Universidad de las Ciencias Informáticas

Facultad 1

“Generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova”

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias de la Informática.

Autora: Ariagna Caridad Oramas Interian
Tutores: Ing. Hector Pérez Baranda
Ing. Jesús Rabelo Pérez

Habana, junio de 2015

Carretera San Antonio de los Baños. Km 2 ½ .Torrens. Municipio Boyeros. La Habana. Cuba

Declaración de Autoría

Declaro ser la única autora del presente trabajo y autorizo a la Facultad 1 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Ariagna Caridad Oramas Interian

_____ _____

Ing. Hector Pérez Baranda Ing. Jesús Rabelo Pérez


'Hay que tener fe en uno
mismo.Ahí reside el secreto'

Agradecimientos

A mis tutores Hector Pérez Baranda y Jesus Rabelo Pérez por confiar en mí y brindarme su ayuda en todo momento.

A mi familia, mi papá y mis hermanos por confiar siempre en mí.

A mis amigos y hermanos Rayco y Raúl por brindarme su amistad y apoyo incondicional.

A mis amigas Mayvis y Junet por acompañarme en esta etapa de mi vida.

A mi compañera de cuarto Yileni por brindarme su apoyo en los momentos tensos de la tesis.

A mis compañeros de aula por recibirme cada día con una sonrisa.

A mi novio por apoyarme siempre, ayudarme y enseñarme con paciencia cada día.

A los profesores que me formaron durante mis años de estudio en la Universidad, especialmente a Yadier, Osiris, Monica, Zumeta, Gendri, Orlando Cárdenas, a todos muchas gracias.

Al tribunal por ayudarme en cada dificultad que presenté a Alan, Mónica, Briseis y mi oponente Amaury.

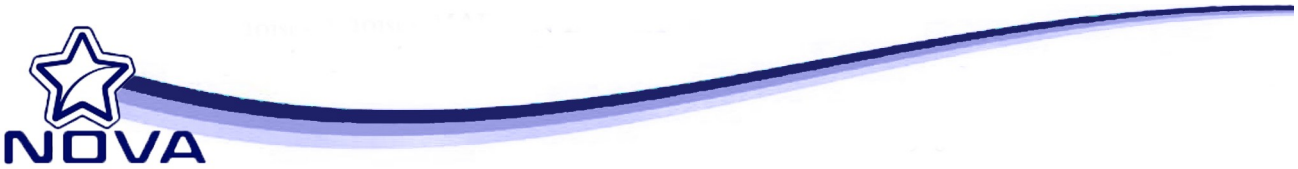
A todos los que de una forma u otra me han apoyado durante toda mi vida, alentándome a seguir alcanzando mis metas paso a paso.

Dedicatoria

A mi madre Sara M Interian Hernández por ser lo más grande que tengo en la vida, por apoyarme en cada momento y confiar en mi.

A mi papá toso Ariel Veiga Cabrera por haberme guiado siempre.

A mi abuela Ada Hernández Rovira y mi tio Freddy Interian Hernández por brindarme su amor incondicional.



Resumen

Cuba, como parte de la estrategia de informatización de la sociedad que lleva a cabo, se encuentra desarrollando el sistema operativo GNU/Linux Nova, que apuesta por la utilización del software libre y el estándar de código abierto. En el sistema operativo se está desarrollando una Plataforma que permitirá mejorar el proceso de construcción de la distribución cubana GNU/Linux Nova. El propósito del presente trabajo consiste en la implementación de una herramienta que permita la creación de dicha distribución. Esta herramienta posibilitará eliminar tiempo y esfuerzo de los desarrolladores que actualmente crean las distribuciones del sistema operativo de forma manual. Dentro de los servicios que brindará se encuentra el de poder conectarse a un repositorio de paquetes y seleccionar los que se deseen tener instalados una vez terminado el proceso de construcción del sistema operativo GNU/Linux Nova. Una vez realizada la investigación se desarrolló el análisis y diseño de las características del sistema y con una base tecnológica bien definida se comenzó con la implementación de la aplicación. Todo el proceso finalizó con una etapa de pruebas para verificar el correcto funcionamiento de la herramienta de Generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova.

PALABRAS CLAVE: Distribución, Integración Continua, Plataforma, Repositorio, Sistema Operativo.

Índice de contenido

Índice

Índice de figuras

Índice de figuras

Índice de tablas

Índice de tablas

Introducción

En los últimos años los sistemas operativos basados en GNU/Linux están siendo utilizados en diversos dispositivos como computadoras, reproductores de multimedia, teléfonos, *tablet* y equipos de propósito general, gracias al apoyo y respaldo de importantes empresas y corporaciones tales como IBM, Novell, Dell, Oracle, Adobe, Google, Yahoo, entre otras. Otra muestra significativa del alcance que tiene GNU/Linux hoy en día es cuando se observa que Google lo utiliza tanto en sus terminales como en sus servidores. Estos últimos utilizan una versión especialmente modificada y optimizada para su motor de búsqueda y el resto de los servicios que ofrece la compañía [1].

Las nuevas tecnologías de la información y las comunicaciones favorecen la productividad y facilitan las relaciones comerciales, un elemento vital de estas nuevas tecnologías es el software. Existe una alternativa para acceder al uso del mismo que permite a los usuarios su estudio, distribución, modificación y ejecución, denominada Software Libre [2].

El software libre contribuye a la igualdad entre los pueblos al permitir el libre acceso de todos a la sociedad del conocimiento. Para los países en vías de desarrollo es una limitante el excesivo costo de la licencia del sistema operativo Microsoft Windows. Además, se le suma a esto el precio de las licencias de los programas específicos tales como Microsoft Office, Corel-Draw, Adobe Photoshop y otros. Teniendo en cuenta que cada licencia solo puede ser utilizada en una única computadora, el precio final del software está en función del número de computadoras que se tenga.

Cuba -como parte de la estrategia de informatización de la sociedad que lleva a cabo- se encuentra desarrollando un sistema operativo que apuesta por la utilización del software libre y estándar de código abierto, debido a sus ventajas con respecto a los sistemas operativos privativos.

Algunas de las ventajas que representa el uso de software libre en lo económico son: no implica gastos

adicionales por concepto de cambio de plataforma de software, por cuanto es operable en el mismo soporte de *hardware* que tiene el país. La adquisición de cualquier aplicación de software libre puede hacerse de forma gratuita o en algunos casos a muy bajos precios. No requiere de licencia de uso, distribución y/o modificación, las cuales son generalmente muy caras [3].

En lo tecnológico hay que destacar que permite lograr una gran adaptación a los contextos de su utilización, lo que responde a la necesidad de socio-adaptabilidad y fomenta la innovación tecnológica del país. Al contar con el código fuente de las aplicaciones se garantiza un mayor por ciento de efectividad en la corrección de errores y obtención de actualizaciones, asegurando una soberanía tecnológica porque diversos sectores pueden contribuir a las mejoras de las aplicaciones [3].

Cuba desde febrero del 2009 cuenta con Nova, una distribución cubana de GNU/Linux que busca maximizar la soberanía tecnológica, socio-adaptabilidad, seguridad y sostenibilidad, desarrollada y mantenida por el Departamento de sistema operativo del Centro de Soluciones Libres (CESOL) de la Universidad de las Ciencias Informáticas (UCI) [4].

Una distribución de GNU/Linux es un conjunto de aplicaciones de software unido al núcleo Linux, las variaciones de estas aplicaciones dan origen a ediciones domésticas, empresariales y de servidores, aumentando así su proliferación. Su construcción está basado en los aportes de sus desarrolladores, que son aprobados por las instituciones patrocinadoras, de ahí que existan diferentes distribuciones: Debian, Ubuntu, Red Hat, entre otras; que aunque sean distintas, todas tienen como propósito común solución de algún problema en específico [5].

Las distribuciones basadas en GNU/Linux poseen su propio sistema de gestión de paquetes de software[1] y de productos. Un sistema de gestión de paquetes es un conjunto de herramientas que son utilizadas por los desarrolladores de la distribución con la posibilidad de automatizar el proceso de creación, actualización y eliminación de paquetes de software. Es importante aclarar que este sistema de gestión se

1 Paquete de software: es una serie de programas que se distribuyen conjuntamente.

refiere a la gestión física del paquete en el repositorio no a la creación, actualización y eliminación en un sistema instalado. Estas distribuciones también cuentan con su propio sistema de gestión de productos, dígase la generación de ISOS[2], catalogando a estos como uno de los resultados principales del proceso de construcción de la distribución que permite automatizar esta tarea. Como resultado de este proceso se logra la liberación de carga de los desarrolladores, además permite la creación de personalizaciones que aumenta la utilización y visibilidad de la distribución [6].

Contar con un gestor de productos que dosifique el factor tiempo en el proceso de desarrollo evitaría atrasos e incumplimientos en los cronogramas. Aunque el proceso de generación de ISOS no es complicado, en ocasiones resulta extenso, atentando directamente a la productividad del equipo de desarrollo.

El actual desarrollo de la Plataforma de Desarrollo e Integración Continua de Nova (PDICN) no permite el proceso de construcción de la distribución cubana de GNU/Linux Nova. Su principal objetivo va enfocado a aplicar los principios de integración continua en el desarrollo de sistemas operativos libres y de código abierto. Actualmente las tareas del proceso de construcción de una imagen de instalación del tipo ISO-9660 se realiza de forma manual influyendo negativamente en el tiempo, esfuerzo y evitando la proyección a tareas de desarrollo, mantenimiento y soporte del sistema operativo de los desarrolladores.

Una de las tareas que retrasa el desarrollo de la distribución es la generación de los productos a partir del repositorio de paquetes de forma manual, se divide en dos subtareas: la creación de metapaquetes[3] que proveen las funcionalidades para satisfacer los requisitos de un producto, y la generación de una imagen de instalación del tipo ISO-9660 para la creación de CD, DVD y memorias USB de instalación. Actualmente en el estado de puesta en marcha de la PDICN esta funcionalidad no se encuentra disponible.

2 ISOS**:** archivo donde se almacena una copia o imagen exacta de un sistema de ficheros. Se rige por el estándar ISO-9660 que le da nombre.

3 Metapaquetes**:** es un paquete que referencia a uno o varios paquetes pero no realiza ninguna funcionalidad.

Tomando como punto de partida la situación descrita se plantea como **problema científico**: ¿Cómo automatizar el proceso de creación de una imagen de instalación ISO-9660 de la distribución cubana de GNU/Linux Nova en la Plataforma de Desarrollo e Integración Continua de Nova?

Por lo que se establece como **objetivo general:** Automatizar el proceso de creación de un ISO a partir de un repositorio en la Plataforma de Desarrollo e Integración Continua de Nova para la distribución cubana de GNU/Linux Nova.

Para darle cumplimiento al objetivo general fueron definidos los siguientes **objetivos específicos:**

- Caracterizar los procesos de construcción de distribuciones de GNU/Linux.
- Implementar extensiones y/o componentes del proceso de creación de imágenes de instalación del tipo ISO-9660 en la Plataforma de Desarrollo e Integración Continua de Nova.
- Evaluar la solución propuesta.

Para dar cumplimiento a los objetivos generales y específicos se identificaron las siguientes **tareas de investigación**:

- Revisión de bibliografía sobre las tecnologías y herramientas utilizadas para el desarrollo de la PDICN para tener una base de conocimientos sobre que herramientas trabajar.
- Revisión de bibliografía sobre paquetes, repositorios y sistemas de instalación para conocer como funciona en procesos de creación de imágenes.
- Diseño de los componentes y/o aplicaciones que den solución al problema, teniendo en cuenta aplicaciones pre-existentes que pueden integrarse a la solución propuesta.
- Implementación de la solución propuesta.
- Diseño de los casos de prueba para la evaluación de la solución.

- Realización de la evaluación de la aplicación.
- Corrección de los posibles errores detectados.

El **objeto de estudio** de la investigación se centra en el proceso de construcción de sistemas operativos GNU/Linux Nova, enmarcando el **campo de acción** en el proceso de creación de imágenes de instalación del tipo ISO-9660 para el sistema operativo GNU/Linux Nova.
En el presente trabajo de diploma se plantea como **idea a defender** que la automatización del proceso de creación de imágenes del tipo ISO-9660 en la PDICN influirá directamente en el proceso de desarrollo de la distribución cubana GNU/Linux Nova ahorrando tiempo y esfuerzo de sus desarrolladores.

Métodos Utilizados

Con el objetivo de obtener los conocimientos a partir de los datos de la práctica y de la teoría precedente y elaborar una estrategia general para enfrentar el problema que se investiga, se utilizan los siguientes métodos de la investigación científica.

- Histórico-Lógico: utilizado para conocer acerca de la evolución del proceso de construcción de algunas de las distribuciones de GNU/Linux más usadas, enfocándose en la tarea de generación de productos para conocer la lógica de su desarrollo, qué elementos de esencia incidieron en los cambios operados en cada distribución y así lograr una mejor comprensión de la problemática existente.
- Analítico-Sintético: en el análisis y síntesis de la información consultada sobre el proceso de construcción de distribuciones y la generación de productos para así elaborar ideas que se apliquen a la solución propuesta.
- Modelación: utilizado en la representación, mediante el uso de diagramas, de las características del sistema, relaciones entre objetos; y las actividades que intervienen en los procesos implementados por la herramienta.

