Universidad de las Ciencias Informáticas
Facultad 1

Herramienta web para la creación de personalizaciones de Nova Servidores

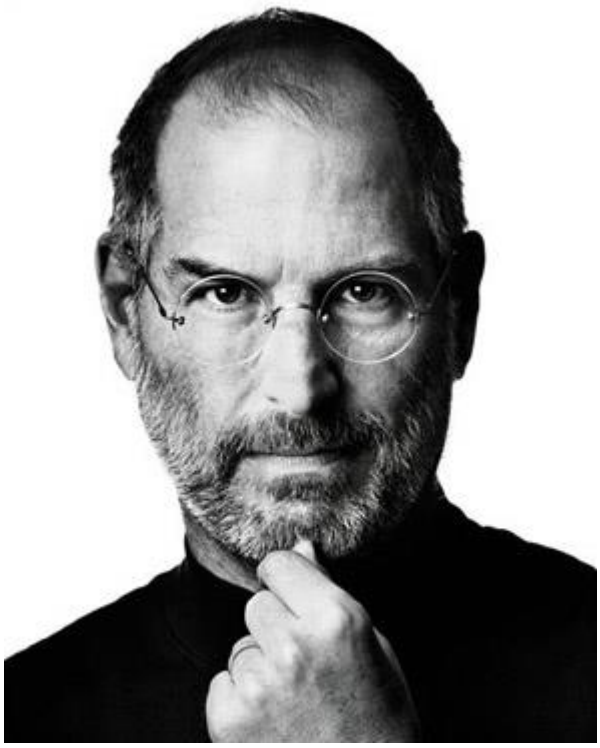
Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Javier Piñeiro Cárdenas

Tutores: Ing. Ivaniet Díaz Romeu

Ing. Gustavo Quezada Arévalo

La Habana, Junio del 2017



"Tu trabajo va a llenar gran parte de tu vida, y la única forma de estar realmente satisfecho con él es hacer lo que creas que es un gran trabajo. Y la única manera de hacer un trabajo genial es amar lo que haces. Si no lo has encontrado, sigue buscando..."

STEVE JOBS

Declaración de autoría

Declaro por este medio que yo Javier Piñeiro Cárdenas, con carné de identidad 93011411962 soy el autor principal del trabajo titulado Herramienta web para la creación de personalizaciones de Nova Servidores y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firman la presente a los 16 días del mes de junio del año 2017.

_____

Javier Piñeiro Cárdenas

Autor

_____ _____

Ing. Ivaniet Díaz Romeu

Tutor

Ing. Gustavo Quezada Arévalo

Tutor

Datos de contacto

Ing. Ivaniet Díaz Romeu
Universidad de las Ciencias Informáticas, La Habana, Cuba
correo: ivaniet@uci.cu

Ing. Gustavo Quezada Arévalo
Universidad de las Ciencias Informáticas, La Habana, Cuba
correo: gquezada@uci.cu

Agradecimientos:

A **mis padres** porque sin ellos nada de esto hubiese sido posible, gracias por ser los padres que siempre han sido, a **mi hermano** por ser el mejor hermano que existe, a mi abuela **Chicha,** la viejita que más quiero en este mundo. A mi tía **América** por siempre estar ahí para mí y ayudarme en todo. A mis tías **Marlen** y **Silvia** por estar pendientes de mí y mis estudios. A **mis primos** y **toda mi familia** en general por el cariño y el apoyo, los quiero mucho.

A **Nailee** por ser la hermanita hembra que nunca he tenido, sabes que siempre voy a estar aquí para ti.

A **Aidee** por ser esa persona que siempre ha estado ahí en las buenas y en las malas durante estos 5 años, a su mamá **Nena** y su abuela **Gertrudis**, por tratarme como si fuera de la familia, muchas gracias por todo.

A **Stephany** y **su familia** por acogerme en su casa y preocuparse por mí.

A **Claudia** porque a pesar de la distancia nunca hemos perdido la comunicación.

A **Rachel** por ser una gran amiga en todo momento.

A **Dareyna**, **Zulemys, Ivet y Liset** por ser mis amigas y aguantarme todo este tiempo que no es tarea fácil.

