Facultad 1

Título: Desarrollo del Centro de Software de Nova Escritorio 5.0.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor: Yosvany Hernández Amaro

Tutores: Ing. Juan Manuel Fuentes Rodríguez

Ing. Dairelys García Rivas

Ing. Jesse Daniel Cano Otero

Ciudad de La Habana

Junio de 2014

Declaración de autoría

Declaro que soy el autor del presente Trabajo de Diploma y reconozco a la Facultad 1 de la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo.
Para que así conste firmo la presente a los ____ días del mes de _____ del año 2010.

_____

Yosvany Hernández Amaro

_____ _____

Ing. Juan Manuel Fuentes Rodríguez Ing. Dairelys García Rivas

_____

Ing. Jesse Daniel Cano Otero

Síntesis de los Tutores

Nombre y Apellidos: Ing. Juan Manuel Fuentes Rodríguez.

Email: jfuentesr@uci.cu

Especialidad de graduación: Ingeniería en Ciencias Informáticas.

Años de experiencia en el tema: 3 años.

Años de graduado: 2 años.

Nombre y Apellidos: Ing. Dairelys García Rivas

Email: dgrivas@uci.cu

Especialidad de graduación: Ingeniería en Ciencias Informáticas.

Años de experiencia en el tema: 2 años.

Años de graduado: 2 años.

Nombre y Apellidos: Ing. Jesse Daniel Cano Otero

Email: jdcano@uci.cu

Especialidad de graduación: Ingeniería en Ciencias Informáticas.

Años de experiencia en el tema: 4 años.

Años de graduado: 5 años.

Agradecimientos

A mis padres por no dejarme solo en ningún momento, por apoyar y respetar mis decisiones, respaldarme ante las necesidades que exige esta carrera y por complacer mis caprichos.

A mi chiquilla por su cariño inocente en todo momento, y en especial, las tantas veces que lo necesité y encontré en ella sin que apenas se diera cuenta.

A mis hermanos Anailys e Idelvis por tantos momentos que necesité de su ayuda y recibí su respaldo, sin decir que no.

A Elsita por haberme aceptado no solo como uno más de la familia sino como un hijo.

A mi primo Samuel por estar pendiente de mí formación y apoyarme en todo momento.

A Katy por haberse convertido en esa persona tan especial que necesitamos tener al lado en momentos que exigen de ser fuerte, por ayudarme a sacar lo mejor de mí y por sus regaños oportunos.

A mis amigos Hanoi, Javier, Isabel, Yamilka y Nestor por todos los momentos juntos que me dieron la dicha de ganar su amistad.

A Iván y Wendy por haber sido mi pequeña familia universitaria.

A mis compañeros de aula por ayudarme a recorrer el camino hasta aquí durante estos 5 años.

A mis tutores, sin ustedes nada hubiera sido posible, gracias por su guía y su tiempo.

Y en especial a la Revolución que me permitió formarme como profesional.

A todos ¡Muchas Gracias!

Yosvany

Resumen

La gestión de aplicaciones es uno de los factores que ha afectado la adaptabilidad de usuarios novel en sistemas basados en GNU/Linux. La presente investigación surge como respuesta a los problemas detectados en el *software* encargado de este proceso, en la Distribución Cubana de GNU/Linux Nova Escritorio 4.0. Estos son provocados principalmente por no poder disponer de los servicios que requiere el *software* para su funcionamiento íntegro, lo que trae consigo, entre otros elementos, que se dificulte su actualización y mantenimiento por parte de los desarrolladores del departamento de Sistema Operativo Nova.

La realización de un estudio de las soluciones actuales existentes para la gestión de aplicaciones en sistemas operativos de interés, arrojó como resultado que estas no resultan factibles para la distribución cubana Nova Escritorio, dadas las deficiencias detectadas durante su análisis, pero permitió determinar las novedades enmarcadas en el objeto de estudio, haciendo uso de los diferentes métodos científicos de investigación.

Se utilizó el lenguaje de programación Python, así como varias bibliotecas nativas de este y de GTK+ principalmente, quedando documentadas estas y el resto de las bibliotecas y herramientas necesarias para el desarrollo del *software*. Este proceso estuvo guiado por la metodología OpenUp, respetando y dando solución a los requisitos definidos para el Centro de *Software* de Nova Escritorio 5.0.

Como resultado de la investigación se obtuvo la primera versión de un *software* para la gestión de aplicaciones desarrollado para Nova Escritorio, aunque puede ser utilizado en sistemas operativos de características similares. Se eliminaron los servicios inservibles para la distribución y se logró optimizar el sistema operativo con la incorporación al Centro de *Software* de una sección para gestionar las actualizaciones. Las pruebas de funcionalidad permitieron asegurar la calidad del *software* y el cumplimiento de las funcionalidades requeridas, siendo un paso de avance en la gestión de aplicaciones de este sistema.

Índice

Contenido

Tablas

Figuras

Introducción

Cuba no está exenta de los diversos problemas que acarrea el dominio de las Tecnologías de la Información y las Comunicaciones (TIC) (Monteagudo Peña, 2004) por parte de las grandes compañías que privatizan el *software*. Por tal motivo desde hace unos años avanza en pos de un desarrollo autónomo en este campo, y lo más importante, por tener total capacidad para decidir sobre la forma en que se desarrollan y usan las tecnologías en el territorio nacional. En el año 2002 surge la Universidad de las Ciencias Informáticas (UCI), denominada en sus inicios “Proyecto Futuro”, como apoyo al proceso de informatización del país.

Con el objetivo de alcanzar la soberanía tecnológica, se inicia un proceso de migración hacia *Software* Libre y Código Abierto protagonizado por el Centro de *Software* Libre (CESOL) de la Universidad de las Ciencias Informáticas, en el que se considera a la Distribución Cubana de GNU/Linux Nova como su plataforma operativa oficial. Este producto es destinado al usuario final y está enfocado a la búsqueda de alternativas para contrarrestar, ante todo, la resistencia ante el cambio tecnológico que ofrecen usuarios familiarizados con Microsoft Windows (Lafita Mosqueda, 2014). Además, representa un paso de avance en la conquista de la seguridad nacional y autonomía sobre el uso de las Tecnologías de la Información y las Comunicaciones, disminuyendo la dependencia hacia productos y *software* privativos.

En el 2011, Nova en su versión 3.0 lanzó un producto más sólido y sencillo de utilizar, enfocado en la estabilidad y soporte del sistema por concepto de actualizaciones en un largo período de tiempo. Para los usuarios de escritorio se desarrollaron 2 variantes, Nova Escritorio y Nova Ligero, esta última orientada a las máquinas de escritorio con bajas prestaciones de hardware. Para los usuarios estándares se desarrolló Nova Escritorio, siendo éste un sistema que permite explotar las potencialidades del hardware, incorpora mejoras visuales, de usabilidad y basa su interfaz predeterminada en el entorno de Gnome 2.30.

Con la salida de Nova Escritorio 4.0 se produce el salto a la arquitectura de Gnome 3.0 y con esta se incorporan a la distribución, entre otros elementos, la forma novedosa del manejo de ventanas, facilitando la interacción con el usuario, creando un ambiente de escritorio sencillo y libre de distracciones, haciendo un mayor uso de la pantalla (Fuentes Rodríguez, 2012).

Un elemento que ha afectado la funcionalidad del sistema operativo Nova es la gestión de aplicaciones. Esto constituye un problema fundamental de los sistemas basados en GNU/Linux desde su surgimiento. Estrategias adoptadas por las diferentes distribuciones han posibilitado el desarrollo de *software* que realizan esta función de forma gráfica permitiendo buscar, instalar y desinstalar aplicaciones en el sistema operativo, además de ofrecer funcionalidades adicionales para realizar una mejor especificación de las aplicaciones disponibles en sus repositorios. Uno de los servicios más difundidos en este tipo de *software*

ha sido el de *ranking*, el cual permite determinar la aceptación y competencia de las aplicaciones mediante las calificaciones de los usuarios que interactúan con el *software*.

En sus inicios, Nova Baire en sus versiones 1.0 y 2.0 optó por utilizar Summon para la gestión de aplicaciones. Este *software* se desarrolló como variante para Nova del Gestor de Paquetes Entropy utilizado por Sabayón (Bodnar, 2013a). Uno de sus principales inconvenientes estaba determinado por la forma precaria de interactuar con los usuarios. Su apariencia y la carencia de una intuitiva interfaz gráfica hacen que tenga poca aceptación en la comunidad. Además, este gestor es ineficiente en la realización de búsquedas, ya que no diferencia entre aplicaciones y paquetes, lo cual dificulta el proceso de instalación al listar todas las dependencias de cada *software* (Sabayon, 2014). El uso de Summon llega su fin en la distribución cubana con el lanzamiento de Nova 3.0.

Entonces se decide utilizar la herramienta para gestionar paquetes Gnome *PackageKit*. Uno de sus problemas fundamentales radicaba en la configuración de sus canales de *software*, ya que *PackageKit* no ofrece una manera de añadir nuevos repositorios a su sistema, sólo le permite activar o desactivar los conocidos incluidos en su instalación por defecto. Para añadir uno nuevo es necesario instalar un paquete con toda la información necesaria, haciendo que este proceso no esté al nivel de todos los usuarios. Además, ofrece una respuesta lenta ante las acciones solicitadas por el usuario (Hughes, 2012)

En Nova 4.0 se decide recurrir al *software* para la gestión de aplicaciones de una de las principales distribuciones de GNU/Linux, Ubuntu. Para utilizar este Centro de *Software* se realizaron modificaciones en la interfaz, ajustándola a la identidad de Nova y fue necesario deshabilitar servicios consumidos por este, para evitar errores en el funcionamiento del *software* que pudieran afectar la gestión de aplicaciones del sistema.

Una desventaja fundamental de utilizar el Centro de *Software* de Ubuntu en Nova Escritorio 4.0 son los servicios que utiliza, ya que estos son gestionados y administrados por Canonical LTD (ALEGSA, 2014), por lo que se requiere tener acceso a internet para su uso. Esto constituye una limitante en nuestro país y se vería afectada la gestión de aplicaciones al efectuar el proceso de migración a *Software* Libre en el territorio nacional. Estos servicios se enfocan en lograr un mejor producto, apoyados por las diversas comunidades de usuarios a través de servicios de *ranking*, recomendaciones, compra de aplicaciones, videos descriptivos de las aplicaciones a instalar; todos pensados para interactuar en la red de redes, por lo que no serían de utilidad en la distribución de GNU/Linux Nova Escritorio 5.0, ya que en su base no se tienen en cuenta las necesidades de los cubanos, con lo que se ve afectada la adaptabilidad de estos sistemas a la distribución.

En cada actualización, Ubuntu realiza mejoras de sus productos que son subidas a los repositorios. Estas resultan necesarias para el *software* utilizado en la Distribución Cubana de GNU/Linux Nova Escritorio 5.0, ya que se incorporan mejoras de seguridad y otros aspectos funcionales. Al no disponer de los servicios

antes mencionados, no es posible actualizar el *software* desde la versión modificada para Nova; es necesario descargarlo nuevamente, realizar las modificaciones y subirlo a los repositorios de la distribución cubana para su utilización por el resto de los usuarios. Esto trae consigo que sea poco sostenible mantener el *software* actualizado, teniendo en cuenta además el esfuerzo que esto requiere para los responsables de dicha tarea.

La facilidad de uso, navegabilidad y diseño son elementos usados con frecuencia por los usuarios para medir la calidad de una aplicación, fundamentados a partir del cumplimiento de los objetivos y funcionalidades que debe realizar la misma. El diseño es un elemento que limita la aceptación y usabilidad del actual Centro de *Software*. El espacio es desaprovechado en las vistas para el manejo y realización de operaciones con las aplicaciones, lo cual reduce el atractivo del *software*. Un ejemplo claro se puede apreciar a la hora de instalar una aplicación: cuando esta es seleccionada aparece su logotipo en la parte izquierda y al otro extremo de la pantalla, en la parte derecha es que tiene un botón para interactuar con el usuario, mostrando la opción a realizar (Ver Anexo 1). Además, el no contar con los servicios mencionados anteriormente provoca que se muestren secciones deshabilitadas en la interfaz de algunas categorías (Ver Anexo 2).

En el Centro de *Software* de la Distribución Cubana de GNU/Linux Nova no se gestionan las actualizaciones y el manejo de versiones de las aplicaciones existentes, sólo se habilita para la instalación la última versión existente en el repositorio. Esto implica que el usuario tenga que mantener su *software* actualizado y con frecuencia esto no es posible debido al soporte que algunas compañías brindan a sus aplicaciones, creando en ocasiones incompatibilidades entre programas con el lanzamiento de nuevas versiones. Conjuntamente con el Centro de *Software* de Nova, se ejecuta un Gestor de Actualizaciones que ofrece la opción de instalar las novedades subidas al repositorio. Este podría ser eliminado de la instalación por defecto de Nova al integrarlo con el *software* a desarrollar, reduciendo de esta forma la carga del sistema operativo.

Una vez analizado cuánto afecta la carencia de un *software* propio para la gestión de aplicaciones en la Distribución Cubana de GNU\Linux Nova Escritorio, surge el siguiente **problema a resolver:** ¿Cómo facilitar la gestión de aplicaciones en el sistema operativo Nova Escritorio 5.0?

El **objeto de estudio** del presente trabajo se centra en la gestión de aplicaciones en sistemas operativos y su **campo de acción** incide en la gestión de aplicaciones del sistema operativo Nova Escritorio 5.0.

Como **objetivo general de la investigación** se determina**:** desarrollar un Centro de *Software* que solucione las deficiencias presentes en la gestión de aplicaciones de la Distribución Cubana de GNU/Linux Nova Escritorio 4.0, para facilitar la gestión de aplicaciones en Nova Escritorio 5.0.

Para complementar el objetivo de la investigación, se han formulado como **objetivos específicos:**

- Sistematizar el estudio sobre los programas para la gestión de aplicaciones en los sistemas operativos modernos.

- Desarrollar el Centro de *Software* de la Distribución Cubana de GNU/Linux Nova Escritorio 5.0.
- Comprobar el funcionamiento del Centro de *Software* para validar el trabajo realizado.

Para darle cumplimiento a los objetivos planteados, se realizarán las siguientes **tareas de investigación:**

- Estudio de los diferentes *software* conocidos para la gestión de aplicaciones en sistemas operativos modernos.
- Selección de la metodología y las herramientas a utilizar para el desarrollo del Centro de *Software* de la Distribución Cubana de GNU/Linux Nova Escritorio 5.0.
- Realización del análisis y diseño del *software* propuesto.
- Implementación del Centro de *Software* de la Distribución Cubana de GNU/Linux Nova Escritorio 5.0.
- Realización de pruebas de aceptación al Centro de *Software* implementado para validar el trabajo realizado.

Para sustentar el propósito de la investigación se plantea la siguiente **Idea a defender**: si se desarrolla un Centro de *Software* que solucione las deficiencias presentes se facilitará la gestión de aplicaciones en la Distribución Cubana de GNU\Linux Nova Escritorio 5.0.

Esta investigación está sustentada sobre la base de la utilización de diferentes **métodos científicos**. Se decidió recurrir a los métodos tanto teóricos como empíricos. De los teóricos se utilizaron el histórico-lógico, el analítico-sintético y el inductivo-deductivo. Como método empírico se utilizó la observación.

Métodos Teóricos:

Histórico-Lógico: Permitió el estudio de los principales *software* para la gestión de aplicaciones utilizados actualmente en el mundo, así como sus antecedentes y evolución en la Distribución Cubana de GNU/Linux Nova. Facilitó la obtención de elementos de interés de cada uno para tener en cuenta en el *software* a desarrollar.

Analítico-Sintético: Se utilizó en el análisis de la bibliografía relacionada con los sistemas que permiten la gestión de aplicaciones en los sistemas operativos GNU/Linux, posibilitando su comprensión y permitiendo la captura de aquellos elementos que se relacionan con el objeto de estudio.

Inductivo-Deductivo: Permitió extraer los métodos y funcionalidades del *software* utilizado para el desarrollo, así como sus patrones de diseño para resolver problemas particulares del *software* en desarrollo.

Métodos Empíricos:

Observación: Este método permitirá mediante la constatación visual, chequear el estado actual de las diferentes propuestas usadas para la gestión de aplicaciones en sistemas operativos.

Resultados esperados:

Se espera obtener un Centro de *Software* que mejore la gestión de aplicaciones en la Distribución Cubana de GNU/Linux Nova Escritorio 5.0 y posea una interfaz moderna e intuitiva.

Este documento se encuentra estructurado en 3 capítulos descritos a continuación:

En el **primer capítulo** titulado **Fundamentación teórica**, se definen los conceptos fundamentales asociados al objeto de estudio, necesarios para el desarrollo y comprensión de la investigación. Se hace una descripción de las herramientas y tecnologías a utilizar para el desarrollo del Centro de *Software*.

En el **segundo capítulo** titulado **Requisitos y Arquitectura del Centro de *Software* de Nova Escritorio 5.0**, se identifican los requisitos y la arquitectura del *software*. Además se muestran y describen los artefactos a obtener en cada una de las iteraciones propuestas por la metodología seleccionada.

En el **tercer capítulo** titulado **Implementación y Evaluación del Centro de *Software* de Nova Escritorio 5.0**, se realiza un enfoque en la implementación y pruebas al *software*. Se hace una explicación de la implementación de la solución propuesta. Además, se realizan casos de prueba para validar el cumplimiento de los requisitos definidos para confirmar la calidad del *software*.

Capítulo 1: Fundamentación Teórica.

Introducción

En este capítulo se realizará un estudio de los principales *software* utilizados para la gestión de aplicaciones, así como la descripción de los más reconocidos en la actualidad. Se llevará a cabo una conceptualización de los términos y procesos más importantes que rodean el problema a resolver. Además se realizará un análisis de las herramientas, bibliotecas y tecnologías existentes que serán usadas en el desarrollo del Centro de *Software*.

1.1. Principales conceptos

Paquete

Un paquete es un grupo de uno o más archivos que son necesarios tanto para la ejecución de un programa de computadora, como para agregar características a un programa ya instalado en la computadora o en una red de computadoras. El término paquete de *software* también es utilizado en la programación orientada a objetos, para nombrar a un grupo de clases relacionadas de un programa. En este significado, los paquetes son especialmente útiles para medir y controlar el acoplamiento inherente de un programa. (ALEGSA, 2009)

Paquete en Linux

Los paquetes son la agrupación instalable de los diferentes archivos necesarios para que un programa funcione. Se pueden parecer a los programas que se descargan de internet para Windows, es decir, un archivo ejecutable que contiene un programa. La diferencia es que en GNU/Linux los paquetes no son ejecutables, sino que son gestionados por terceras aplicaciones. Además, los paquetes de Linux suelen ser muchísimo más compactos y reducidos, ya que no traen consigo sus dependencias y, llegado el caso de necesitarse, serán instaladas o se le informará de la necesidad de hacerlo, según el gestor de paquetes que se esté utilizando. (UTLAI, 2014)

Las aplicaciones Linux se suministran normalmente en varios formatos o tipos de paquetes:

- Paquetes de Fuentes (comprimidos pero no empaquetados): se reconocen por la extensión: *.tgz*, *.tar.gz, .tar.bz2*.
- Paquetes binarios (*software* pre-compilado): se reconocen por la extensión: *.deb*, *.rpm*, *.mdk*. Las distribuciones basadas en *Red Hat* y Fedora utilizan los .rpm, mientras que los *.deb* se utilizan en distribuciones basadas en Debian como es el caso de Ubuntu, Nova, entre otras.
- Paquetes de fuentes empaquetadas (*software* instalable): se reconocen por la extensión *.ebuild*.

Aplicación

Una aplicación es un programa informático que se ejecuta sobre el sistema operativo creado para llevar a cabo o facilitar una tarea en un dispositivo informático (ALEGSA, 2013). En general, es un programa compilado (aunque a veces interpretado) que puede tener distintas licencias de distribución mostrando algún tipo de interfaz, ya sea de texto o gráfica (o ambas) (Mastemagazine, 2013).

Repositorio

Un repositorio es una colección de paquetes de programas de una distribución de Linux específica que generalmente contiene archivos binarios pre-compilados que pueden ser descargados e instalados por los usuarios de la distribución correspondiente. Es posible también encontrar paquetes de código fuente. (Depizzol, 2013)

Gestor de paquetes

Un Gestor de paquetes es una colección de herramientas que sirven para automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de *software*, manteniendo una base de datos en la que se refleja el *software* instalado, versiones, dependencias y otros datos. Estos pueden funcionar en distribuciones en las que no vienen instalados por defecto, aunque lo recomendable es utilizar el que la distribución aconseja. En un sistema con varios gestores de paquetes, se puede instalar varias veces el mismo *software*, aplicación, biblioteca, una vez por cada gestor de paquetes, ya que no comparten las bases de datos, lo que puede ocasionar inestabilidades en el sistema e incluso su colapso y quedar inutilizable. Cada Gestor de paquetes tiene sus propios ficheros de configuración en los cuales, entre algunas cosas más, se guardan los lugares donde buscar el *software*, llamados repositorios. (López Torrijos, 2012)

1.2. Análisis de sistemas homólogos

Una vez definidos los conceptos relacionados con el objeto de estudio se da paso al análisis de algunos de los principales sistemas encargados de la gestión de aplicaciones, con el fin de obtener una panorámica de las propuestas utilizadas actualmente en varias distribuciones de GNU/Linux y sistemas operativos de interés. Su análisis estructural y funcional apoyado en el diseño permitirá decidir si alguna cumple con las expectativas planteadas para el Centro de *Software* de Nova Escritorio 5.0, de no ser así se tendrán en cuenta elementos de interés y novedades encontradas en la investigación.

1.2.1. *Gestor de paquetes Synaptic*

El Gestor de paquetes Synaptic ha sido uno de los programas más importantes para las distribuciones basadas en Debian. Permite instalar *software* en el equipo y gestionar el *software* que está ya instalado. El Gestor de paquetes Synaptic resuelve estas dependencias automáticamente sin necesidad de ser indicadas manualmente por el usuario, haciendo más eficiente el proceso de instalación. Su funcionamiento está basado en el gestor de paquetes APT (Herramienta Avanzada de Empaquetamiento) y proporciona

funciones que son similares a las de la herramienta de línea de comandos *apt-get* en un entorno gráfico. (INTEF, 2010)

En detalle, el Gestor de paquetes Synaptic proporciona las características siguientes:

- Instalar, eliminar, configurar, actualizar y descargar paquetes.
- Actualizar el sistema completo.
- Administrar repositorio de paquetes.
- Buscar paquetes por nombre, descripción y otras propiedades.
- Seleccionar paquetes por estado, sección, nombre o un filtro personalizado.
- Ordenar paquetes por nombre, estado, tamaño o versión.
- Examinar toda la documentación en línea relacionada con un paquete.
- Bloquear paquetes a la versión actual.
- Forzar la instalación de una versión de paquete específica.