El presente trabajo de diploma está estructurado por tres capítulos, descritos brevemente a continuación:

Capítulo 1: Generación de productos: Explica los principales conceptos que se tratan para lograr un mayor entendimiento del tema tratado. Estudio de las herramientas que realizan la función de generación de productos en algunas de las distribuciones más conocidas para determinar elementos que puedan ser de utilidad para una solución propuesta.

Capítulo 2: Proceso de desarrollo de la aplicación: Se definen los procesos necesarios del desarrollo de la aplicación, así como el diseño de la estructura del producto, para empezar la implementación según la metodología trazada.

Capítulo 3: Implementación y validación de la solución propuesta: Se explican las tareas de implementación, los resultados obtenidos, se exponen las funcionalidades alcanzadas en el período de implementación y se diseñan, realizan y documentan las pruebas para lograr la verificación del producto.

Capítulo 1: Generación de productos

El objetivo de este capítulo consiste en exponer los conceptos fundamentales para lograr una mejor comprensión sobre los términos tratados en el proceso de Generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova, así como un estudio sobre los sistemas de generación de productos en las distribuciones más utilizadas como Debian, Ubuntu, entre otras. Por último, se presenta tanto la metodología como las herramientas y tecnologías a utilizar en la implementación del sistema.

1.1 Conceptos fundamentales

Paquete de código fuente

Un paquete fuente es aquel que incluye código escrito en un lenguaje de programación, y generalmente puede ser utilizado por cualquier tipo de máquina si el código se compila de manera correcta [7].

Paquete Binario

Un paquete binario es el que está construido específicamente para algún tipo de ordenador o arquitectura que puede ser x86 (i386-i686), AMD64, ARM, MIPSEL, u otras [7].

Dependencia de paquetes

Los programas a menudo utilizan los mismos archivos que otras aplicaciones. En vez de poner esos archivos en un mismo paquete, se puede distribuir por separado en diferentes paquetes para proporcionarlos a todos los programas que los necesiten. Por eso, al instalar programas que necesitan esos archivos, el paquete que los contiene debe ser instalado. Cuando un paquete depende de otro de esa manera, esto se conoce como dependencia de paquete [7].

Repositorio de paquetes

Un repositorio es una estructura organizativa similar a las bases de datos que contiene los paquetes, ya sea binario o fuente, así como la información de los mismos, que están disponibles para ser descargados e instalados [7].

Producto de *software*

Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación [8].

Metapaquetes

Un metapaquete es una especie de paquete que no tiene ninguna funcionalidad, simplemente tiene como dependencia otros paquetes para su instalación y configuración, permitiendo agrupar varias aplicaciones para una funcionalidad en específico en un solo paquete [9].

Imagen de instalación

Una imagen ISO es un archivo donde se almacena una copia o imagen exacta de un sistema de ficheros, normalmente un disco compacto, un disco óptico, como un CD, un DVD, pero también soportes USB. Se rige por el estándar ISO-9660 que le da nombre [9].

1.2 Herramientas para la generación de productos

A continuación se realizará un análisis acerca de las herramientas para la generación de productos de las distintas distribuciones de GNU/Linux. El análisis profundo de dichas herramientas permite que se tenga una visión más clara de las funcionalidades que se pueden aprovechar y otras que se puedan mejorar, a la hora de la elaboración de la solución propuesta.

1.2.1 Remastersys

Se utiliza para crear un Live CD/DVD personalizado de Ubuntu y algunos de sus derivados, permite además hacer una copia de seguridad de todo un sistema, incluyendo los datos de usuario. Si hay una herramienta importante para los que utilizan Ubuntu en grandes despliegues empresariales es Remastersys. Con esta herramienta se toma Ubuntu de una PC con todas las aplicaciones instaladas y el usuario puede hacer su propia versión y utilizar esta para instalar cuantas máquinas se deseen [10].

Características

- La herramienta de generación imágenes de instalación Remastersys, dispone tanto de una versión de línea de comandos para la creación de su imagen de instalación como de una versión en interfaz gráfica que se adapte a las necesidades de los usuarios inexpertos en software libre. Al finalizar el proceso de construcción de una imagen solo queda instalarla en la máquina deseada, a través de la utilización de una unidad CD/DVD o creando una memoria USB.
- Remastersys permite crear diferentes tipos de sistemas con soporte de instalación, a través de la utilización de dos opciones diferentes en el proceso de personalización de la misma [10].
- **Dist:** Esta opción permite crear una imagen sin los ficheros personales de los usuarios del sistema (/home) ni configuraciones propias. Es recomendable eliminar el historial y las caches así como basarse en el directorio /etc/skel que es el utilizado por defecto para la configuración de los usuarios del sistema; característica que explota al máximo esta herramienta [10].
- **Backup:** Es utilizada para realizar la imagen con configuraciones incluidas del /home, básicamente lo que se realiza es una copia del mismo, es decir, una copia de seguridad completa del sistema, incluyendo las configuraciones de los usuarios [10].
- Por defecto remastersys crea la imagen de instalación con el nombre customdist.iso en el directorio /home/remastersys; sin embargo, esta característica se puede modificar cambiando el

lugar donde se debe generar la imagen de instalación y el nombre que se decida indicar a la misma.

- En el proceso de creación de la imagen también se permite personalizar el instalador, modificándole características como que imagen de fondo a utilizar en el menú de arranque y la selección de un usuario al cual se le realizara la salva de los documentos.

Desventajas

- Una de las desventajas de gran impacto es que el autor de Remastersys anuncia en su sitio web que abandonó el proyecto en abril del 2013.
- Un elemento importante a tener en cuenta es que cuando se crea un Live CD/DVD a partir de un sistema instalado trae consigo el almacenamiento de los controladores presentes en ese momento además de los que trae en el núcleo; por lo tanto, a la hora de exportarlo para otro dispositivo hay que prever que sea compatible con el anterior porque podría ocasionar un mal funcionamiento del sistema creado. Se recomienda que la imagen a generar no debe sobrepasar el tamaño de un DVD porque no puede generarla el programa.

1.2.2 Ubuntu Customization Kit (UCK)

Es una herramienta que ayuda al usuario a crear un Live CD/DVD personalizado de la distribución Ubuntu, así como también de sus derivadas por ejemplo: Kubuntu, Xubuntu y Edubuntu [11].

Características

Por defecto la herramienta utiliza un directorio (tmp) predefinido donde almacena todos los datos utilizados durante el proceso, aunque puede ser cambiado y reubicado según las necesidades del usuario final. Dentro del mismo se encuentran otros como "remaster-iso-cache", donde se almacenan las modificaciones realizadas al Live CD, esto aumenta la rapidez de ejecución de la tarea, requiriendo menos

espacio en disco y ordenando la instalación de software adicional; pero puede no funcionar para todos los escenarios.

UCK brinda una interfaz visual basada en zenity[4] lo cual posibilita al usuario una mejor integración con la herramienta, pero su mayor utilización radica en la interfaz de línea de comandos la cual brinda un conjunto de herramientas mucho más potente. A continuación se describen las líneas de comando:

- **UCK-remaster-chroot-rootfs:** Esta operación invoca un proceso, cambiando el directorio raíz del sistema, entrando en un entorno virtual donde se realizara todo el proceso de personalización.
- **UCK-remaster-clean:** Elimina todos los archivos temporales extraídos en el ISO, pero guarda los *scripts* de personalización del caché apt y el build.log.
- **UCK-remaster-clean-all:** Elimina todos los directorios creados durante el proceso de remasterización.
- **UCK-remaster-finalize-alternate:** Genera un nuevo paquete llave con la firma GPG[5] del usuario, los archivos de la versión y los índices. Este script es el último en ejecutarse cuando se trabaja en una imagen ISO.
- **UCK-remaster-mount:** Monta el ISO así como el sistema de archivos comprimidos dentro del Live CD a través de una serie de bucle invertido para crear instancias grabables de las imágenes en los directorios remaster-iso y remaster-root respectivamente. Es importante aclarar que esto puede no funcionar para todos los escenarios.
- **UCK-remaster-pack-iso:** Genera el nuevo archivo ISO a partir de la carpeta de trabajo.
- **UCK-remaster-pack-rootfs:** Genera el nuevo archivo de imagen rootfs a partir de la carpeta de

4 Zenity: es un programa para desplegar ventanas de diálogo que reciben o entregan información procesable con scripts en bash.

5 GPG: por sus siglas en inglés GNU Privacy Guard herramienta de cifrado y firmas digitales.

trabajo.

- **UCK-remaster-prepare-alternate:** Crea un repositorio extra. Este script tiene que ser llamado antes de poder empezar a añadir paquetes en los repositorios nuevos generados.
- **UCK-remaster-remove-win32-files:** Elimina todos los archivos relacionados con win32 (autorun, menú, Wubi, software ventanas) extraídos del ISO.
- **UCK-remaster-unpack-iso:** Este comando monta temporalmente el Live CD en un directorio para luego volcar todo su contenido en el directorio remaster-iso.
- **UCK-remaster-unpack-rootfs:** Extrae el sistema de archivos raíz, almacenado en forma comprimida, de una imagen ISO desempaquetada (uck-remaster-unpack-iso).

Desventajas

No es capaz de generar un ISO completamente a partir de un repositorio, su función radica en modificar un liveCD ya existente, lo cual trae consigo la imposibilidad de integrarlo a un proceso continuo de construcción de paquetes y la posterior creación del sistema operativo, también implica un mayor esfuerzo en tiempo y recursos para lograr una personalización ya que todo el proceso se realiza sobre un entorno simulado.

1.2.3 Reconstructor

Es un programa que permite a cualquier usuario construir una imagen ISO de Debian y derivados (Ubuntu, Xubuntu, Kubuntu). Reconstructor está presente como una aplicación de escritorio desde hace bastante tiempo; pero ahora hace su aparición como aplicación web que permite crear y personalizar un Live CD de Debian o Ubuntu [12].

Características

- Brinda la posibilidad de personalizar la imagen de arranque, fondos de escritorio, pantalla de GNOME[6], tema GDM[7] y otras configuraciones.
- Reconstructor es un programa escrito en Python y posee como una de sus opciones que permite añadir nuevas aplicaciones a los repositorios para su posterior compilación.
- Si se desea colaborar con otra persona el programa permite añadirlos en la sección Miembros, y el proyecto se mostrará en la sección de proyectos compartidos, lo cual posibilita el desarrollo de forma comunitaria. También presenta elementos avanzados que puede realizar con el Editor de secuencias de comandos.
- Para el uso de la herramienta Reconstructor es necesario tener una cuenta en la misma, la cual se gestionará a través del correo electrónico, también es necesario la creación de un proyecto para cada una de las personalizaciones a realizar. Por defecto presenta la configuración de los repositorios de Ubuntu con facilidades de búsqueda y localización de sus paquetes en los mismos.
- Reconstructor también ofrece una serie de otras personalizaciones, como cambiar el fondo de pantalla por defecto o añadir una imagen gráfica.
- Permite incluso agregar una clave Gconf[8] si se desea mover los botones de la ventana en Ubuntu. Las personalizaciones mencionadas anteriormente se encuentran en la sección Módulos.

Desventajas

Actualmente solo cuenta con una interfaz de escritorio y su implementación web aún se encuentra en fase experimental y desarrollo.

6 GNOME: es un entorno de escritorio e infraestructura de desarrollo para sistemas operativos GNU/Linux.
7 GDM: por sus siglas en inglés GNOME Display Manager es un gestor de configuración.
8 Gconf: sistema utilizado por GNOME para almacenar configuraciones del entorno gráfico.

1.2.4 SUSE Studio

Es un dispositivo de software o constructor de software, simple y rápido, basado en una interfaz de usuario web, donde se puede recrear y armar a medida desde cero un sistema operativo basado en SUSE Linux Enterprise u openSUSE, tanto para la arquitectura de 32 bits, como para la de 64 bits [13].

Características

- Posee una unidad de prueba que puede arrancar, configurar y probar el dispositivo en una ventana del navegador web, tras haber construido el sistema el cual puede ser descargado posteriormente.
- Permite crear múltiples distribuciones y mantener almacenados los "perfiles" para recrearlos cuando se necesite.
- SUSE Studio le permite cargar software empaquetado en el formato RPM[9]. Si se desea subir varios archivos puede que resulte más fácil crear un comprimido (.tar, .tar.gz, .tgz, .tar.bz2, .tbz, o .zip) en lugar de subir cada paquete individual. El sistema luego se encarga de extraer este archivo y agregar los paquetes RPM individuales a su colección de software [13].
- Genera diferentes formatos de imagen, para entorno físicos, virtualizados o incluso para servicios de computación en la nube.
- Se pueden compilar discos virtuales en los formatos estándar de VMware o Xen, archivos ISO para Live CDs, imágenes para almacenar directamente en disco, en memorias USB o tarjetas SD, y hay anunciado que pronto soportará imágenes para Amazon EC2[10] [14].
- Aparte de las aplicaciones disponibles en SUSE Studio, se puede buscar y añadir paquetes concretos de los repositorios oficiales de SUSE o acudir a repositorios de terceras partes,

9 RPM: por sus siglas en inglés Red Hat Package Manager es una herramienta de administración de paquetes.
10 Amazon EC2: por sus siglas en inglés Amazon Elastic Compute Cloud es una parte central de la plataforma de cómputo en la nube.

automáticamente el sistema buscará y añadirá las dependencias.