A **Humberto, Elvis, Yunier, Yosbel, Nelson** por ser mis amigos y compañeros de estudio, dominó y fiestas.

A **Lexy**, **Roberto** y **Edel** las personas más responsables del grupo, de los que aprendí muchas cosas en estos 5 años.

A ese trio de **Raisa, Lisbet y Betsy** por tantos buenos momentos que pasamos juntos.

A **Daynelis,** gracias por tus consejos que me ayudaron mucho.

A **Danita** que a pesar del poco tiempo que no conocemos siempre está pendiente de mí.

A mis tutores **Ivaniet y Gustavo** por el apoyo, la ayuda y sobre todo la paciencia que tuvieron conmigo.

A **todos los profesores** que me prepararon durante estos 5 años y a todas las personas que de una forma u otra contribuyeron a que llegara este día, a todos muchas gracias.

Dedicatoria:

A **mis padres** por el apoyo, el amor y el cariño que siempre me han dado.

Resumen

GNU/Linux es un sistema operativo tipo Unix, de código abierto, con grandes capacidades para el procesamiento y la administración de los recursos, que mantiene la seguridad como pilar importante. En la actualidad muchos países principalmente de América Latina están optando por esta alternativa, entre ellos se encuentra Cuba que está inmerso en un proceso de migración a software libre donde la distribución cubana de GNU/Linux Nova, es el sistema operativo seleccionado para ser desplegado en los Organismos de Administración Central del Estado. Este sistema operativo cuenta con una versión para servidores a la que frecuentemente se le realizan personalizaciones que consumen mucho tiempo debido a que se desarrolla de forma manual. El objetivo del presente trabajo de diploma es desarrollar una herramienta web que permita agilizar la construcción de personalizaciones de Nova Servidores en el proceso de desarrollo de soluciones a la medida. Para guiar el proceso de construcción de la propuesta de solución se utilizó la metodología de desarrollo de software Variación de AUP para la universidad y en la implementación se emplearon tecnologías y herramientas de software libre. La evaluación de la propuesta de solución se realizó a partir de la aplicación de técnicas, métricas y pruebas que garantizan el correcto funcionamiento de la aplicación y demostraron la satisfacción del cliente hacia el sistema desarrollado. Al culminar la presente investigación se obtuvo una herramienta web que permite la construcción de personalizaciones de Nova Servidores.

Palabras claves: GNU/Linux, herramienta, personalizaciones, sistema operativo.

Índice

Índice de figuras

Índice de tablas

Introducción

Un sistema operativo es el software básico de una computadora que provee una interfaz entre el resto de los programas del ordenador, los dispositivos hardware y el usuario. Sus funciones básicas son administrar los recursos y las tareas de la máquina, coordinar el hardware y organizar archivos y directorios en dispositivos de almacenamiento (Carmona, 2012). La parte más importante de un sistema operativo es su núcleo. En un sistema GNU/Linux, Linux es el núcleo, el resto del sistema consiste en otros programas, algunos de los cuales fueron escritos por o para el proyecto GNU[1], este en sí mismo no forma un sistema operativo funcional, se prefiere utilizar el término "GNU/Linux" para referirse a los sistemas que la mayor parte de las personas llaman de manera informal "Linux" (Debian, 2016).

GNU/Linux es un sistema operativo tipo Unix, diseñado para aprovechar al máximo las capacidades de las computadoras basadas en el microprocesador i386 y posteriores. Es un sistema con capacidades de multiprocesamiento, multitarea y multiusuario. Garantiza la protección de la memoria entre procesos, carga de ejecutables por demanda, utiliza memoria virtual de paginación (sin intercambio de procesos completos) y control de tareas POSIX (*Portable Operating System Interface*, Interfaz Portable de Sistema Operativo) (Linux-es, 2015).

Este sistema operativo se ha distinguido por su estabilidad de operación, su velocidad, su seguridad como pilar importante y su capacidad para la administración eficiente de los recursos de la computadora como la memoria, poder de la unidad central de procesamiento y espacio en disco. Todas estas características lo han llevado a diferenciarse del resto de los sistemas operativos del mundo, pero sin dudas, la más interesante de todas es que se compone de código abierto, es decir, su desarrollo se hace abiertamente y el código puede ser modificado y distribuido libremente. Esto ha propiciado que en la actualidad las distribuciones GNU/Linux hayan alcanzado una significativa popularidad ampliando su uso a varias esferas de la sociedad, convirtiéndose en una alternativa a nivel mundial (Linux-es, 2015).