Para aplicar cambios sobre el sistema de paquetes es necesario tener permisos de administración. Al iniciarse solicita la autenticación del usuario para determinar sus privilegios en el sistema. Synaptic debe ser utilizado con cuidado, ya que el sistema puede quedar inestable.

La ventana principal muestra los siguientes componentes (Ver Anexo 3):

- Barra de menú: Contiene menús que proporcionan acceso a todas las funciones de Synaptic.
- Barra de herramientas: Proporciona acceso directo a las funciones principales de “Recargar”, “Marcar todas las actualizaciones”, “Aplicar y Buscar”.
- Selector de categoría: Proporciona categorías para reducir la lista de paquetes mostrada.
- Lista de paquetes: Lista todos los paquetes conocidos. Puede reducirse utilizando filtros.
- Descripción del paquete: Proporciona información acerca del paquete seleccionado.
- Barra de estado: Visualiza el estado actual de Synaptic. (LliureX, 2011)

Nota: “Recargar” permite consultar todos los repositorios y reconstruir la lista de paquetes conocidos.

El uso del Gestor de paquetes Synaptic disminuyó considerablemente con el lanzamiento del Centro de *Software* de Ubuntu y su eliminación de la instalación por defecto de esta distribución. A pesar de proporcionar al usuario las características antes mencionadas, es importante resaltar que este gestor de paquetes no es tan fácil de usar como el *software* que lo reemplazó en Ubuntu. Este es un elemento importante a tener en cuenta a fin de mejorar la gestión de aplicaciones en Nova Escritorio 5.0, por lo que se decide no utilizar este *software*.

1.2.2. *Gnome PackageKit*

PackageKit es un sistema que permite instalar, desinstalar y actualizar *software* de forma sencilla. Está diseñado para unificar todas las herramientas gráficas de *software* utilizadas en diferentes distribuciones. Utiliza herramientas de otras distribuciones como *yum* y *apt* mediante scripts o interfaces compiladas. Esto

es posible ya que PackageKit no las reemplaza, en cambio provee un conjunto de abstracciones que pueden ser utilizadas por otros administradores de interfaz gráfica o de texto. (Hughes, 2007)

Este Gestor de Paquetes en sí en un demonio activado por el sistema, llamado *packagekitd*. Ser "activado por el sistema" significa que se ejecuta como un proceso del sistema sin estar vinculado a ninguna sesión de usuario, posibilitando que no se interrumpan instalaciones o actualizaciones de larga duración si el usuario cierra la sesión (Gómez Savino, 2010a). PackageKit no carga el sistema, ya que solo se ejecuta cuando se hace uso de la herramienta visual o la lista de comandos y se detiene una vez concluida la acción a realizar, por lo que no demora el inicio de sesión ni consume memoria cuando no es utilizado. (Ver Anexo 4)

PackageKit permite además:

- Poner tareas en cola, lo que significa que varios usuarios pueden usarlo en paralelo sin bloquear completamente el uno al otro, ya que las tareas se ejecutan por orden.
- Instalar actualizaciones de seguridad en el arranque del sistema.
- Remover dependencias de archivos.
- Abrir formatos de archivo desconocidos.

Utiliza PolicyKit para la autenticación de usuarios, lo que permite a administradores controlar el nivel de acceso sobre operaciones privilegiadas. Esto posibilita que no sea necesario correr la interfaz gráfica como administrador, solo hay que ejecutar el subproceso que hará la tarea de instalar, desinstalar o actualizar en el momento especificado (Petersen, 2013).

Este *software* presenta problemas con la intercomunicación de procesos, ya que cuando se encuentra en ejecución algún subproceso en segundo plano no pueden ser iniciadas nuevas transacciones desde la herramienta gráfica, produciéndose un error donde se le notifica al usuario que debe esperar que concluya la acción que se esté realizando. La configuración de los canales de *software* no está al nivel de todo tipo de usuarios, ya que para agregar fuentes adicionales a las incluidas por defecto en la configuración del *software*, es necesario instalar desde la Terminal un paquete con toda la información referente al nuevo repositorio (Hughes, 2012). Además, poseer varias capas de abstracción para el trabajo con diferentes herramientas para la gestión de paquetes, como es el caso de *yum* y *apt*, provoca que el *software* se vea cargado con capas innecesarias, creando ambigüedad en su funcionamiento y lentitud.

1.2.3. *Centro de Software de Ubuntu*

El Centro de *Software* de Ubuntu es un programa informático utilizado como sistema de gestión de paquetes para este sistema operativo. Surge como estrategia para unificar en una sola aplicación las herramientas para el manejo de paquetes empleadas hasta el momento de forma simultánea (Añadir y quitar programas, Gestor de paquetes Synaptic, el Gestor de actualizaciones y Gdebi), evitando así posibles confusiones por

parte del usuario final. Esta decisión también está dada por el interés de reducir el espacio de los programas que ocupan la instalación por defecto de Ubuntu, a fin de continuar distribuyéndolo en un CD.

Dentro de sus principales funcionalidades permite buscar, instalar y eliminar aplicaciones del sistema operativo haciendo uso del demonio APT, el cual permite realizar acciones sobre los paquetes y sus dependencias de forma autónoma una vez que el usuario decide la operación a realizar sobre las aplicaciones, además de facilitar las actualizaciones automáticas al sistema e incluso actualizar la distribución a una superior.

Tiene la opción de añadir repositorios de terceros para instalar aplicaciones que no se encuentren en los repositorios oficiales de Ubuntu. Cuenta con más de 62000 elementos (bibliotecas, aplicaciones, y paquetes varios) disponibles en los repositorios de esta distribución. Las aplicaciones se dividen en 13 categorías agrupadas en la parte izquierda, las cuales son: Accesorios, Acceso Universal, Ciencia e Ingeniería, Educación, Gráficos, Internet, Juegos, Oficina, Sonido y vídeo, Temas y ajustes, Tipografías, Herramientas para desarrolladores, y Sistema. Algunas de estas categorías cuentan con subcategorías, para tener un mejor orden en su jerarquía de aplicaciones y facilitar su uso. Ofrece aplicaciones libres, de código abierto, aplicaciones privativas, y aplicaciones de pago, estas últimas incorporadas en la versión 3.0 del *software* a través de un sistema de compra. En la portada se muestran al usuario los programas añadidos recientemente a los repositorios y los mejor valorados, establecido por un servicio de ranking entre las puntuaciones de las aplicaciones. Además, permite ver el *software* ya instalado en el sistema y un historial de sucesos (Matthew, 2005).

El uso de Xapian como gestor de base de datos permite realizar búsquedas y actualizaciones simultáneas una vez que aparecen nuevos elementos visibles en el repositorio, garantizando la integridad y permanencia en estado coherente de la base de aplicaciones en caso de fallar la transacción que se estuviera realizando (Xapian, 2014). También ofrece una respuesta rápida al usuario debido a sus criterios de búsqueda definidos por frases, proximidad entre elementos, clasificación probabilística de elementos recientes, entre otros.

Para establecer un control en el sistema de privilegios y autenticación de usuarios, hace uso de la aplicación de herramientas de desarrollo *PolicyKit*, permitiendo tener un mayor nivel de control centralizado del sistema y facilitando la administración de privilegios de los usuarios. (Petersen, 2013)

El lanzamiento del Centro de *Software* proporcionó a los usuarios de Ubuntu un *software* con una interfaz gráfica de usuario más atractiva que la mostrada por su predecesor Synaptic. La navegación y realización de búsquedas, la presencia de iconos en las aplicaciones, las categorías para agrupar el *software* y los servicios incorporados hicieron que se decidiera aprovechar las facilidades de este *software* en Ubuntu desde su versión 12.04, y captaron la atención de los desarrolladores de Nova, tomando este *software* para encargarse de la gestión de aplicaciones en Nova Escritorio 4.0.

1.2.4. *Gestor de paquetes Yellow dog Updater, Modified (YUM)*

YUM es el gestor de paquetes por defecto usado en *CentOS* y *Fedora*. Este se encarga de instalar, eliminar y actualizar paquetes en sistemas basados en *Red Hat Package Manager* (RPM). Estos sistemas utilizan la extensión .rpm para identificar sus paquetes. Al hacer uso de RPM, hace fácil el proceso de actualización de un grupo de máquinas sin necesidad de realizarlo de forma manual en cada una de ellas. Automáticamente calcula las dependencias y da la opción de mostrar al usuario lo que ocurre por detrás de la herramienta gráfica. Al ser una herramienta genérica con funcionalidades básicas para la gestión de *software* puede mostrar, según las preferencias del usuario, la interfaz de Gnome *PackageKit*, *KpackageKit* o *Yumex*, esta última la usada en la instalación por defecto de YUM en Fedora (Gómez Savino, 2010b). Otras de sus principales características son:

- Soporte de múltiples repositorios.
- Configuración simple.
- Operaciones rápidas.
- Comportamiento coherente con los RPM.
- Soporta grupos de paquetes, incluyendo repositorios de grupos múltiples.
- Posee una interfaz simple.

YUM usa un repositorio en línea por defecto pero puede ser configurado para usarlo de forma local.

Como principales vistas para la navegación presenta (Ver Anexo 5):

Vista por paquetes:

En esta se pueden seleccionar diferentes opciones para el listado de paquetes, las cuales pueden ser combinadas entre ellas:

- Categoría: Permite seleccionar una categoría para clasificar los paquetes. Estas pueden ser agrupadas por Grupos RPM, Repositorios, Arquitectura, Tamaños y Edad. Al ser seleccionada una categoría se agrega a la vista un panel con la clasificación correspondiente a la misma.
- Actualizaciones: Sólo se listan los paquetes que actualizan alguno ya instalado.
- Disponibles: Se listan todos los paquetes disponibles para instalar en su sistema.
- Instalados: Se listan sólo los paquetes instalados en el sistema.

Vista por grupos:

En esta vista se pueden seleccionar grupos completos de paquetes a instalar, remover o actualizar.

Vista selección de repositorios:

En esta vista se permite seleccionar cuáles repositorios habilitar o deshabilitar.

Vista salida:

En esta vista se refleja la salida en consola de las tareas realizadas por YUM en el sistema.

Vista cola de paquetes:

En esta vista se muestran los paquetes seleccionados sobre los cuales va a actuar YUM cuando el usuario se lo indique ejecutando el botón “Procesar Cola”. (Proyecto Fedora, 2012b)

Actualmente en el Proyecto Fedora se trabaja en el desarrollo de una nueva herramienta que será incluida como solución alternativa y experimental en Fedora, aunque YUM seguirá siendo su gestor de paquetes principal. Hasta el momento se han observado beneficios en la herramienta, como mejor soporte de idiomas, menor consumo de memoria, mayor rendimiento y velocidad, solucionando las dependencias basándose en RPM. DNF, como se identifica la herramienta está basada en la misma sintaxis de YUM, aunque utiliza algunas bibliotecas específicas como *libsolv* y *hawkey*. (Proyecto Fedora, 2012a)

Una desventaja de utilizar el Gestor de Paquetes YUM en Nova Escritorio está dada por el hecho de que este no es compatible con distribuciones basadas en Debian, por lo que su utilización en Nova Escritorio representaría un cambio significativo en la arquitectura y la reimplementación de muchas funcionalidades en la gestión de paquetes.

1.2.5. *Deepin Software Center*

Deepin *Software* Center (DSC) es una de las aplicaciones más populares de Linux Deepin. La razón por la que esta distribución se ha vuelto una de las más reconocidas y desarrolladas es por el familiar entorno de escritorio de Gnome, trabajado en función de incluir características únicas para la comunidad que lo usa. Presenta un esquema novedoso que rompe con los tradicionales estándares de diseño, resaltando la interfaz con colores e imágenes llamativas. Incluye varios temas para seleccionar a gusto del usuario. (Linux Deepin, 2011)

En la página principal se muestra una lista actualizada de aplicaciones destacadas y ofrece sus recomendaciones basadas en las calificaciones otorgadas por los usuarios mediante un servicio de ranking (Ver Anexo 6).

Para buscar las aplicaciones aparece el botón “Repositorio”, donde éstas se muestran agrupadas por categoría. Para instalarlas se muestra el botón “Instalar”. Una vez instalada se muestra un botón “Inicio”, dando la opción de iniciarla desde el Centro de *Software*. Para desinstalar se listan en una pestaña “Desinstalar” solo las aplicaciones instaladas por el usuario. (Unixmen, 2008)

DSC tiene un Gestor de Actualizaciones que se encarga de gestionar las últimas actualizaciones disponibles y notificarlo al usuario. Permite realizar descargas en paralelo, pausar y reanudar descargas y limpiezas de caché, realizando estas tareas con un multi-hilo de descarga. Para descargas rápidas, DSC incorpora fuentes oficiales de Ubuntu.

Al realizar un estudio de este *software* se encontraron varios elementos que resultan de interés para Nova Escritorio 5.0 como son la sencillez, facilidad de uso y su interfaz intuitiva, pero su uso en la distribución cubana traería consigo problemas con los servicios que utiliza, por lo no se solucionarían las causas que dieron lugar al planteamiento del objeto de estudio de esta investigación. Además, el Deepin *Software*

Center está basado en GTK 2, por lo que sería un retroceso en cuanto a la versión de GTK utilizada en Nova Escritorio.

1.2.6. *Gestor de Paquetes Entropy*

Provee una forma moderna y fácil de utilizar sistemas basados en *Gentoo*, añadiendo un alto nivel de inteligencia artificial en el mundo de los gestores de paquetes, debido a su sistema de instalación. *Entropy* descarga el paquete de código binario de la aplicación y en dependencia de la arquitectura del sistema se instala, a diferencia del *Portage* de *Gentoo* que descarga el paquete de código fuente completo y por recetas de compilación lo compila para después instalarlo. Esto hace que sea un Gestor de Paquetes rápido, eficiente y a la vez consuma menos recursos. Aunque es independiente de *Portage*, a la vez es compatible 100% con éste. (Ver Anexo 7)

Utiliza la herramienta de línea de comando *equo*. Esta juega un papel importante, ya que se encarga al final de la instalación del sistema de remover los paquetes que el instalador de Sabayón reconoció que no serían necesitados para el sistema (Sabayon, 2014).

Uno de sus principales inconvenientes es la ineficiencia en la realización de búsquedas, ya que lista todas las dependencias de cada aplicación, lo que dificulta el proceso de instalación al no estar bien definidas aplicaciones y paquetes. Además, Entropy no es compatible con Gnome 3, por lo que solo puede ser utilizado en versiones inferiores de Gnome y sistemas basados en Gentoo. Para Nova Escritorio 5.0, utilizar este Gestor de Paquetes representaría un retroceso en cuanto a la versión de Gnome, o implicaría basar el sistema en *Gentoo*, siendo esto un cambio significativo en la arquitectura, por lo que no es factible su utilización.

1.2.7. *App Store de Mac*

Mac OS X es una serie de sistemas operativos basados en Unix desarrollado, comercializado y vendido por *Apple Inc*. Para facilitarles a los usuarios la búsqueda y descarga de aplicaciones informáticas, esta compañía creó el servicio App Store para los diferentes dispositivos y sistemas que desarrolla. La *App Store* de Mac fue incluida en el Mac OS X v10.6.6, siendo compatible solo con esta versión o posteriores actualizaciones del *software* (Apple, 2014). Como requisitos mínimos necesarios para su funcionamiento están:

- Tener un ordenador Mac con procesador Intel.
- Tener acceso a internet (puede ser de pago).
- Tener una cuenta de *iTunes* o *iCloud*, (ofrece la opción de crear un nuevo ID de *Apple* en el *App Store*).

Se puede ejecutar desde el icono de la *App Store* en el *Dock* o desde el *Finder* en la carpeta Aplicaciones. Para navegar por las aplicaciones, el usuario puede seleccionar la vista deseada:

- Destacado.

- Categorías.
- Compras.
- Actualizaciones.

(Ver Anexo 8) También presenta la opción de buscar aplicaciones por título, creador o editor, categoría o descripción, escribiendo en el campo de búsqueda en la esquina superior derecha de la ventana del *App Store*. Los programas se encuentran almacenados en la carpeta Aplicaciones, accesible desde cualquier ventana del *Finder* y fáciles de encontrar; con sólo comenzar a teclear su nombre, se realiza una búsqueda que devuelve el resultado al instante.

Un concepto algo diferente con respecto a otros sistemas, pero de forma sencilla, es el de instalar y desinstalar programas. Una vez comprado o descargado, este puede presentarse de las siguientes formas:

- En formato comprimido (*.zip*, *.rar*, *.tar.gz*, etc): se representan con el icono de una caja o "paquete". Al hacer doble clic se abren con el programa adecuado, mostrando el contenido en formato *.dmg* o *.app*.
- Como imagen de disco (*.dmg*, *.iso*): al ejecutarlo se monta la imagen a la izquierda del *Finder*. Generalmente el contenido es un archivo *.app*.
- Como paquete de extensión *.app*: se arrastra el fichero a la carpeta "Aplicaciones" o hacia el disco duro y se ejecuta.
- Como instalador ejecutable: algunos programas requieren instalarse a nivel de administrador o requieren componentes especiales, por ejemplo el Office de Microsoft para Mac, la *suite* de Adobe (Photoshop, Dreamweaver, etc), entre otros. Para instalarlos solo hay que ejecutarlos y posteriormente se agregan a la carpeta "Aplicaciones".

También permite hacer copias de seguridad de las aplicaciones descargadas. Para desinstalar un programa, si este no tiene un des instalador, basta con arrastrar el icono de la carpeta Aplicaciones a la "Papelera".

La *App Store* también permite comprobar si hay actualizaciones para OS X o para las aplicaciones instaladas y permite gestionarlas desde una de sus secciones. El usuario es informado desde el botón "Actualizaciones" en la barra de herramientas y el icono de la *App Store* en el *Dock*, mostrando la cantidad disponible. Interrumpir, reanudar o cancelar no son acciones posibles a realizar una vez iniciada la actualización. (Apple Support, 2012)

Este es uno de los *software* más novedosos de los estudiados, pero al ser privativo no se tiene acceso al código fuente, por lo que no es posible utilizarlo en la distribución cubana ni reutilizar sus funcionalidades, aunque se tendrán en cuenta aspectos significativos de su interfaz y funcionamiento.

1.3. Aporte del análisis de sistemas homólogos

Una vez realizado el estudio de la situación actual de los principales sistemas usados para la gestión de aplicaciones, se determinó no usar ningún *software* existente por las razones expuestas en la problemática

y epígrafes anteriores, decidiéndose desarrollar un nuevo Centro de *Software*. Se recomienda utilizar el Centro de *Software* de Ubuntu como base para el desarrollo, dado que Nova Escritorio 5.0, al igual que su versión anterior, estará basado en Debian y utilizará GTK 3, siendo este *software* compatible con ambos requisitos del sistema. Cabe destacar que desde el lanzamiento de Nova Escritorio 4.0 se ha estado trabajando con este *software*, por lo que los desarrolladores del centro encargados de su mantenimiento se encuentran familiarizados con su estructura y funcionamiento, por lo que resulta más sostenible reutilizar las funcionalidades de este *software* conocido que hacer un análisis de otro de los mencionados, o empezar el desarrollo desde cero. Se tuvo en cuenta además las facilidades que brinda Xapian para mantener la consistencia en la base de aplicaciones y su rapidez para realizar búsquedas en el repositorio, por lo que se propone mantener este gestor de aplicaciones en el *software* a desarrollar.

En la base de este software se definirán funcionalidades básicas que podrán ser utilizadas para el desarrollo de una herramienta gráfica, basada en la biblioteca multiplataforma *Qt*, para el sistema operativo Nova Ligero.

Para el diseño del *software* a desarrollar se tendrá en cuenta la propuesta del Deepin *Software* Center.

1.4. Metodología de desarrollo

Para el desarrollo del *software* se decidió utilizar la metodología OpenUp, ya que esta es utilizada en el proceso de desarrollo de programas de código abierto. Además, es la metodología orientada en el departamento de sistema operativo para el desarrollo de todos sus productos, ya que permite detectar errores tempranos a través de un ciclo de desarrollo iterativo y también por ser una metodología centrada en el cliente. Con el tiempo espera cubrir un amplio conjunto de necesidades en el campo del desarrollo de programas. Permite un abordaje ágil al programa en análisis con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles y tareas. Es un proceso interactivo de desarrollo de *software* simplificado, completo y extensible, para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y de los involucrados, por encima de las formalidades innecesarias. (López Castellano, 2012)

1.5. Lenguaje de programación y modelado

Se seleccionó para la implementación del Centro de *Software* de Nova 5.0 el lenguaje de programación Python por las siguientes razones:

- La manera de organizar sus instrucciones, Python permite dividir su programa en módulos reutilizables desde otros programas en este lenguaje.
- El lenguaje incorpora una gran colección de módulos estándar que se pueden utilizar como base de los programas.
- Python posee todos los módulos necesarios para el trabajo con APT, *Aptdaemon*, *PolicyKit*, GTK, entre otros.

El uso de este lenguaje también ha sido establecido como política del proyecto.

Para el modelado de los artefactos arquitectónicos e ingenieriles del sistema, fue empleado el lenguaje UML[1]. (López Castellano, 2012)

1.6. Herramienta CASE

Para el modelado del proyecto se usó **Visual Paradigm** en su versión 8.0. Esta es una aplicación que soporta el trabajo con UML y otros lenguajes de descripción de sistemas. Entre sus principales funcionalidades pueden mencionarse la fácil confección y manejo de una amplia gama de diagramas ingenieriles, que permiten la documentación de cualquier *software* orientado a objetos. Como funciones adicionales, provee la capacidad de generar reportes y posibilita la reingeniería del código. (Visual Paradigm, 2007)

1.7. Plataforma de desarrollo

Se decidió utilizar el entorno integrado de desarrollo Eclipse en su versión 3.8.1, que integrado a un conjunto de plugins[2], incluye una gran cantidad de funcionalidades utilizadas tanto para el desarrollo del Centro de *Software* como para la gestión del mismo. Los componentes adicionados fueron los siguientes:

- *PyDev*: integra las funcionalidades de Eclipse para el desarrollo con Python. Permite la utilización de las bibliotecas de este lenguaje, reconocimiento sintáctico de código, refactorización, *debugger*[3] y consola interactiva. (López Castellano, 2012)

1.8. Otras herramientas

DevHelp: es un navegador de documentación de API de código abierto para los entornos de escritorio de GTK+ y Gnome**.** Es compatible con *gtk-doc*, el cual es un marco de referencia de *Application Programming Interface*[4] (API) diseñado para GTK+ y utilizado mediante el entorno de escritorio de Gnome para la documentación de API. (Hallendal, 2011)

Glade: es una herramienta para el Desarrollo Rápido de Aplicaciones (RAD) que permite el desarrollo fácil de interfaces de usuario para GTK+ y el entorno de escritorio Gnome. Las interfaces de usuario diseñadas en Glade se guardan como XML, y usando la función *GtkBuilder* de la biblioteca GTK+ pueden ser cargadas por las aplicaciones de forma dinámica. Estos archivos XML generados con Glade pueden utilizarse en numerosos lenguajes de programación como C, C++, C#, Vala, Java, Perl, Python y otros. Glade es un *software* libre publicado bajo la Licencia GNU GPL.

