Especialista.
Resumen:	El Caso de Uso comienza cuando se selecciona en el menú Edición / Editar subsistema la opción Ficheros, este permite adicionar o eliminar ficheros al subsistema seleccionado. El Caso de Uso finaliza cuando el usuario haya modificado o no la estructura de ficheros del subsistema seleccionado.
Precondiciones:	Debe haber un subsistema seleccionado.
Referencias	RF2.2
Prioridad	Crítico.

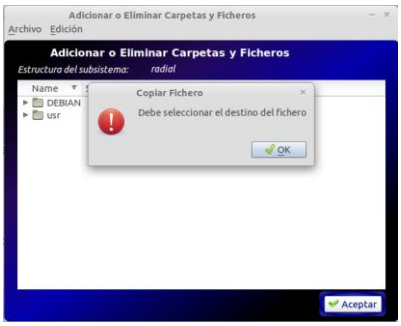
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Caso de Uso comienza cuando se selecciona en el menú Edición / Editar subsistema la opción Ficheros.	1.1. El sistema muestra un formulario donde se encuentra la estructura actual del subsistema seleccionado. 1.2. El sistema ejecuta alguna de las siguientes opciones: • Crear Directorio, ir a la sección “Crear Directorio”. • Copiar Fichero, ir a la sección

	"Copiar Fichero". • Copiar Directorio, ir a la sección "Copiar Directorio". • Eliminar, ir a la sesión "Eliminar".
Sección "Crear Directorio"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Crear Directorio". 1.3. El usuario introduce el nombre del directorio.	1.1. El sistema verifica que el usuario haya seleccionado donde desea crear el directorio. 1.2. El sistema muestra un campo de texto. 1.4. El sistema crea el directorio. 1.5. El sistema muestra el mensaje: "El directorio ha sido creado exitosamente".
Prototipo de Interfaz	
	
Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	1.1. Si el usuario no tiene seleccionado donde crear el directorio el sistema muestra el mensaje "Debe seleccionar el destino del directorio". 1.4. Si ocurrió algún error durante la creación del directorio el sistema muestra el mensaje: "Error al crear el

	directorio”.
Prototipo de Interfaz 	
Poscondiciones	Se crea o no un nuevo directorio.
Sección “Copiar Fichero**”**	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Copiar Fichero”. 1.3. El usuario busca el fichero que desea copiar.	1.1. El sistema verifica que el usuario haya seleccionado el destino del fichero. 1.2. El sistema muestra un campo de búsqueda. 1.4. El sistema copia el fichero. 1.5. El sistema muestra el mensaje: “El archivo ha sido copiado exitosamente”.
Prototipo de Interfaz 	
Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	1.1. Si el usuario no tiene seleccionado donde copiar el archivo el sistema muestra el mensaje “Debe seleccionar el destino del fichero”. 1.4. Si ocurrió algún error durante la copia del fichero el sistema muestra el

	mensaje: “Ha ocurrido un error al copiar el archivo”.

Prototipo de Interfaz

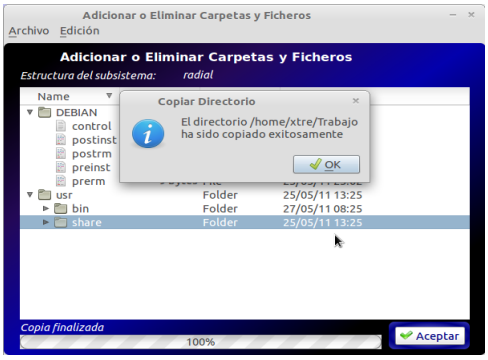


Poscondiciones	Se copia o no el fichero seleccionado.

Sección “Copiar Directorio**”**

Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Copiar Directorio”.	1.1. El sistema verifica que el usuario haya seleccionado el destino del directorio.
	1.2. El sistema muestra un campo de búsqueda.
1.3. El usuario el directorio que desea copiar.	1.4. El sistema copia el directorio.
	1.5. El sistema muestra el mensaje: “El directorio ha sido copiado exitosamente”.

Prototipo de Interfaz

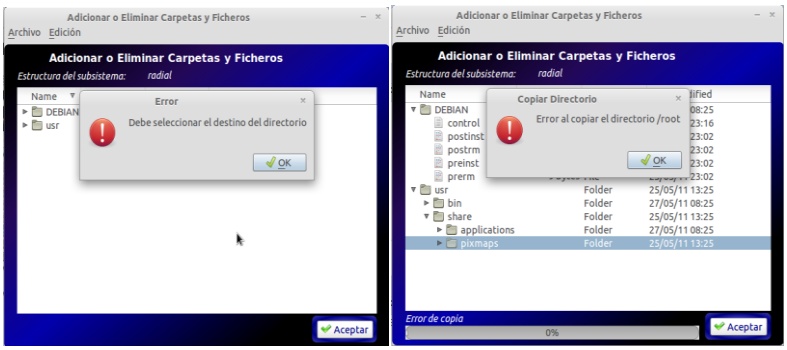


Flujo alterno

Acción del Actor	**Respuesta del Sistema**
	1.1. Si el usuario no tiene seleccionado donde copiar el directorio el sistema

	muestra el mensaje “Debe seleccionar el destino del directorio”. 1.4. Si ocurrió algún error durante la copia del directorio el sistema muestra el mensaje: “Error al copiar el directorio”.

Prototipo de Interfaz

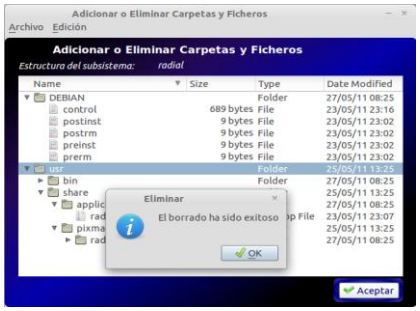


Poscondiciones	Se copia o no la carpeta seleccionada.

Sección “Eliminar**”**

Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Eliminar”	1.1. El sistema verifica que el usuario haya seleccionado lo que desea eliminar. 1.2. El sistema elimina lo seleccionado en la estructura actual. 1.3. El sistema muestra el mensaje: “El borrado ha sido exitoso”.

Prototipo de Interfaz



Flujo alterno

Acción del Actor	**Respuesta del Sistema**
	4.1. Si el usuario no tiene seleccionado

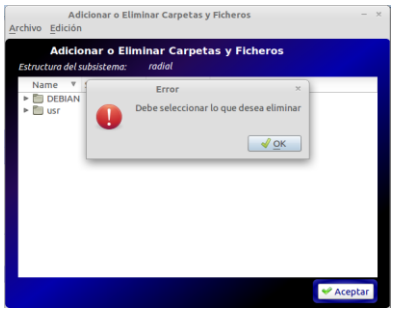
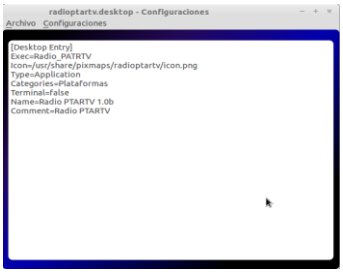
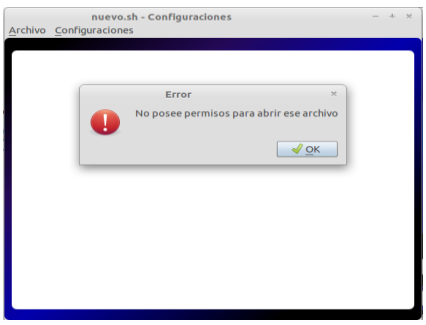
	lo que desea eliminar el sistema muestra el mensaje “Debe seleccionar lo que desea eliminar”.
Prototipo de Interfaz 	
Poscondiciones	Se elimina o no lo que este seleccionado.