GNU/Linux no ha conseguido introducirse en los ordenadores domésticos y solo tiene el 2,31% de uso en la cuota mundial (Market Share Reports, 2016). Pero reina en móviles, empresas, centros de datos y en la infraestructura de la red. El 80% de las transacciones bursátiles se realizan en sus plataformas y 9 de

[1] GNU es un acrónimo recursivo que significa GNU No es Unix (*GNU is Not Unix*).

cada 10 superordenadores llevan el símbolo del pingüino en su interior, también ha expandido su uso a televisiones y automóviles (Reventós, 2012).

En América Latina varios países están optando por esta alternativa; el Gobierno de Venezuela estableció por decreto presidencial el uso preferente del Software Libre (SL) y de GNU/Linux en toda la administración pública, incluso en los ministerios y oficinas gubernamentales (Chávez, 2004). En este país se desarrolla la distribución de GNU/Linux, Canaima, usada en el proyecto Canaima Educativo y en la producción de computadoras de escritorio que se ofrecen a bajo precio (Gavasa, 2014). En el 2008 el gobierno de Ecuador mediante el Decreto 1014 estableció la obligatoriedad del uso de tecnologías de código abierto en entes públicos, este marco legal permitió el desarrollo de eCURUL un programa que permite realizar votaciones electrónicas y controla los debates gubernamentales en las sesiones de la Asamblea Nacional (Correa, 2008).

Uruguay aprobó en diciembre de 2013 la Ley de Software Libre y Formatos Abiertos en el Estado. Esta ley establece la obligación de uso de formatos abiertos y estándares en las instituciones gubernamentales, además prioriza el desarrollo del software libre para las soluciones informáticas (El Senado y la Cámara de Representantes de la República Oriental del Uruguay, 2014).

La República Dominicana a través de distintas ONG (Organización No Gubernamental) apoya y promueve el uso del SL en la educación y en el desarrollo científico. En Argentina existen iniciativas estatales para promover el uso del SL y en Chile el Ministerio de Educación y la Universidad de la Frontera crearon EduLinux, programa que se aplica en las escuelas chilenas y en el 90% de las bibliotecas (Gavasa, 2014).

Cuba en aras de ganar en soberanía tecnológica y socio-adaptabilidad, así como garantizar la informatización de todas las esferas de la sociedad está inmersa en un proceso de migración a software libre donde la distribución cubana de GNU/Linux Nova, es el sistema operativo seleccionado para ser desplegado en los Organismos de Administración Central del Estado (OACE). Este sistema operativo desarrollado por el proyecto Nova, perteneciente al Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas, cuenta con una versión para servidores a la que frecuentemente se le realizan personalizaciones con el objetivo de facilitar la adaptabilidad del sistema a las necesidades del cliente. Este engorroso proceso actualmente consume mucho tiempo ya que se desarrolla de forma manual, haciéndose necesario la participación de varios especialistas para realizar la tarea, los cuales

deben poseer amplios conocimientos sobre el tema debido a la cantidad y complejidad de los comandos que se utilizan en él. Todo esto trae consigo la pérdida oportunidades de trabajo para CESOL.

Dada la situación problemática anteriormente planteada, se define como **problema de la investigación:** ¿Cómo agilizar el proceso de construcción de personalizaciones de Nova Servidores en el desarrollo de soluciones a la medida? Se define como **objeto de estudio:** herramientas web de personalizaciones de distribuciones GNU/Linux, enmarcado en el **campo de acción:** herramientas web de personalizaciones para Nova Servidores.

El presente trabajo tiene como **objetivo general:** Desarrollar una herramienta web que permita agilizar la construcción de personalizaciones de Nova Servidores en el proceso de desarrollo de soluciones a la medida.