- Permite añadir sus propios archivos y especificar dónde se instalarán en el sistema, incluso sobreescribiendo los de algunos paquetes ya existentes.
- SUSE Studio permite incluir su propio logo y fondo de escritorio para la distribución, además ofrece un panel para personalizar las configuraciones de red, idioma, cuentas de usuario y grupos pregenerados. También al añadir MySQL permite subir contenido a las bases de datos.
- La selección de la arquitectura que se elija está condicionada al material del que se disponga, para a partir de ahí tomar una decisión correcta. Si se dispone de máquinas muy antiguas puede que no tengan CPUs de 64 bits, esto restringe la elección a 32-bit. Todo este tipo de elementos deben ser considerados al momento de elegir la arquitectura.

SUSE Studio posee una página en la cual se puede editar el *template* base. Cuenta con seis pestañas disponibles que se describen a continuación.

- ***Start***: La pestaña de bienvenida donde se pueden nombrar/renombrar la aplicación.
- ***Software***: La pestaña en la que puede seleccionar los paquetes de los repositorios de openSUSE.
- ***Configuration***: Como su nombre lo indica, es aquí en donde será posible el ir configurando los diferentes aspectos del sistema.
- ***Files***: Los archivos que aquí se añadan será copiados en la aplicación después de instalar los paquetes. Adding files (Agregar Archivos) es opcional. Los permisos y las jerarquías serán preservados.
- ***Build***: Esta pestaña ayuda a armar la aplicación.
- ***Share***: Le permite compartir su aplicación por todo el mundo.

Desventajas

- Para tener acceso a la página y crear el ISO es obligatorio tener una cuenta online de las llamadas OpenID[11], lo cual dificulta su uso en un entorno que carezca de conectividad y de la implementación del protocolo necesario para lograr la autenticación.

1.2.5 Genisoimage

Genisoimage es un programa de pre-masterización para generar imagen del tipo ISO-9660. Anteriormente se denominaba mkisofs y a partir de la versión 4 de Debian cambió de nombre a acrónimo de Generate Iso Image. Es capaz de describir los ficheros en el sistema de archivos ISO-9660 a un Host UNIX y proporciona información como nombres de archivos largos, permisos POSIX[12], enlaces simbólicos y dispositivo de bloque. Es un programa que permite crear imágenes de sistemas de archivos ISO-9660 que pueden grabarse en CD, DVD o memorias usb. El sistema incluye herramientas útiles para trabajar con las imágenes ISO como [15]:

- **mkzftree:** Crea la imagen ISO-9660 con contenidos comprimidos.
- **Dirsplit:** Separa fácilmente grandes contenidos de un directorio en discos de un tamaño predefinido.

Características

- Genera sistemas de ficheros híbridos (HFS[13]), estos se observan cuando se accede desde un Macintosh[14] y se observan los archivos en formato ISO-9660 cuando se accede desde otras máquinas.

11 OpenID: estándar de identificación digital descentralizado, con el que un usuario puede identificarse en una página web a través de una URL.
12 POSIX: por sus siglas en ingles Portable Operating System.
13 HFS: por sus siglas en ingles Hierarchical File System, es un sistema de archivos desarrollado por Apple Inc.
14 Macintosh: línea de ordenadores personales diseñada, desarrollada y comercializada por Apple Inc.

- Como alternativa, genisoimage puede generar las extensiones de Apple ISO-9660 para cada archivo. Estas extensiones proporcionan a cada fichero algunas banderas (*Finder*[15]) cuando se accede desde un Macintosh.
- Genisoimage asigna ponderaciones a cada nombre de archivo, y si se encuentran dos nombres que si son iguales, el nombre con la prioridad más baja se modifica para incluir un número de 3 dígitos (garantizado ser único). Por ejemplo, los dos archivos foo.bar y foo.bar.~1~ podrían interpretarse como FOO.BAR; 1 y FOO000.BAR; 1.
- El sistema toma lo que se encuentra en el directorio seleccionado y genera una imagen binaria que corresponderá a un sistema de archivos ISO-9660 y/o sistema de ficheros HFS cuando escriben en un dispositivo de bloque.
- Cuando se utilizan varias opciones de HFS, genisoimage intentará un reconocimiento de archivos almacenados en un número de formatos de Apple/Unix y copiará los datos y las bifurcaciones de recursos, así como cualquier información relevante del Finder.
- Se debe tener en cuenta que genisoimage no está diseñado para comunicarse directamente con el *hardware*. La mayoría de los quemadores tienen un conjunto de comandos propietarios que pueden variar de un fabricante a otro, y necesita una herramienta especializada para quemar el disco.
- El directorio donde se copiará el sistema de archivos ISO-9660 es el especificado. Se pueden especificar múltiples rutas y genisoimage combinará los ficheros encontrados en todos los componentes de la ruta de acceso especificada, para formar el sistema de archivos de imagen.

15 Finder: para el desarrollo de la investigación se utilizará el término Finder para referirse a bandera.

Valoración del estudio de las herramientas para la generación de productos

De las cinco herramientas de gestión de productos estudiadas (Remastersys, Ubuntu Customization Kit, Reconstructor, SUSE Studio y Genisoimage) ninguna cumple con las características del país y del proyecto, porque tienen como limitantes que algunas requieren de la conexión a Internet para subir los fuentes y crear los ISOS como es el caso de Suse Studio y Reconstructor. Por otra parte la versión liberada de Remastersys, es obsoleta y descontinuada, también es el caso de Ubuntu Customization Kit, adicionándole que solamente permite modificar un ISO ya existente, la herramienta Genisoimage también genera imágenes del tipo ISO-9660, pero ninguna de las herramientas estudiadas permite la integración de todo el proceso de compilado de paquetes y creación del sistema operativo, por lo que el uso de algunas de estas herramientas es ineficiente. Como la mayoría de ellas realizan la gestión en Internet, el proyecto Nova desconoce el estado real de cada paquete existente en sus repositorios, lo que impide lograr la soberanía tecnológica, no garantiza la sostenibilidad en el tiempo así como la seguridad. Por tal motivo se hace necesario la creación de una herramienta que permita integrar todo el proceso de construcción de una imagen de instalación del tipo ISO-9660 a partir de un repositorio de paquetes.

Tabla 1: Comparación de las herramientas estudiadas.

	Características			
Herramientas	**Uso de Internet**	**Generación de ISOS**	**Personalización de ISOS**	**Construcción de ISOS a partir de Repositorios**
Remastersys		x	x	
Ubuntu Customization Kit		x	x	
Reconstructor	x	x	x	x
Suse Studio	x	x	x	x
Genisoimage		x		

1.3 Metodología de desarrollo

Para poder obtener mayores resultados en el desarrollo de software, lograr hacer un trabajo organizado y ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto, surge una solución, las metodologías de desarrollo, que tienen como objetivo presentar un conjunto de técnicas tradicionales y modernas de modelado de sistemas que permitan desarrollar software de calidad, incluyendo heurísticas de construcción y criterios de comparación de modelos de sistemas.

Existen dos tipos de metodologías:

Pesadas o tradicionales: Hace énfasis en la planificación del proceso de desarrollo, imponiendo una mayor disciplina, luego de que todo esté en orden se comienza el desarrollo. Se emplea en el desarrollo de sistemas grandes, lo cual hace que no se adapte a entornos de trabajos cambiantes, donde los requisitos o no se pueden predecir o están en constante cambio [16].

Ágiles: Apuestan por un desarrollo de software más rápido, ya que se adaptan con mayor facilidad a cualquier cambio que se pueda presentar, intenta evitar los caminos de las metodologías pesadas, enfocándose más en la producción rápida de software [16].

Características de la metodología OpenUp

OpenUp (Open Unified Process en español Proceso Unificado Abierto) es un *framework*[16] de procesos de desarrollo de programas de código abierto, que 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,

16 framework: marco de trabajo que define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular.

completo y extensible; para pequeños equipos de desarrollo, que valoran los beneficios de la colaboración y los involucrados, con el resultado del proyecto, por encima de formalidades innecesarias. Ciclo de Vida de la metodología OpenUp [17].

Figura 1: Ciclo de vida de la metodología OpenUP

En la Figura 1 se puede observar las cuatro fases de desarrollo que propone la metodología, las cuales se definen a continuación:

1. **Fase de Concepción:** realizar los planteamientos del problema, elaborar la justificación, definir objetivo general y objetivos específicos, obtener toda la información relacionada con el proyecto y capturar las necesidades de los clientes en los objetivos del ciclo de vida para el proyecto.

2. **Fase de Elaboración:** definir los riesgos significativos para la arquitectura y establecer la base para la elaboración de la arquitectura del sistema.

3. **Fase de Construcción:** diseñar, implementar y realizar las pruebas de las funcionalidades para realizar un sistema completo y completar el desarrollo del sistema basado en la arquitectura definida.

4. **Fase de Transición:** asegurar que el sistema sea entregado a los usuarios y evaluar la funcionalidad y

escena del último entregable de la fase de construcción.

Después del análisis realizado se determina el uso de una metodología ágil porque es muy idónea para las entregas de productos en pequeños espacios de tiempo. Se desea desarrollar una aplicación que se integrará con la Plataforma de Desarrollo e Integración Continua de Nova y como este proyecto utiliza la metodología OpenUp para lograr una integración exitosa se decide utilizar esta metodología de desarrollo. Es importante conocer que OpenUp ayuda a la creación, distribución y mantenimiento de las aplicaciones de software, además los requisitos son variables y el equipo de desarrollo es pequeño.

1.4 Lenguajes y herramientas

1.4.1 Herramienta CASE

Como herramienta CASE[17] se utilizó el Visual Paradigm en su versión 8.4, que utiliza el lenguaje Unificado de Modelado (UML[18]) profesional que soporta el ciclo de vida completo de desarrollo del software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El lenguaje de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad y generación automática de código. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Visual Paradigm soporta los principales estándares de la industria tales como SysML, BPMN, XMI, entre otros. Ofrece un completo conjunto de herramientas de los equipos de desarrollo de software necesario para los requisitos de la captura, software de planificación, la planificación de controles, el modelado de clases y datos, entre otros [19].

Las características básicas de la herramienta CASE son:

17 CASE: por su siglas en inglés Computer Aided Software Engineering.

18 UML: por sus siglas en inglés Unified Modeling Language.

- Proporcionar topologías de aplicación flexibles. La herramienta debe proporcionar facilidades de construcción que permita separar la aplicación entre el cliente, el servidor y más importante, entre servidores.
- Proporcionar aplicaciones portátiles. La herramienta debe generar código para Windows, OS/ 2, Macintosh, Unix y todas las plataformas de servidores conocidas. Debe ser capaz, a tiempo de corrida, desplegar la versión correcta del código en la máquina apropiada.
- Control de Versión. La herramienta debe reconocer las versiones de códigos que se ejecutan en los clientes y servidores, y asegurarse que sean consistentes. También, la herramienta debe ser capaz de controlar un gran número de tipos de objetos incluyendo texto, gráficos, mapas de bits, documentos complejos y objetos únicos, tales como definiciones de pantallas y de informes, archivos de objetos y datos de prueba y resultados.
- Trabajar con una variedad de administradores de recurso. La herramienta debe adaptarse ella misma a los administradores de recurso que existen en varios servidores de la red; su interacción con los administradores de recurso debería ser negociable a tiempo de ejecución.
- Trabajar con una variedad de software intermedio. La herramienta debe adaptar sus comunicaciones cliente / servidor al software intermedio existente. Como mínimo la herramienta debería ajustar los temporizadores basándose en, si el tráfico se está moviendo en una LAN o WAN.

1.4.2 Lenguaje de Programación Python

Se utiliza como lenguaje de programación Python en su versión 2.7, es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1991. Su implementación se encuentra bajo la licencia de código abierto. Su filosofía de diseño enfatiza en la legibilidad del código y su sintaxis es clara y expresiva. Es un lenguaje multiparadigma, que no fuerza un estilo de programación en particular, sino

que permite la programación orientada a objeto, la programación estructurada, así como la programación funcional y la orientada a aspectos. Incluye módulos que proporcionan entradas y salidas de ficheros, llamadas al sistema y hasta interfaces gráficas de usuario (GUI) como TK, GTK, QT, y PythonCard. Python se utiliza como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, al no tener que compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa [20].

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Las características básicas de Python son:

- Python usa tipado dinámico y conteo de referencias para la administración de memoria.
- Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).
- Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

1.4.3 Herramienta Eclipse 4.3

Se utiliza como herramienta de desarrollo Eclipse en su versión 4.3, es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma. El

entorno de desarrollo integrado (IDE) de Eclipse emplea módulos para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido permitiéndole a Eclipse extenderse usando otros lenguajes de programación como Python [21].

Las características principales del IDE son:

- Dispone de un Editor de texto con un analizador sintáctico.
- La compilación es en tiempo real.
- Posee su propio control de versiones.

1.4.4 Marco de trabajo para aplicaciones web Django

Se utiliza como marco de trabajo para la web Django en su versión 2.0 de alto nivel y de código abierto escrito en Python. Se trata de una modificación del patrón arquitectónico Modelo-Vista-Controlador (MVC), a la que han llamado Modelo-Plantilla-Vista (MTV por sus siglas en inglés). Su meta fundamental consiste en el desarrollo de sitios complejos conducidos por la base de datos. Utiliza el lenguaje de programación Python para las configuraciones, archivos y modelos de datos. Provee una interfaz administrativa opcional generada dinámicamente y configurada mediante modelos de administración [22].