---

[1] El Lenguaje Unificado de Modelado (UML), es el lenguaje estándar para la representación del desarrollo de *software* y sistemas. Para lo anterior utiliza una serie de estereotipos que ilustran los diferentes procesos, objetos y entidades de un *software*, así como sus relaciones.

[2] Los “plugins” son fragmentos de *software* que añaden funcionalidad a una aplicación determinada. Generalmente son instalados de manera adicional.

[3] Una herramienta que permite monitorizar la ejecución de una aplicación en su totalidad o un fragmento de la misma, observando los valores que toman sus variables, llamadas realizadas, objetos creados y otras acciones, con el objetivo de detectar fallas y observar su comportamiento.

[4] Término en ingles que significa Interfaz de Programación de Aplicaciones.

1.8.1. Bibliotecas utilizadas

GTK+: Es una herramienta multiplataforma para crear interfaces gráficas de usuario. Ofrece un conjunto completo de *widgets* o dispositivos, adecuados para proyectos que van desde pequeñas herramientas hasta completas suites de aplicaciones. Se ha diseñado para permitir programar en lenguajes como C, C++, C#, Ruby, Perl, PHP o Python. Utiliza la licencia LGPL del proyecto de *Software* Libre GNU. Está respaldado por una gran comunidad de desarrolladores y mantenedores de núcleo de empresas como *Ret Hat*, *Novell*, *Intel*, entre otras. Se ajusta a un gran número de características buscadas por los desarrolladores como son el aspecto nativo con una gran cantidad de temas de apoyo, el enfoque orientado a objetos, accesibilidad, la disponibilidad de documentación referente al tema. GTK+ está construido en la cima de *GLib*, lo que le permite proporcionar las construcciones del lenguaje algorítmico fundamentales comúnmente duplicados en las aplicaciones. Ha sido utilizado en dispositivos móviles como el Nokia 770/N800/N810/N900 como parte de la iniciativa de GMAE (Gnome Mobile Embedded) (Paul, 2007). (GTK + team, 2012)

***Advanced Package Tool* (APT) o Herramienta Avanzada de Empaquetado:** Es una herramienta que permite manejar automáticamente las dependencias. Es usada básicamente para 3 cosas fundamentales, actualizar paquetes instalados, buscar nuevos paquetes para instalar e instalar nuevos paquetes, utilizando como fuente uno o varios repositorios.

La actualización de paquetes sirve para mantener el sistema con todos los parches de seguridad al día, o simplemente para pasar a versiones más actualizadas de las aplicaciones favoritas. Para saber si hay actualizaciones disponibles, basta ejecutar el comando *apt-get upgrade*. APT se encarga de revisar la información de los repositorios y mostrar todos los paquetes que pueden ser actualizados.

Para buscar alguna aplicación se puede usar el comando *apt-cache search algo*, en donde algo corresponde a alguna característica a buscar. Por ejemplo *apt-cache search dvd* mostrará todos los paquetes relacionados con DVD.

Una vez que se decide la o las aplicaciones a instalar, bastará ejecutar el comando *apt-cache install algo* en donde *algo* corresponde a la aplicación deseada. Por ejemplo *apt-get install mplayer* instalará el reproductor de multimedia "*mplayer*". En caso de que haya alguna dependencia, APT se encargará de alertar e instalar todos los paquetes requeridos por la aplicación. (Equipo APT, 2008)

***Dbus*:** es un bus de mensajes, utilizado para el envío de mensajes entre aplicaciones. Utiliza un servidor centralizado que utiliza información de manera que la latencia en el flujo de la misma es muy baja, comparado con los demás mecanismos IPC[5] (Pennington, 2013). Brinda la libertad de no usar *libdbus*, una librería predeterminada para la comunicación entre aplicaciones, sino que brinda la posibilidad de poder escribir nuevas librerías, siempre y cuando estas cumplan las especificaciones requeridas, trabaja como un

[5] Inter Process Comunication por sus siglas en inglés, Intercomunicación de procesos

framework, permitiéndole al sistema operativo extenderse con un servicio de mensajería aplicado a la comunicación entre un nuevo dispositivo y las aplicaciones del sistema. En cuanto a la gestión de memoria, en caso de que se necesite compartir grandes cantidades de memoria usando las primitivas de compartición de memoria, se obtienen mejores resultados, pero se debería programar o utilizar una librería portable de memoria compartida, Dbus facilita este proceso realizándolo de manera nativa. (Pérez, 2013)

***Python-apt*:** proporciona acceso a casi todas las funciones soportadas por las librerías *apt-pkg* y *apt-inst* subyacentes. Esto significa que es posible volver a escribir los programas de *frontend*[6] como *apt-CD-ROM* en Python. Al pasar por la biblioteca, los dos primeros módulos son *apt_pkg* y *apt_inst*. Estos módulos son enlaces directos a las bibliotecas *apt-pkg* y *apt-inst* y la base para el resto de *python-apt*. El paquete APT utiliza módulos *apt_pkg* y *apt_inst* para proporcionar fáciles formas de manipular la memoria caché, en la obtención e instalación de nuevos paquetes. También ofrece clases de progreso útiles, para el texto y las interfaces GTK+. El paquete *aptsources* proporciona clases y funciones para leer archivos como */etc/apt/sources.list* y modificarlos. (Klode, 2011)

***Aptdaemon*:** es un servicio de gestión de paquetes en transacciones. Permite a usuarios normales realizar tareas de gestión de paquetes, por ejemplo: actualizar la cache, mejorar el sistema, instalar o eliminar paquetes de software. En el lenguaje Python con GTK (*PyGtk*), como se conoce, permite supervisar y controlar la transacción que esté en ejecución. Este se ejecuta de forma independiente a la interfaz *Dbus* que se ejecute en el sistema. (Heinlein, 2009)

***Gobject Introspection*:** esta biblioteca actualmente está en desarrollo y su primera versión estable fue lanzada junto con GTK 3. Esta sirve de puente para comunicar a los distintos lenguajes de programación usados en GNU/Linux, por lo que mediante esta es posible que una biblioteca hecha en C pueda ser utilizada sin mayor esfuerzo por otros lenguajes y por otras aplicaciones. *Gobject Introspection* pone los metadatos de la implementación dentro de la misma librería *Gobject*, usando anotaciones dentro de los comentarios que son interpretados luego. *Gobject Introspection* también crea una base de datos llamada *Gobject Introspection Repository (gir)* en la que se guarda un fichero con formato XML y con extensión *.gir* por cada biblioteca que puede ser usada a través de *Gobject Introspection*, con la información sobre las funciones, procedimientos y variables que se encuentran en dicha librería. Al ser compiladas estas librerías, un escáner revisa el fuente en busca de las anotaciones en los comentarios y crea un el fichero *.gir*, que es compilado y situado en el repositorio *(/usr/share/gir-1.0*), donde podrá ser usado por otras librerías. (Fuentes Rodríguez, 2012)

***GLib*:** es una biblioteca de propósito general que se usa para implementar muchas funciones no gráficas, es básica para el trabajo con GTK+. La misma proporciona los bloques necesarios para construir

[6] Es la parte del *software* que interactúa con el o los usuarios

aplicaciones y bibliotecas escritas en C. Proporciona el sistema de objetos básico usado en Gnome, la implementación del bucle principal, y un gran conjunto de funciones de utilidad para cadenas y estructuras de datos comunes. (GNOME Project, 2013)

Conclusiones parciales

A partir del estudio realizado en el capítulo, se puede concluir que las distribuciones de GNU/Linux no cuentan con un software para la gestión de aplicaciones que se ajuste a las características que se espera contenga el Centro de Software de Nova 5.0. Este estudio permitió tener una visión más amplia de las peculiaridades encontradas en varios software de este tipo, tomando elementos de interés para el software a desarrollar. Utilizar como base para el desarrollo el Centro de Software de Ubuntu constituye un adelanto para el desarrollo, ya que no es necesario implementar todas las funcionalidades de cero, dando la posibilidad de incorporarle otras nuevas que serán manejadas desde la propuesta de interfaz realizada para Nova. Como metodología de desarrollo se asumió Nova OpenUp y el lenguaje de programación Python, que son los utilizados en el proyecto de sistema operativo. Como herramienta CASE se seleccionó Visual Paradigm, ya que esta brinda soporte para el trabajo con el lenguaje de modelado unificado (UML), además de incluir otros lenguajes de descripción de sistemas. Como IDE de desarrollo se utilizará Eclipse por la posibilidad que este ofrece de incluir *plugins* y trabajar con Python.

Capítulo 2: Requisitos y Arquitectura del Centro de Software de Nova Escritorio 5.0.

Introducción

Un elemento necesario para sentar las bases de todo proyecto de desarrollo de software, es definir como se desenvuelven los procesos que se desean informatizar. Es importante determinar el flujo de trabajo a seguir para lograr cada objetivo propuesto, guiado por la metodología de desarrollo seleccionada, a fin de ajustarse a las características del proyecto y el grupo de trabajo. En el presente capítulo se realiza una descripción de los requisitos del sistema, así como sus niveles de prioridad, se modelan y describen los casos de uso de manera que se pueda entender mejor el producto que se desea obtener. Se define la arquitectura a utilizar, así como los patrones de diseño aplicados en la obtención de la propuesta de solución.

2.1. Propuesta del sistema a desarrollar

El Centro de Software de Nova 5.0 es un sistema que permitirá la gestión de aplicaciones en la Distribución Cubana de GNU/Linux Nova. Se propone con su desarrollo lograr suprimir los problemas encontrados en el actual software utilizado en la distribución cubana, incorporando novedades detectadas en sistemas homólogos, a fin de satisfacer las necesidades de la comunidad, guiado por los requerimientos definidos para el software. El sistema contará con un *banner*[7] dinámico en la portada, buscando cambiar la imagen estática de la anterior propuesta. Se incluirá una propuesta del software recomendado para la migración, a fin de apoyar este proceso que se lleva a cabo en el país. Para lograr una imagen más atractiva e intuitiva para el usuario, se reducirán los espacios desaprovechados en la vista, donde se muestran las especificaciones del software, basando su diseño en la creación de un ambiente moderno y novedoso. Con esta propuesta se facilitará el mantenimiento del software, eliminando los servicios inservibles para la distribución cubana de GNU/Linux Nova que presenta el actual Centro de *Software*. Para filtrar las búsquedas, además de contar con categorías que agrupan el software correspondiente, se incorpora un buscador que es mostrado al instante de teclear el criterio especificado por el usuario; este busca las coincidencias en todo el software y si está seleccionada una categoría solo el software correspondiente a la misma.

[7] Es un gráfico, generalmente elaborado en un formato de archivo *.GIF* o *.JPG* que se coloca como encabezado incluyendo piezas publicitarias de un producto.

2.2. Especificación de requisitos

Requisitos Funcionales

Prioridad Muy Alta

1. Instalar *software*.
2. Desinstalar *software*.
3. Listar *software*.
4. Buscar *software*.
5. Actualizar sistema.
6. Ver descripción del *software*.

Prioridad Alta

7. Listar categorías.
8. Crear y actualizar base de datos de aplicaciones.
9. Mostrar software recomendado por Nova (para la migración).
10. Mostrar vista de instalación.
11. Mostrar vista actualización del sistema.
12. Mostrar vista aplicaciones instaladas.
13. Configuración de los canales de *software*.
14. Navegación.
15. Categorías especiales (Novedades, Más votadas, Más descargadas).
16. Servicio de *ratings* e *imágenes*.

Requisitos No Funcionales

1. Usar un *banner* dinámico.
2. Utilizar Eclipse como entorno de desarrollo integrado.
3. Presentar una interfaz sencilla e intuitiva al usuario.

2.3. Modelo de casos de uso

Una vez recopilados los requisitos, quedó conformada la primera vista externa del sistema mediante el modelo de casos de uso, como se muestra en la Figura 3. Esta etapa del proceso permitió identificar los principales actores que interactúan con el sistema (Ver Tabla 1), además de agrupar los requisitos en fragmentos funcionales de la aplicación que respondan a las peticiones de un actor determinado.

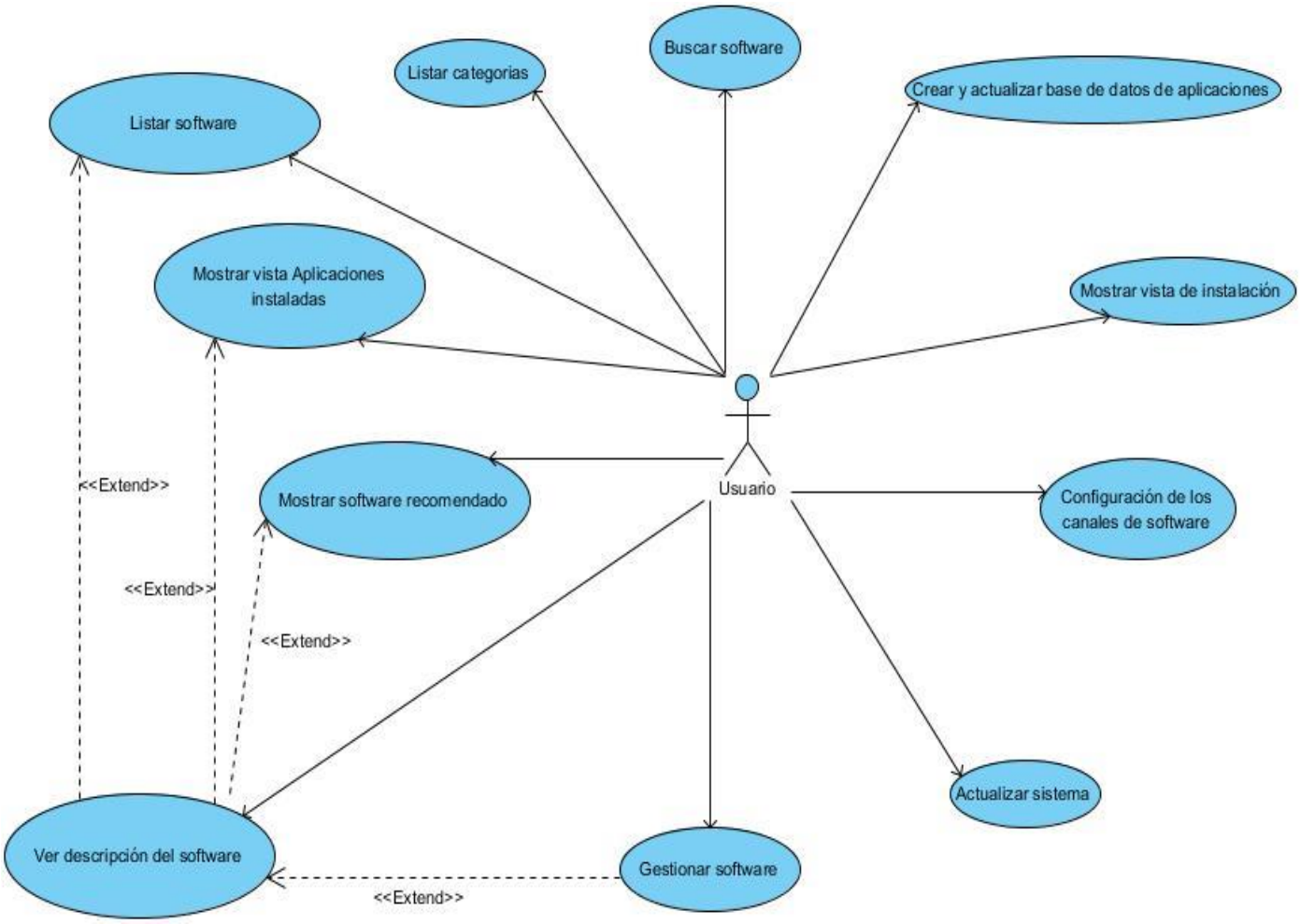


Figura 1. Diagrama de casos de uso del sistema.

Actor	Descripción
Usuario	Interactúa con el sistema para lograr la gestión de aplicaciones.

Tabla 1. Actores del sistema

2.4. Descripción de casos de uso del sistema

La descripción de los casos de uso constituye una actividad fundamental para obtener un mayor entendimiento sobre los procesos que estos representan y a la vez comprender a profundidad la interacción de cada usuario con el sistema. A continuación se describen los casos de uso que engloban las funcionalidades propuestas al alcance de esta investigación.

CUS Listar categorías

Objetivo	Listar las categorías que agrupan al *software*.
Actores	Usuario

Resumen	El sistema muestra las categorías que agrupan el *software*.
Complejidad	Alta
Prioridad	Alta
Precondiciones	
Postcondiciones	Las categorías son agregadas a las vistas de *Software* e *Instalados*.
Flujo de eventos	
Flujo básico Listar categorías	

Actor	**Sistema**
1. Ejecuta el Centro de Software	2. Obtiene las categorías definidas para agrupar el *software* y las muestra.
	3. Termina el caso de uso.

Prototipo de interfaz:

Tabla 2. CUS Listar categorías.

CUS Listar *software*

Objetivo	Listar el *software* correspondiente a la categoría seleccionada.
Actores	Usuario
Resumen	El sistema muestra el *software* correspondiente a la categoría seleccionada.
Complejidad	Alta

Prioridad	Muy Alta
Precondiciones	
Pos condiciones	Se muestra una lista con el *software* correspondiente a la categoría seleccionada.
Flujo de eventos	
Flujo básico Listar *software*	

Actor	**Sistema**
1. Ejecuta el Centro de *Software*	2. Muestra la portada y las categorías que agrupan el *software* del sistema.
3. Selecciona una categoría.	4. Muestra el *software* correspondiente a la categoría seleccionada.
5. Selecciona una aplicación.	6. Se lanza el Caso de Uso **Ver descripción del *software***.
	5. Termina el caso de uso.

Prototipo de interfaz:

Tabla 3. CUS Listar software.

CUS Mostrar *software* recomendado

Objetivo	Mostrar el *software* recomendado por Nova.
Actores	Usuario

Resumen	El sistema muestra en la portada el *software* recomendado por Nova y la categoría a la que pertenece.
Complejidad	Alta
Prioridad	Alta
Precondiciones	
Pos condiciones	Se muestra el *software* recomendado.
Flujo de eventos	
Flujo básico Mostrar *software* recomendado	
Actor	**Sistema**
1. Ejecuta el Centro de *Software*	2. Obtiene la lista de aplicaciones recomendadas y las muestra en la portada.
	3. Termina el caso de uso.

Prototipo de interfaz:

Tabla 4. CUS Mostrar software recomendado.

CUS Mostrar vista de aplicaciones instaladas

Objetivo	Mostrar en una vista el *software* instalado en el sistema.
Actores	Usuario

Resumen	El sistema muestra en una vista el *software* instalado.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Debe estar seleccionada la vista de *Instalados*.
Pos condiciones	Se muestran las aplicaciones instaladas en el sistema.

Flujo de eventos	
Flujo básico Mostrar vista de aplicaciones instaladas	
Actor	**Sistema**
1. Selecciona la vista del *software* instalado.	
2. Selecciona la categoría de "Todo el *software*". **Ver sección 1**. Selecciona otra categoría. **Ver sección 2**.	
Sección1. Categoría de "**Todo el *software***" seleccionada.	
Actor	**Sistema**
	3. Muestra todo el *software* instalado en el sistema.
	4. Termina el Caso de Uso.
Sección 2. Otra categoría seleccionada.	
Actor	**Sistema**
	3. Muestra el *software* instalado en el sistema correspondiente a la categoría seleccionada.
	4. Termina el Caso de Uso.

Prototipo de interfaz:

Tabla 5. CUS Mostrar vista de aplicaciones instaladas.