Para garantizar el cumplimiento del objetivo general, se ha desglosado en los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación acerca de las herramientas web de personalizaciones de distribuciones GNU/Linux.
2. Diseñar una herramienta web que permita la construcción de personalizaciones de Nova Servidores.
3. Implementar la herramienta web para la construcción de personalizaciones de Nova Servidores.
4. Evaluar la herramienta web para la construcción de personalizaciones de Nova Servidores mediante la aplicación de técnicas, métricas y pruebas.

Se definen como **preguntas científicas:**

1. ¿Cuáles son los presupuestos teóricos que fundamentan el proceso de personalización del sistema operativo Nova Servidores?
2. ¿Qué aspectos se deben tener en cuenta para diseñar una aplicación web para la personalización del sistema operativo Nova Servidores?
3. ¿Cuáles son las herramientas y tecnologías más adecuadas para implementar la aplicación web para la personalización del sistema operativo Nova Servidores?
4. ¿Qué técnicas, métricas y pruebas aplicar para la evaluación de la aplicación web de personalización del sistema operativo Nova Servidores?

Para el desarrollo de la investigación se utilizaron los siguientes **métodos de la investigación científica:**

Métodos teóricos:

- **Histórico lógico:** este método permitió conocer la evolución de las herramientas para las personalizaciones de software así como los elementos que las caracterizan.
- **Análisis y síntesis:** este método se utilizó para el estudio y análisis de diferentes fuentes bibliográficas con el objetivo de obtener un amplio conocimiento acerca las características y peculiaridades del proceso de creación de personalizaciones para distribuciones GNU/Linux.
- **Modelación:** se utilizó para modelar los diagramas de requisitos, análisis y diseño e implementación en la construcción de la herramienta para la creación de personalizaciones para Nova Servidores.

Métodos empíricos:

- **Entrevista:** este método se utilizó para conocer las características del proceso de desarrollo de personalizaciones de Nova Servidores en la actualidad (Ver anexo 1).
- **Observación**: este método se utilizó en el análisis del proceso de personalización de Nova Servidores. Para su aplicación se utilizó la guía de observación para el proceso de creación de personalizaciones de Nova Servidores (Ver anexo 2).

El presente trabajo de diploma está estructurado en: resumen, introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

Capítulo 1. Fundamentación teórica del proceso de personalización de distribuciones GNU/Linux: en este capítulo se presentan los conceptos fundamentales asociados al problema planteado, el estudio sobre las principales características y peculiaridades del procesos de personalización de distribuciones GNU/Linux, la metodología de desarrollo de software a utilizar y las distintas herramientas y tecnologías que serán empleadas en el desarrollo de la solución.

Capítulo 2. Análisis y diseño de la propuesta de solución: en este capítulo, se desarrolla el análisis y diseño de la propuesta de solución tomando como punto de partida las características del proceso de personalización y los elementos planteados por la metodología Variación de AUP para la UCI en el escenario 3. Se definen los requisitos del sistema y la arquitectura del mismo.

Capítulo 3. Implementación, pruebas y evaluación de la propuesta de solución: este capítulo se enfoca en la construcción del sistema a partir de los resultados del Análisis y Diseño. En él se elaboran los diagramas de componentes y de despliegue y se define los estándares de codificación a utilizar en la implementación de la solución. Además, se realizan pruebas de software con el objetivo de descubrir y corregir errores y se evalúa la propuesta de solución.

Capítulo 1: Fundamentación teórica del proceso de personalización de distribuciones GNU/Linux

El presente capítulo contiene los conceptos fundamentales asociados al problema planteado, el estudio sobre las principales características y peculiaridades del proceso de personalización de distribuciones GNU/Linux, así como la metodología de desarrollo de software a utilizar y las distintas herramientas y tecnologías que serán empleadas en el desarrollo de la solución.

1.1 Conceptos fundamentales

1.1.1 Software libre

Según la Fundación de Software Libre, el software es libre cuando sus usuarios gozan de las siguientes libertades (Ubuntu-es, 2016):

- La libertad de usar el programa, con cualquier propósito.
- La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a cada necesidad.
- La libertad de distribuir copias del programa, con lo cual se puede ayudar al prójimo.
- La libertad de mejorar el programa y compartir esas mejoras con los demás, de modo que toda la comunidad se beneficie.