Algunas de sus características más relevantes son:

- Mapeador Objeto-Relacional: Permite definir los modelos de datos mediante código Python, y permite acceder a ellos mediante una API para las diferentes bases de datos soportadas.
- Diseño elegante de URL: Permite crear URLs sencillas y claras mediante expresiones regulares.
- Sistema de plantillas: Posee un lenguaje de plantilla robusto y eficiente, que separa el diseño, contenido y código Python de una manera elegante.
- Internacionalización: Presenta soporte total para crear aplicaciones multi-lenguajes.

Django unido al lenguaje de programación Python constituyen las tecnologías base en la Plataforma de Desarrollo e Integración Continua de Nova. De ahí que jueguen un papel relevante en la implementación de la solución del lado del servidor.

1.4.5 Herramientas para la construcción de metapaquetes

El proceso de “debianizar” fuentes consiste en adaptar estos ficheros para que funcionen de acuerdo con el sistema de paquetes de Debian. Cualquier persona puede realizar un paquete, aunque para que éste forme parte de la distribución debe ser un desarrollador oficial. Una de las razones por las que se utiliza el proceso de “debianizar” es cuando se desea integrar un paquete a una aplicación existente.

Una herramienta fundamental en el proceso de “debianizar” es el comando denominado dh_make encargado de crear el directorio debian con todos los ficheros necesarios para poder “debianizar” posteriormente. Este comando posee varios tipos de paquetes a generar y a continuación se presenta una breve descripción de ellos:

single binary: Permite generar un paquete binario, es el caso más frecuente.

indep binary: Permite crear un paquete binario que es independiente de la plataforma destino.

multiple binary: Cuando se tiene un proyecto muy grande y se compone de librerías, scripts y/o varios ejecutables. Con esta opción se le comunica a dh_make que desde el paquete fuente se crearán varios paquetes binarios.

library: Si se desea empaquetar una librería esta es la mejor opción, dh_make generará los archivos necesarios para construir la típica pareja de paquetes libloquesea.deb y libloquesea-dev.deb.

Una vez dh_make nos pedirá que confirmemos los datos introducidos y se creará el directorio debian conformado principalmente por los siguientes ficheros:

Copyright: Incluye la licencia del programa y del empaquetador.

Control: Contiene la descripción de los paquetes Debian. El primero de todos es el paquete fuente, con sus “build-depends”, las dependencias necesarias para compilar y construir los paquetes binarios. Tras éste, vienen listados todos los paquetes binarios que se generarán (separados por una línea en blanco).

Rule: Se trata de un *Makefile*[19] que será el encargado de construir el paquete. Normalmente, el código que genera será el necesario para realizar todas las tareas.

Changelog: Es donde se va poniendo los sucesivos empaquetamientos del programa (cuando se actualiza la versión, se incluyen parches) y el fichero *coyright*.

1.4.6 Herramienta Selenium

Selenium en su versión 2.9.2 es un entorno de pruebas de software para aplicaciones basadas en la web. Provee una herramienta de grabar/reproducir para crear pruebas sin usar un lenguaje de *scripting* para pruebas. Incluye también un lenguaje específico de dominio de pruebas para escribirlas en un amplio número de lenguajes de programación populares incluyendo Java, C#, Ruby, Groovy, Perl, Php y Python. Las pruebas pueden ejecutarse usando la mayoría de los navegadores web modernos en diferentes sistemas operativos como Windows, Linux y OSX [23].

Conclusiones parciales

El estudio de las principales herramientas de generación de productos presentes en diferentes distribuciones de GNU/Linux, evidenció la necesidad de que la Plataforma de Desarrollo e Integración Continua de Nova, incluya su propio sistema de gestión de productos para lograr una mejor integración del proceso de creación de la distribución GNU/Linux Nova, tomando como punto de partida algunos de estos sistemas ya creados.

19 Makefile*:* son los ficheros de texto que utiliza make para llevar la gestión de la compilación de programas.

Capítulo 2: Proceso de desarrollo de la aplicación

El presente capítulo se centra en esclarecer el proceso de desarrollo de la solución al problema planteado mediante el transcurso de las diferentes fases de la metodología OpenUp, y un grupo de artefactos ingenieriles que posibilitan la obtención de un producto de calidad.

2.1 Propuesta de solución

La solución propuesta consiste en construir un sistema de gestión de productos para la Plataforma de Desarrollo e Integración Continua de Nova. Se desea crear una interfaz de usuario web, donde se puede recrear y armar a medida desde cero un sistema operativo, tanto para la arquitectura de 32 bits, como para la de 64 bits. La aplicación debe ser capaz de conectarse a un repositorio ya existente para que muestre todos los paquetes del mismo. Debe tener la capacidad de gestionar cada metapaquete de la plataforma y crear nuevos en caso de ser requerido, permitiendo más adelante generar un ISO-9660 seleccionando los metapaquetes necesarios y de esta manera almacenar los "perfiles" para recrearlos cuando se necesite. Partiendo de que la Plataforma de Desarrollo e Integración Continua de Nova proporciona la funcionalidad de poder añadir nuevos software a los repositorios para su posterior compilación, el sistema de gestión de productos a desarrollar dará la posibilidad de que al crear el sistema operativo ya tendrá instalado los programas existentes en la Plataforma que sean seleccionados por el usuario.

2.2 Modelo de dominio

El diagrama que se elabora en el modelo de dominio de la Figura 2 representa los objetos relacionados con los principales conceptos que se trabajarán en el desarrollo de este trabajo. El diagrama permitirá a los usuarios, desarrolladores e interesados lograr una mayor comprensión de los conceptos que se

manejan, permitiéndoles utilizar un vocabulario común para comprender el contexto en el que se encuentra el sistema.

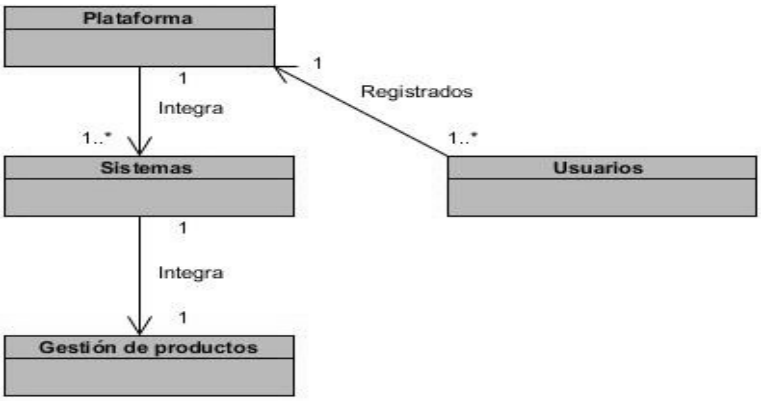


Figura 2: Modelo del dominio de la solución propuesta.

Descripción de Clases del Modelo de Dominio

La modelación del dominio constituye la herramienta fundamental para garantizar la comprensión y descripción de las clases o conceptos y sus relaciones más importantes dentro del contexto del problema. A continuación se presenta la descripción de los conceptos identificados en la presente investigación.

Tabla 2: Descripción del modelo de dominio.

Conceptos	**Descripción**
Usuarios	Persona que realiza las peticiones a la herramienta de Gestión de productos.
Plataforma	Sistema que sirve como base para hacer funcionar determinados módulos.
Sistemas	Sistemas que poseen las distintas funcionalidades de la Plataforma.
Gestión de productos	Constituye la herramienta de Gestión de productos.

2.3 Requisitos del sistema

Un requisito es una descripción detallada, en el cual el cliente describe brevemente las características y cualidades que un sistema informático debe tener. Para una mejor especificación se dividen en dos grupos, requisitos funcionales y no funcionales.

2.3.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, la manera en que éste debe reaccionar a entradas específicas y de cómo se debe comportar en situaciones particulares.

En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer, como también describen lo que el sistema debe hacer. Estos requerimientos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requerimientos. Cuando se expresan como requerimientos del usuario, habitualmente se describen de una forma bastante abstracta, sin embargo, los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas, salidas y excepciones. A continuación se describen los requisitos funcionales del sistema de Generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova:

1. Crear un metapaquete.
2. Modificar un metapaquete.
3. Listar un metapaquete.
4. Eliminar un metapaquete.
5. Crear ISO-9660.
6. Eliminar ISO-9660.
7. Listar ISO-9660 creados.

8. Seleccionar paquetes que incluirá la imagen ISO-9660.
9. Incluir configuraciones predeterminadas para la gestión de los metapaquetes.

2.3.2 Requisitos no funcionales

Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema, incluyen restricciones de tiempo sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema. A continuación se describen los requisitos no funcionales del sistema de Generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova:

Restricciones de diseño

1. Para la implementación del sistema se utilizará el lenguaje de programación Python.
2. Para la implementación del sistema del lado del servidor se utilizará el marco de trabajo para el desarrollo web Django.

Seguridad

1. Garantizar el tratamiento de excepciones.
2. Garantizar el control de acceso al sistema.
3. La información debe viajar por un canal cifrado.

Requisitos de licencia

1. Se implementará bajo la licencia GPL.

2.4 Modelo de casos de uso del sistema

Los casos de uso de un sistema, describen un conjunto de actividades desde el punto de vista de los actores, produciendo un resultado concreto y tangible. Son descriptores de las interacciones que se producen entre los usuarios y el sistema. Los casos de usos presentes en la solución son:

1. Gestionar metapaquetes.
2. Gestionar ISO-9660.

Actores del sistema

Los actores del sistema intercambian información con él, aunque no forman parte de este. Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado. A continuación se muestran los actores y la justificación que tienen en el sistema.

Administrador: Es el encargado de monitorizar todo las funcionalidades de la aplicación en el servidor, también gestiona los productos.

Usuarios: Representa a una persona que puede utilizar las funcionalidades del sistema.

Diagrama de casos de uso del sistema

En la figura 3 se representan los casos de uso, que son fragmentos de funcionalidades que agrupan los requisitos que debe cumplir el sistema. El modelo de casos de uso describe las funciones que realizan en el sistema y la interacción de estos con el actor.

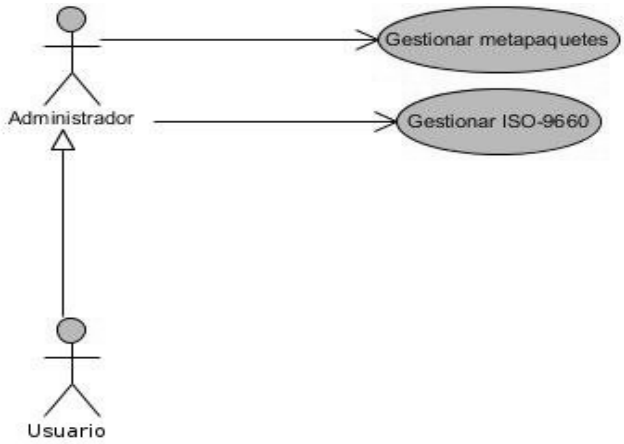


Figura 3: Diagrama de caso de uso de la solución propuesta.

2.5 Especificación de los casos de usos críticos

Tabla 3: Caso de uso: Gestionar metapaquetes.

Objetivo	Adicionar, modificar, eliminar y listar metapaquetes.	
Actores	Administrador, usuarios.	
Resumen	Adicionar, modificar, eliminar y listar metapaquetes que se necesitan para la creación de la imagen ISO-9660.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	-	
Postcondiciones	El metapaquete queda eliminado, modificado o creado según la opción escogida.	
Flujo de eventos		
Flujo básico <Gestionar metapaquete>		
	Actor	**Sistema**
1	Accede al formulario gestionar metapaquetes.	Permite realizar varias acciones: • Adicionar un nuevo metapaquete. Ver

		Sección 1: “Adicionar metapaquete” • Modificar un metapaquete. Ver Sección: 2 “Modificar metapaquete” • Eliminar un metapaquete. Ver Sección 3: “Eliminar metapaquete”
2		Termina el caso de uso
Sección 1: “Adicionar metapaquete”		
Flujo básico <Adicionar metapaquete>		
	Actor	**Sistema**
1	Selecciona la opción adicionar metapaquete.	Se muestra el formulario para adicionar el metapaquete.
2	Introduce los datos del formulario.	Se validan los datos introducidos.
3		Adiciona un nuevo metapaquete.
4		Termina el caso de uso.
Flujos alternos		
2.a <Los datos son incorrectos>		
	Actor	**Sistema**
1		Muestra un mensaje indicando el error cometido así como el formulario y campo en que se produjo.
Sección 2: “Modificar metapaquete”		
Flujo básico <Modificar metapaquete>		
	Actor	**Sistema**
1	Selecciona el metapaquete a modificar.	Muestra el formulario para modificar metapaquete.
2	Modifica el metapaquete.	Se validan los datos introducidos.
3		Termina el caso de uso.
Flujos alternos		
2.a <Los datos son incorrectos>		

	Actor	Sistema
1		Muestra un mensaje indicando el error cometido así como el formulario y campo en que se produjo
Sección 3: “Eliminar metapaquete”		
Flujo básico <Eliminar metapaquete>		
	Actor	**Sistema**
1	Selecciona el metapaquete a eliminar	Elimina el metapaquete seleccionado.
2		Termina el caso de uso.
Requisitos no funcionales	Para la implementación del sistema se utilizará el lenguaje de programación Python. Para la implementación del sistema del lado del servidor se utilizará el marco de trabajo para el desarrollo web Django. Garantizar el tratamiento de excepciones.	
Prototipos		

Tabla 4: Caso de uso: Gestionar ISO-9660.