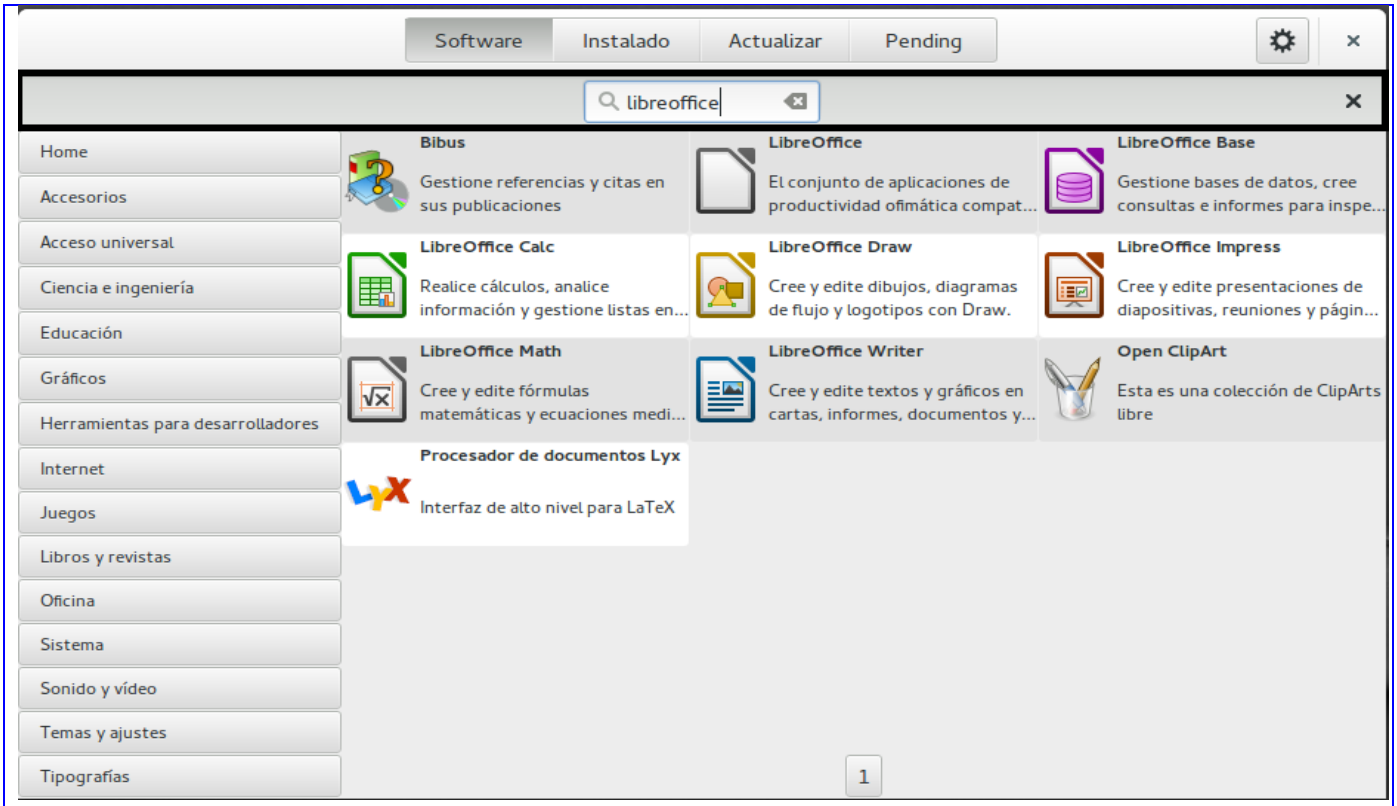
CUS Buscar *software*

Objetivo	Buscar el *software* deseado.
Actores	Usuario
Resumen	El CUS se inicia cuando el usuario comienza a introducir el criterio para filtrar la búsqueda del *software* deseado.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Debe haberse ejecutado el CUS Listar *software*. Debe estar seleccionada la vista de *Software* o Instalados.
Pos condiciones	Se muestran las coincidencias correspondientes al criterio de búsqueda introducido y la categoría seleccionada.
Flujo de eventos	
Flujo básico Buscar *software*	
Actor	**Sistema**

1. Selecciona la vista “*Software*”. **Ver sección 1**. Selecciona la vista “Instalados”. **Ver sección 2**.	
Sección 1. *Software*.	
2. Comienza a teclear introduciendo el criterio de búsqueda.	3. Muestra un área de texto contenida dentro de una barra de búsqueda con los caracteres introducidos por el usuario y 2 botones de “cerrar”, uno en el área de texto y otro en la barra de búsqueda.
3.1a Selecciona el botón “cerrar” del área de texto. Saltar al flujo alterno **Limpiar área de texto**. 3.2a Selecciona el botón cerrar de la barra de búsqueda. Saltar al flujo alterno **Cerrar barra de búsqueda**.	
	4. Verifica que no esté seleccionada alguna categoría. Si hay una categoría seleccionada: **Saltar al flujo alterno 4.1a.**
	Verifica coincidencias con el criterio de búsqueda introducido en todo el *software* y muestra el resultado.
	5. Termina el caso de uso.
Flujos alternos	
3.1a Limpiar área de texto	
Actor	**Sistema**
	4. Muestra el área de texto en blanco.
	5. Muestra la barra de búsqueda y la vista del sistema antes de iniciar la búsqueda.
	6. Termina el Caso de Uso.
Flujos alternos	
3.2a Cerrar barra de búsqueda	
Actor	**Sistema**
	4. Oculta la barra de búsqueda.

	5. Muestra la vista del sistema antes de iniciar el Caso de Uso.
	6. Termina el Caso de Uso.
Flujos alternos	
4.1a Está seleccionada una categoría	
Actor	**Sistema**
4.1a Selecciona una categoría.	5. Verifica coincidencias con el criterio de búsqueda introducido en el *software* correspondiente a la categoría seleccionada y muestra el resultado.
	6. Termina el caso de uso.
Sección 2. Instalados.	
2. Comienza a teclear introduciendo el criterio de búsqueda.	3. Muestra un área de texto contenida dentro de una barra de búsqueda con los caracteres introducidos por el usuario.
3.1b Selecciona el botón “cerrar” del área de texto. Saltar al flujo alterno **Limpiar área de texto**. 3.2b Selecciona el botón cerrar de la barra de búsqueda. Saltar al flujo alterno **Cerrar barra de búsqueda**.	
	4. Verifica que no esté seleccionada alguna categoría. Si hay una categoría seleccionada: **Saltar al flujo alterno 4.1b.**
	Verifica coincidencias con el criterio de búsqueda introducido en todo el *software* instalado y muestra el resultado.
	5. Termina el caso de uso.
Flujos alternos	
3.1b Limpiar área de texto	
Actor	**Sistema**
	4. Muestra el área de texto en blanco.

	5. Muestra la barra de búsqueda y la vista del sistema antes de iniciar la búsqueda.
	6. Termina el Caso de Uso.
Flujos alternos	
3.2b Cerrar barra de búsqueda	
Actor	**Sistema**
	4. Oculta la barra de búsqueda.
	5. Muestra la vista del sistema antes de iniciar el Caso de Uso.
	6. Termina el Caso de Uso.
Flujos alternos	
4.1b Está seleccionada una categoría	
Actor	**Sistema**
4.1b Selecciona una categoría.	5. Verifica coincidencias con el criterio de búsqueda introducido en el *software* instalado correspondiente a la categoría seleccionada y muestra el resultado.
	6. Termina el caso de uso.



Prototipo de interfaz:

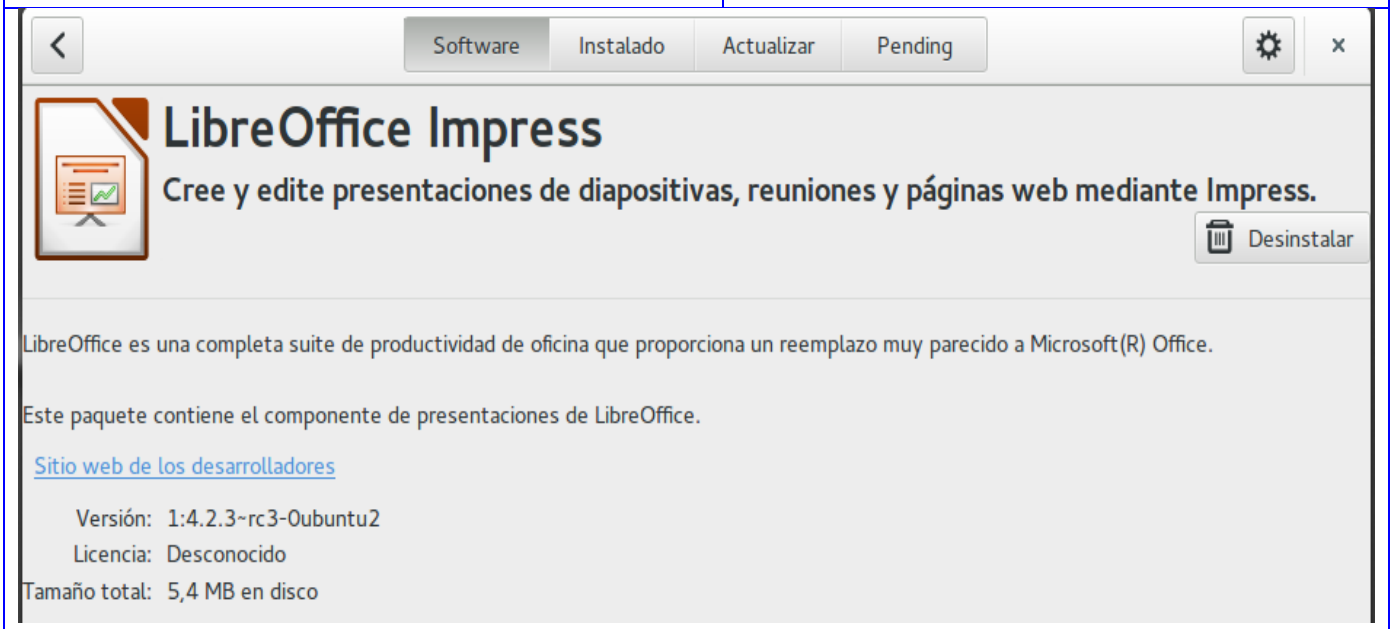
Tabla 6. CUS Buscar software.

CUS Ver descripción del *software*

Objetivo	Mostrar en una vista la descripción detallada del *software*.
Actores	Usuario
Resumen	El caso de uso inicia cuando el usuario selecciona una aplicación y el sistema muestra una vista con datos específicos que la describen.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Debe haberse ejecutado el CUS Listar *software* o el CUS Mostrar *software* recomendado.
Pos condiciones	Se muestra una vista con la descripción del *software*.
Flujo de eventos	
Flujo básico Ver descripción del software	

	Actor	**Sistema**

1. Selecciona una aplicación.	2. Obtiene los datos de la aplicación seleccionada y su estado en el sistema (instalada o no).
	3. Muestra una vista con la descripción detallada de la aplicación, un botón para volver a la vista anterior, y un botón con la acción a realizar sobre el *software* (Instalar o Eliminar).
4. Selecciona el botón (Instalar o Eliminar). **Ver sección 1 "Instalar o Eliminar"**. Selecciona el botón de volver a la vista anterior. **Ver sección 2 "Regresar a la vista anterior"**.	
Sección 1 "Instalar o Eliminar"	
7. Se lanza el **CUS Gestionar *software***.	8. Ejecuta la acción y regresa al paso 3.
Sección 2 Regresar a la vista anterior.	
9. Selecciona el botón para regresar a la vista anterior.	10. Se lanza el **CUS Listar *software*** o **CUS Mostrar *software* recomendado**.
	11. Termina el Caso de Uso.



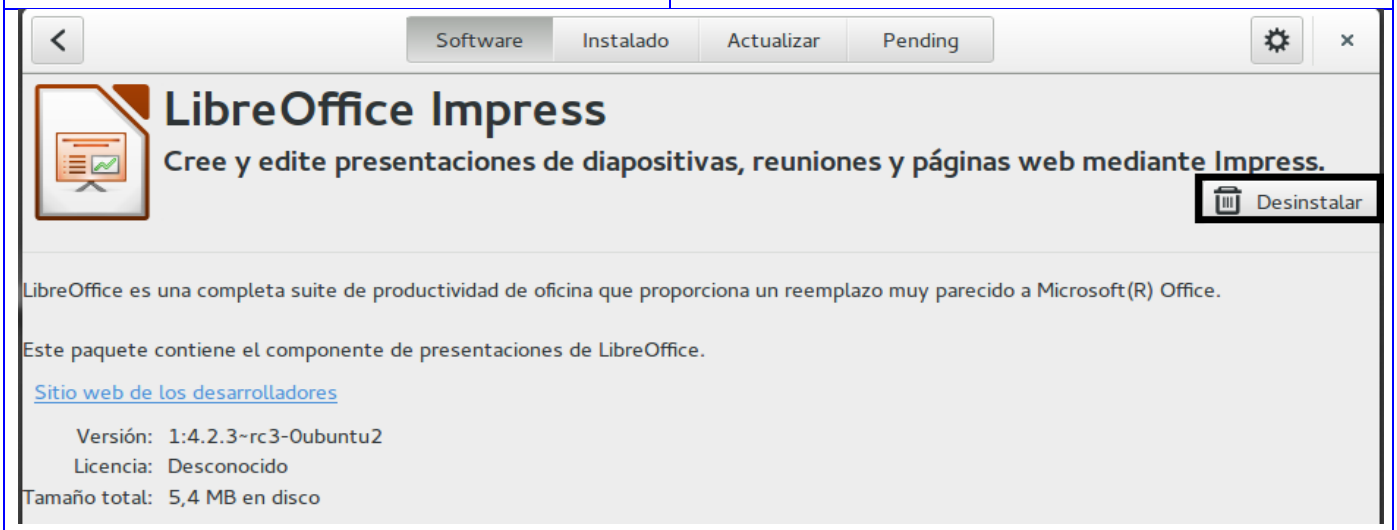
Prototipo de interfaz:

Tabla 7. CUS Ver descripción del software.

CUS Gestionar *software*

Objetivo	Permite realizar sobre el *software* las acciones de instalar y eliminar del sistema.
Actores	Usuario
Resumen	Se instala o elimina una aplicación en dependencia de su estado en el sistema.
Complejidad	Muy Alta
Prioridad	Alta
Precondiciones	Debe haberse ejecutado el CUS Ver descripción del *software*.
Pos condiciones	Se actualiza el estado del *software*.
Flujo de eventos	
Flujo básico Gestionar software	
Actor	**Sistema**
1. Instalar *software*. Ver sección 1 Desinstalar *software*. Ver sección 2.	
Sección 1: Instalar *software*	
1. Se ejecuta el **CUS Ver descripción del *software***.	2. Muestra un botón con la opción de instalar el *software* mostrado.
3. Selecciona el botón.	4. Verifica que el usuario que solicita la transacción tiene los permisos para realizarla.
	5. Obtiene del repositorio la aplicación y los paquetes necesarios para su instalación.
	6. Oculta el botón y muestra en su lugar una barra con progreso de instalación y se lanza el **CUS Mostrar vista de instalación**.
	7. Cuando finaliza la transacción muestra el botón nuevamente actualizado con la opción de desinstalar al terminar la transacción.
	8. Termina el caso de uso.
Sección 2: Desinstalar *software*	
1. Se ejecuta el **CUS Ver descripción del *software***.	2. Muestra un botón con la opción de desinstalar el *software* mostrado.

3. Selecciona el botón.	4. Verifica que el usuario que solicita la transacción tiene los permisos para realizarla.
	5. Busca la aplicación en el sistema y sus dependencias.
	9. Oculta el botón mostrando en su lugar una barra con el progreso de desinstalación y se lanza el **CUS Mostrar vista de instalación**.
	10. Cuando finaliza la transacción muestra el botón nuevamente actualizado con la opción de instalar.
	6. Termina el caso de uso.



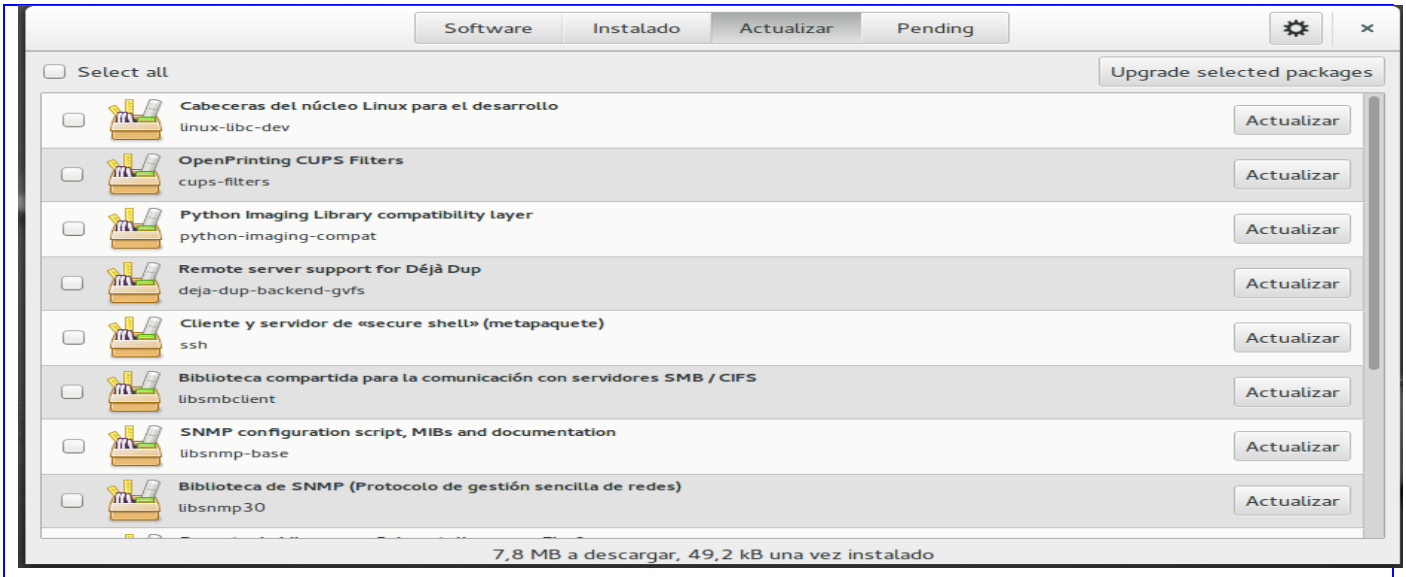
Prototipo de interfaz:

Tabla 8. CUS Gestionar software.

CUS Actualizar sistema

Objetivo	Permite actualizar el *software* y sus dependencias instaladas en el sistema.
Actores	Usuario
Resumen	El usuario puede actualizar los paquetes seleccionados o actualizar un paquete específico.
Complejidad	Muy Alta
Prioridad	Alta
Precondiciones	Deben existir actualizaciones en el repositorio de los paquetes instalados en el sistema operativo.

Pos condiciones	Se actualiza el *software* y se elimina de la lista de paquetes actualizables.
Flujo de eventos	
Flujo básico Actualizar sistema	
Actor	**Sistema**
1. Selecciona la vista *Actualización.*	2. Muestra la lista de paquetes actualizables y el espacio que ocupa en disco antes y después de instalar.
3. Selecciona el botón Actualizar. **Ver sección 1 Actualizar**. Selecciona los paquetes que desea actualizar y presiona el botón de Actualizar el *software* seleccionado. **Ver sección 2 Actualizar paquetes seleccionados.**	
Sección 1: Actualizar.	
	4. Inicia la transacción y lanza el **CUS Mostrar vista de instalación**.
	5. Cuando finaliza la transacción actualiza la vista.
	6. Termina el Caso de Uso.
Sección 2: Actualizar paquetes seleccionados.	
	4. Inicia la transacción y lanza el **CUS Mostrar vista de instalación**.
	5. Cuando finaliza la transacción actualiza la vista.
	6. Termina el Caso de Uso.
	7. Inicia la transacción y lanza el **CUS Mostrar vista de instalación**.
	8. Termina el caso de uso.



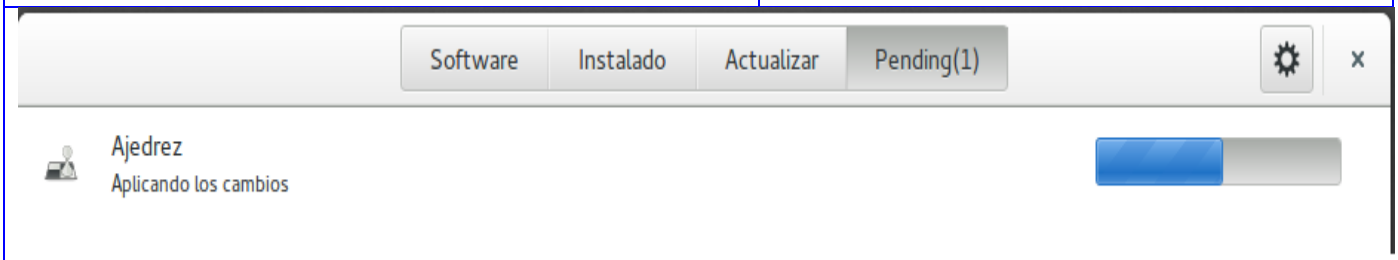
Prototipo de interfaz:

Tabla 9. CUS Actualizar Sistema.

CUS Mostrar vista de instalación

Objetivo	Mostrar el progreso de una transacción iniciada.
Actores	Usuario
Resumen	Se muestra el progreso de las transacciones iniciadas.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Deben ejecutarse los Casos de Uso Gestionar *software*, o Actualizar el sistema.
Pos condiciones	Queda vacía la lista de transacciones pendientes.
Flujo de eventos	
Flujo básico Mostrar vista de instalación	
Actor	**Sistema**
1. Se ejecutan los CUS Gestionar *software*, o Actualizar sistema.	2. Detecta que inició una transacción y muestra su progreso y la opción de “cancelar” en la vista *Pendiente*. **Ver Flujo alterno 2.1.**
	3. Cuando termina la transacción se elimina de la lista de transacciones pendientes.

	4. Termina el Caso de Uso.
Flujos alternos	
2.1 Se selecciona la opción de “cancelar”	
Actor	**Sistema**
2.1 Selecciona la opción de cancelar la transacción.	3. Detiene la transacción y la elimina de la lista de transacciones pendientes.
	4. Termina el caso de uso.



Prototipo de interfaz:

Tabla 10. Mostrar vista de Instalación.

2.5. Modelo de diseño

El modelo de diseño del sistema recoge la realización de los principales Casos de Uso reflejados en el epígrafe 2.4. Serán reflejadas en este documento sólo la realización de los casos de uso críticos que dan paso al cumplimiento de los objetivos de la investigación, prescindiendo de las funcionalidades del sistema cuyos cambios no hayan sido arquitectónicamente significativos.

Listar categorías

Para dar cumplimiento a la funcionalidad descrita en el CUS Listar categorías fue necesario la colaboración de clases de la figura 2. La clase principal que interviene en la solución del CUS es “CategoryView”, la cual hereda de “Gtk.Bin”. En el método “_init_view” se inicializa la vista donde se muestran las categorías y sus aplicaciones asociadas. Desde aquí se llama al método “append_departmets”, siendo este el encargado de adicionar las categorías ordenadas por nombre y el “*home*”, para mostrar el *banner* y el *software* recomendado por Nova. Estas son de tipo “CategoryItemButton” y están contenidas en una caja vertical.

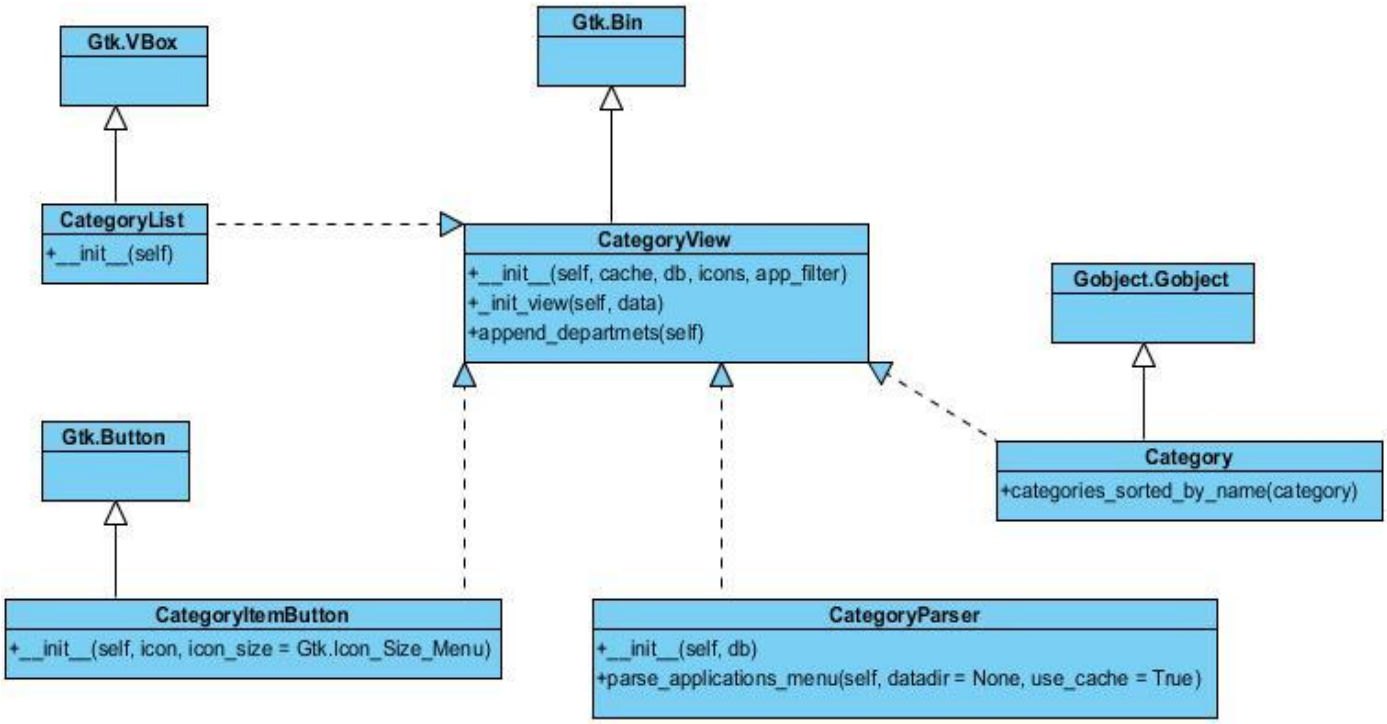


Figura 2. Diagrama de clases del CUS Listar categoría.