Tabla 7. Descripción detallada del Caso de Uso Editar Ficheros.

3.4.3.3 Gestionar Configuraciones

Caso de Uso:	Gestionar Configuraciones.
Actores:	Especialista.
Resumen:	El Caso de Uso comienza cuando se selecciona en el menú Edición / Editar subsistema la opción Configuraciones, este permite adicionar o eliminar configuraciones al subsistema seleccionado. El Caso de Uso finaliza cuando el usuario haya creado o no, modificado o no nuevas configuraciones.
Precondiciones:	Debe haber un subsistema seleccionado.
Referencias	RF2.3
Prioridad	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Caso de Uso comienza cuando se selecciona en el menú Edición / Editar subsistema la opción Configuraciones.	1.1. El sistema ejecuta alguna de las siguientes opciones: • Nuevo, ir a la sección “Nuevo”. • Abrir, ir a la sección “Abrir”. • Guardar, ir a la sección “Guardar”. • Guardar Como, ir a la sesión

	"Guardar Como".
Sección "Nuevo"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Nuevo".	1.1. El sistema muestra un campo de texto en blanco.
Sección "Abrir"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Abrir". 1.2. El usuario busca la configuración que desea abrir.	1.1. El sistema muestra un campo de búsqueda. 1.3. El sistema abre la configuración mostrando su contenido en un campo de texto.
Prototipo de Interfaz	
	
Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	1.3 Si la configuración seleccionada no posee permisos de lectura el sistema muestra el mensaje "No posee permisos para abrir ese archivo".
Prototipo de Interfaz	
	

Poscondiciones	Se abre o no la configuración seleccionada.
Sección "Guardar"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Guardar".	1.1. El sistema verifica que el documento se haya modificado. 1.2. Si lo que se desea guardar es una configuración nueva el sistema ejecuta la opción "Guardar Como", sino guarda el documento con los cambios realizados.
Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
	1.1 Si el documento no ha sido modificado el sistema deshabilita la opción "Guardar".
Poscondiciones	Se guarda la configuración.
Sección "Guardar Como"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Guardar Como". 1.2. El usuario selecciona el nombre y la dirección donde desea guardar la configuración.	1.1. El sistema muestra un campo de búsqueda.
Poscondiciones	Se guarda o no la configuración en la dirección seleccionada.

Tabla 8. Descripción detallada del Caso de Uso Editar Configuraciones.

3.4.3.5 Crear instalador

Caso de Uso:	Crear instalador.
Actores:	Especialista.
Resumen:	El Caso de Uso comienza cuando el usuario selecciona la opción crear. El Caso de Uso finaliza cuando el sistema termine de crear el instalador de los subsistemas seleccionado o cuando ocurra algún error durante esta creación.

Precondiciones:	Se debe haber seleccionado el o los subsistemas que se desean instalar.
Referencias	RF3, 3.1
Prioridad	Crítico

Flujo Normal de Eventos

Acción del Actor	**Respuesta del Sistema**
1. El Caso de Uso se inicia cuando el usuario selecciona la opción “Crear instalador”.	1.1. El sistema verifica si hay algún subsistema seleccionado. 1.2. El sistema valida que el usuario haya seleccionado la ubicación donde se creará el instalador.
	2.1. El sistema crea el instalador de los subsistemas seleccionados. 2.2 El sistema muestra el mensaje: “La creación ha sido un éxito”.

Prototipo de Interfaz

Flujos alternos

Acción del Actor	**Respuesta del Sistema**
	1.1. Si no se encuentra seleccionado ningún subsistema el sistema muestra el mensaje: “Debe seleccionar al menos un subsistema para crearle un instalador”. 1.2. Si no se selecciona ninguna dirección el sistema muestra el mensaje: “Debe seleccionar una ubicación”.

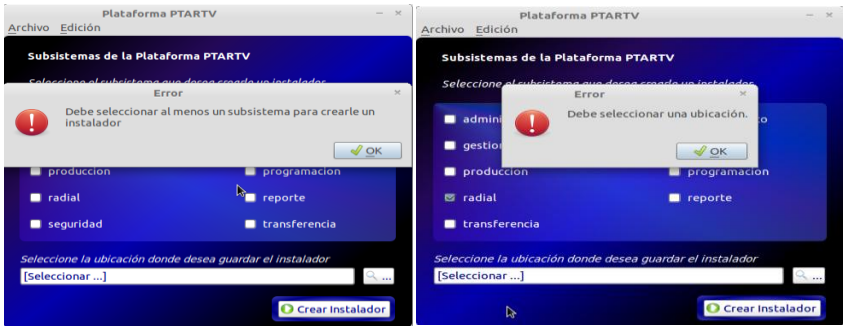
	2.1. Si ocurre algún error durante la creación del instalador el sistema termina este proceso y muestra el mensaje: “Ha ocurrido un error durante la creación del instalador de alguno de los subsistemas seleccionados”.
Prototipo de Interfaz 	
Poscondiciones	Se crea o no el instalador de los subsistemas definidos por el usuario.

Tabla 9. Descripción detallada del Caso de Uso Crear instalador.

3.5 Conclusiones parciales

En el presente capítulo se realizó el análisis del modelo de dominio donde se abordaron los conceptos y principales eventos del entorno, presentando el Diagrama de Clases del Modelo de Dominio así como su glosario de términos para una mejor comprensión del mismo.

Se elaboró el levantamiento de los requisitos funcionales siendo estos los que definen las funciones que el sistema será capaz de realizar y los requisitos no funcionales los cuales son las características que pueden limitar el sistema de una forma u otra. También se realizó una descripción general del sistema propuesto donde se describieron los actores que interactúan con el sistema, así como la presentación del Diagrama de Caso de Uso del Sistema. Se realizó además una detallada descripción de los Casos de Uso presentes, que describen la interacción típica entre el usuario y el sistema, mostrando información de cómo trabaja este sistema o cómo se desea que trabaje.

Capítulo 4: Construcción de la solución propuesta.

4.1 Introducción

En este capítulo se abordarán los aspectos relacionados con la construcción de la solución propuesta. Se definirá la arquitectura del sistema donde se describen las principales características de los patrones del diseño a utilizar. Luego de analizar en el capítulo anterior las funcionalidades que debe cumplir el sistema se realizará el diseño y seguidamente la implementación del sistema. Se elaborarán los diagramas de clases del diseño del sistema y por último se realizarán las pruebas a la solución presentada.

4.2 Arquitectura de Software

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada Arquitectura de Software o Arquitectura lógica. Se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para guiar el desarrollo de software dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado.

Algunos objetivos dentro de un esquema de Arquitectura de Software pueden ser: el software debe ser mantenible, esto es, fácilmente analizable, modificable, corregible; también puede ser un objetivo el nivel de interacción con otros sistemas informáticos, o su escalabilidad. Estas Arquitecturas están definidas muchas veces por el tipo de tecnología a la cual se enfrenta un programador o grupo de programadores, por lo cual algunos tipos de Arquitectura son más recomendables que otras para ciertas tecnologías. (21)

Para la modelación de la arquitectura del sistema se seleccionó el estilo arquitectónico basado en Capas. Como variante de este estilo se utilizará el “dos capas”: Presentación y Lógica del negocio.