1.1.2 Distribución GNU/Linux

El sitio web CodeGamers en su sección “Linux: conceptos básicos” plantea:

Una distribución es un conjunto de programas y un sistema de gestión de paquetes (archivos) que se montan sobre el *kernel*[2] de GNU/Linux. Dependiendo de la distribución el rendimiento del sistema operativo también puede variar debido a que algunas distribuciones compilan su propio núcleo a partir del original de GNU/Linux. Además, se diferencian en el entorno de escritorio, aunque la mayoría tienen una versión para cada entorno (CodeGamers, 2014).

Manuel Alejandro Sánchez en su tesis de grado plantea, una distribución de GNU/Linux puede ser descrita como una distribución de software basada en el núcleo Linux que incluye determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios. Las mismas pueden contener o no aplicaciones o controladores privativos (Sánchez, 2015).

[2] Es núcleo del sistema operativo, el cual se encarga de manejar los recursos del ordenador.

En la presente investigación se asume el concepto dado por Manuel Alejandro Sánchez sobre la distribución GNU/Linux.

1.1.3 Personalización de un sistema operativo

La Real Academia de la Lengua Española define la palabra personalizar como: “Dar carácter personal a algo”.

El sitio web Gesoft plantea que la personalización del software es el desarrollo de sistemas, aplicaciones, programas, sitios web, páginas web o cualquier software, de acuerdo con las especificaciones del cliente, es decir, se construye y forma a las necesidades únicas y gustos del cliente (Gesoft, 2015).

De lo anteriormente planteado el autor establece como personalización de un sistema operativo, la modificación del mismo a través de acciones realizadas sobre el conjunto de aplicaciones y configuraciones que lo conforman, con el objetivo de agregar características al sistema y adaptarlo a las necesidades y gustos de sus usuarios finales.

En el proceso de investigación se asumen los conceptos anteriores con el objetivo de conocer las características particulares del software libre, distribuciones y personalizaciones de un sistema operativo.

1.2 Herramientas de personalización de distribuciones

1.2.1 Instalinux

Instalinux (Ver anexo 3) es un servicio en línea que permite elegir entre varias distribuciones de GNU/Linux, configurarlas y personalizarlas. Para ello posee un asistente en inglés, en el que se ingresan los datos de configuración de la instalación y al finalizar se crea la imagen ISO[3] para descargar según las especificaciones del usuario. Instalinux cuenta con dos versiones, una simple para la mayoría de usuarios y una avanzada para usuarios más experimentados. Permite elegir entre siete distribuciones de Linux como base: CentOS, Debian, Fedora, Mint, OpenSUSE, Scientific y Ubuntu. De cada una, dispone la versión más reciente y algunas anteriores, opción muy útil para aprovechar equipos antiguos de bajas prestaciones de hardware (Linux en español, 2017).

En cuanto a la arquitectura, básicamente se puede optar por procesadores de 32 y de 64 bits. Respecto a la instalación, permite configurar una instalación vía internet o una más clásica con CD (*Compact Disc*,

[3] **ISO** archivo donde se almacena una copia o imagen exacta de un sistema de ficheros.

Disco Compacto), así como aspectos más locales como la hora, el idioma de la distribución, además de definir una contraseña de administrador. El proceso es relativamente rápido, más configurable cuantos más conocimiento se posea (Linux en español, 2017).

1.2.2 SUSE Studio

SUSE Studio (Ver anexo 4) es una aplicación web creada por SUSE (*Software und Systementwicklung*, Desarrollo de Sistemas y de Software) para el diseño de versiones personalizadas de esta conocida distribución de GNU/Linux. Para usarla es necesario crear una cuenta o usar credenciales de *Google, Twitter, Facebook o Yahoo*. Una vez iniciada la sesión, cuenta con un asistente que guía el proceso paso a paso. Permite elegir qué versión de SUSE utilizar, la versión gratuita OpenSUSE o las versiones profesionales de SUSE en sus distintas variantes. También se puede optar por versiones más personalizadas de la galería SUSE, donde hay propuestas interesantes pensadas para públicos muy concretos con necesidades muy específicas. El siguiente paso será seleccionar los paquetes a instalar y las dependencias de estos se añaden automáticamente (Linux en español, 2017).