Listar software

Para dar solución al CUS Listar *software*, una vez que se encuentran listadas las categorías que agrupan el *software*, es necesario obtener la lista de aplicaciones correspondiente a cada categoría. Desde “CategoryView ", a través del método “update_app_list” se manda a la clase “AppEnquire” a ejecutar una consulta a la base de datos Xapian, para obtener las aplicaciones. Esta lista es actualizada cada vez que el usuario selecciona una categoría. Las especificaciones de cada aplicación se obtienen en “AppDetails”. Para construir el botón de tipo “GridButton” se toma el nombre, un resumen y el icono de la aplicación. Estos botones son adicionados a la vista una vez seleccionada la categoría y en dependencia de la cantidad de aplicaciones correspondientes se actualiza un selector de páginas.

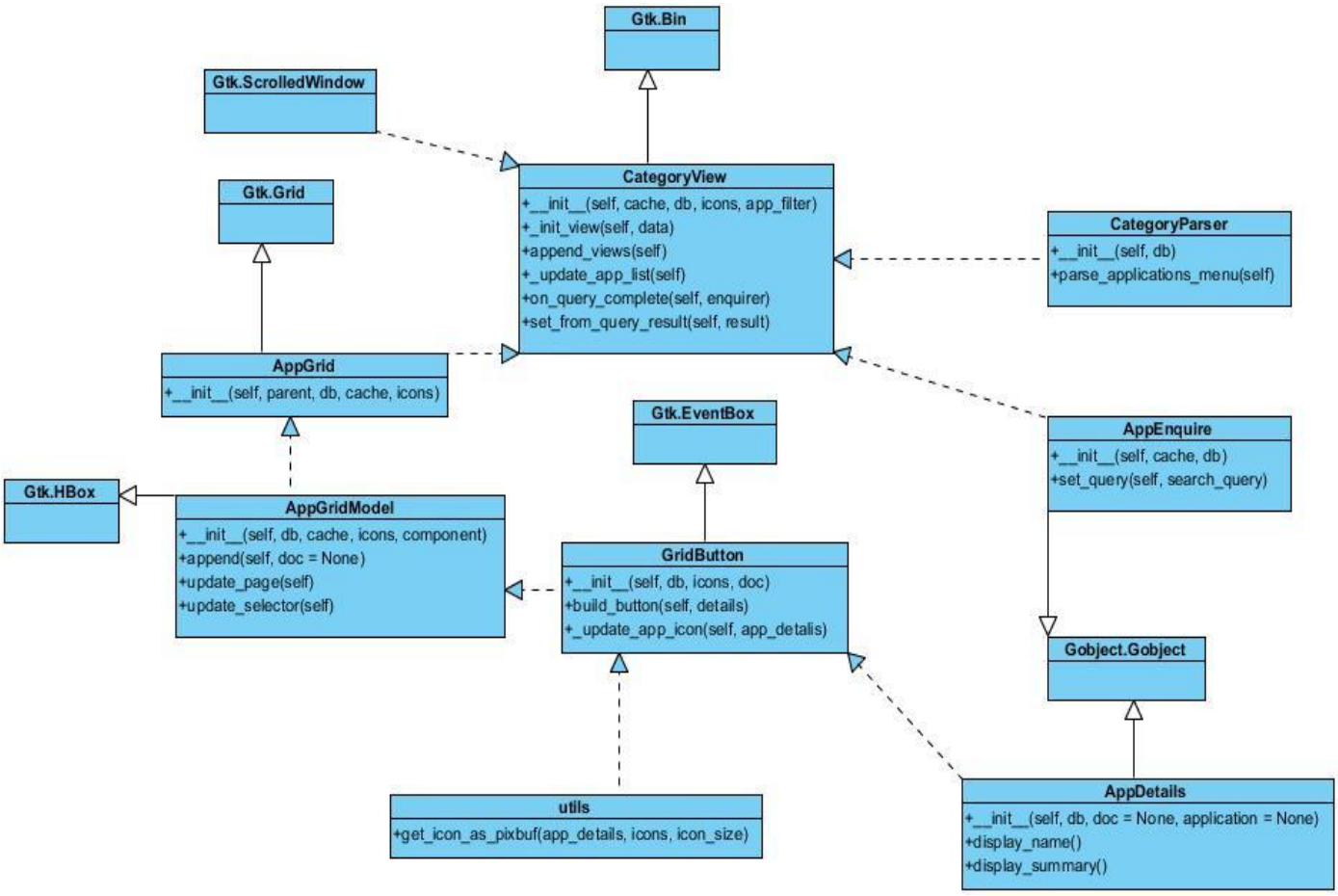


Figura 3. Diagrama de clases del CUS Listar Software.

Buscar *software*

La colaboración de clases que se muestra en la figura 4 es la encargada de satisfacer la solución del CUS Buscar *software*. Desde “NSCApp” se adiciona al “StackPane” una barra de búsqueda que se muestra al instante de escribir. El texto se almacena en un buffer una vez emitida la señal que detecta el cambio en los términos introducidos por el usuario. El sistema realiza una consulta a la base de datos con el método “get_query” y muestra el resultado en correspondencia con la vista actual, la cual es manejada por “ViewManager”.

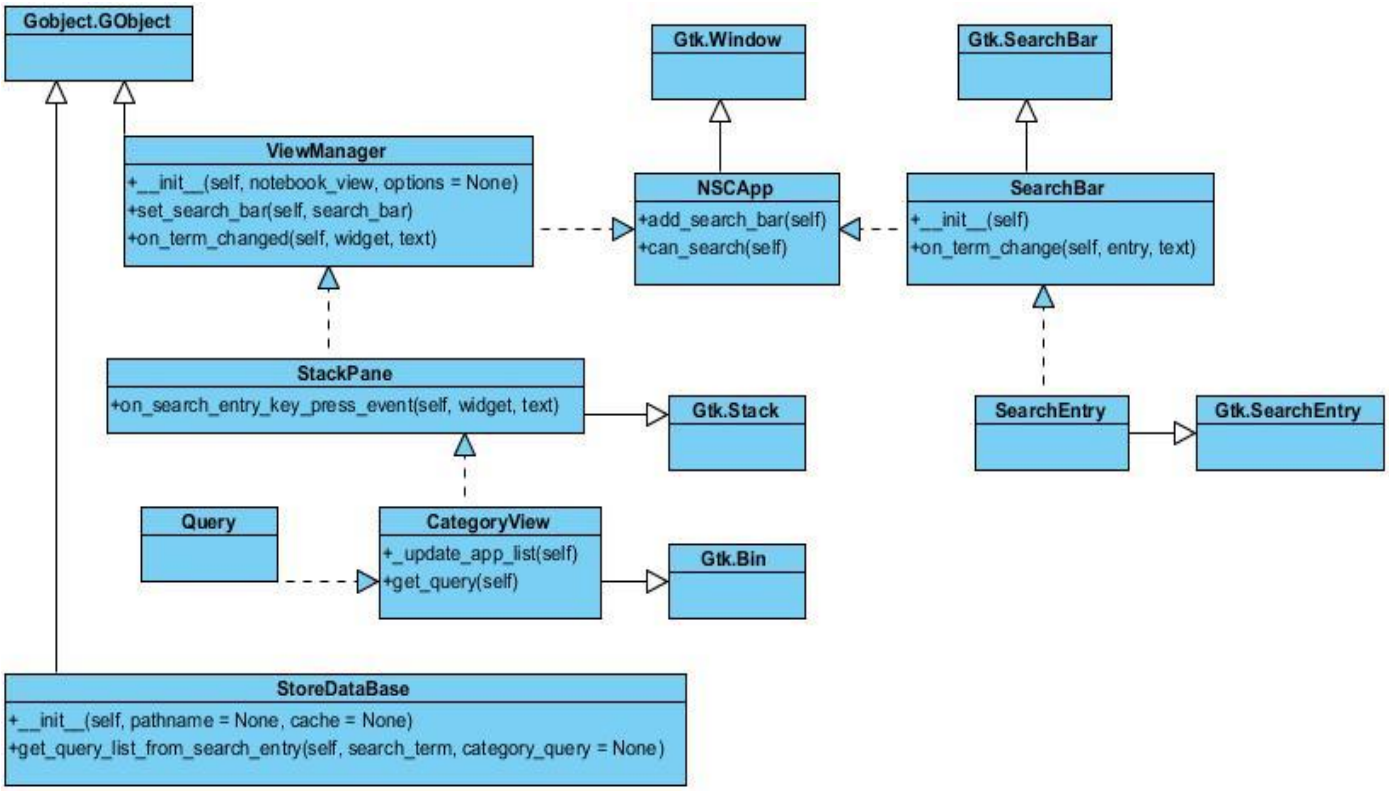


Figura 4. Diagrama de clases del CUS Buscar Software.

Mostrar vista con los detalles de la aplicación

Esta vista se muestra al seleccionar una aplicación. Desde el “StackPane” se emite la señal, pasando como parámetro la vista y el documento. La vista “AppDetailsView” se diseñó con la herramienta “Glade” y se construye utilizando un “Gtk.Build”. Ésta se actualiza con una serie de datos descriptivos de la aplicación al llamar al método “update”. Estos datos son tomados de “AppDetails”.

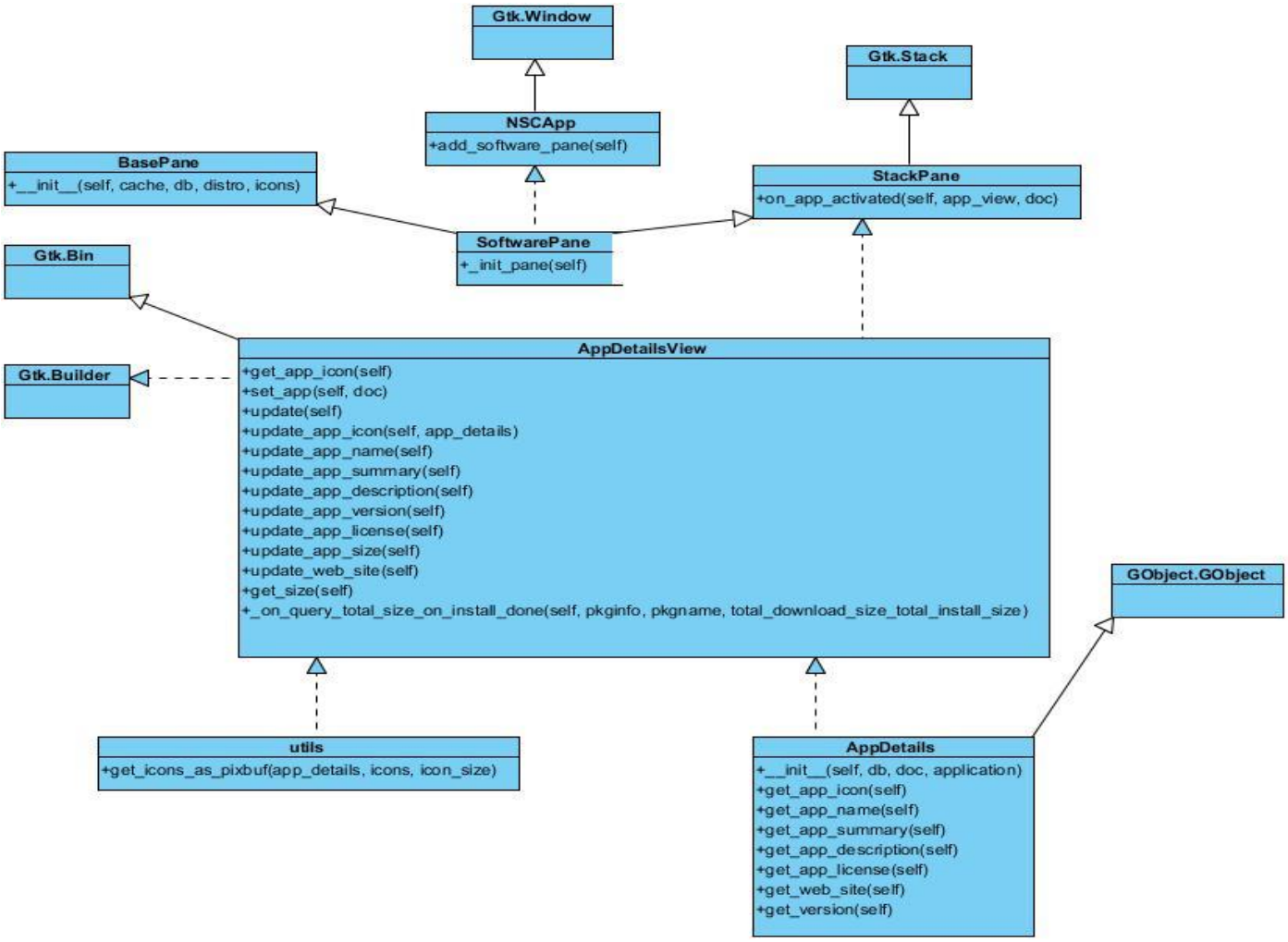


Figura 5. Diagrama de clases del CUS Mostrar vista con los detalles de la aplicación.

Mostrar vista de instalados

Para mostrar la vista con el *software* instalado en el sistema, es necesario adicionarla al StackPane desde la clase “NSCApp”. La clase “InstalledCategoryView” es la encargada de actualizar la lista con el *software* instalado, la cual se obtiene de ejecutar el método “set_installed_only” de la clase “AppFilter”. El sistema brinda la opción de mostrar el *software* instalado agrupado por categoría o todo a la vez. El método “get_query” es el encargado de consultar la base de datos y de buscar coincidencias en el cajón de búsqueda, devolviendo el resultado basándose en la vista actual.

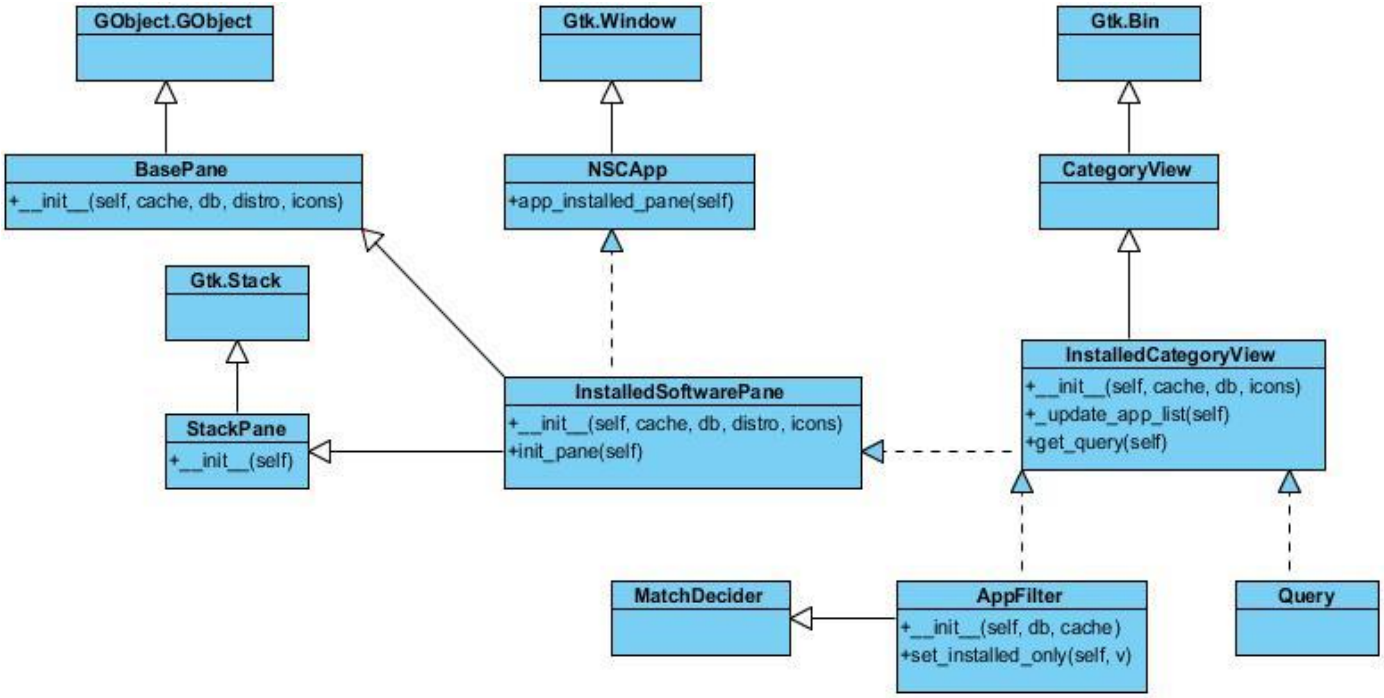


Figura 6. Diagrama de clases del CUS Mostrar vista de instalados.

Gestionar *software*

Para dar solución al CUS Gestionar *software*, es necesaria la colaboración de clases de la figura 7. Una vez que la vista con los detalles de la aplicación es actualizada con sus datos correspondientes, el sistema muestra un botón con la acción a realizar (Instalar o Remover), actualizado al llamar al método “configure” desde “update_app_action”. Al seleccionar el botón, desde la clase “AppAction” se notifica a “ApplicationManager” la acción a realizar y se verifican los permisos del usuario para realizar la acción solicitada. En el caso de Instalar, antes de iniciar la transacción el sistema verifica que la aplicación cumple con los requerimientos de hardware necesarios para su funcionamiento. En caso de que el *software* dependa de otros paquetes, se notifica al usuario y se habilitan estas dependencias. Al concluir la transacción se actualiza el botón.

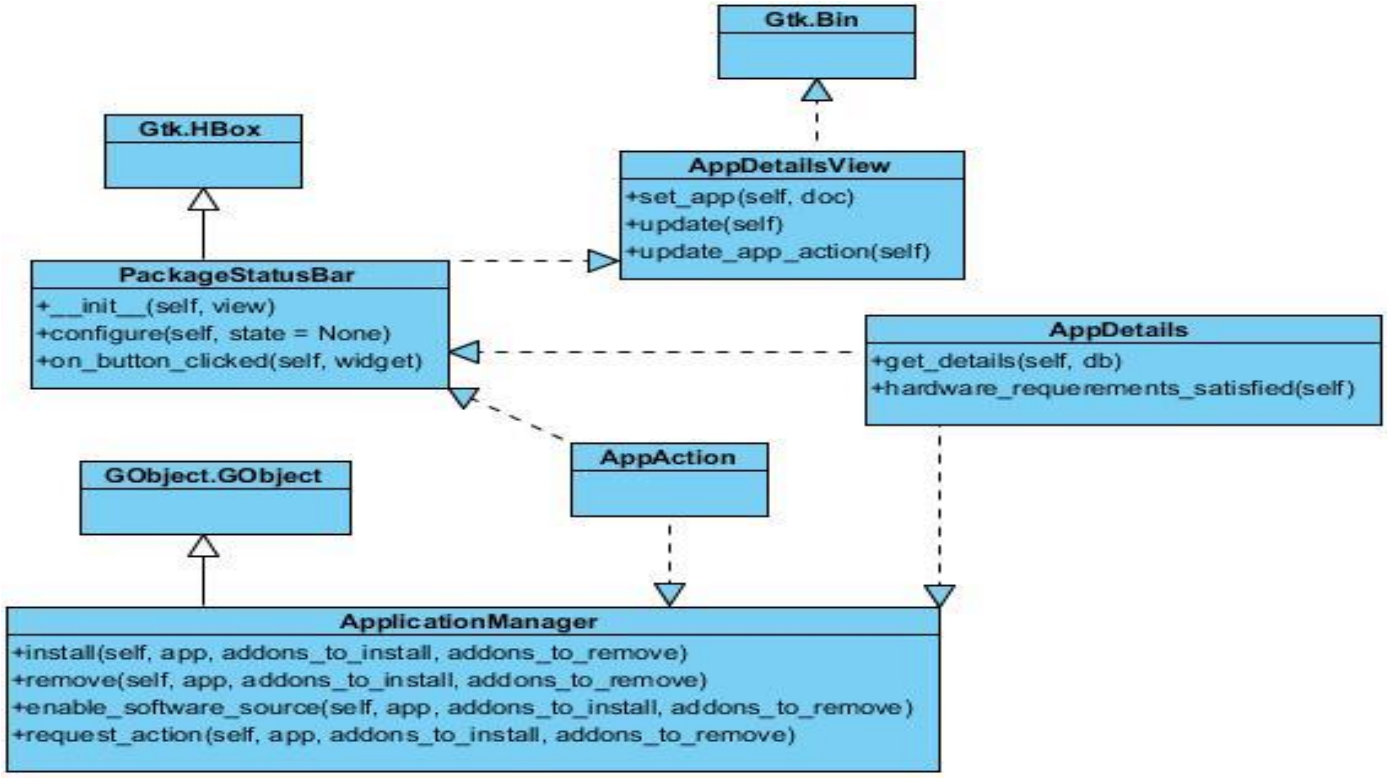


Figura 7. Diagrama de clases del CUS Gestionar software.

Actualizar sistema

Para actualizar el sistema se requiere la colaboración de clases de la figura 8. Las transacciones se realizan utilizando un servicio de *Dbus* que es manejado desde la clase “AptClient” a través del método “upgrade_package”. *Dbus* optimiza el funcionamiento del sistema, ya que permite guardar una lista con el nombre de los paquetes a actualizar e ir actualizando las transacciones pendientes una vez concluida la que esté en ejecución, sin tener que ejecutar el método “upgrade” por cada paquete actualizable. Desde la clase “AppManager” se determina si la actualización corresponde a un solo paquete, ejecutando el método “upgrade”, o en caso de que estén seleccionados varios paquetes se ejecuta el método “upgrade_multiple”. Al concluir la transacción, se actualiza la lista de paquetes en la vista *Actualización* o se muestra vacía en caso de que hayan sido actualizados todos.