Objetivo	Adicionar, modificar, eliminar y listar ISO-9660.	
Actores	Administrador, usuarios.	
Resumen	Adicionar, modificar, eliminar y listar imágenes del tipo ISO-9660.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	Tienen que existir los metapaquetes.	
Postcondiciones		
Flujo de eventos		
Flujo básico <Gestionar ISO-9660>		
	Actor	**Sistema**
1	Accede al formulario gestionar ISO-9660.	Permite realizar varias acciones con el ISO-9660: • Crear un nuevo ISO-9660. Ver Sección 1: “Adicionar ISO-9660” • Modificar un ISO-9660. Ver Sección: 2 “Modificar ISO-9660” • Eliminar un ISO-9660. Ver Sección 3: “Eliminar ISO-9660”
2		Termina el caso de uso
Sección 1: “Crear ISO-9660”		
Flujo básico <Crear ISO-9660>		
	Actor	**Sistema**
1	Selecciona la opción crear ISO-9660.	Se crea el ISO-9660.
2		Adiciona un nuevo ISO-9660.
3		Termina el caso de uso.
Flujos alternos		
2.a <Los datos son incorrectos>		
	Actor	**Sistema**

1		Muestra un mensaje de error .
Sección 2: “Modificar metapaquete”		
Flujo básico <Modificar ISO-9660>		
	Actor	**Sistema**
1	Selecciona el ISO-9660 a modificar.	Muestra el formulario para modificar el ISO-9660.
2	Modifica el ISO-9660.	Se validan los datos introducidos.
3		Termina el caso de uso.
Flujos alternos		
2.a <Los datos son incorrectos>		
	Actor	**Sistema**
1		Muestra un mensaje indicando el error cometido así como el formulario y campo en que se produjo
Sección 3: “Eliminar ISO-9660”		
Flujo básico <Eliminar ISO-9660>		
	Actor	**Sistema**
1	Selecciona el ISO-9660 a eliminar	Elimina el ISO-9660 seleccionado.
2		Termina el caso de uso.
Requisitos no funcionales	Para la implementación del sistema se utilizará el lenguaje de programación Python. Para la implementación del sistema del lado del servidor se utilizará el marco de trabajo para el desarrollo web Django. Garantizar el tratamiento de excepciones.	

Prototipos	

2.6 Arquitectura

Patrón de arquitectura

Los patrones arquitectónicos, tienen como objetivo fundamental asegurar que los sistemas informáticos cumplan con los objetivos de las cualidades de software; mantenibilidad, escalabilidad, reutilización y disponibilidad. A continuación se presenta el patrón más relevante en la implementación de la solución.

Modelo-Vista-Plantilla (MTV[20])

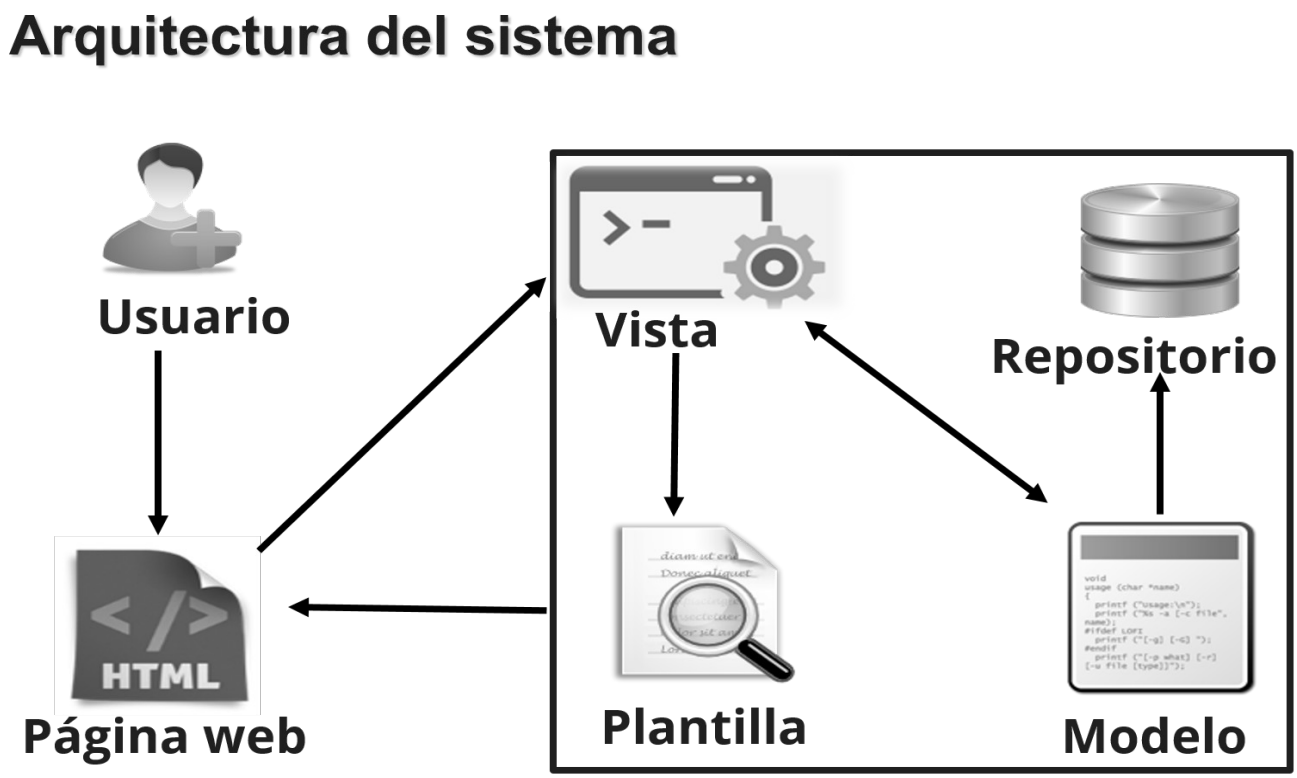


Figura 4: Ejemplo del funcionamiento modelo vista plantilla.

En la Figura 4 se muestra como funciona el patrón arquitectónico modelo-vista-plantilla. La utilización de este patrón, se encuentra asociado a la utilización del marco de trabajo para el desarrollo web Django que lo implementa. Debido a que la clase controladora es manejada por el mismo *framework* y la parte más importante se produce en los modelos, las plantillas y las vistas, Django es conocido como un Framework MTV.

20 MTV: Por sus siglas en ingles Model Template View.

2.7 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Para el diseño de la generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova se aplicaron las bases e ideas que proporcionan los patrones GRAPS[21], los cuales describen los principios fundamentales de asignación de responsabilidades a los objetos o clases.

Patrón Experto: Establece que la responsabilidad de realizar una determinada labor es de la clase que contiene todos los datos necesarios para ello.

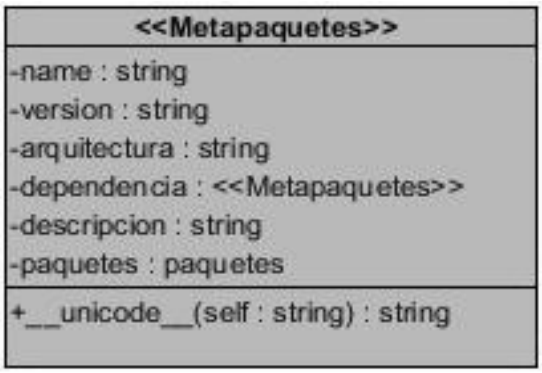


Figura 5: Ejemplo de la utilización de patrones.

Patrón bajo acoplamiento: Propone el diseño de clases con pocas dependencias, lo que reduce el impacto del cambio y facilita la reutilización en otros sistemas. En la Figura 6 se muestra como se depende únicamente de la clase Metapaquetes.

21 GRAPS: por sus siglas en ingles General Responsibility Assignment Software Patterns: Patrones de Software de Asignación de Responsabilidades.

Patrón Alta Cohesión: Detalla cómo cada clase debe realizar una labor única dentro del sistema, teniendo responsabilidades moderadas en un área funcional y colaborando con las otras para llevar a cabo las tareas. La responsabilidad de este patrón es evitar que se asignen demasiadas responsabilidades a las clases. En la Figura 6 se evidencia mediante el desempeño de métodos con funcionalidades únicas y altamente relacionadas.

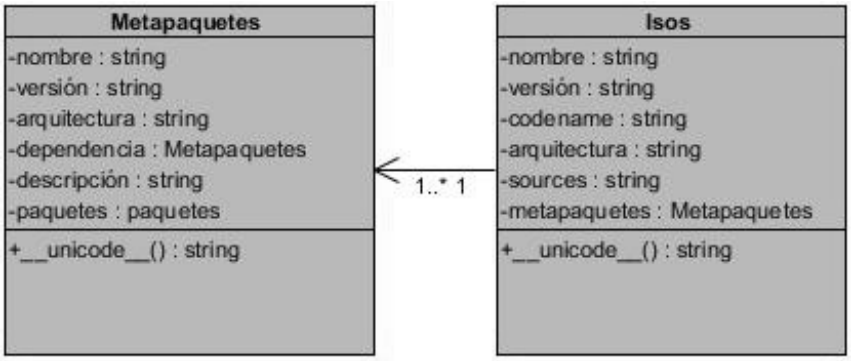


Figura 6: Diagramas Graps.

2.8 Diagrama de clases del diseño

Un diagrama de clases del diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Para una mayor comprensión de la realización física de los casos de usos se muestra en la Figura 7 el diagrama de clases del diseño de estereotipos web.

2.9 Diagrama de interacción

Los diagramas de interacción son utilizados para modelar los comportamientos dinámicos que caracterizan un sistema informático. Estos suelen representar un conjunto de objetos o clases y sus

relaciones, así como los mensajes que se pueden enviar entre ellos. En la Figura 8 y 9 se muestran los diagramas de secuencia correspondientes a algunos de los escenarios reflejados en los caso de uso. Estos a diferencia de los de colaboración, destacan el orden temporal de los mensajes.

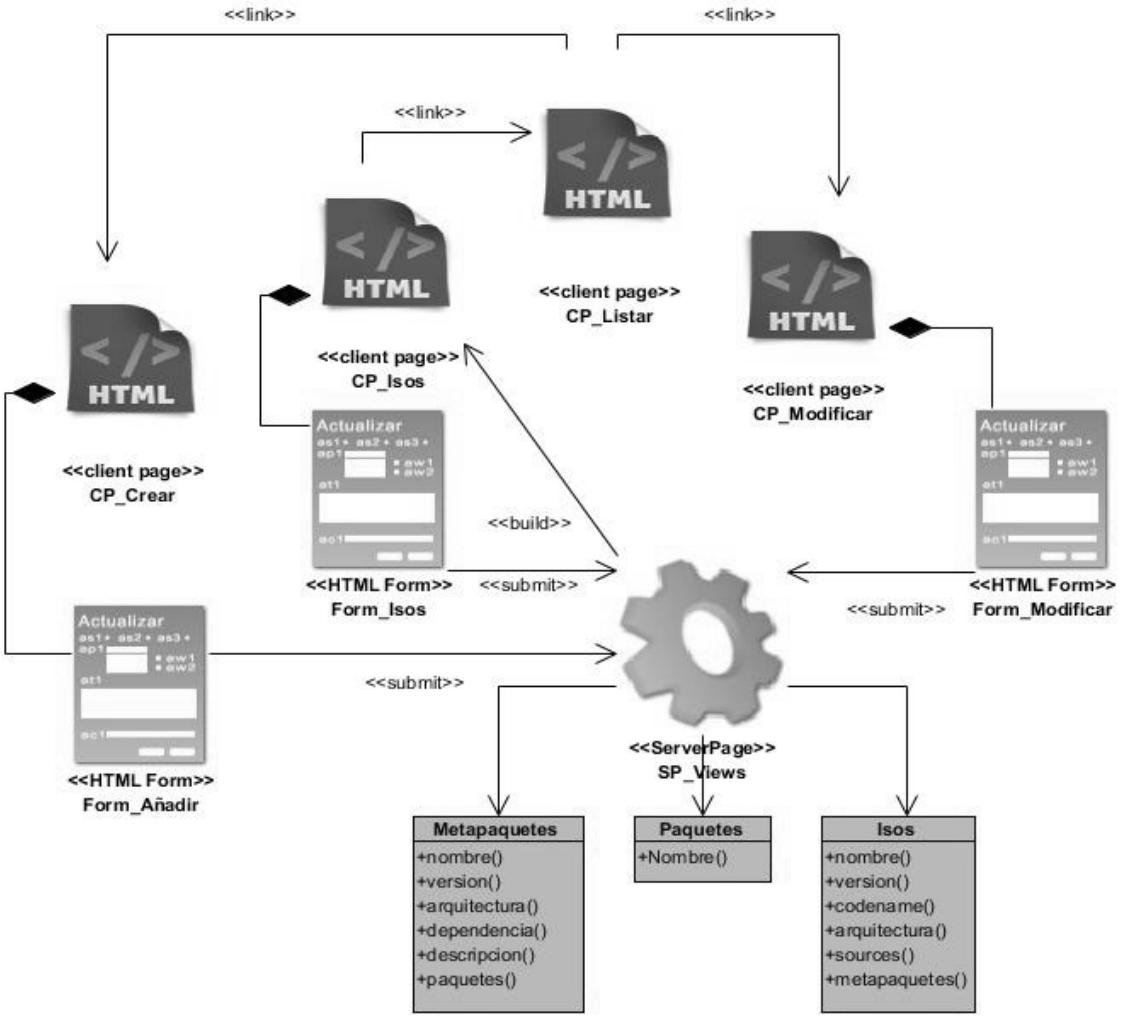


Figura 7: Diagrama de clases del diseño de la solución propuesta.