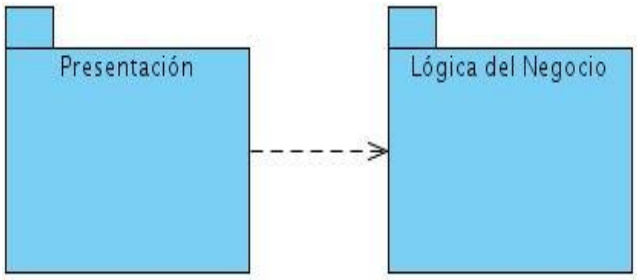


Figura 6. Variante de Arquitectura en Capas: "2 capas".

Este patrón arquitectónico soporta un diseño basado en niveles de abstracción creciente. Su objetivo principal es separar la lógica de diseño de la lógica del negocio. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Admite una amplia reutilización del código generado, incluyendo refinamiento.

Capa de presentación: es la capa que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" para el usuario. Esta capa se comunica únicamente con la capa lógica del negocio.

Capa lógica del negocio: es la capa que contiene los procesos a realizar con la información recibida desde la capa de presentación, las peticiones que el usuario ha realizado, y responsabilizándose de que se le envíen las respuestas adecuadas a la capa de presentación.

4.3 Principios del Diseño

El diseño es la manera de representar un objeto que se está creando. Es la información de base que describe aspectos de este objeto. En el diseño de software se realizan una serie de procesos para definir la arquitectura, componentes, interfaces y otras características de un sistema. Este diseño es una descripción de la estructura del software que se va a implementar como las interfaces entre los componentes del sistema y en ocasiones los algoritmos utilizados. Los principios básicos de diseño proporcionan un marco de trabajo necesario para lograr que se realice correctamente. Estos diseños favorecen la gestión de la complejidad de los sistemas de software lo que permite la obtención de la calidad que estos sistemas han de presentar. Roger S. Pressman sugiere,

en su libro “Ingeniería del Software, Un enfoque práctico”, un conjunto de principios para el diseño del software los cuales son mencionados a continuación:

- En el proceso de diseño no deberá utilizarse orejeras.
- El diseño no deberá inventar nada que ya esté inventado.
- El diseño deberá presentar uniformidad e integración.
- El diseño deberá estructurarse para admitir cambios.
- El diseño deberá evaluarse en función de la calidad mientras se va creando, no después de terminado.
- El diseño deberá revisarse para minimizar los errores conceptuales.

4.3.1 Estándares de la Interfaz de la Aplicación

La interfaz de usuario es un elemento clave en cualquier aplicación, esta debe ser legible, intuitiva y lo más eficiente posible, en vistas de lograr que el sistema cumpla con los requisitos expresados por el usuario. La interfaz del componente fue diseñada siguiendo los estándares de usabilidad expuestos en los requerimientos no funcionales, con el objetivo de lograr que el cliente pueda trabajar con mayor facilidad en la aplicación desarrollada.

4.4 Patrones de Diseño

“Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.” (22)

En la construcción del modelo de diseño del sistema fueron utilizados los patrones GRASP (Patrones de asignación de responsabilidades) y GoF (Grupo de los cuatro), ambos dirigidos al desarrollo de sistemas orientados a objetos.

4.4.1 Patrones GRASP

El patrón Experto: propone la asignación de responsabilidades específicas a las clases creadas; es un principio básico que suele utilizarse en el diseño orientado a objetos. Expresa simplemente la "intuición" de que cada clase contiene los métodos relacionados

con la información que posee. (23) Este patrón se aplicó a todas las clases creadas en la implementación del sistema, en las que en algunos casos se evidencian las funcionalidades correspondientes al manejo de los atributos que pertenecen a cada una, y en otros casos, solamente aquellas que son necesarias para cumplir el objetivo con que fueron creadas.

El patrón Creador: se utilizará este patrón debido a las propiedades de todo sistema orientado a objetos. Se aplicará en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. Este patrón se evidencia en la clase CC_Plataformas la cual crea una instancia de la clase LN_Plataforma y esta crea una instancia de la clase LN_Subsistema.

El patrón Bajo Acoplamiento: propone la independencia de las clases del diseño con el fin de reducir el impacto de los cambios y permitir una mayor reutilización del código. Asigna responsabilidades de forma tal que las clases del diseño sean más independientes y que la comunicación sea la menor posible. (23) Esto favorece a la flexibilidad del diseño y a la actualización de los cambios del sistema pues las clases son menos dependientes entre sí.

El patrón Alta cohesión: propone cuán relacionadas y orientadas están las responsabilidades de una clase. Planteando la contribución entre clases para realizar tareas de elevada complejidad. Soporta un aumento de la capacidad de reutilización, siendo una gran funcionalidad, ya que una clase muy cohesiva puede destinarse a un propósito muy específico. (23) Este patrón se evidencia en el diseño de la clase controladora CC_Plataformas, la clase LN_plataforma y la clase LN_Subsistema.

El patrón Controlador: define quién deberá encargarse de atender un evento del sistema. El controlador es un intermediario entre la interfaz de usuario y el núcleo de las clases donde se encuentra la lógica del sistema. (23) Es una clase que, para el diseñador, representa de algún modo al sistema global.

4.4.2 Patrones GoF

Composición: este patrón permite a los clientes tratar uniformemente a los objetos simples y compuestos de una estructura jerárquica recursiva. Este es usado cuando se

pretende que los clientes no reparen en las diferencias entre objetos simples y compuestos. (24)

Fachada: este patrón proporciona una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La “fachada” satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas. Este es usado cuando se pretende proporcionar una interfaz simple para un subsistema complejo. (24)

4.5 Modelo del Diseño

Durante el diseño se modela el sistema y su arquitectura para que soporte los requisitos funcionales y no funcionales. El diseño es el centro de atención al final de la fase de elaboración y comienzo de las iteraciones de construcción. El modelo de diseño es un modelo de objetos que describe la realización física de los Casos de Uso centrándose en los requisitos funcionales como en los no funcionales. Las abstracciones del modelo de diseño tienen una correspondencia directa con los elementos físicos del ambiente de implementación. (25)

4.5.1 Diagrama de Clases de Diseño

Un diagrama de clases es un diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas para lograr una mejor interpretación de la implementación del sistema. Estos diagramas describen de las clases sus atributos, métodos y las relaciones entre ellas.

A continuación se muestran los diagramas de clases del diseño correspondientes a cada Caso de Uso del Sistema.

4.5.1.1 Gestionar Subsistema

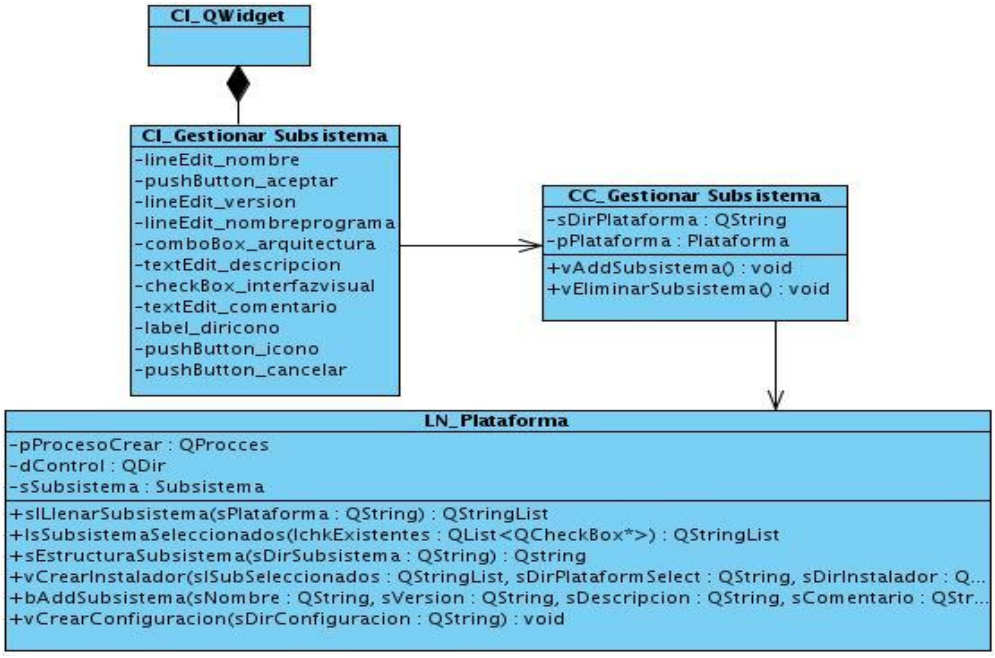


Figura 7. Diagrama de Clases del Diseño del Caso de Uso Gestionar Subsistema.