Otras opciones a tener en cuenta son las relacionadas con la configuración del sistema operativo, pudiendo configurar prácticamente cualquier aspecto. Además, se tiene la opción de incluir archivos personales, como manuales o documentos, en directorios concretos y finalmente, guardar el instalador en una imagen ISO normal o en otros formatos VMDK[4] (Virtual Machine Disk, Disco de Máquina Virtual) o VHD[5] (Virtual Hard Disk, Disco Duro Virtual) propios de máquinas virtuales (Linux en español, 2017).

Otras características:

- Agregar y eliminar paquetes y repositorios.
- Configurar opciones de localización, inicio, el uso de bases de datos, gestión de almacenamiento, y otras opciones como la apariencia a través del logo y fondos de pantalla.
- Permite la construcción de una imagen de disco, LiveCD/DVD, imagen de VMware (empresa estadounidense de software y hardware, filial de *EMC Corporation* que proporciona software de virtualización disponible para ordenadores compatibles X86), imagen de Xen (máquina virtual de

4 VMDK (.vmdk) es un formato de archivo que describe los contenedores usados en discos duros virtuales para ser utilizados en máquinas virtuales.

5 VHD es un formato de archivo que representa una unidad de disco duro virtual.

código abierto desarrollada por la Universidad de Cambridge) y Formato OVF (*Open Virtualization Format*, Formato de Virtualización Abierta).

1.2.3 Reconstructor

Reconstructor (Ver anexo 5) está presente como una aplicación de escritorio desde hace algún tiempo, pero ahora hace su aparición como aplicación web que permite crear y personalizar un LiveCD de Debian o Ubuntu, al más puro estilo de SUSE Studio. Primeramente se debe seleccionar la versión base a modificar y los paquetes que se pretenden usar, esto último desde repositorios (incluidos universe y multiverse), luego se puede personalizar la "nueva" distribución en la sección "módulos". Una vez concluidos los pasos, se podrá crear y descargar la imagen (Infonucleo, 2017).

Otras características (Infonucleo, 2017)**:**

- Posibilidad de personalizar la imagen de arranque, el usplash (aplicación que dibuja el logo de Ubuntu con su barra de progreso durante la carga del sistema operativo), fondos de escritorio, pantalla de GNOME (*GNU Network Object Model Environment,* Entorno de Modelo de Objetos de Red GNU).
- Tema GDM (*GNOME Display Manager*, Gestor de Visualización de GNOME).
- Permite configurar el repositorio para la personalización generada.

1.3 Análisis de las herramientas de personalización

A continuación, se hace un análisis de las herramientas de personalización estudiadas en el epígrafe 1.2 con el objetivo de conocer si cumplen con las necesidades existentes en la creación de personalizaciones para la distribución de GNU/Linux Nova en su versión para servidores. En la comparación se utilizaron cuatro criterios de análisis.

Criterios de análisis:

- Tecnología web
- Selección de paquetes
- Personalizaciones para servidores
- Personalizaciones con distribución GNU/Linux Nova como base.

Tabla 1. Resultado del análisis de las herramientas de personalización de distribuciones

(Fuente: elaboración propia)

Criterios de análisis	**Herramientas seleccionadas**		
	Instalinux	SUSE Studio	Reconstructor
Tecnología web	Sí	Sí	Sí
Selección de paquetes	Sí	Sí	Sí
Personalizaciones para servidores	No	Sí	No
Personalizaciones con la distribución GNU/Linux Nova como base	No	No	No

El análisis anterior sobre las diferentes herramientas de personalización demuestra la necesidad de desarrollar una nueva solución ya que estas herramientas no resuelven las carencias que presenta el proyecto Nova Servidores para la personalización de su sistema operativo. El estudio realizado permitió definir un conjunto de aportes y limitaciones, visibles en la Tabla 2, que fueron utilizados como referencia para el diseño de la propuesta de solución.