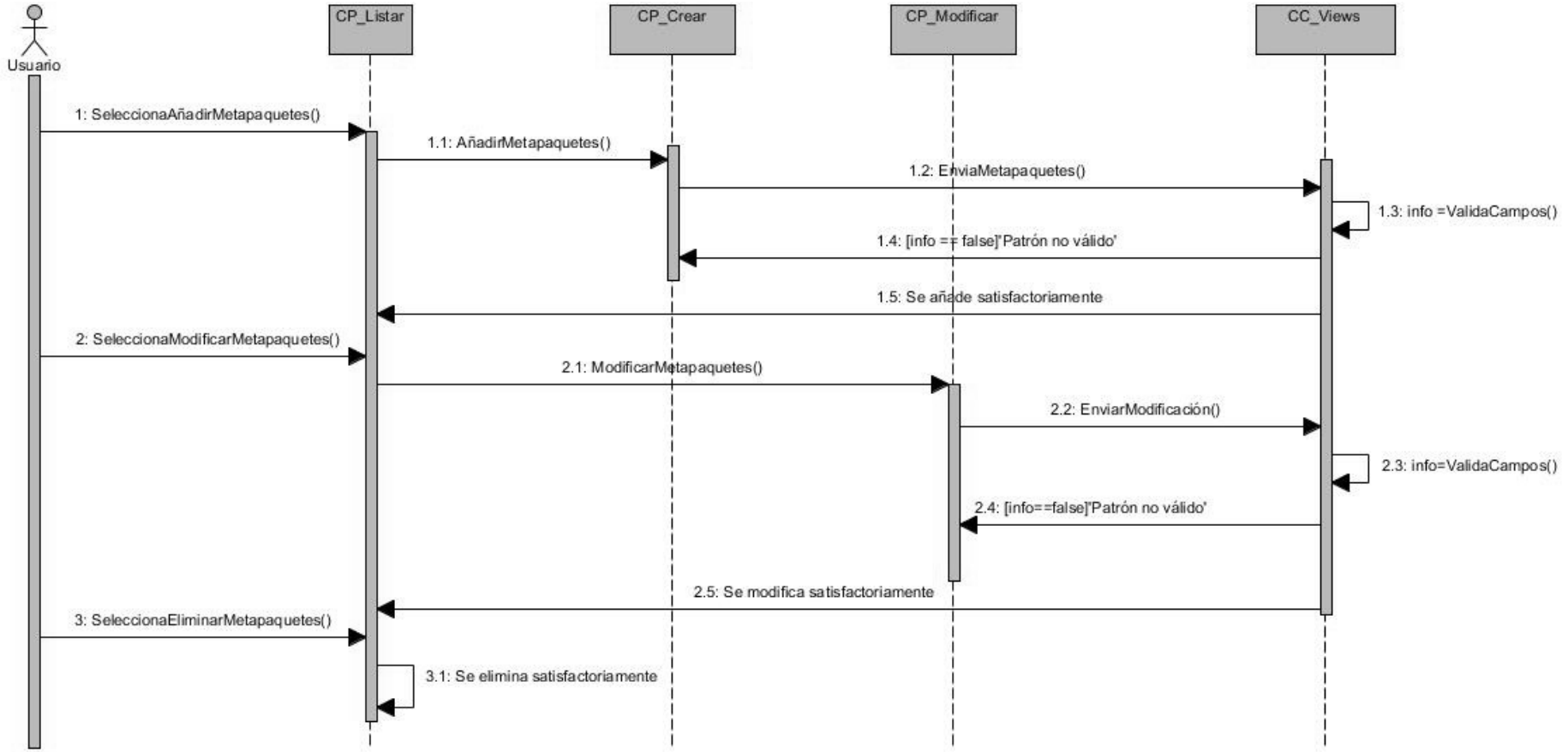


Figura 8: Diagrama de secuencia: Gestión de metapaquetes.

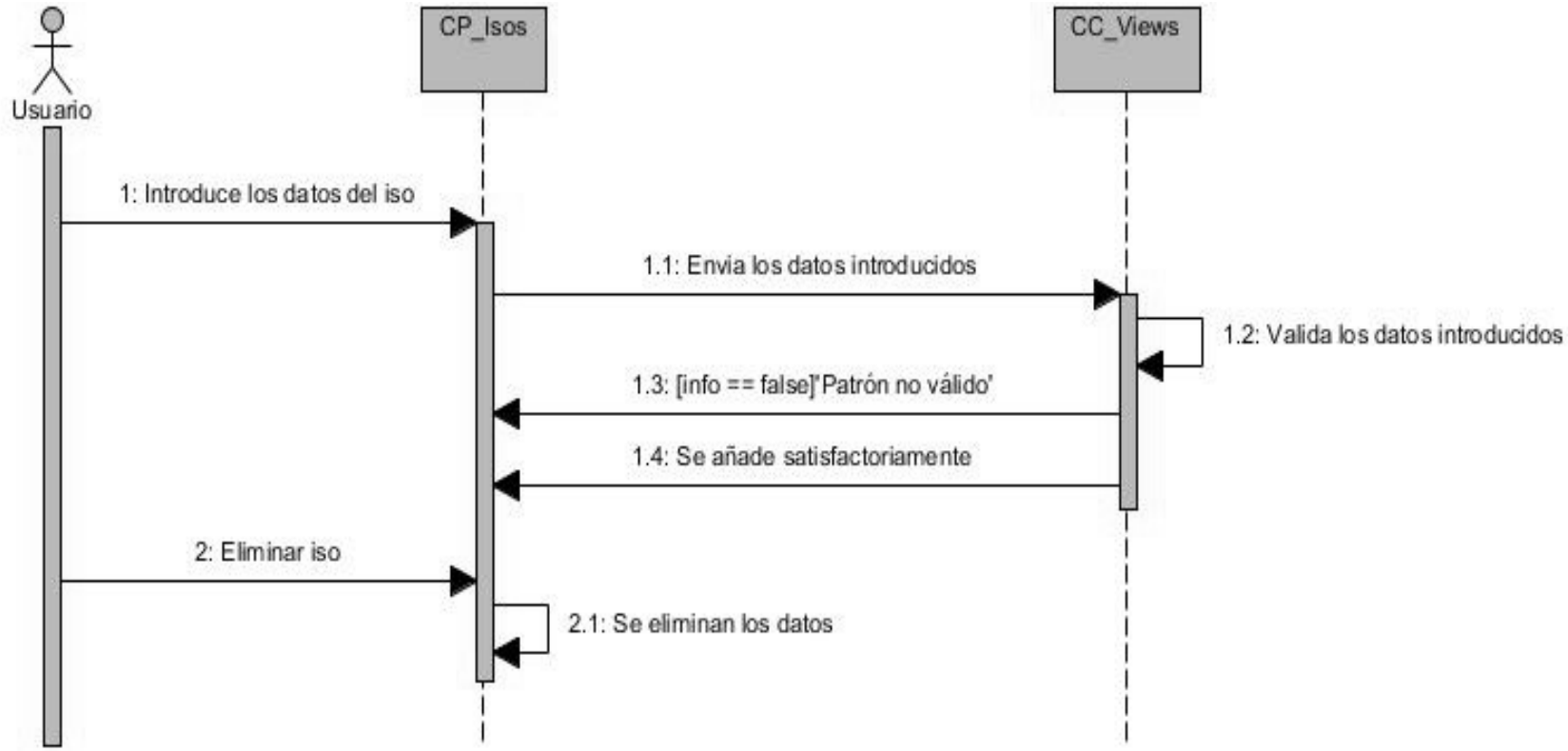


Figura 9: Diagrama de secuencia: Generación de ISOS-9660.

2.10 Modelo de despliegue

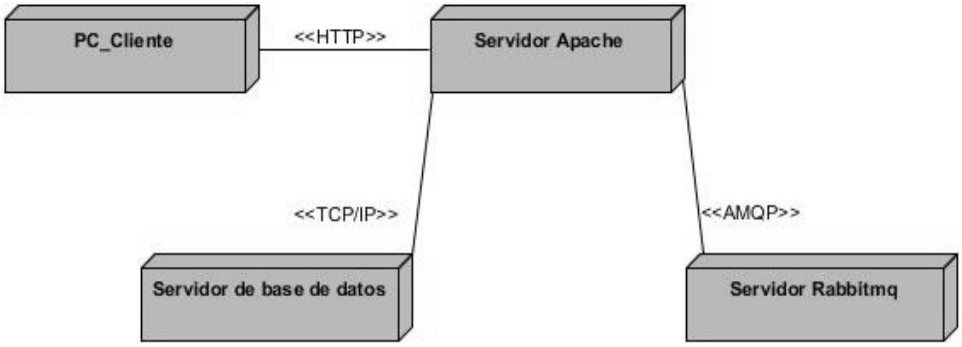


Figura 10: Diagrama de despliegue de la solución propuesta.

Para representar las interconexiones entre los nodos físicos de la solución propuesta y el software que los componen, se muestra el siguiente diagrama de despliegue.

PC_Cliente: Son las máquinas donde se encuentran los usuarios que acceden al servidor Apache.

Servidor Apache: Es el servidor donde se encuentra la herramienta de Generación de productos que le brindara el servicio a los clientes.

Servidor de base de datos: Es donde se encuentra almacenada la base de datos que maneja e *framework* django.

Servidor Rabbitmq: Es un intermediario para la mensajería de código abierto.

Conclusiones parciales

- Al definir la especificación de los requisitos funcionales y no funcionales del sistema, se obtiene una mejor comprensión que sirvió de guía para la implementación del sistema.
- Con la realización del Diagrama de Casos de Uso del Sistema, el Diagrama de Clases del Diseño y los Diagramas de Interacción (colaboración) se logró una mejor comprensión de las funcionalidades del sistema.
- Con la realización de los principales artefactos que rigen la metodología OpenUp para las fases de elaboración y construcción, se permitió un mayor control y entendimiento del desarrollo de la solución propuesta.

Capítulo 3: Implementación y Validación de la Solución Propuesta

La implementación del sistema es una de las fases imprescindibles dentro del proceso de desarrollo de software. En el presente capítulo se describe el diseño de las pruebas realizadas al sistema de Gestión de productos de la Plataforma de desarrollo e Integración Continua de Nova.

3.1 Modelo de componentes que integran la herramienta informática

El modelo de componentes representa la forma en que es estructurado un sistema informático atendiendo a las diferentes partes que lo componen. Cada componente debe ser tratado como una unidad de composición independiente e indispensable dentro de un sistema, y que puede contraer relaciones de dependencia con otros componentes. Algunos ejemplos de componentes físicos lo constituyen los archivos, módulos, librerías, ejecutables, binarios, entre otros [24].

3.2 Diagrama de componentes

En la Figura 11 se muestra el diagrama de componentes que permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Además muestra la organización y las dependencias entre un conjunto de componentes.

A continuación se describe el paquete principal que contiene el diagrama de componentes correspondiente a la herramienta de Generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova:

- **GPNova**: Agrupa en su interior todos los componentes, estableciendo una estructura organizativa

acorde al patrón de arquitectura MVT.

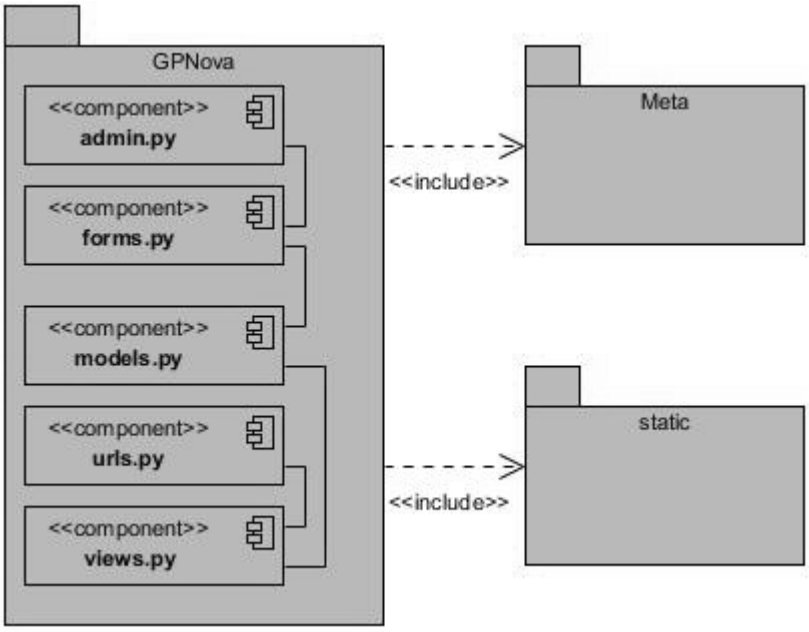


Figura 11: Diagrama de componentes.

3.3 Resultados obtenidos

La solución informática obtenida posee un conjunto de funcionalidades encaminadas al ahorro en tiempo y esfuerzo de los desarrolladores que actualmente crean las distribuciones del sistema operativo de forma manual. A continuación se describe como funciona el flujo de información en la solución propuesta.