4.5.1.2 Gestionar Ficheros

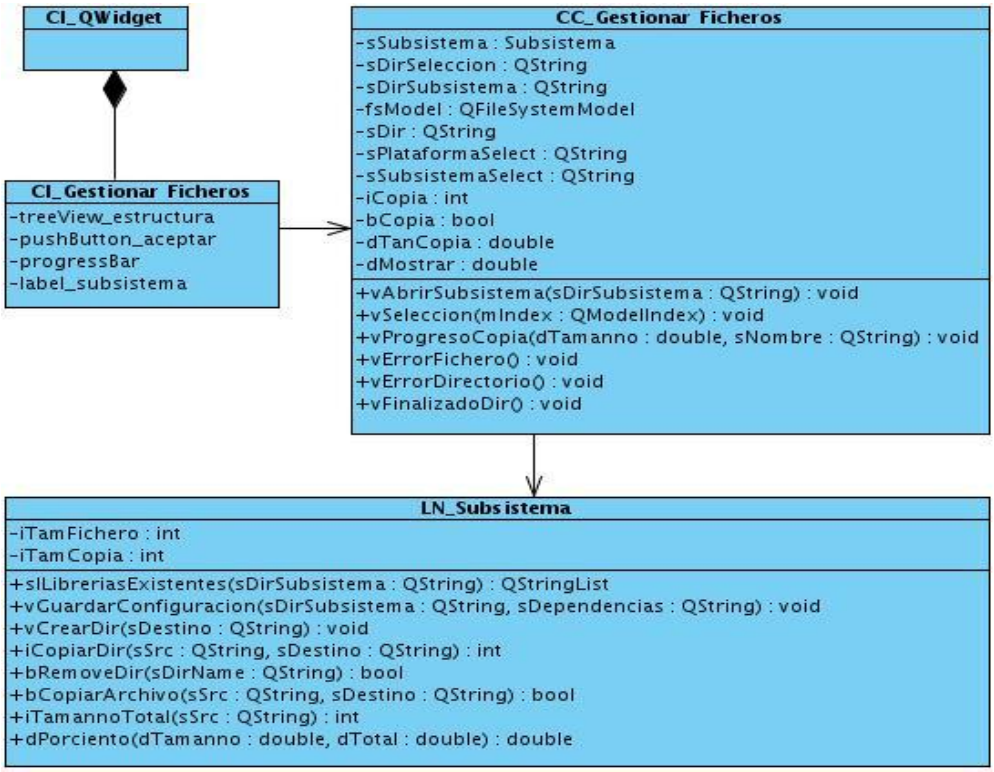


Figura 8. Diagrama de Clases del Diseño del CU "Gestionar Ficheros".

4.5.1.3 Gestionar Dependencias

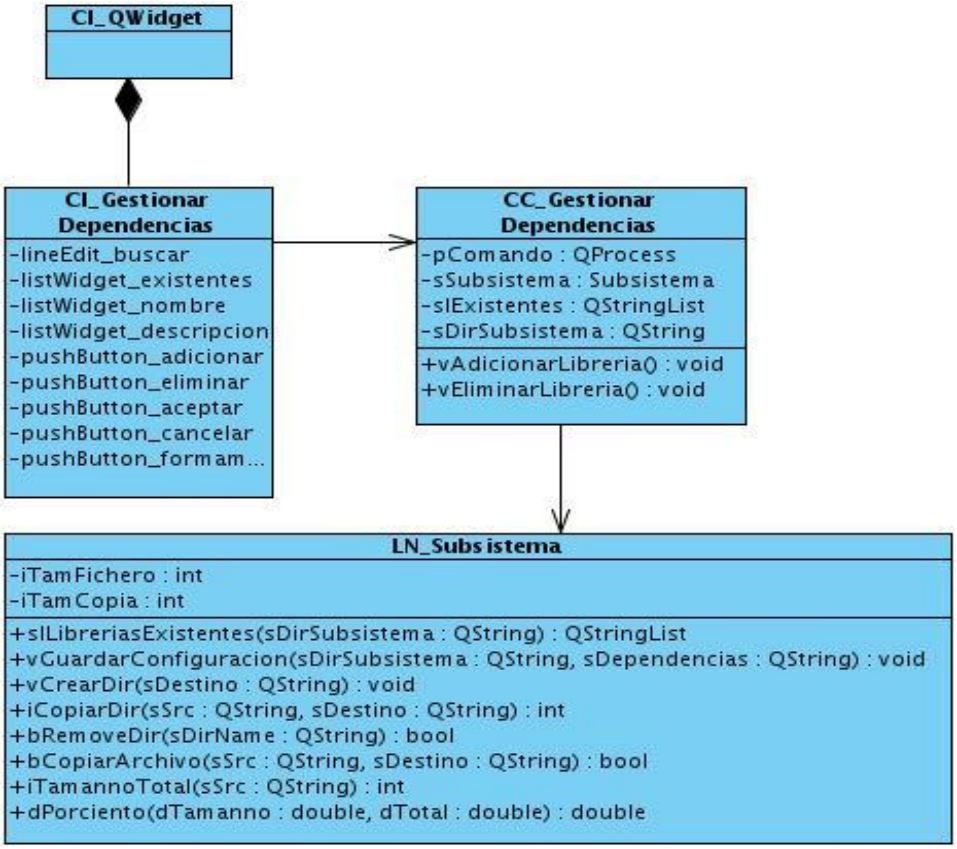


Figura 9. Diagrama de Clases del Diseño del CU "Gestionar Dependencia".

4.5.1.4 Gestionar Configuraciones

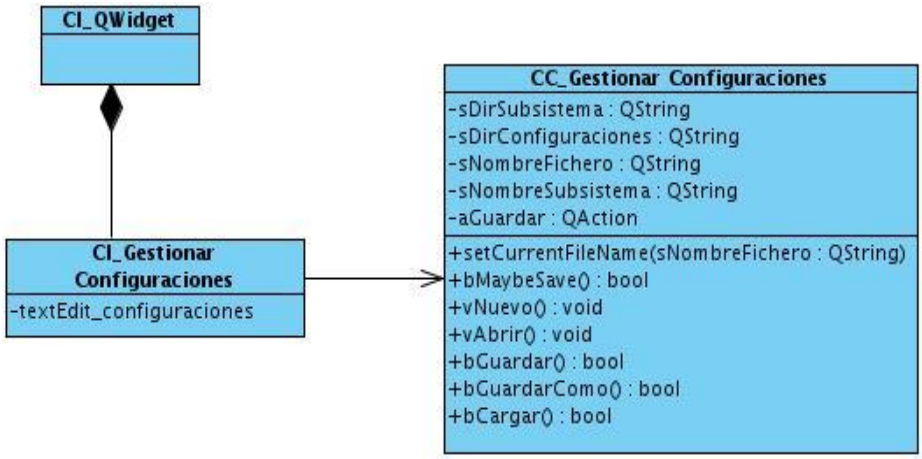


Figura 10. Diagrama de Clases del Diseño del CU "Gestionar Configuraciones".