Tabla 2. Aportes y limitaciones de las herramientas analizadas

(Fuente: elaboración propia)

Herramientas	**Aportes**	**Limitaciones**
Instalinux	Permite crear el usuario administrador y configurar las opciones de red.	No permite realizar una personalización de una distribución que no sean las que están predefinidas.
SUSE Studio	Las dependencias de los paquetes seleccionados se añaden automáticamente.	Solo permite hacer personalizaciones de OpenSuse.
Reconstructor	Permite configurar el repositorio para la personalización generada.	Solo permite realizar personalizaciones de Debian y Ubuntu.

1.4 Metodología de desarrollo de software

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un

producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado (González y otros, 2016).

La metodología seleccionada para el desarrollo de la propuesta de solución es **Variación de AUP para la UCI,** la cual fue elaborada en la Universidad de las Ciencias Informáticas. Esta metodología logró estandarizar el proceso de desarrollo de software en la universidad, dando cumplimiento a las buenas prácticas y logrando hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. Variación de AUP para la UCI es la metodología de desarrollo de software usada en CESOL para el cual se desarrolla este trabajo de diploma. La misma cuenta con tres fases (Rodríguez, 2015):

- **Inicio:** durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
- **Cierre:** en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

El presente trabajo de diploma se desarrolla en la fase de Ejecución y transitará por las siguientes disciplinas propuestas en la metodología: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de Aceptación. Los productos de trabajo generados durante la elaboración de la solución estarán basados en el Escenario No 3 ya que se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez y su continuidad. Este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y las relaciones entre los procesos identificados (Rodríguez, 2015).

1.5 Lenguajes y herramientas para el modelado de la solución

En el modelado de la propuesta de solución se utilizaron los siguientes lenguajes y herramientas.

1.5.1 Lenguaje de modelado

UML 2.0 (*Unified Modeling Language*, Lenguaje de Modelado Unificado) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Ofrece un estándar para describir los modelos, incluyendo aspectos conceptuales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. UML cuenta con un conjunto de notaciones y diagramas para modelar sistemas orientados a objetos y describe la semántica esencial de lo que estos diagramas y símbolos significan. Algunas de las principales ventajas que tiene la utilización de UML en la construcción de sistemas de software son (Cabrera, 2012):

- Puede usarse en las diferentes etapas del ciclo de vida del desarrollo de sistemas.
- Es independiente del proceso o metodología de desarrollo y del lenguaje de implementación.
- Permite crear y modificar modelos expresivos mediante la combinación de los 13 tipos de diagramas que suministra en esta versión.
- Es posible extender la funcionalidad de la notación gráfica mediante estereotipos y proveer una base formal para los diagramas.

BPNM (*Business Process Modeling Notation*, Notación para el Modelado de Procesos de Negocio) es un estándar de modelado de procesos de negocio, en donde se presentan gráficamente las diferentes etapas por las que trascurre el proceso. La notación ha sido diseñada específicamente para coordinar la secuencia de procesos y los mensajes que fluyen entre los diferentes procesos participantes. Está dirigido a gerentes, directores, dueños de empresas, ingenieros de procesos, analistas de negocios, analistas de sistemas, administradores de proyectos, responsables de calidad y todo aquel que necesita definir, documentar y hacer más eficientes sus procesos de negocio con uno de los estándares más avanzado y aceptado a nivel internacional (Aviztar, 2017).

Características de BPMN (Analítica, 2014):

- Proporciona un lenguaje gráfico común, con el fin de facilitar su comprensión a los usuarios de negocios.
- Integra las funciones empresariales.
- Utiliza una Arquitectura Orientada a Servicios, con el objetivo de adaptarse rápidamente a los cambios y oportunidades del negocio.

- Combina las capacidades del software y la experiencia de negocio para optimizar los procesos y facilitar la innovación del negocio.

1.5.2 Herramientas de modelado

Visual Paradigm 8.0 para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para ingenieros de software, analistas de sistemas y arquitectos de sistemas que están interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos (Targetware, 2016).

Visual Paradigm también ofrece (Targetware, 2016):

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/HTML.
- Documentación automática.
- Ambiente visualmente superior de modelado.

Otra herramienta seleccionada como parte del modelado del sistema es ForeUI para la creación de prototipos de interfaces.