El sistema de Gestión de productos para la Plataforma de Desarrollo e Integración Continua de Nova está basado en el patrón modelo-plantilla-vista. El flujo de información funciona de la siguiente forma: Primeramente el usuario introduce los datos en el navegador web enviando la solicitud que es recibida por

la clase vista que a su vez interactúa con el modelo para obtener los datos del Repositorio. Una vez obtenido los datos la clase vista interactúa con la plantilla mostrando, esta última, la respuesta a la petición realizada por el usuario.

La solución informática obtenida provee un conjunto de funcionalidades, a continuación se describen los casos de uso implementados:

Gestionar metapaquetes: Proceso en el cual se permite al usuario poder crear, modificar y eliminar los metapaquetes para la creación del producto final.

Gestionar Isos: Proceso en el cual se permite al usuario poder crear y eliminar las imágenes de instalación de la distribución GNU/Linux Nova.

3.4 Pruebas de Unidad

Las pruebas unitarias representan la verificación realizada a pequeños fragmentos de código, con el objetivo de comprobar que cada procedimiento o clase funciona correctamente. De esta manera los desarrolladores pueden elaborar un código más confiable, donde se detectan fallos a medida que se programa, sin necesidad de esperar al final, lo que resulta menos engorroso y más eficiente. Para la implementación de la solución propuesta se realizan las pruebas unitarias mediante la herramienta Selenium IDE. De acuerdo a la metodología utilizada, todo este proceso se lleva a cabo a través del desarrollo y solo a las funcionalidades críticas. A continuación se ilustran algunos de los resultados obtenidos.

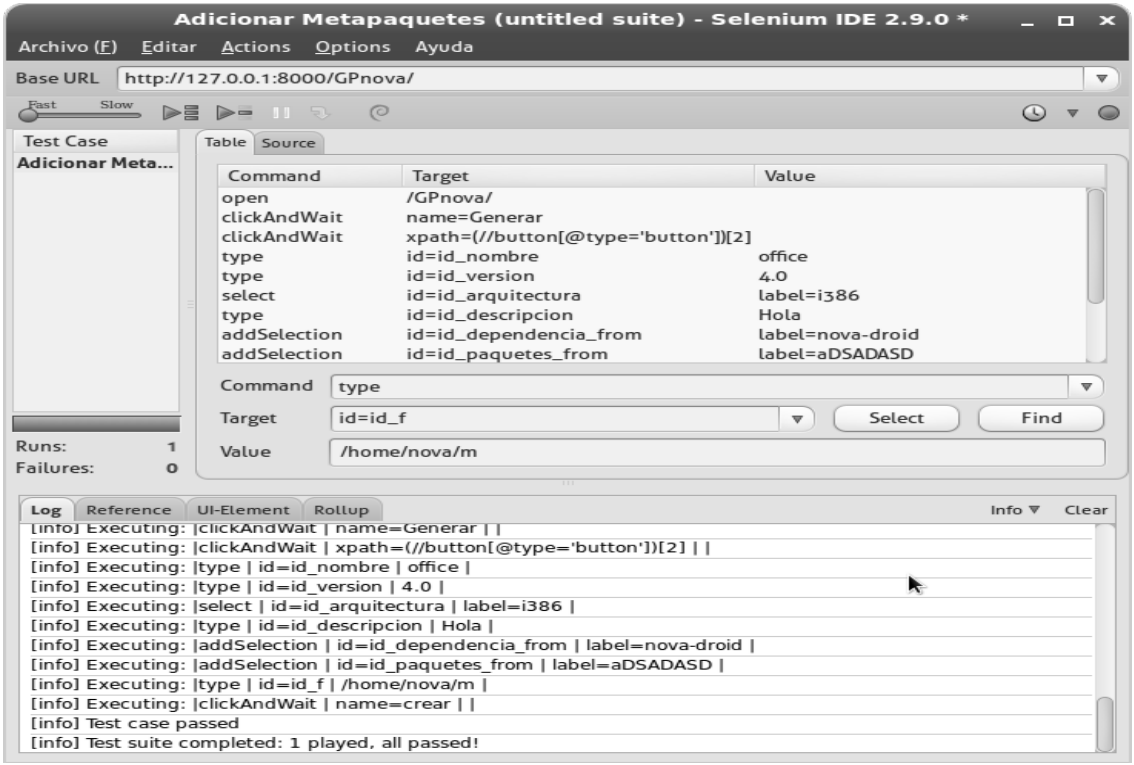


Figura 12: Prueba unitaria a la funcionalidad añadir metapaquetes.

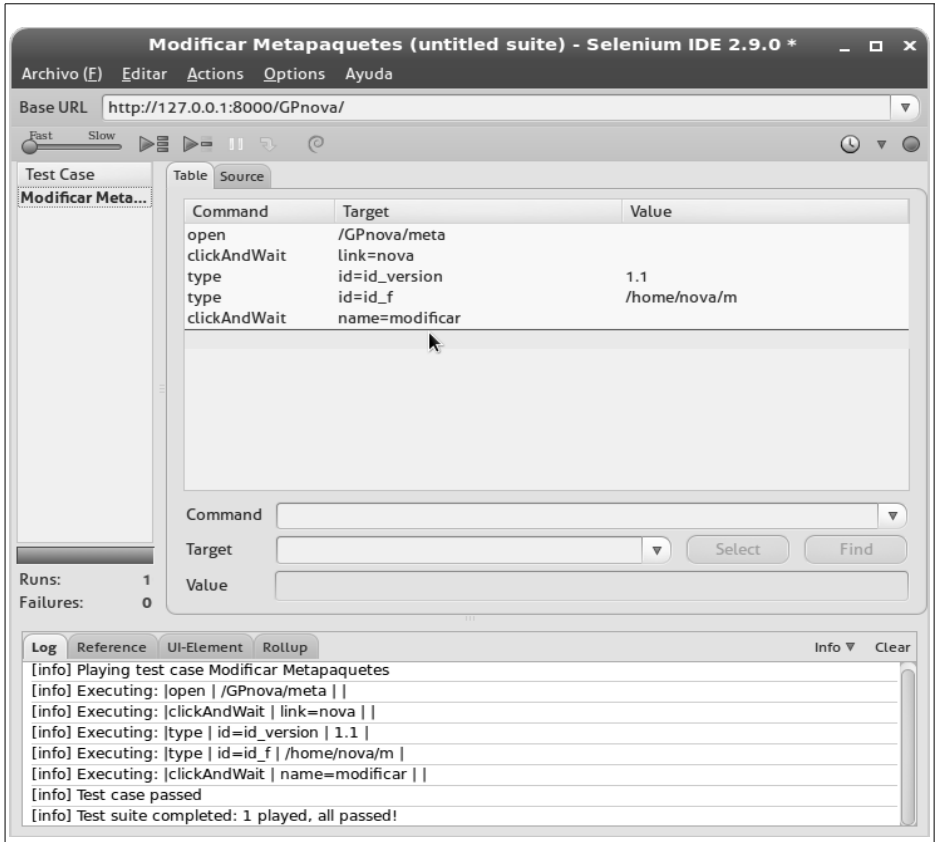


Figura 13: Pruebas unitarias a la funcionalidad modificar metapaquetes.

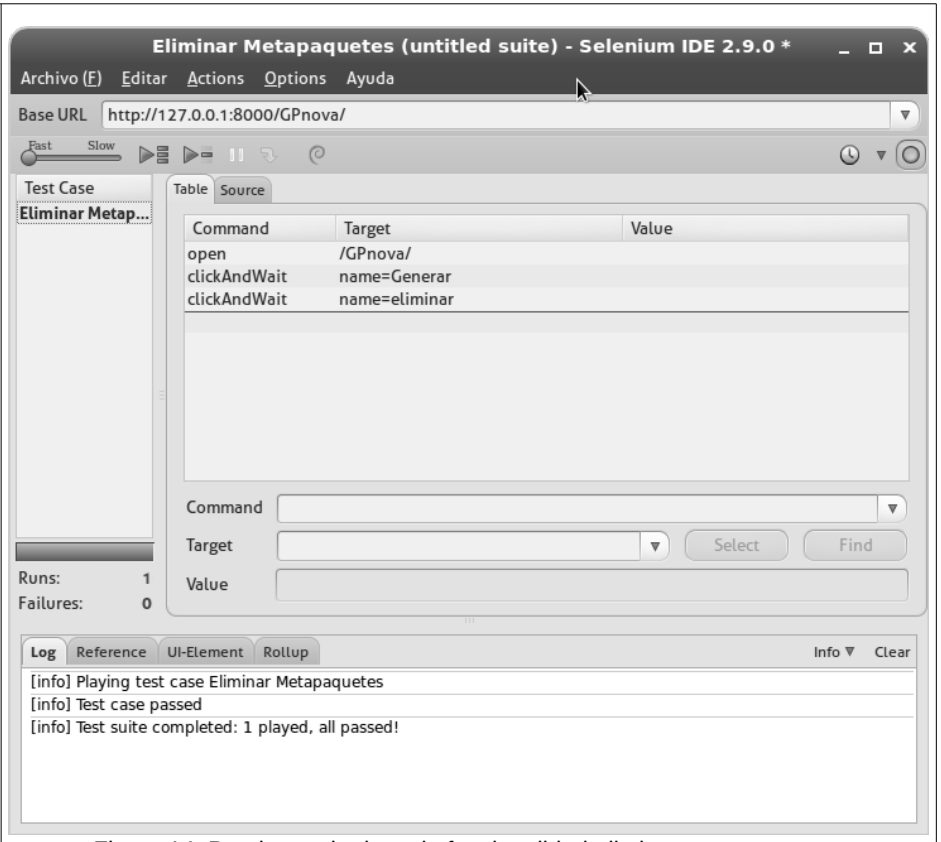


Figura 14: Pruebas unitarias a la funcionalidad eliminar metapaquetes.

Resultados de las pruebas de unidad

Para llevar a cabo la detección de las no conformidades presentes en la aplicación desarrollada, se realizaron las pruebas de unidad con la herramienta Selenium. En la Tabla 5 se muestran los resultados obtenidos a cada funcionalidad de prueba a la herramienta de Generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova.

Tabla 5: Resultados de las pruebas de unidad.

Funcionalidades	No conformidades
Añadir metapaquete	No se detectan errores
Modificar metapaquete	No se detectan errores
Eliminar metapaquete	No se detectan errores

3.5 Pruebas funcionales

Las pruebas funcionales son aquellas que se aplican a un software determinado, con el objetivo de validar que las funcionalidades implementadas estén de acuerdo a las especificaciones de los requisitos definidos con anterioridad. Para la ejecución de este tipo de pruebas, suelen emplearse dos métodos fundamentales: el método de Caja Blanca y el método de Caja Negra. El primero se centra en las pruebas al código de las aplicaciones; mientras que el segundo permite a los probadores enfocar su atención en el funcionamiento de la interfaz, a través del análisis de los datos de entrada y los de salida.

En este epígrafe se exponen los aspectos concernientes a las pruebas funcionales realizadas utilizando el método de Caja Negra. Se realizaron 5 casos de prueba basados en casos de uso, a continuación se muestra en las Tablas 6 y 7 un fragmento del diseño del caso de prueba “Añadir metapaquete” basado en caso de uso Gestionar metapaquetes. El resto de los casos de prueba se pueden encontrar en el Anexo 1.

Tabla 6: Variables empleadas en el diseño del caso de prueba “Adicionar metapaquete”.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	NO	Se admiten solo caracteres de texto.
2	Versión	Campo numérico	NO	Se admiten solo caracteres numéricos.
3	Arquitectura	Campo alfanumérico	NO	Se admiten solo caracteres alfanuméricos.
4	Descripción metapaquete	Campo de texto	NO	Se admiten solo caracteres de texto.
5	Dependencia	Campo de selección	SI	Se seleccionan las dependencias.
6	Paquete	Campo de selección	SI	Se seleccionan los paquetes.
7	Examinar	Campo de selección	NO	Se selecciona el archivo de configuración.

Tabla 7: Fragmento del caso de prueba “Adicionar metapaquete”.

Escenario	Descripción	Nombre	Versión	Arquitectura	Descripción metapaquetes	Dependencias	Paquetes	Examinar	Respuesta del sistema	Flujo central
EC 1.1 Adicionar metapaquete correctamente.	El usuario introduce la información de forma correcta.	V	V	V	V	V	V	V	El sistema adiciona el metapaquete creado.	1. El usuario selecciona la opción Adicionar. 2. El sistema activa un formulario para introducir la información. 3. El usuario registra los datos y presiona el botón crear.
		Nova-meta	2.3	i386	Distribución de GNU/Linux Nova	Nova-base	Nova-droid	D:\Estudio\Tesis\Documento		
EC 1.2 Adicionar metapaquete incorrectamente.	El usuario introduce la información de forma incorrecta.	V	I	V	V	V	V	V	El sistema no adiciona el metapaquete creado y muestra un mensaje de error en el campo versión.	
		office	eeee	amd64	Paquete de Office	Nova-meta	Nova-droid	D:\Estudio\Tesis\Documento		
EC 1.3 Adicionar metapaquete dejando campos vacíos.	El usuario introduce la información dejando campos vacíos.	I	I	V	V	V	V	I	El sistema no adiciona el metapaquete creado y muestra un mensaje de error en los campos vacíos.	
		Vacío	Vacío	i386	Sistema de gestión	Nova-meta	Nova-droid	Vacío		

Resultados de las pruebas funcionales

Para llevar a cabo la detección de las no conformidades presentes en la aplicación desarrollada, se realizaron 3 iteraciones de pruebas funcionales. En la Tabla 8 se muestran los resultados obtenidos en cada iteración de prueba a la herramienta de Generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova, así como la corrección de cada una de las no conformidades encontradas.