4.5.1.5 Crear Instalador

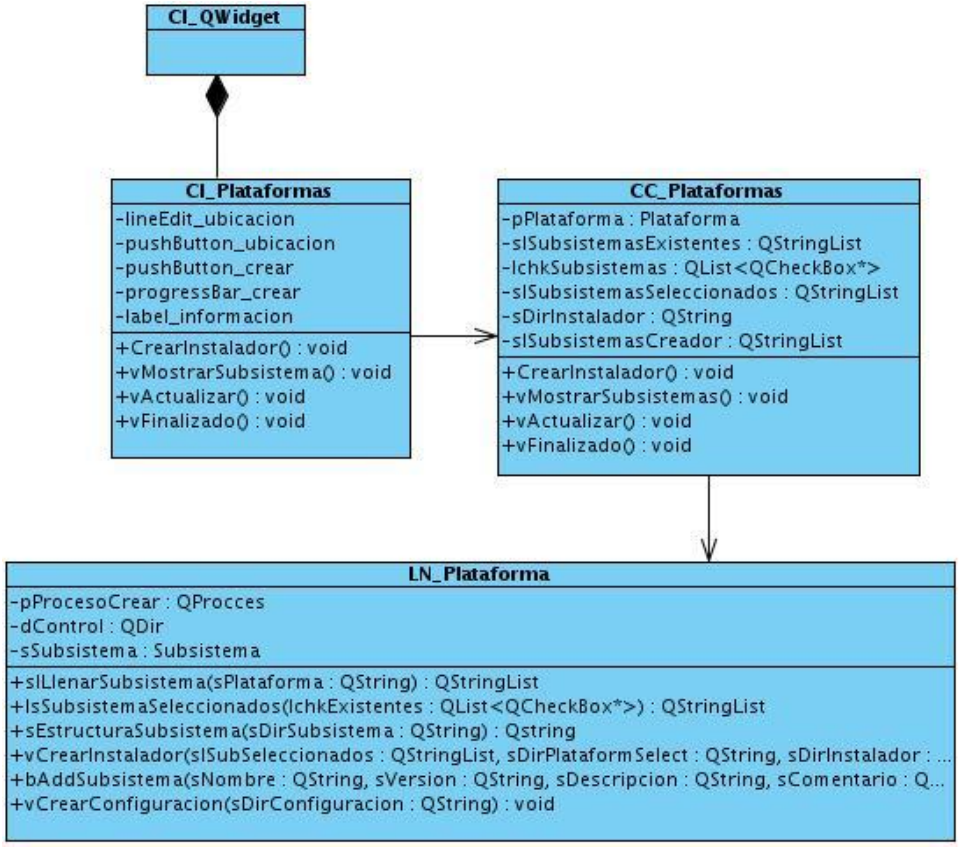


Figura 11. Diagrama de Clases del Diseño del CU "Crear Instalador".

4.6 Modelo de Despliegue

El modelo de despliegue está compuesto por el Diagrama de Despliegue y es utilizado para capturar los elementos de configuración del procesamiento y las relaciones físicas entre los componentes hardware y software en el sistema final. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. En la siguiente figura se muestra el modelo de despliegue correspondiente a la aplicación.

Figura 12. Diagrama de Despliegue.

4.7 Modelo de Implementación

El modelo de implementación está compuesto por un conjunto de subsistemas y componentes que establecen la composición física de la implementación del sistema. Para una mejor organización de la implementación del sistema, se organizaron las clases por componentes lo cual permitió una mejor reutilización de código.

4.7.1 Diagrama de Componente

Los diagramas de componentes son usados para estructurar el modelo de implementación y mostrar las relaciones entre sus elementos. Los componentes representados pueden ser datos, archivos, ejecutables, código fuente y directorios. También se puede decir que los Diagramas de Componentes muestran los componentes de software que constituyen una parte reusable, sus interfaces y sus interrelaciones. A continuación se muestra el Diagrama de Componentes del sistema.

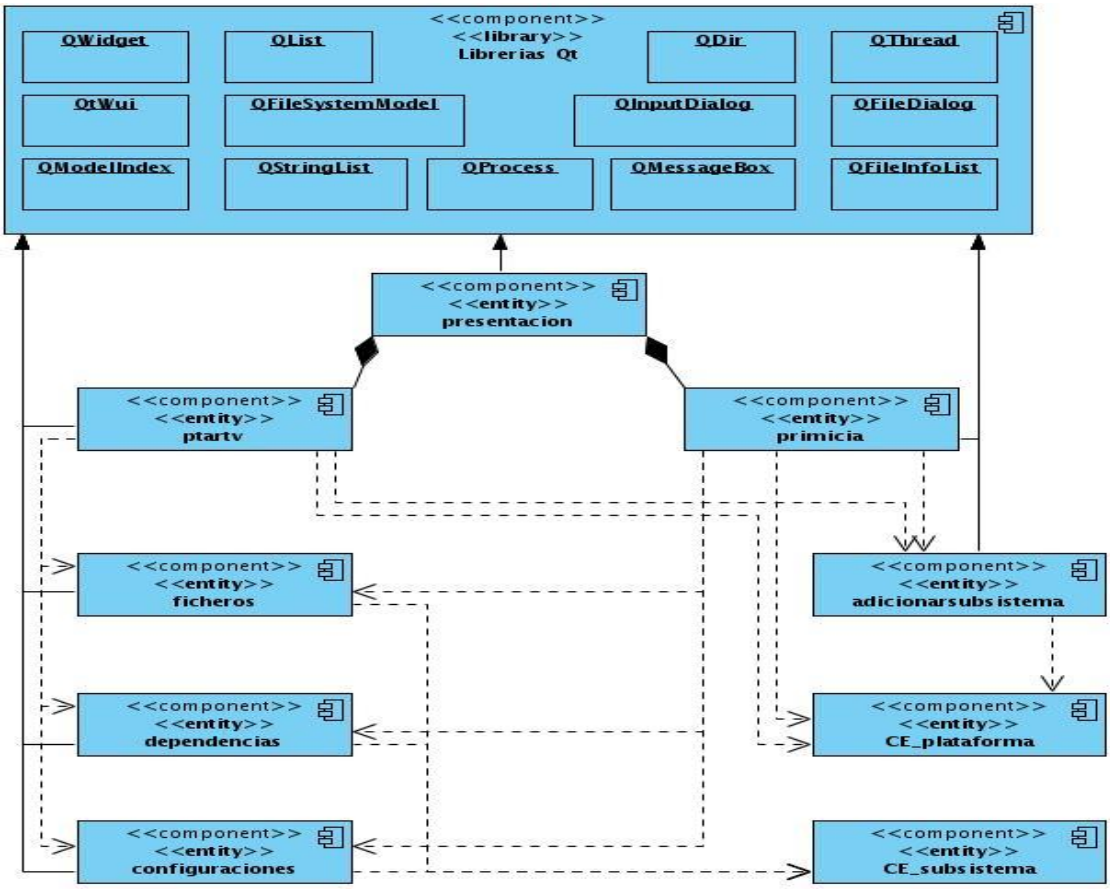


Figura 13. Diagrama de Componentes.