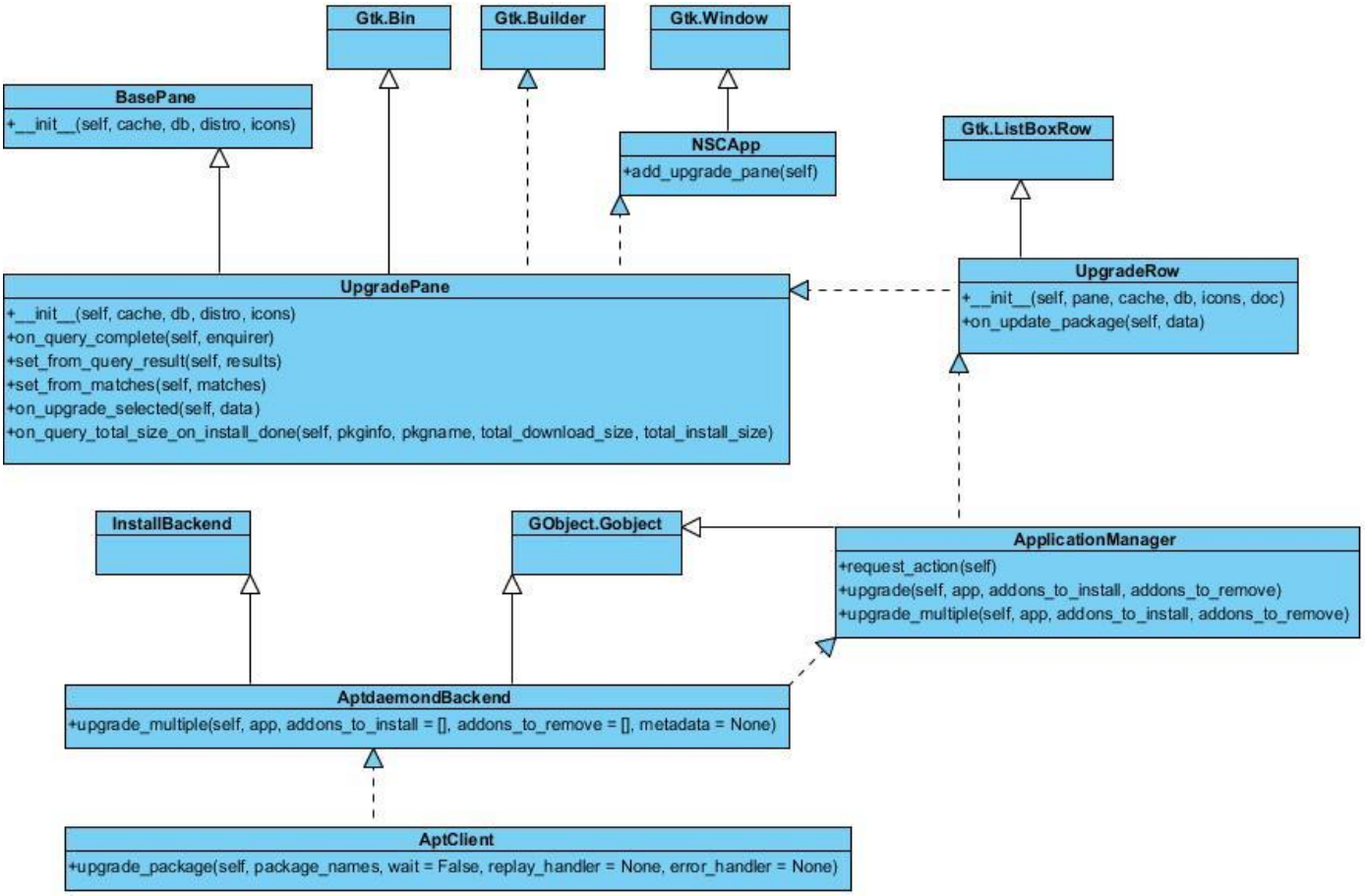


Figura 8. Diagrama de clases del CUS Actualizar sistema.

2.6. Arquitectura del software

La arquitectura de *software* se refiere a la estructuración del sistema que, idealmente, se crea en etapas tempranas del desarrollo. Esta estructuración representa un diseño de alto nivel del sistema que tiene dos propósitos primarios: satisfacer los atributos de calidad, desempeño, seguridad, modificabilidad, y servir como guía en el desarrollo. (Cervantes, 2010)

Durante la fase de Elaboración de la metodología de desarrollo, se definió utilizar la arquitectura en 3 capas.

Arquitectura en 3 capas

La arquitectura 3 capas permite estructurar un proyecto en:

- Capa de Presentación
- Capa de Negocio
- Capa de Datos

Esta separación entre capas permite llevar a cabo un desarrollo paralelo en varios niveles, facilitando el mantenimiento, soporte y la flexibilidad para dotar al sistema con nuevas funcionalidades (Ginarte González, 2011).

En la Capa de Presentación están contenidos los componentes visuales con los que el usuario interactúa. Esta se comunica con la Capa de Negocio, la cual es encargada de solicitar a la Capa de Datos las aplicaciones a mostrar en cada vista y realizar las transacciones solicitadas por el usuario. La Capa de Datos abstrae a las capas superiores para comunicarse con los repositorios, satisfacer las transacciones solicitadas, y obtener las aplicaciones con toda la información registrada en la base de aplicaciones Xapian. Se utilizó esta arquitectura, ya que es recomendada cuando existen capas de una aplicación anterior que pueden reutilizarse o integrarse a la nueva aplicación.

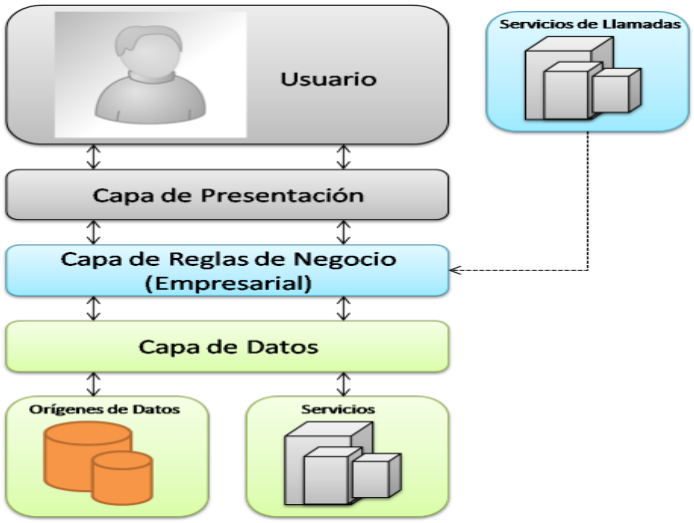


Figura 9. Estructura clásica del patrón arquitectónico 3 capas.

2.7. Estilos arquitectónicos

Un estilo arquitectónico define una familia de sistemas de *software* en términos de su organización estructural. Un estilo arquitectónico representa los componentes y las relaciones entre ellos con las restricciones de su aplicación, las asociaciones y reglas de diseño para su construcción. (Pérez y Mendoza, 2013)

2.8. Patrones de diseño

Un patrón de diseño es una descripción de un problema y su solución, que recibe un nombre y puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Dentro de estos patrones se encuentran los Patrones Generales de *Software* para Asignar Responsabilidad o GRASP, como se identifican. Estos describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (Larman, 1999)

Durante la realización del análisis y diseño del Centro de *Software* de Nova 5.0 se aplicaron estos patrones de la forma siguiente:

Patrón Experto

El patrón experto permite asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Como principales beneficios permite conservar el

encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. (Larman, 1999)

Este patrón es aplicado en el Caso de Uso Mostrar vista de aplicaciones instaladas, donde "InstalledCategoryView" es la clase experta en información para obtener las aplicaciones instaladas en el sistema, ya que esta es la que puede acceder a la clase "AppFilter", que es la encargada de filtrar las aplicaciones y determinar su estado en el sistema.

Creador

El patrón creador permite asignarle a una clase la responsabilidad de crear instancias de otra, permitiendo acceder de esta forma a los objetos de la clase instanciada desde otra clase. Al escoger un creador se da soporte al bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. (Larman, 1999)

Un ejemplo de su uso se refleja en el Caso de Uso Listar *software* donde la clase "CategoryView" crea una instancia de "CategoryParser" para acceder al método "parse_aplications_menu" y obtener las categorías del *software*.

Bajo acoplamiento

El acoplamiento es la medida con que una clase esa conectada a otras clases, por lo que una clase con bajo acoplamiento no depende de muchas otras, lo que facilita su entendimiento por separado, su facilidad de reutilización y que cambios en otros componentes no la afecten. (Larman, 1999)

Este patrón es aplicado a la clase "SearchBar", ya que esta no depende de muchas otras clases para realizar su función en la aplicación, y tomándose como una clase separada del resto no se afecta el entendimiento de cómo funciona, ya que queda bien definido que esta clase es la que construye la barra de búsqueda del sistema.

2.9 Modelo de despliegue

El Diagrama de despliegue es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue o distribución de los artefactos del *software* en los destinos de despliegue. (Sarmiento, 2013)

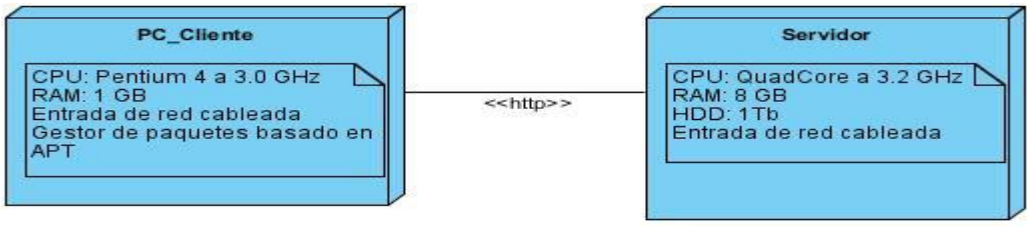


Figura 10. Modelo de despliegue.

Conclusiones parciales

En este capítulo se realizó una propuesta de solución, teniendo en cuenta los requisitos funcionales y no funcionales, los cuales quedaron plasmados en la descripción de Casos de Uso realizada como parte de la

etapa de Concepción del proyecto definida por la metodología OpenUp. Una vez definido el modelo de diseño del sistema, quedó representada la colaboración de clases que intervienen en la solución de los Casos de Uso definidos, lo que facilitó el entendimiento de la lógica del sistema. Quedó definida la arquitectura a utilizar así, como los patrones de diseño utilizados y su aplicación en la solución propuesta. Se realizó un modelo de despliegue, quedando representado cómo será la interacción del *software* con el entorno donde incide, así como el protocolo usado para la comunicación con el repositorio de aplicaciones.

Capítulo 3: Implementación y Evaluación del Centro de Software de Nova Escritorio 5.0

La implementación es la etapa del desarrollo del *software* donde se llevan a la práctica los resultados obtenidos en el diseño del sistema. Las clases y subsistemas identificados en el diseño son convertidos en ficheros y componentes que contienen el código fuente de la aplicación. En este capítulo se realizarán las pruebas al *software* desarrollado, guiado por la estrategia de prueba definida. Se realizará una observación y registro de los resultados para evaluar las respuestas negativas del sistema ante los casos de prueba diseñados, realizando una descripción de los mismos para darle seguimiento a las no conformidades encontradas.

5.1. Diagrama de componentes del sistema

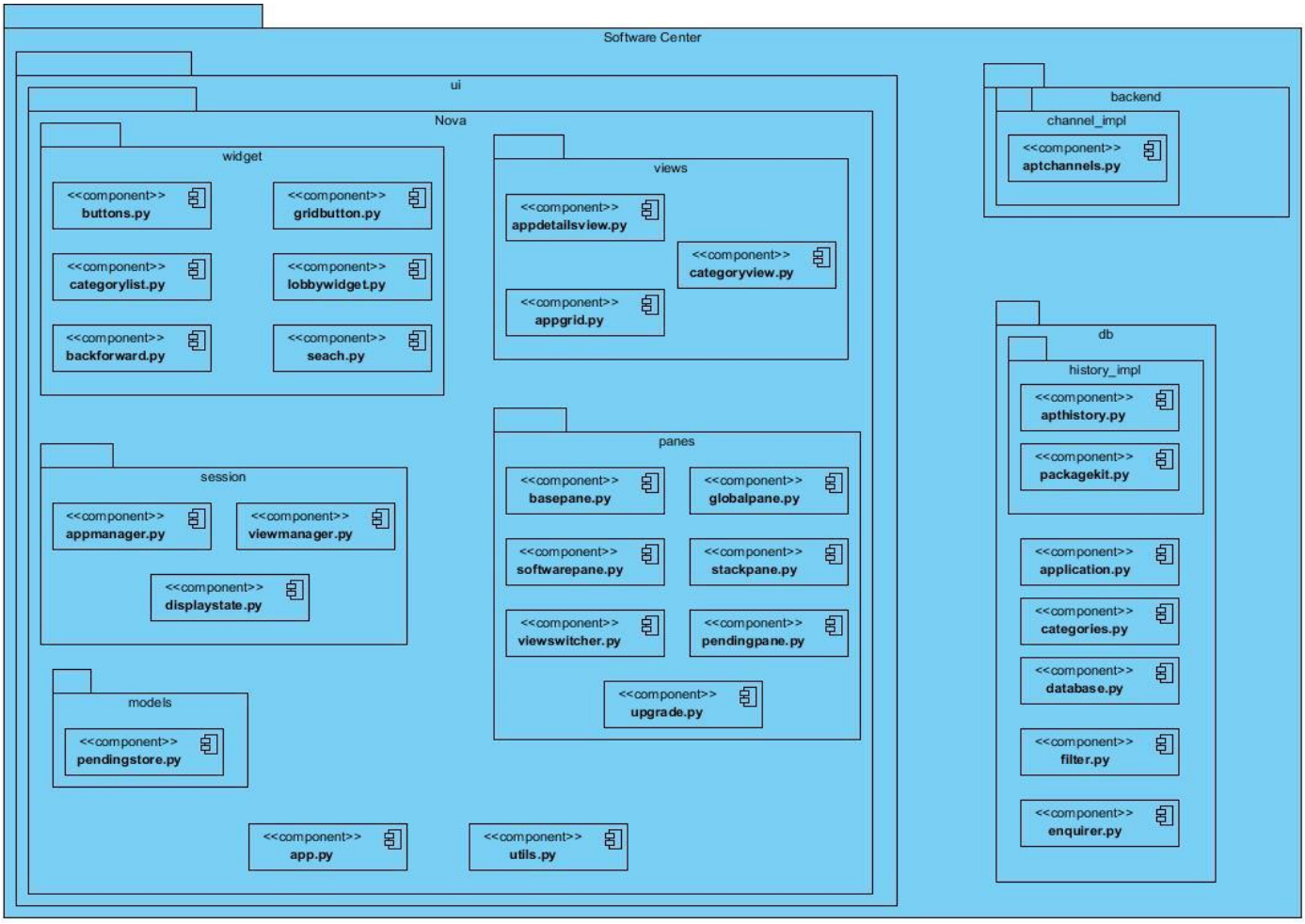


Figura 11. Diagrama de componentes del sistema

3.1.1 Componentes principales del negocio

A fin de mantener consistencia entre la arquitectura y representación del sistema, fueron situados los componentes principales de la capa de negocio en el paquete *softwarecenter*, así como se muestra en la Figura 12. De esta manera, son contenidos en el mismo los siguientes subsistemas:

***backend*:** Permite la realización de transacciones utilizando un servicio de *Dbus*. Actúa como intermediario entre los componentes visuales y la base de datos.

***db*:** Contiene toda la información referente a cada aplicación, su estado en el sistema, la categoría donde se agrupa. Permite filtrar las aplicaciones y realizar consultas a la base de datos.

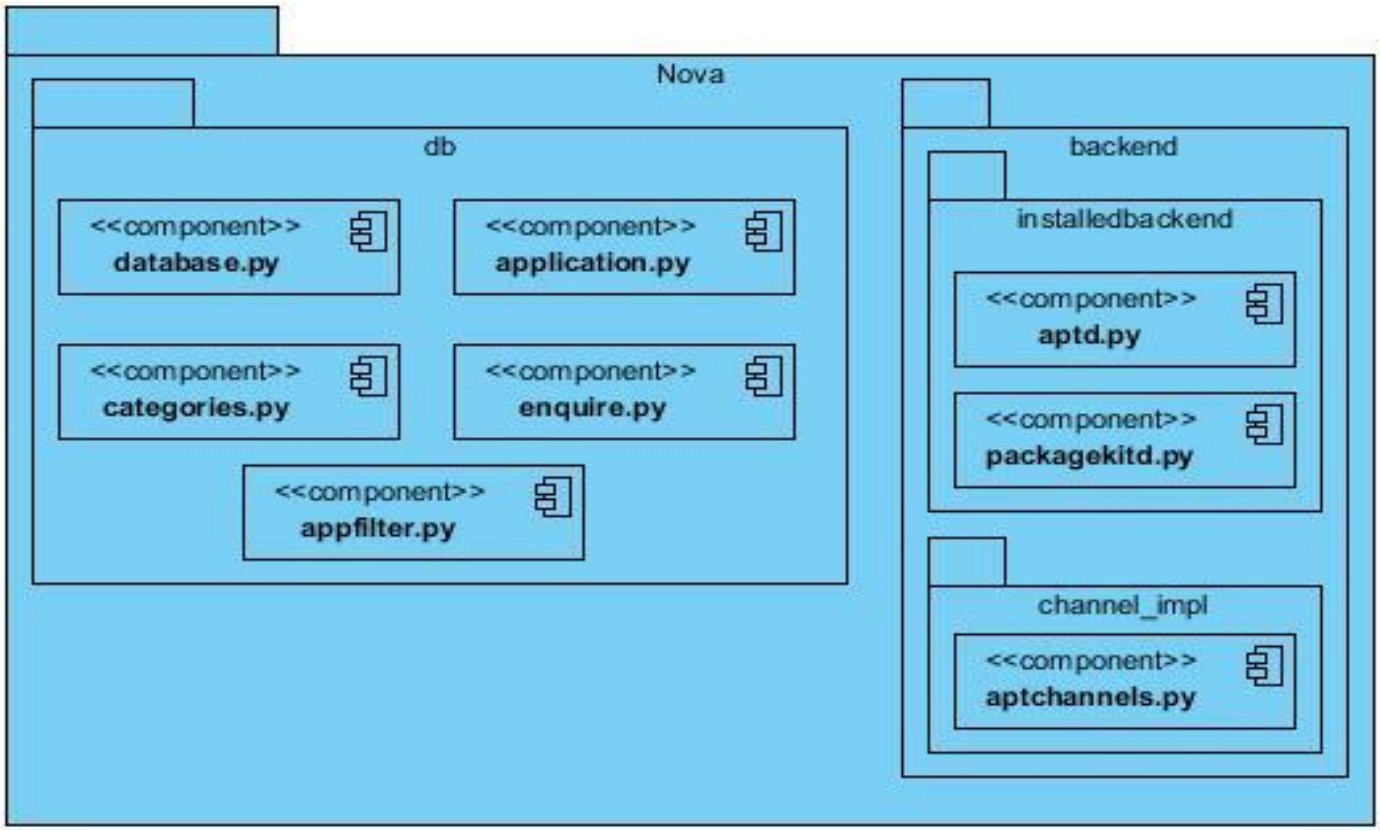


Figura 12. Diagrama de Componentes principales del negocio.

3.1.2 Componentes de interfaz

En el paquete *ui* se agruparon los componentes relacionados con la interfaz de la aplicación. Se especificó en el paquete Nova los componentes visuales relacionados con la aplicación. Se mantuvo la estructura y jerarquía de paquetes utilizada por la aplicación base, facilitando el entendimiento y la reutilización de componentes, permitiendo tomar funcionalidades útiles para la aplicación desarrollada.

***app.py*:** es el encargado de crear la aplicación y lanzar los componentes en tiempo de ejecución una vez que estos son llamados o instanciados.

***utils.py*:** constituye un componente genérico con funcionalidades que se utilizan en varias clases para optimizar el código.

***widget*:** son los dispositivos auxiliares que se utilizan para construir las vistas, los botones, la portada del sistema, las categorías para agrupar el *software*, y la barra de búsqueda.

***panes*:** estos realizan las funciones sugeridas en el nombre del componente. Se encargan de adicionar las vistas mostradas en el sistema.

***views*:** contiene los componentes que se utilizan en la creación de las vistas.

***session*:** contiene los componentes encargados de manejar las vistas y aplicaciones.

***models*:** contiene el componente encargado de manejar el estado de las transacciones apoyado en funciones nativas de GTK+.

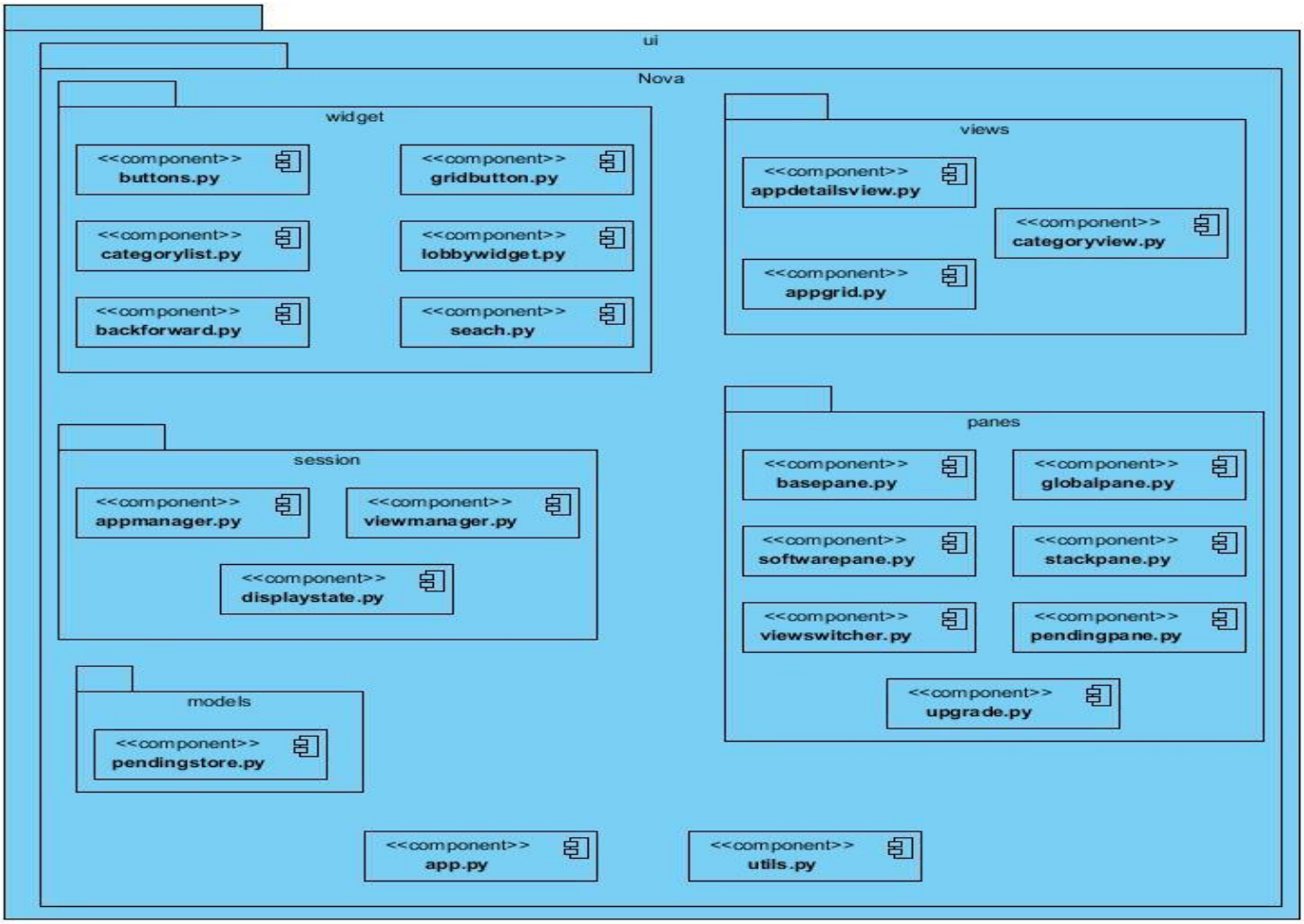


Figura 13. Diagrama de Componentes de interfaz.