Tabla 8: Resultado de las pruebas funcionales.

No conformidades	Iteración 1	Iteración 2	Iteración 3
Detectadas	8	12	6
Resueltas	8	12	6
Pendientes	0	0	0

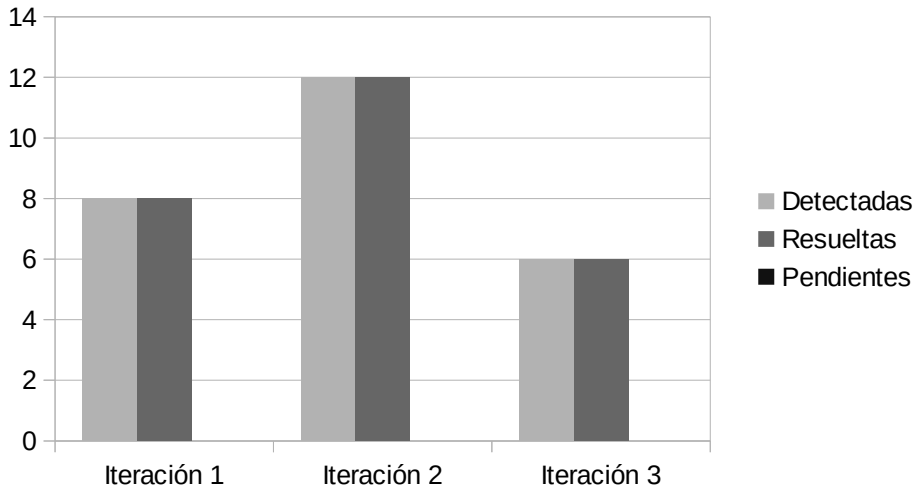


Figura 15: Iteraciones realizadas a la aplicación.

3.6 Pruebas de aceptación

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcional pactada previamente con el cliente.

La prueba de aceptación de la herramienta de Gestión de productos para la Plataforma de Desarrollo e Integración Continua de Nova se encuentra en el Anexo 2.

Conclusiones parciales

- Las pruebas unitarias permitieron la obtención de un código legible y optimizado durante toda la etapa de desarrollo.
- Las pruebas funcionales realizadas permitieron detectar y corregir fallos en las interfaces, garantizando una correcta validación de los datos que se introducen en el sistema.
- La prueba de aceptación permitió la aprobación del producto por parte del cliente.

Conclusiones Generales

- Se desarrolló un sistema de Generación de productos en la Plataforma de Desarrollo e Integración Continua de Nova, para lograr la integración de todo el proceso de creación de la distribución cubana GNU/Linux Nova.

- Para lograr la aceptación de la aplicación por el cliente se realizaron las pruebas de *software*, documentándose las mismas y sus resultados, lográndose un producto probado y aceptado por el cliente dando fe del cumplimiento de sus exigencias.

Recomendaciones

- Utilizar el sistema de Gestión de productos de la Plataforma de Desarrollo e Integración Continua para contribuir al desarrollo en comunidad de la distribución cubana de GNU/Linux Nova.
- Considerar las nuevas actualizaciones de las tecnologías utilizadas para la prevención de posibles huecos de seguridad.

Bibliografía

1. Guillem Alsina González. Auge de teléfonos móviles que utilizan GNU/Linux | Empresa y economía. *Empresas & Economia* [online]. 11 August 2007. [Accessed 4 December 2014]. Available from: http://www.empresayeconomia.es/aplicaciones-para-empresas/auge-de-telefonos-moviles-que-utilizan-gnulinux.html
2. Richard Stallman. ¿Qué es el software libre? - Proyecto GNU - Free Software Foundation. *Software libre* [online]. 13 November 2014. [Accessed 4 December 2014]. Available from: https://www.gnu.org/philosophy/free-sw.es.html
3. PÉREZ VILLASÓN, Yoandy, et al. Buenas prácticas para la migración a código abierto. Cuba, UCI, 2014.
4. Pierra, Fuentes Allan. Nova distribución Cubana de GNU/LINUX, Tesis (Maestría en informática aplicada). La Habana, Cuba: Universidad de la Ciencias Informáticas, 2013. pp. 29.
5. Richard Stallman. Acerca del proyecto GNU - Proyecto GNU - Free Software Foundation. *El sistema operativo GNU* [online]. 30 May 2014. [Accessed 4 December 2014]. Available from: http://www.gnu.org/gnu/the-gnu-project.html.
6. Debian -- El sistema operativo universal. [online]. 7 October 2014. [Accessed 4 December 2014]. Available from: https://www.debian.org/distrib/packages.
7. Ubuntu Comunity. Instalar software – doc.ubuntu-es. [En Linea]. [Accedido 1 Enero 2015]. Disponible en: http://doc.ubuntu-es.org/Instalar_software
8. Linux ISO Image Downloads | Linuxlookup. [online]. [Accessed 10 June 2015]. Available from: http://linuxlookup.com/linux_iso

9. Glossary - Community Help Wiki. [online]. 29 March 2015. [Accessed 4 June 2015]. Available from: https://help.ubuntu.com/community/Glossary.

10. «Remastersys - EcuRed», *EcuRed*. [En línea]. [Accedido: 16-feb-2015]. Disponible en: http://www.ecured.cu/index.php/Remastersys.

11. klichota balliano, «Ubuntu Customization Kit | SourceForge.net», *Ubuntu Customization Kit*, 2015. [En línea]. [Accedido: 16-feb-2015]. Disponible en: http://sourceforge.net/projects/uck/.

12. «Reconstructor: crea un LiveCD a tu medida » MuyLinux», *MuyLinux*, 21-jul-2010. [En línea]. [Accedido: 12-feb-2015]. Disponible en: http://www.muylinux.com/2010/07/21/reconstructor-crea-un-livecd-a-tu-medida.

13. «Adding Software», *Suse Studio Help*, 2013. [En línea]. [Accedido: 16-feb-2015]. Disponible en: http://susestudio.com/help/quickstart/adding-software.html.

14. «SUSE Studio a fondo: crea tu distro a medida, y pruébala», Software y Web, 2010. [En línea]. [Accedido: 23-feb-2015]. Disponible en: http://www.genbeta.com/linux/suse-studio-a-fondo-crea-tu-distro-a-medida-y-pruebala.

15. genisoimage(1) - Linux man page. [online]. [Accessed 14 May 2015]. Available from: http://linux.die.net/man/1/genisoimage

16. Metodologias Agiles vs Pesadas, MSF, MOF, ITIL: Metodologia agiles vs pesadas. [online]. [Accessed 11 June 2015]. Available from: http://metodologiaagilesvspesadas.blogspot.com/2012/05/metodologia-agiles-vs-pesadas.html.

17. Open UP: Metodología Open UP. [online]. [Accessed 11 June 2015]. Available from: http://openupeaojmp.blogspot.com/2013/09/metodologia-open-up.html

18. SANTIAGO SALGADO. Microsoft Word - Articulo Tecnico - Santiago Rios .docx - AC-SISTEMAS-

ESPE-047042.pdf. [online]. [Accessed 17 May 2015]. Available from: http://repositorio.espe.edu.ec/bitstream/21000/6316/1/AC-SISTEMAS-ESPE-047042.pdf

19. Herramienta CASE - EcuRed. [online]. [Accessed 11 June 2015]. Available from: http://www.ecured.cu/index.php/Herramienta_CASE.

20. Gonzales, Duque Raúl. Python para todos. España , Mayo del 2007.

21. «Getting Started with Eclipse». [En línea]. [Accedido: 05-may-2015]. Disponible en: https://eclipse.org/users/.

22. Holovaty, Adrian y Kaplan Jacob. The Definitive Guide to Django: Web Development Done Right. Apress, Diciembre de 2007.

23. SeleniumHQ/selenium · GitHub. [online]. [Accessed 4 June 2015]. Available from: https://github.com/seleniumhq/selenium.

24. SOMMERVILLE, I. *INEGINERÍA DE SOFTWARE.* Madrid: Pearson Educación S.A, 2005. 84-7829-074-5.

25. SELECTING A DEVELOPMENT APPROACH. Revalidated: March 27, 2008. Retrived 27 Oct 2008.

Anexo1

Caso de prueba

Tabla 9: Fragmento del caso de prueba “Modificar metapaquetes”.

Escenario	Descripción	Nombre	Versión	Arquitectura	Descripción metapaquetes	Dependencias	Paquetes	Examinar	Respuesta del sistema	Flujo central
EC 1.1 Modificar metapaquete correctamente.	El usuario introduce la información de forma correcta.	V	V	V	V	V	V	V	El sistema modifica el metapaquete creado.	1. El usuario selecciona la opción Modificar. 2. El sistema activa un formulario para introducir la información. 3. El usuario registra los datos y presiona el botón modificar.
		Nova-meta	2.3	i386	Distribución de GNU/Linux Nova	Nova-base	Nova-droid	D:\Estudio\Tesis\Documento		
EC 1.2 Modificar metapaquete incorrectamente.	El usuario introduce la información de forma incorrecta.	V	I	V	V	V	V	V	El sistema no modifica el metapaquete creado y muestra un mensaje de error en el campo versión.	
		office	eeee	amd64	Paquete de Office	Nova-meta	Nova-droid	D:\Estudio\Tesis\Documento		
EC 1.3 Modificar metapaquete dejando campos vacíos.	El usuario introduce la información dejando campos vacíos.	I	I	V	V	V	V	I	El sistema no modifica el metapaquete creado y muestra un mensaje de error en los campos vacíos.	
		Vacío	Vacío	i386	Sistema de gestión	Nova-meta	Nova-droid	Vacío		

Tabla 10: Fragmento del caso de prueba “Eliminar metapaquetes”.

Escenario	Descripción	Flujo central
EC 1.1 Seleccionar la opción eliminar.	El usuario selecciona el botón eliminar y el metapaquete se elimina satisfactoriamente.	1. El usuario selecciona el botón eliminar. 2. El sistema elimina el metapaquete.

Tabla 11: Fragmento del caso de prueba “Adicionar ISO-9660”.

Escenario	Descripción	Nombre	Versión	Codename	Arquitectura	Sources	Metapaquetes	Respuesta del sistema	Flujo central
EC 1.1 Adicionar Iso correctamente.	El usuario introduce los datos correctamente.	V	V	V	V	V	V	El sistema adiciona el Iso creado correctamente.	El usuario introduce los datos en el formulario para crear un Iso y finalmente presiona el botón Generar Iso.
		Nova	2.2	2013	i386	http://nova.f10.uci.cu	Nova-kiosko Nova-base		
EC 1.2 Adicionar Iso incorrectamente.	El usuario introduce los datos incorrectamente.	V	I	V	V	V	V	El sistema lanza un mensaje de error y no crea el Iso.	
		Server	eeee	2013	i386	http://nova.f10.uci.cu	Nova-kiosko Nova-base		
EC 1.3 Adicionar Iso dejando datos vacíos.	El usuario introduce los datos dejando campos vacíos.	V	V	V	I	V	V	El sistema lanza un mensaje de error y no crea el Iso.	
		Server	2.2	2013	Vacío	http://nova.f10.uci.cu	Nova-kiosko Nova-base		

Tabla 12: Fragmento del caso de prueba “Eliminar ISOS-9660”.

Escenario	Descripción	Flujo central
EC 1.1 Seleccionar la opción eliminar.	El usuario selecciona el botón eliminar y el ISO-9660 se elimina satisfactoriamente.	1. El usuario selecciona el botón eliminar. 2. El sistema elimina el ISO-9660.

Tabla 13: Variables empleadas en el diseño del caso de prueba “Modificar metapaquetes”.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	NO	Solo caracteres de texto.
2	Versión	Campo numérico	NO	Solo caracteres numéricos.
3	Arquitectura	Campo alfanumérico	NO	Solo caracteres alfanuméricos.
4	Descripción metapaquete	Campo de texto	NO	Solo caracteres de texto.
5	Dependencia	Campo de selección	SI	Se seleccionan las dependencias.
6	Paquete	Campo de selección	SI	Se seleccionan los paquetes.

Tabla 14: Variables empleadas en el diseño del caso de prueba “Adicionar ISO-9660”.

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	NO	Solo caracteres de texto
2	Versión	Campo alfanumérico	NO	Solo caracteres alfanuméricos
3	Codename	Campo numérico	NO	Solo caracteres numéricos
4	Arquitectura	Campo texto	NO	Solo caracteres de texto
5	Sources	Url	NO	Solo dirección Url
6	Metapaquetes	Seleccionar	NO	Se seleccionan los metapaquetes

Anexo2

La Habana, 25 Mayo 2015

La Herramienta de Gestión de Productos para la Plataforma de Desarrollo e Integración Continua de Nova en su versión 1,0 cumple con los requerimientos establecidos para su desarrollo.

La presente versión:

1. Se integra como una aplicación de la Plataforma de Desarrollo e Integración Continua de Nova.
2. Genera mas de una imagen de instalación ISO-9660 de forma simultanea.
3. Gestiona metapaquetes que luego serán utilizados para la generación de los isos.
4. Permite la personalización y construcción de imágenes de instalación ISO-9660 a partir, totalmente, de un repositorio.
5. Reutiliza elementos de la Plataforma como es la librería celery para la realización de tareas asíncronas.

Ing. Hector Pérez Baranda

Ing. Dairelis García Rivas