4.8 Métodos de Prueba

Al desarrollar cualquier software es de vital importancia la realización de pruebas. Las pruebas no son más que técnicas de validación predominantes o procesos de ejecución de un programa con la intención de verificar y revelar la calidad del mismo. Estas pruebas son utilizadas para descubrir errores en la aplicación que antes no se habían descubierto Los métodos de prueba son utilizados para realizarle pruebas a un software. Con el objetivo de validar las funcionalidades del sistema y mostrar una indicación de la calidad del producto se realizarán pruebas funcionales y unitarias.

4.8.1 Pruebas Funcionales

Se denominan pruebas funcionales, a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados. A este tipo de pruebas se les denomina también pruebas de comportamiento o pruebas de caja negra, el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas.

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. Estas pruebas se enfocan en los requerimientos establecidos y en la funcionalidad del sistema. Esto significa que 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.

4.8.1.1 Casos de Prueba

Los casos de prueba permiten detallar la forma en que se va a probar el sistema, los que incluyen los datos de entrada con las que se realizará la prueba correspondiente, las condiciones de ejecución y resultados obtenidos.

Al realizar un caso de prueba se debe verificar:

- Que el componente cumpla con los requerimientos del usuario tal y como se describe en la especificación de requisitos.
- Si el producto se comporta como se desea, tal y como se describe en las especificaciones de los Casos de Uso.

A continuación se muestran los casos de pruebas realizados en el sistema:

Caso de Prueba #1: Caso de Uso Gestionar Subsistema.

- **Secciones a probar en el Caso de Uso Gestionar Subsistema.**

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC1: Adicionar subsistema	EC 1.1: Adicionar subsistema con éxito.	El sistema muestra un formulario. El usuario llena todos los campos del formulario. El sistema agrega el subsistema nuevo.
	EC 1.2: Adicionar subsistema falla.	El sistema verifica que los datos estén correctos y que no existan campos vacios. El sistema no agrega el subsistema y muestra un mensaje indicando el error.
SC 2: Eliminar subsistema.	EC 2.1: Eliminar subsistema con éxito.	El sistema elimina los subsistemas seleccionados.
	EC 2.2: Eliminar subsistema falla.	El sistema comprueba que exista un subsistema seleccionado. El sistema no elimina ningún subsistema y muestra un mensaje indicando que debe seleccionar al menos un subsistema.

Tabla 10: Secciones a probar en el Caso de Uso Gestionar Subsistema.

- **Descripción de las variables del Caso de Uso Gestionar Subsistema.**

No	Nombre de campo	Clasificación	Valor Nulo	Descripción

1	Nombre del subsistema	Campo de texto	No	El usuario introducirá el nombre del subsistema.
2	Versión	Campo de texto	No	El usuario introducirá la versión del subsistema.
3	Descripción	Campo de texto	No	El usuario introducirá una breve descripción del subsistema.
4	Interfaz visual	CheckBox	Si	El usuario lo seleccionará si el instalador posee interfaz visual.
5	Nombre del programa	Campo de texto	Si	El usuario introducirá el nombre del instalador del subsistema.
6	Comentario del programa	Campo de texto	Si	El usuario introducirá un breve comentario del instalador del subsistema.
7	Dirección del icono del programa	Campo de búsqueda	Si	Permite buscar el ícono para el instalador del subsistema.
8	Dirección del binario del programa	Campo de búsqueda	Si	Permite buscar el binario que ejecuta el instalador del subsistema.
9	Subsistemas seleccionados	CheckBox	No	Permite eliminar los subsistemas seleccionados.

Tabla 11: Descripción de las variables del Caso de Uso Gestionar Subsistema.

- **Matriz de datos del Caso de Uso Gestionar Subsistemas.**

1. **Sección Adicionar subsistema.**

Escenario	EC 1.1: Eliminar subsistema con éxito.	EC 1.2: Eliminar usuario falla.
Nombre del subsistema	Válido (V)	V
Versión	V	Inválido (I)
Descripción	V	V
Interfaz visual	V	V
Nombre del programa	V	V
Comentario del programa	V	V
Dirección del icono del programa	V	V
Dirección del binario del programa	V	V
Respuesta del sistema	El sistema verifica que los datos introducidos por el usuario sean correctos, adiciona el subsistema y muestra un mensaje al usuario indicando que la acción fue ejecutada con éxito.	El sistema verifica que los datos introducidos por el usuario sean correctos, el sistema muestra un mensaje informando el error ocurrido.
Resultado de la prueba	Satisfactoria	Satisfactoria

Tabla 12: Matriz de datos para la sección Adicionar subsistema.

2. **Sección Eliminar subsistema.**

Escenario	EC 2.1: Adicionar subsistema con éxito.	EC 2.2: Insertar usuario falla.
Subsistemas seleccionados	V	I
Respuesta del sistema	El sistema verifica que se haya	El sistema verifica que se

	seleccionado algún subsistema y elimina los subsistemas seleccionados.	haya seleccionado algún subsistema, no elimina ningún subsistema y muestra un mensaje indicando que debe seleccionar al menos un subsistema.
Resultado de la prueba	Satisfactoria	Satisfactoria

Tabla 13: Matriz de datos para la sección Eliminar subsistema.

Caso de Prueba #2: Caso de Uso Crear Instalador.

- **Secciones a probar en el Caso de Uso Crear instalador.**

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC1: Crear instalador.	EC 1.1: Crear instalador con éxito.	El sistema verifica que haya algún subsistema seleccionado, el sistema verifica que el usuario haya seleccionado una dirección para guardar el instalador, el sistema crea el instalador de los subsistemas seleccionados.
	EC 1.2: Crear instalador falla.	El sistema verifica que haya algún subsistema seleccionado, el sistema verifica que el usuario haya seleccionado una dirección para guardar el instalador, El sistema muestra un mensaje indicando que existe un error en la creación.

Tabla 14: Secciones a probar en el Caso de Uso Crear Instalador.

- **Descripción de las variables del Caso de Uso Gestionar Subsistema.**

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Dirección donde se guardará el instalador	Campo de búsqueda	No	El usuario buscará donde se guardarán los instaladores creados.
2	Subsistemas seleccionados	CheckBox	No	Permite crear el instalador a los subsistemas seleccionados.

Tabla 15: Descripción de las variables del Caso de Uso Crear Instalador.

- **Matriz de datos del Caso de Uso Gestionar Subsistemas.**

Escenario	EC 1.1: Crear instalador con éxito.	EC 1.2: Crear instalador falla.
Dirección donde se guardará el instalador	V	V
Subsistemas seleccionados	V	I
Respuesta del sistema	El sistema verifica que el usuario haya buscado una dirección para guardar los instaladores a crear, el sistema verifica que el usuario haya seleccionado al menos un subsistema, el sistema crea el instalador de los subsistemas seleccionados y muestra un mensaje al usuario indicando que la acción fue ejecutada con éxito.	El sistema verifica que el usuario haya buscado una dirección para guardar los instaladores a crear, el sistema verifica que el usuario haya seleccionado al menos un subsistema, el sistema muestra al usuario un mensaje indicando que ha ocurrido algún error durante la creación del instalador.
Resultado de la prueba	Satisfactoria	Satisfactoria

Tabla 16: Matriz de datos para la sección Crear instalador.

Los casos de prueba anteriormente realizados fueron satisfactorios, el sistema cumple con los requerimientos del usuario y se comporta de la manera deseada tal y como se

describe en las especificaciones de los Casos de Uso. Estas pruebas demuestran la calidad del sistema desarrollado.