5.2. Casos de pruebas

Los casos de pruebas tienen como objetivo fundamental determinar si los requisitos de la aplicación son parcial o completamente satisfactorios a través de funciones determinadas. Estos casos de pruebas proporcionan una descripción de puntos importantes de observación, con pruebas positivas y negativas

para lograr que la mayoría de los requisitos de la aplicación sean debidamente probados. (González Méndez, 2010)

5.3. Diseño de casos de prueba de caja negra

Las pruebas de caja negra permiten comprobar que las funcionalidades del sistema son operativas y se obtiene el resultado esperado después de la ejecución de cada caso de prueba. Como método de prueba se utilizó el de partición equivalente.

5.3.1. Diseño de caso de prueba Listar categorías

Escenario	Descripción	Respuesta del sistema	Flujo central
EP: 1 Listar categorías	Muestra las categorías que agrupan el *software*.	Se deben mostrar las categorías que agrupan el *software*.	➢ Se ejecuta el Centro de *Software*.

Tabla 11. Diseño de caso de prueba Listar categorías.

5.3.2. Diseño de caso de prueba Listar *software*

Escenario	Descripción	Respuesta del sistema	Flujo central
EP: 1 Mostrar portada del sistema.	Muestra la vista principal del sistema.	Se debe mostrar la portada del *software* con un *banner* dinámico, el *software* recomendado y las categorías que agrupan el *software*.	➢ Se selecciona la categoría "*Home*".
EP: 2 Listar *software* correspondiente a una categoría.	Muestra el *software* correspondiente a la categoría seleccionada.	Se deben mostrar las aplicaciones correspondientes a la categoría seleccionada.	➢ Se selecciona una categoría distinta a "*Home*".

Tabla 12. Diseño de caso de prueba Listar software.

5.3.3. Diseño de caso de prueba Mostrar vista de aplicaciones instaladas

Escenario	Descripción	Respuesta del sistema	Flujo central
EP: 1 Mostrar vista de aplicaciones instaladas.	Se muestra en una vista *Instalados* el *software* instalado en el sistema.	Se debe mostrar todo el *software* instalado en el sistema.	➢ Se selecciona la vista *Instalados.* ➢ Se selecciona la categoría *Todo el software.*

		Se debe mostrar solo el *software* correspondiente a la categoría seleccionada.	➢ Se selecciona la vista *Instalados.* ➢ Se selecciona una categoría excepto *Todo el software.*

Tabla 13. Diseño de caso de prueba Mostrar vista de aplicaciones instaladas.

5.3.4.Diseño de caso de prueba Buscar *software*

Escenario	Descripción	Respuesta del sistema	Flujo central
EP: 1 Buscar en todo el *software*.	El usuario comienza a introducir el criterio para filtrar la búsqueda del *software* deseado.	Se debe mostrar en la vista *Software* una barra de búsqueda con la cadena de texto introducida y las coincidencias encontradas en todo el *software* de acuerdo al criterio de búsqueda introducido.	➢ Se verifica que esté seleccionada la vista *Software*. ➢ Se comienza a teclear para introducir el criterio de búsqueda.
EP: 2 Buscar en el *software* correspondiente a la categoría seleccionada.	El usuario comienza a introducir el criterio para filtrar la búsqueda del *software* deseado.	Se debe mostrar en la vista *Software* una barra de búsqueda con la cadena de texto introducida y las coincidencias encontradas en el *software* de acuerdo al criterio de búsqueda introducido correspondientes a la categoría seleccionada.	➢ Se verifica que esté seleccionada la vista *Software*. ➢ Se comienza a teclear para introducir el criterio de búsqueda.
EP: 3 Buscar en todo el *software* instalado.	El usuario comienza a introducir el criterio para filtrar la búsqueda del *software* deseado.	Se debe mostrar en la vista *Instalados* una barra de búsqueda con la cadena de texto introducida y las coincidencias encontradas en el *software* instalado de acuerdo al criterio de búsqueda introducido.	➢ Se verifica que esté seleccionada la vista *Instalados*. ➢ Se comienza a teclear para introducir el criterio de búsqueda.

EP: 4 Buscar en el *software* instalado correspondiente a la categoría seleccionada.	El usuario comienza a introducir el criterio para filtrar la búsqueda del *software* deseado.	Se debe mostrar en la vista *Instalados* una barra de búsqueda con la cadena de texto introducida y las coincidencias encontradas en el *software* instalado de acuerdo al criterio de búsqueda introducido correspondiente a la categoría seleccionada.	➢ Se verifica que esté seleccionada la vista *Instalados*. ➢ Se comienza a teclear para introducir el criterio de búsqueda.
EP: 5 Limpiar área de texto.	Se muestra el área de texto vacía.	Se debe limpiar el área de texto y actualizar el *buffer* de búsqueda.	➢ Una vez mostrada la barra de búsqueda y el área de texto se selecciona el botón cerrar de área de texto.
EP: 6 Cerrar barra de búsqueda.	Se oculta la barra de búsqueda.	Se debe ocultar la barra de búsqueda en la vista donde se encuentre.	➢ Una vez mostrada la barra de búsqueda se selecciona el botón cerrar.

Tabla 14. Diseño de caso de prueba Buscar software.

5.3.5. Diseño de caso de prueba Mostrar *software* recomendado

Escenario	Descripción	Respuesta del sistema	Flujo central
EP: 1 Mostrar *software* recomendado	Debe mostrar en la página inicial las aplicaciones recomendadas por Nova.	Se debe mostrar en la página principal el icono de las aplicaciones recomendadas por Nova con su nombre y la categoría del *software* donde se encuentra.	➢ Se ejecuta el Centro de *Software*.

Tabla 15. Diseño de caso de prueba Mostrar software recomendado.

5.3.6.Diseño de caso de prueba Ver descripción del *software*

Escenario	Descripción	Respuesta del sistema	Flujo central
EP: 1 Ver descripción del *software*	Debe mostrar una vista con una descripción detallada de la aplicación.	Se deben ocultar las categorías y el *software* correspondiente a la misma, mostrando una vista con las especificaciones del *software* seleccionado y un botón con la opción de *Instalar* o *Eliminar*, en dependencia del estado de la aplicación en el sistema operativo.	➢ Se ejecuta el CUS Listar *software* o el CUS Mostrar *software* recomendado. ➢ Se selecciona una aplicación.
EP: 2 Regresar a la vista anterior	Debe regresar a la vista donde se encontraba antes de seleccionar la aplicación.	Debe ocultar la vista con las especificaciones de la aplicación y volver a la vista donde se encontraba antes de seleccionar la aplicación.	➢ Se selecciona el botón de retroceso mostrado en la parte superior izquierda.

Tabla 16. Diseño de caso de prueba Ver descripción del software.

5.3.7. Diseño de caso de prueba Gestionar software

Escenario	Descripción	Respuesta del sistema	Flujo central
EP: 1 Instalar *software*	Debe instalar el *software* mostrado al seleccionar el botón de *Instalar*.	Debe ocultar el botón mostrando una barra de progreso y agregar la transacción a la vista de *Pendiente*. Al terminar la transacción el *software* debe estar instalado en el sistema operativo y el botón debe mostrarse nuevamente con la opción de *Eliminar*.	➢ Se ejecuta el CUS Ver descripción del *software*. ➢ Se selecciona el botón de *Instalar*.
EP: 2 Eliminar *software*	Debe desinstalar el *software* mostrado al seleccionar el botón de *Eliminar*.	Debe ocultar el botón mostrando una barra de progreso y agregar la transacción a la vista de	➢ Se ejecuta el CUS Ver descripción del *software*. ➢ Se selecciona el botón de *Eliminar*.

		Pendiente. Al terminar la transacción el *software* debe estar instalado en el sistema operativo y el botón debe mostrarse nuevamente con la opción de *Instalar*.	

Tabla 17. Diseño de caso de prueba Gestionar software.

5.3.8. Diseño de caso de prueba Actualizar sistema

Escenario	Descripción	Respuesta del sistema	Flujo central
EP: 1 Actualizar *software*	Actualiza el paquete al seleccionar el botón actualizar.	➢ Deshabilita el botón y muestra el progreso en la vista de *Pendiente*. ➢ Al terminar de actualizar el paquete, recarga la vista y no muestra el paquete actualizado.	➢ Se selecciona la vista de *Actualización*. ➢ Se muestra una lista con los paquetes actualizables. ➢ Se selecciona el botón Actualizar de un paquete específico.
EP: 2 Actualizar paquetes seleccionados	Actualiza los paquetes seleccionados al hacer clic en el botón “actualizar los paquetes seleccionados”.	➢ Deshabilita el botón y muestra el progreso en la vista de *Pendiente*. ➢ Al terminar de actualizar los paquetes, recarga la vista y no muestra los paquetes actualizados.	➢ Se selecciona la vista de *Actualización*. ➢ Se muestra una lista con los paquetes actualizables. ➢ Se marcan los paquetes que se desean actualizar y se selecciona el botón “actualizar los paquetes seleccionados”.

Tabla 18. Diseño de caso de prueba Actualizar sistema.

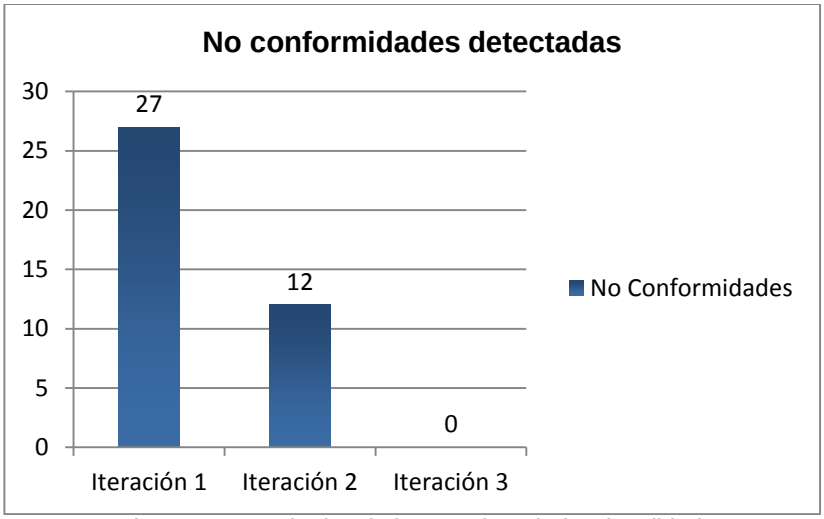


Figura 14. Resultados de las Pruebas de funcionalidad.

Resultados esperados: El proceso de pruebas de funcionalidad se lleva a cabo en tres iteraciones y las No Conformidades detectadas se relacionan principalmente, con la conectividad del *software* con el repositorio de aplicaciones, la navegación y construcción de las vistas, y con el uso de las funciones nativas de GTK+, al bloquear o impedir la ejecución del *software* en ocasiones. En la primera iteración se detectan 27 no conformidades, que se corrigen en su totalidad. En la segunda iteración se detectan 12 NC que también son corregidas y en la tercera no se detectan NC. La resolución de los errores detectados en esta etapa permitió el aseguramiento de una mayor calidad y funcionalidad en el producto, basado en el diseño y la arquitectura propuestos.

Conclusiones

Al culminar la presente investigación se dio cumplimiento a los objetivos planteados, arribándose a las siguientes conclusiones:

- El estudio realizado de las soluciones existentes arrojó que las mismas presentan incompatibilidades en cuanto a servicios o *software*, y por lo tanto, no satisfacen las necesidades de la Distribución Cubana de GNU/Linux Nova Escritorio.
- El modelado del problema mediante herramientas de análisis y diseño permitió implementar una solución que aproveche los beneficios de la arquitectura tres capas, agilizando el proceso de desarrollo y potenciando la reutilización de componentes.
- La implementación de la solución propuesta permitió obtener un *software* compatible con el sistema operativo Nova Escritorio y centralizar la gestión de paquetes, reduciendo la carga del mismo.
- Las pruebas realizadas al *software* implementado demostraron que el mismo cumple con los requerimientos definidos al inicio de la investigación y permitieron la corrección de errores encontrados en las iteraciones.

Recomendaciones

El autor del presente trabajo recomienda:

- Profundizar en el estudio del Gestor de Paquetes YUM para el desarrollo de una herramienta gráfica basada en Qt para el sistema operativo Nova Ligero, teniendo en cuenta la capacidad de abstracción de este *software* para utilizar varias herramientas gráficas.
- Desarrollar servicios para el Centro de *Software*, como por ejemplo el de *ranking*, posibilitando a los usuarios de la comunidad de Nova calificar sus aplicaciones y así determinar las más relevantes, mejorando de esta forma las aplicaciones favoritas del sistema y el *software* recomendado.

Bibliografía

ALEGSA 2009. Definicion de Paquete de software - ¿qué es Paquete de software? [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://www.alegsa.com.ar/Dic/paquete%20de%20software.php.

ALEGSA 2013. Definicion de Aplicación (informática) - ¿qué es Aplicación? [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://www.alegsa.com.ar/Dic/aplicacion.php.

ALEGSA 2014. Definicion de Canonical Ltd - ¿qué es Canonical Ltd? [en línea]. [Consulta: 1 abril 2014]. Disponible en: http://www.alegsa.com.ar/Dic/canonical%20ltd.php.

APPLE 2013. El ABC del Mac: El Finder. [en línea]. [Consulta: 26 abril 2014]. Disponible en: http://support.apple.com/kb/HT2470?viewlocale=es_ES&locale=es_ES.

APPLE 2014. Mac Basics: Mac App Store is the built in way to purchase and update software. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://support.apple.com/kb/ht4481.

APPLE SUPPORT 2012. Mac App Store: Actualizar OS X y apps. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://support.apple.com/kb/PH11502?viewlocale=es_ES.

BODNAR, L. 2013a. DistroWatch.com: Sabayon Linux. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://distrowatch.com/table.php?distribution=sabayon.

BODNAR, L. 2013b. DistroWatch.com: Sabayon Linux. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://distrowatch.com/table.php?distribution=sabayon.

CENTOS PROJECT 2014. About CentOS. [en línea]. [Consulta: 26 abril 2014]. Disponible en: http://www.centos.org/about/.

CERVANTES, H. 2010. Arquitectura de Software | SG. [en línea]. [Consulta: 15 mayo 2014]. Disponible en: http://sg.com.mx/revista/27/arquitectura-software.

COMPUTACIÓN APLICADA 2010. Historia de Google. [en línea]. [Consulta: 26 abril 2014]. Disponible en: http://www.cad.com.mx/historia_de_google.htm.

CRIADO, S.D. y GAVILÁN, E. 2009. Curso de Introducción a GNU/Linux. [en línea]. [Consulta: 26 abril 2014]. Disponible en: http://www.ant.org.ar/cursos/curso_intro/book1.html.

DEPIZZOL, V. 2013. APTonCD. [en línea]. [Consulta: 5 junio 2014]. Disponible en: http://aptoncd.sourceforge.net/doc-manual.html.

ENCICLOPEDIA UNIVERSAL 2012. Qt (biblioteca). [en línea]. [Consulta: 5 junio 2014]. Disponible en: http://enciclopedia_universal.esacademic.com/1378/Qt_%28biblioteca%29.

EQUIPO APT 2008. Ubuntu Manpage: apt-get - utilidad de manejo de paquetes APT -- interfaz en línea de. [en línea]. [Consulta: 4 abril 2014]. Disponible en: http://manpages.ubuntu.com/manpages/hardy/es/man8/apt-get.8.html.

FUENTES RODRÍGUEZ, J.M. 2012. *Entorno de Escritorio de Nova 4.0*. La Habana: UCI.

GINARTE GONZÁLEZ, Y. 2011. *Desarrollo de una herramienta informática para buscar y recuperar documentos en el repositorio documental del Departamento Señales Digitales.* La Habana: s.n.

GNOME PROJECT 2013. GLib Reference Manual - Centro del desarrollador de GNOME. [en línea]. [Consulta: 16 mayo 2014]. Disponible en: https://developer.gnome.org/glib/.

GÓMEZ SAVINO, G.J. 2010a. 2.3. YUM. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://docs.fedoraproject.org/es-ES/Fedora/14/html/Software_Management_Guide/ch02s03.html.

GÓMEZ SAVINO, G.J. 2010b. Capítulo 3. Las interfases GUI de gnome-packagekit. *Documentación de Fedora* [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://docs.fedoraproject.org/es-ES/Fedora/14/html/Software_Management_Guide/Las_interfases_GUI_de_gnome-packagekit.html.

GONZÁLEZ MÉNDEZ, B.G.M. 2010. *Sistema de Gestión de Datos Geológicos. Módulo: Inventario de Minerales Sólidos Rol Diseñador de Casos de Prueba.* [en línea]. La Habana: UCI. Disponible en: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_03577_10/1/TD_03577_10.pdf.

GTK + TEAM 2012. GTK+ Features. [en línea]. [Consulta: 1 abril 2014]. Disponible en: http://www.gtk.org/features.php.

HALLENDAL, M. 2011. Devhelp for Linux Free Download. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://linux.softpedia.com/get/Utilities/Devhelp-3875.shtml.

HEINLEIN, S. 2009. Aptdaemon in Launchpad. [en línea]. [Consulta: 16 mayo 2014]. Disponible en: https://launchpad.net/aptdaemon.

HUGHES, R. 2007. PackageKit - What is PackageKit? [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://www.packagekit.org/pk-intro.html.

HUGHES, R. 2012. Fuentes de software. *Gnome* [en línea]. [Consulta: 12 febrero 2014]. Disponible en: https://help.gnome.org/users/gnome-packagekit/stable/software-sources.html.es.

INTEF 2010. El gestor de paquetes Synaptic. *Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado* [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://www.ite.educacion.es/formacion/materiales/43/cd/modulo_6/el_gestor_de_paquetes_synaptic.html.

IXNAY 2013. Sabayon | Home. [en línea]. [Consulta: 26 abril 2014]. Disponible en: http://www.sabayon.org/.

KLODE, J. 2011. Python APT Library — python-apt v0.8.0~exp4 documentation. [en línea]. [Consulta: 16 mayo 2014]. Disponible en: http://apt.alioth.debian.org/python-apt-doc/library/index.html.

LAFITA MOSQUEDA, A. 2014. distribución Nova de GNU/Linux » Revista Tino. [en línea]. [Consulta: 1 abril 2014]. Disponible en: http://revista.jovenclub.cu/?tag=distribucion-nova-de-gnulinux.

LARMAN, C. 1999. *UML y patrones. Introduccion al analisis y deseño orientado a objeto*. S.l.: s.n. ISBN 970-17-0261-1.

LINUX DEEPIN 2011. Linux Deepin - Features. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://www.linuxdeepin.com/feature.

LINUX PROJECT 2005. Daemon Definition. [en línea]. [Consulta: 26 abril 2014]. Disponible en: http://www.linfo.org/daemon.html.

LLIUREX 2011. Gestor de paquetes Synaptic. *Iniciación a LliureX* [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://cefire.edu.gva.es/file.php/1/LLiurex_pera_la_tasca_docent/Unidad_5/gestor_de_paquetes_synaptic.html.

LÓPEZ CASTELLANO, J.G. 2012. *Serere 3.0. Instalador del sistema operativo Nova GNU/Linux* [en línea]. Ciudad de la Habana: UCI. Disponible en: http://bibliodoc.uci.cu/RDigitales/2012/noviembre/3/TD_05104_12.pdf.

LÓPEZ TORRIJOS, F.J. 2012. Gestionar Software - Paquetes | www.lopeztorrijos.com. [en línea]. [Consulta: 5 junio 2014]. Disponible en: http://www.lopeztorrijos.com/tutoriales/linux/gestionar-software-paquetes.

MASTEMAGAZINE 2013. Definición de Aplicación - Significado y definición de Aplicación. *Mastemagazine* [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://www.mastermagazine.info/termino/3874.php.

MATTHEW, P.T. 2005. SoftwareCenter - Ubuntu Wiki. [en línea]. [Consulta: 5 junio 2014]. Disponible en: https://wiki.ubuntu.com/SoftwareCenter.

MONTEAGUDO PEÑA, J.L. 2004. Educación Médica - Tecnologías de la Información y Comunicaciones. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://scielo.isciii.es/scielo.php?pid=S1575-18132004000200004&script=sci_arttext&tlng=en.

PAUL, R. 2007. GNOME Foundation announces embedded initiative | Ars Technica. [en línea]. [Consulta: 1 abril 2014]. Disponible en: http://arstechnica.com/uncategorized/2007/04/gnome-foundation-announces-mobile-and-embedded-initiative/.

PENNINGTON, H. 2013. D-Bus Specification. [en línea]. [Consulta: 16 mayo 2014]. Disponible en: http://dbus.freedesktop.org/doc/dbus-specification.html.

PÉREZ, D. 2013. [humanCode] Comunicando aplicaciones con D-Bus (Parte I) | humanOS. [en línea]. [Consulta: 16 mayo 2014]. Disponible en: http://humanos.uci.cu/2013/11/humancode-comunicando-aplicaciones-con-d-bus-parte-i/.