4.8.2 Pruebas Unitarias

Las pruebas unitarias son realizadas para probar el correcto funcionamiento de un módulo de código, estas están dirigidas a analizar clases, métodos y porciones de código que puedan probarse de forma aislada. El uso de las pruebas unitarias permitirá que:

- Los errores sean más fáciles de localizar: bastará con ejecutar la batería de "Colecciones de pruebas", y ver qué módulos no las pasan.
- Los errores están más acotados: cuando un programa falla, muchas veces no se sabe de dónde pueden venir los problemas.

Con las pruebas unitarias se consigue acotar los errores, sabiendo qué módulos no están pasando las pruebas.

- Se reducen los "efectos secundarios": muchas veces, cuando se quiere arreglar algo bajo presión, se cometen otros errores, o no se tiene en cuenta ciertos aspectos, que hacen que el programa deje de funcionar por otro lado. Incluso a veces, es más peligroso arreglar un error que dejarlo como está, ya que se puede subsanar el error, pero generar otros distintos. Al aplicar las pruebas unitarias es más fácil controlar el programa pues se asegura que todo funciona tal y como se esperaba.
- Se da más seguridad al programador: normalmente, la persona que ha programado un módulo no es la misma que la que debe corregir sus errores. Esto crea una sensación de inseguridad al programador pues a la hora de corregir un error, no tiene la certeza de que su corrección no va a afectar a otros módulos que desconoce.

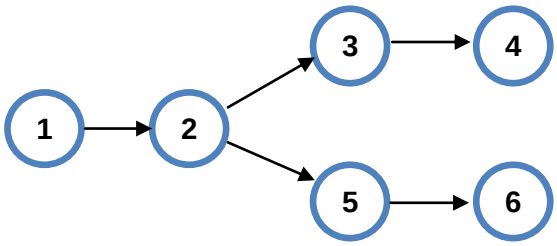
Las pruebas unitarias aseguran que una corrección no repercuta en otros módulos, y permite al programador centrarse en su trabajo. (26)

A continuación se muestran los casos de pruebas realizados a algunas funcionalidades del sistema.

Función 1: Crear directorio.

```
void Subsistema::vCrearDir(QString &sDestino)
{
   QDir dDirSubsistema(sDestino);                                    1
   QString sDirName = QInputDialog::getText(this, tr("Crear Directorio"),
                                                                     1
                                            tr("Nombre del directorio"));
    if (!sDirName.isEmpty())                                         2
    {
       if(!dDirSubsistema.mkdir(sDirName))                           3
          QMessageBox::warning(this, tr("Crear Directorio"),
                                                                     4
                               tr("Error al crear el directorio"));
       else                                                          5
          QMessageBox::information(this, tr("Crear Directorio"),
                tr("El directorio %1 \n ha sido creado exitosamente").arg(sDirName),   6
          QMessageBox::Ok);
    }
}
```

Paso 1: Generar grafo de flujo de datos.



Paso 2: Cálculo de complejidad ciclomática.

V (G)= Número de Aristas - Número de Nodos + 2

V (G)= 5- 6+2= 1

Paso 3: Caminos básicos.

CB1: 1 2 3 4

CB2: 1 2 5 6

Paso 4: Caso de prueba para el camino básico.

CB1: 1 2 3 4

Caso de prueba: Crear directorio.

Resultado esperado: Creación de un directorio.

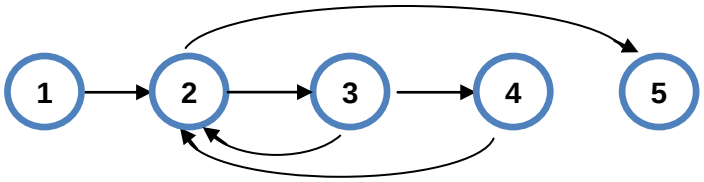
Resultado de la prueba: Satisfactoria.

Función 2: Subsistemas seleccionados

```
QStringList Plataforma::slSubsistemasSeleccionados(QList<QCheckBox *>
lchkExistentes)
{
   QStringList slSubsistemasSeleccionados;   1
   for(int i = 0; i < lchkExistentes.size(); i++)   2
   {
      if(lchkExistentes.at(i)->isChecked())   3
      {
         slSubsistemasSeleccionados.append(lchkExistentes.at(i)->objectName());   4
      }
   }
```

```
    return slSubsistemasSeleccionados;  5
}
```

Paso 1: Generar grafo de flujo de datos.



Paso 2: Cálculo de complejidad ciclomática.

V (G)= Número de Aristas - Número de Nodos + 2

V (G)= 6- 5+2= 3

Paso 3: Caminos básicos.

CB1: 1 2 3 4 2 5

CB2: 1 2 3 2 5

Paso 4: Caso de prueba para el camino básico.

CB1: 1 2 3 4 2 5

Caso de prueba: Subsistemas seleccionados.

Entrada: Lista de CheckBox (chkExistentes).

Resultado esperado: Llenar una lista con los subsistemas seleccionados.

Resultado de la prueba: Satisfactoria.

4.9 Resultados de las pruebas

El resultado de las pruebas realizadas al software fue satisfactorio, se comprobó que las respuestas del sistema son las esperadas y que el mismo cumple con las especificidades planteadas y que a su vez coinciden con las descripciones de los Casos de Usos

planteados con anterioridad. Con las pruebas aplicadas se comprobó que el sistema cumple con lo objetivos propuesto.

4.10 Validación de la solución propuesta

Luego de obtener resultados satisfactorios en las pruebas desarrolladas al sistema se realizó el despliegue de varios subsistemas mediante instaladores creados con la aplicación construida. Este proceso arrojó resultados esperados, los cuales demuestran el cumplimiento del objetivo general de la investigación. Durante la realización del despliegue de los subsistemas de las plataformas televisivas PTARTV y Primicia se recogieron los siguientes datos: tiempo de instalación y tiempo de configuración. Con los datos obtenidos se realizó una comparación (ver tabla 17) donde se evidencia claramente el cumplimiento de la idea a defender en la investigación en curso.

Subsistemas	Tiempo de Instalación (minutos)		Tiempo de configuración (minutos)		Tiempo total (minutos)	
	Manual	Instalador	Manual	Instalador	Manual	Instalador
Web	5	5	10	1	15	6
Programación	5	4	8	1	13	5
Radial	7	3	5	-	12	3
Transmisión	5	3	5	-	10	3
Monitoreo	5	2	4	-	9	2

Tabla 17: Comparación entre despliegue manual y despliegue mediante un instalador.

La comparación realizada en la tabla anterior demuestra que el despliegue o prueba de alguna de las plataformas televisivas mencionadas en la presente investigación se realizará más rápido y efectivo con la utilización de la aplicación construida.

4.11 Conclusiones parciales

En el presente capítulo se realizaron los diagramas de clases del diseño los cuales describen la arquitectura seleccionada para el desarrollo del sistema, siendo esta la arquitectura en 2 capas (Presentación y Lógica del Negocio). Esta arquitectura ha sido de gran ayuda para el desarrollo del sistema porque al estar bien definida, es más fácil estructurar el contenido de trabajo y de las clases, haciendo menos complicada la implementación del sistema. También se realizaron las pruebas al sistema validando los requisitos necesarios para el correcto funcionamiento del mismo, dándole la calidad requerida al producto. Se validó la aplicación mediante los resultados obtenidos en la realización del despliegue de algunos subsistemas de las plataformas PTARTV y Primicia, demostrando que con la utilización de la aplicación construida este proceso se realiza de forma rápida.