PÉREZ DÍAZ, A.J. 2011. Definición GNOME Enciclopedia Proyecto AjpdSoft. [en línea]. [Consulta: 26 abril 2014]. Disponible en: http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=843.

PÉREZ, M.A. y MENDOZA, L.E., 2013. *SISTEMAS DE INFORMACIÓN II TEORÍA* [en línea]. 2013. S.l.: s.n. Disponible en: http://prof.usb.ve/lmendoza/Documentos/PS-6116/Teor%EDa%20PS6116%20Arq.%20de%20Software.pdf.

PETERSEN, R. 2013. PolicyKit | Ubuntu Tutorial. *surfingturtlepress* [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://ubuntu.paslah.com/policykit/.

PROYECTO FEDORA 2012a. Features/DNF - FedoraProject. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: https://fedoraproject.org/wiki/Features/DNF.

PROYECTO FEDORA 2012b. yumex. *Fedora HOSTED* [en línea]. [Consulta: 12 febrero 2014]. Disponible en: https://fedorahosted.org/yumex/.

SABAYON 2014. En:Entropy - Sabayon Wiki. [en línea]. [Consulta: 1 abril 2014]. Disponible en: https://wiki.sabayon.org/index.php?title=Entropy.

UNIXMEN 2008. Linux Deepin Software Center Will become OpenSource's own App Store by End of Year | Unixmen. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://www.unixmen.com/linux-deepin-software-center-will-become-opensources-own-app-store-by-end-of-year/.

UTLAI 2014. Los paquetes y su gestión: DEB, RPM y tar.gz. *Usuarios de Tiflotecnología para el Libre Acceso a la Información.* [en línea]. [Consulta: 1 abril 2014]. Disponible en: http://www.nodo50.org/utlai/joomla/index.php/softlibre/41-apuntes-sobre-linux/128-7-los-paquetes-y-su-gestion-deb-rpm-y-targz.html.

VISUAL PARADIGM 2007. Visual Paradigm para UML - Programación - herramienta para desarrollo de aplicaciones utilizando modelado UML* ideal para los que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad | www.targetware.com.ar. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://www.software.com.ar/visual-paradigm-para-uml.html.

XAPIAN 2014. The Xapian Project. [en línea]. [Consulta: 12 febrero 2014]. Disponible en: http://xapian.org/.

Glosario de términos

Canonical LTD: es una compañía fundada por el empresario sudafricano Mark Shuttleworth, para la promoción de proyectos de *Software* Libre y Código Abierto. Uno de sus principales productos, es la familia de distribuciones Ubuntu.

Gentoo: es una distribución de Linux versátil, rápida y completamente gratuita orientada a desarrolladores y profesionales de redes.

Portage: es un sistema gestor de paquetes constituido por un sistema de puertos, basado en Python con un número de características avanzadas incluyendo una minuciosa gestión de paquetes y de dependencias.

Sabayón: es una distribución basada en Gentoo que tiene como objetivo ofrecer al usuario una gran cantidad de aplicaciones que están listos para su uso y un sistema operativo auto-configurado. Detecta bien el *hardware* donde se instala e incluye un gran número de paquetes de *software* en su instalación por defecto, dejando a disposición del usuario el *software* adicional en un repositorio. Sabayón es compatible con varios entornos de escritorio, *KDE, GNOME, LXDE, Xfce*. (Ixnay, 2013) (Bodnar, 2013b)

CentOS Linux: la distribución CentOS Linux es una plataforma estable, predecible, manejable y reproducible derivado de las fuentes de *Red Hat Enterprise Linux* (RHEL). Es desarrollada por un pequeño grupo de desarrolladores, apoyados por una comunidad de usuarios activa, incluyendo los administradores de sistemas, administradores de redes, gerentes, colaboradores núcleo Linux, y los entusiastas de Linux de todo el mundo. (CentOS Project, 2014)

Finder: el finder permite acceder de forma visual a prácticamente todos los dispositivos y accesorios en la Mac, incluyendo aplicaciones, discos duros, archivos, carpetas y DVD. Puede ser usado para organizar todos los archivos y carpetas, buscar material en todo el equipo, eliminarlo y otras opciones. (Apple, 2013)

Google: Google es una empresa cuyo principal producto es el motor de búsqueda del mismo nombre. (Computación Aplicada, 2010)

Demonio: es un tipo de programa en sistemas operativos tipo Unix que se ejecuta discretamente en segundo plano, en lugar de bajo el control directo de un usuario, a la espera de ser activada por la ocurrencia de un evento o condición específica. (Linux Project, 2005)

GNU Network Object Model Environment (GNOME): es un entorno de escritorio e infraestructura de desarrollo para sistemas operativos Unix y derivados Unix como GNU/Linux, BSD o Solaris; compuesto enteramente de *software* libre. Este permite a los usuarios usar y configurar sus ordenadores de una forma sencilla. (Pérez Díaz, 2011) (Criado y Gavilán, 2009)

Gdebi: es una aplicación gratuita y de código abierto para GNU/Linux que nos permite instalar archivos DEB en nuestro sistema de forma simple. El programa destaca por su facilidad de uso y por la cantidad de

información que da en su sencillo interfaz. Es capaz de resolver las dependencias que necesite un paquete DEB, tanto si contamos con conexión a Internet como si no la tenemos (en este caso, las dependencias deberán estar presentes en el sistema). Gdebi se convierte así en uno de los instaladores de paquetes DEB más sencillos y, a la vez, efectivos que existen actualmente.

Framework: estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, que puede servir de base para la organización y desarrollo de *software*.

Qt: es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario. Fue creada por la compañía noruega Trolltech. Qt es utilizada en KDE, un entorno gráfico para sistemas principalmente Linux. Utiliza el lenguaje de programación C++, pero permite usar también C, Python y Perl. Además cuenta con un soporte (sin interfaz gráfico) para acceder a bases de datos mediante SQL, así como uso de XML y una excelente API para el manejo de ficheros. (Enciclopedia Universal, 2012)

Anexo 1.

Figura 15. Desaprovechamiento de espacios.

Anexo 2.

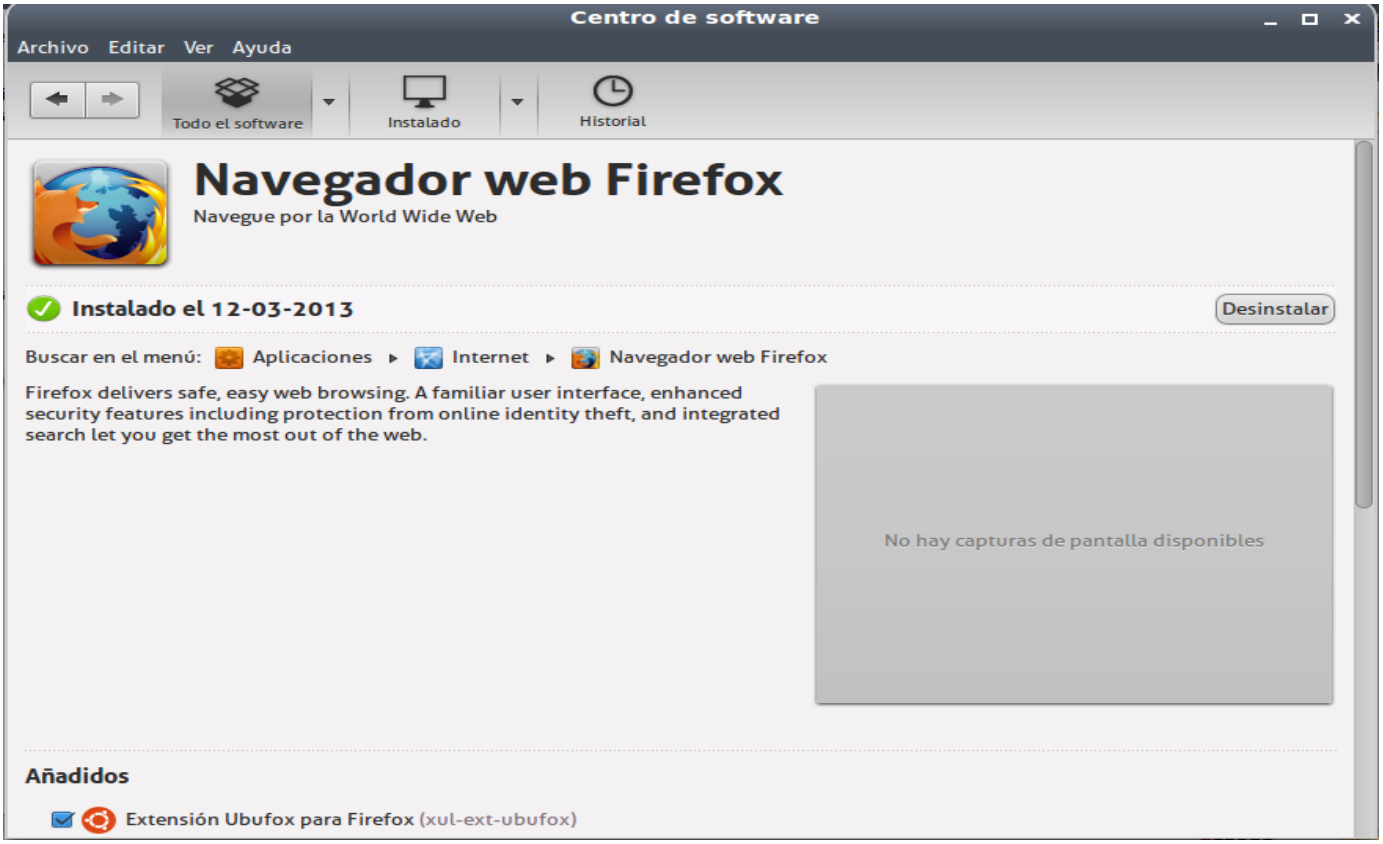


Figura 16. Secciones deshabilitadas.

Anexo 3.

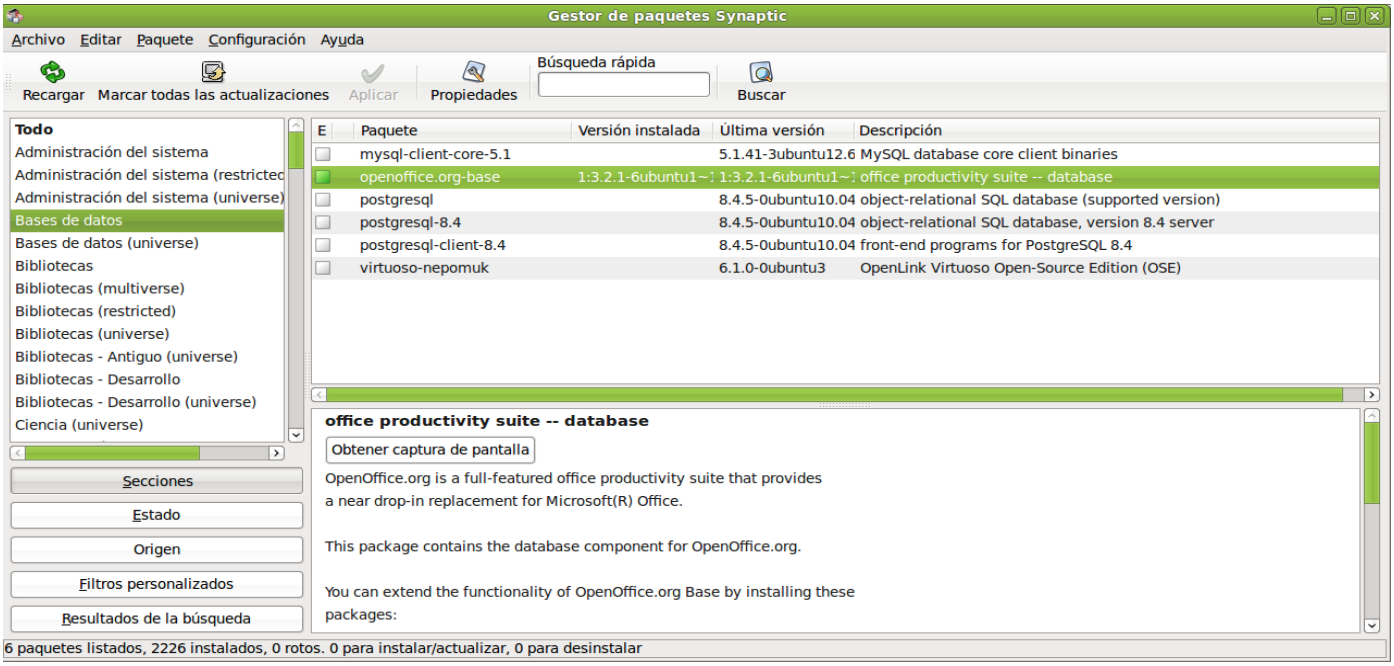


Figura 17. Gestor de paquetes Synaptic.

Anexo 4.

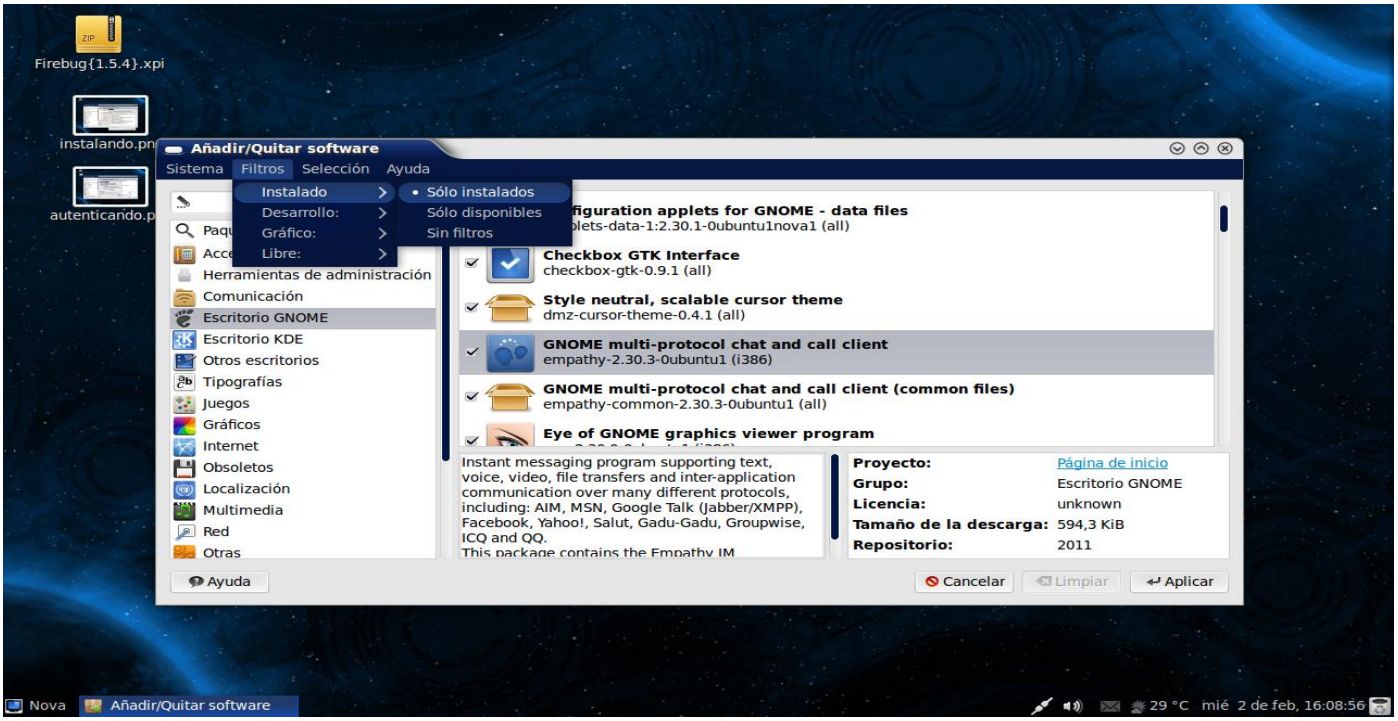


Figura 18. Gestor de paquetes Gnome PackageKit.

Anexo 5.

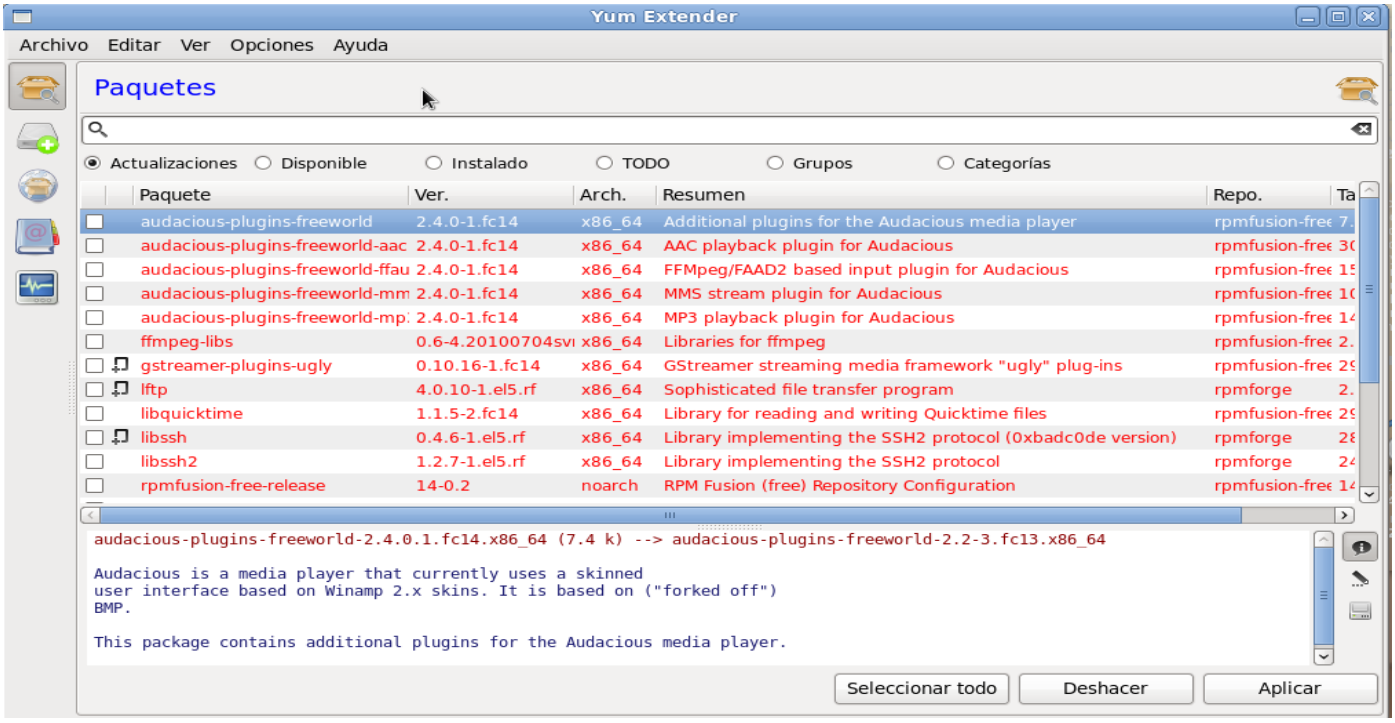


Figura 19. Gestor de paquetes YUM.

Anexo 6.

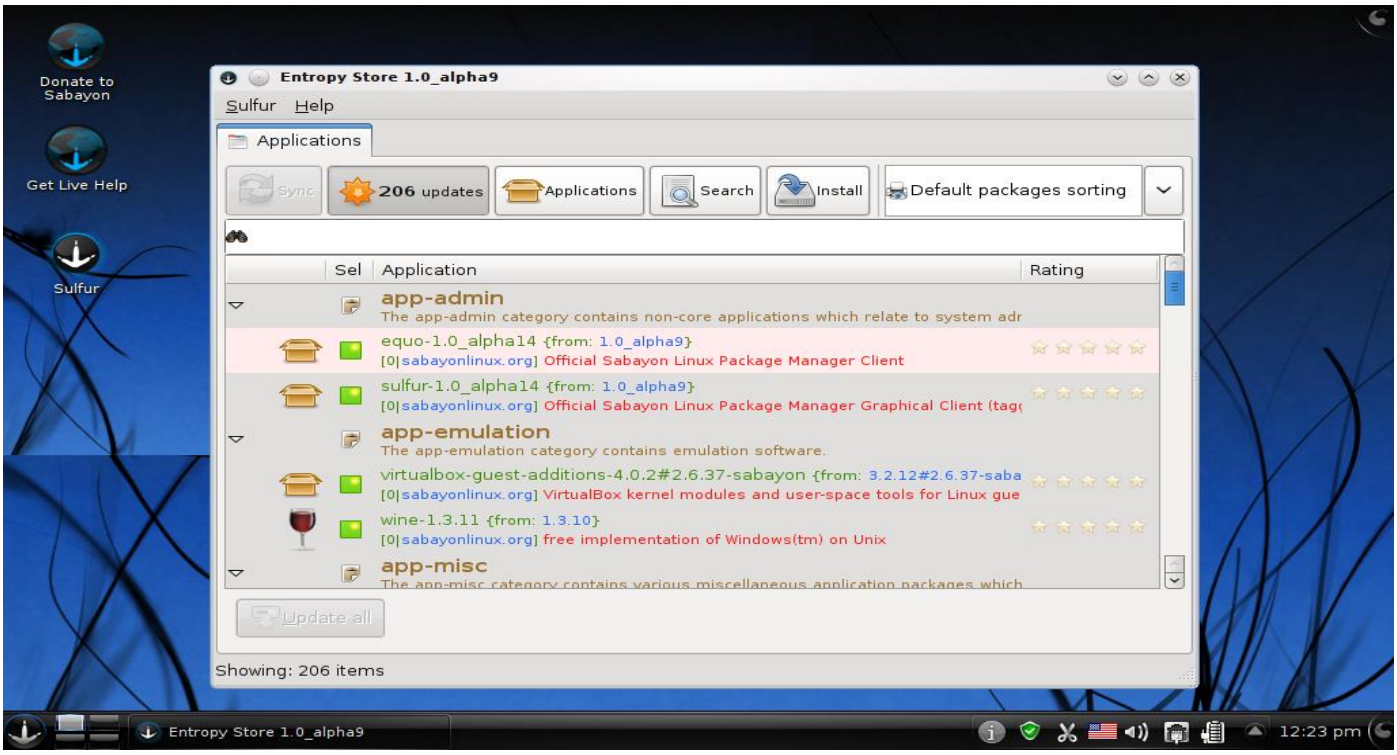


Figura 20. Gestor de paquetes Entropy.

Anexo 7.

Figura 21. Deepin Software Center.

Anexo 8.

Figura 22. App Store de Mac.