Conclusiones generales:

Una vez realizado el desarrollo de la investigación se arriban a las siguientes conclusiones:

- Se utilizaron los métodos de investigación científica, tanto teóricos como empíricos. Estos permitieron conocer el estado del objeto de estudio de la investigación y la comprensión de forma satisfactoria de toda la situación problemática de la investigación.
- Se realizó el estudio de otras aplicaciones existentes, demostrando que no se cuenta con una herramienta capaz de realizar la instalación y configuración completa de plataformas desarrolladas en entornos web y escritorio al mismo tiempo.
- Se realizó un estudio y análisis de las tendencias y tecnologías actuales permitiendo una correcta elección de la metodología de desarrollo, la herramienta CASE, el lenguaje de programación y el IDE de desarrollo.
- El uso de la metodología de desarrollo RUP posibilitó generar artefactos que documentan de forma clara y concisa la solución informática permitiendo que esta pueda ser utilizada y consultada.
- Se diseñaron y aplicaron 2 casos de pruebas funcionales y 2 casos de pruebas unitarias logrando recolectar las no conformidades del sistema. Estas pruebas permitieron la validación de los requisitos necesarios para el correcto funcionamiento del sistema, dándole la calidad requerida al producto.
- Como resultado de la investigación se logró el desarrollo de un asistente de configuración para la instalación de las plataformas televisivas PTARTV y Primicia. Este asistente permite la realización de forma más rápida y efectiva del despliegue o prueba de dichas plataformas.

Recomendaciones:

- Seguir estudiando y mejorando el dominio de las librerías Qt y del lenguaje utilizado en la realización de la aplicación.
- Continuar con el desarrollo y soporte de la aplicación construida.
- Agregar funcionalidades a la aplicación y mejorar las existentes para así dar más facilidad de uso al usuario.
- Ampliar el alcance de la aplicación con el objetivo de que su uso no sea solamente para las plataformas televisivas PTARTV y Primicia.
- Seguir investigando en función de optimizar los resultados obtenidos y robustecer la solución propuesta.
- Realizar otros tipos de pruebas que permitan verificar la calidad del sistema.

Glosario de términos:

Instalación de software: es el proceso por el cual nuevos programas son transferidos a un computador y, eventualmente, configurados, para ser usados con el fin para el cual fueron desarrollados.

Configuración: es la adaptación de una aplicación software o un elemento hardware al resto de los elementos del entorno y a las necesidades específicas del usuario. (27)

Software: equipamiento lógico o soporte lógico de una computadora digital; comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos, que son llamados hardware. (28)

Plataformas televisivas: portales que ofrecen varias emisiones online de canales televisivos.

Aplicación: Es una clase de programa informático creado para facilitar al usuario un determinado tipo de trabajo. Esto lo caracteriza frente a otros programas como los sistemas operativos, las utilidades y los lenguajes de programación.

Entidad: Es cualquier concepto del mundo real con una existencia independiente.

Fichero: Directorio. Agrupación de archivos de datos, atendiendo a su contenido, a su propósito o a cualquier otro criterio.

Clase: Define los atributos y los métodos de una serie de objetos.

Caso de Uso: secuencias de acciones que el sistema puede llevar a cabo interactuando con sus actores.

Prototipo de interfaz: maqueta visual de la futura aplicación en forma de imagen.

Trabajos Citados:

1. **Terra.** *Ejecutable.* [En línea] http://www.terra.es/personal/lermon/cat/articles/evin0369.htm.

2. **Hernández, Vismar Santos.** Gestiopolis.com. *Estudio sobre la industria del Software a nivel mundial. Caracterización en América Latina y Cuba.* [En línea] 23 de Junio de 2009. http://www.gestiopolis.com/administracion-estrategia/estudio-sobre-la-industria-del-software-en-america-latina.htm.

3. **Albet.** Albet Ingeniería y Sistemas. *Albel.* [En línea] www.albet.cu/sobre-albet.

4. **Definición.de.** [En línea] http://definicion.de/.

5. **IBM.** UML, RUP, and the Zachman Framework: Better together. [En línea] http://www.ibm.com/developerworks/rational/library/nov06/temnenco/.

6. **Daily Miranda Pardo, Juniel Tamayo Hernández.** *Actividades del proceso de despliegue de software.* [En línea] http://www.monografias.com/trabajos-pdf/despliegue-soluciones-software/despliegue-soluciones-software.pdf.

7. **Carzaniga, Antonio.** A Characterization of the Software Deployment Process and a Survey of Related Technologies. 1997.

8. **WINE.** WINE HQ. *Package.* [En línea] http://www.winehq.org/site/docs/wineusr-guide/glossary.

9. **MSDN.** *Windows Installer.* [En línea] http://msdn.microsoft.com/en-us/library/cc185688(VS.85).aspx.

10. **Flexera.** *InstallShield.* [En línea] http://www.flexerasoftware.com/products/installshield.htm.

11. **Juan Pablo Gómez Gallego, Jorge Galves.** *Fundamentos de la metodología RUP.* [En línea] 16 de Septiembre de 2007. http://www.scribd.com/doc/297224/RUP.

12. **Booch, G. Rumbaugh, J. y Jacobson, I.** *El Lenguaje Unificado de Modelado pág 11.* 2000.

13. **EVA.** Introducción al Proceso Unificado de Desarrollo de Software (RUP) y al Lenguaje Unificado de Modelado (UML). [En línea] http://eva.uci.cu/file.php/102/Curso_2010-2011/Clases/Semana_02/Conferencia_3/Materiales_complementarios/Introduccion_a_RUP_y_UML.pdf.

14. **S, Rosenblum Nenad Medvidovic y David.** *Domains of Concern in Software Architectures and Architecture Description Languages.* California : s.n., 1997.

15. **Scribd.** Ingeniería de Software 1. *Capítulo 1 Herramientas CASE.* [En línea] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

16. **Altamirano, Ing. Alfonso Valdez.** *Comparativo de Entornos de Desarrollo Integrados (IDE´s).* s.l. : www.ubicuos.com.

17. **Aguilar, Luis Joyanes.** *Fundamentos de programación. Algoritmos, estructuras .Pág 21-34.* España: McGraw Hil : s.n., 2003.

18. **Mora, Sergio Luján.** *C++ paso a paso pág 1-2.*

19. **Carrero, Angel.** *Qt Creator, un completo entorno de desarrollo.* [En línea] 10 de Junio de 2010. http://www.programacion.com/noticia/qt_creator-un_completo_entorno_de_desarrollo_1723.

20. **Garcerant, Iván.** Tecnología y Synergix. *Modelo de Dominio.* [En línea] 10 de Julio de 2008. http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

21. **Mastermagazine.** MASTERMAGAZINE. *Definición de Arquitectura Software.* [En línea] http://www.mastermagazine.info/termino/3916.php.

22. **Larman, Craig.** *"UML y patrones" Tomo I. Capítulo 18, Páginas 185-215.*

23. **Larman, Craig.** *UML y Patrones.Una introducción al análisis y diseño orientado a objetos y al proceso unificado. .* s.l. : 2da edición.

24. **Universidad de Burgos.** Introducción a la Programacion Orientada a Objetos. [En línea] Octubre de 1999. [Citado el: 5 de Diciembre de 2007.] http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_1.htm.

25. **Torossi, Gustavo.** *El Proceso Unificado de desarrollo de Software pág 40-45.*

26. **Domínguez Mora, Dunier.** *Implementación de los Subsistemas Web y Transferencia.* La Habana : s.n., 2009.

27. **Mastermagazine.** Definición de Configurar. [En línea] http://www.mastermagazine.info/termino/4404.php.

28. **WordReference.com.** Definición de Software. [En línea] http://www.wordreference.com/definicion/software.