ForeUI 3.2 es una herramienta que permite diseñar prototipos de interfaces web o de aplicaciones mediante la creación de maquetas de imágenes. La aplicación ofrece un arreglo de componentes que incluye botones, menús, barras de desplazamiento y combo box, además de herramientas gráficas para insertar formas, líneas, textos, imágenes. Una vez construidos los prototipos permite exportarlos como imagen, PDF (*Portable Document Format*, Formato de Documento Portátil) o HTML 5 (*HyperText Markup Language,* Lenguaje de Marcas de Hipertexto) (Foreui.com, 2017).

1.6 Herramientas y tecnologías de implementación

En el desarrollo de la propuesta de solución se utilizaron las siguientes herramientas y tecnologías.

1.6.1 Lenguaje de programación

Un lenguaje de programación es un sistema estructurado y diseñado principalmente para que las máquinas (por ejemplo, los ordenadores) se entiendan con las personas y entre ellas. Contiene un conjunto de acciones consecutivas que el ordenador debe ejecutar. Se utilizan principalmente para controlar cómo se comporta una máquina o crear programas informáticos (Areatecnologia, 2016). Para el

desarrollo de la herramienta de creación de personalizaciones de Nova Servidores se seleccionó Python como lenguaje de programación.

Python 3.5 es un lenguaje potente, flexible y con una sintaxis clara y concisa que no requiere dedicar tiempo a su compilación debido a que es interpretado. Es de código abierto y como tal cualquiera puede contribuir a su desarrollo y divulgación. Además, no es necesario pagar ninguna licencia para distribuir el software desarrollado en este lenguaje. Hasta su intérprete se distribuye de forma gratuita. Puede ser utilizado en diversas plataformas y sistemas operativos, entre los que podemos destacar los más populares, Windows y Linux; pero además puede funcionar en *smartphones* (teléfonos inteligentes) (Codejobs, 2016).

Las principales características que tiene este lenguaje son (Universidad de Granada, 2016):

- Lenguaje de programación multiparadigma (programación orientada a objetos, programación estructurada y programación funcional).
- Utiliza el tipo de dato dinámico *y reference counting* (conteo de referencia) para el manejo de memoria.
- Permite la resolución dinámica de nombres, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado ligadura dinámica de métodos).
- Puede utilizarse como un lenguaje de extensión para módulos y aplicaciones que necesitan de una interfaz programable.

1.6.2 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (*Integrated Development Environment*, IDE), es una aplicación de software que proporciona servicios integrales para facilitarle al programador de computadora el desarrollo de software. Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDEs tienen autocompletado inteligente de código (Suárez, 2015). Para el desarrollo de la propuesta de solución se seleccionó PyCharm.

PyCharm 3.2 es un IDE con un completo juego de herramientas para el desarrollo productivo con el lenguaje de programación Python. Adicionalmente, provee capacidades de alto rango para desarrolladores profesionales de web con el marco de trabajo Django (Pythonízame, 2014).

Las principales características de este IDE son (Pythonízame, 2014):

- Editor inteligente.
- Depurador de código gráfico.
- Inspección de código.
- Integración de control de versiones.
- Ejecución de pruebas.
- Consola Python integrada.
- Soporta los repositorios Git, Mercurial, Subversion y Github.
- Permite los leguajes XML (*Extensible Markup Language*, Lenguaje de Marcado Extensible) y HTML (*HyperText Markup Language*, Lenguaje de Marcas de Hipertexto).
- Posee una terminal local.

1.6.3 Marco de trabajo

Un marco de trabajo (*framework*) es una gran librería o conjunto de librerías donde además de facilitar funciones para su uso, dispone de una sintaxis o metalenguaje específico del marco de trabajo y una forma de organización de su código. Para usar un marco de trabajo no basta con conocer el nombre de la función a utilizar: hay que saber qué sintaxis emplea, qué obligaciones impone (a la hora de organizar el código, archivos) y su lógica o filosofía de trabajo. El uso de marcos de trabajo se ha extendido debido a la gran complejidad de las aplicaciones que se desarrollan ya que facilitan su organización y mantenimiento (Aprenderaprogramar, 2016). En el desarrollo de la herramienta para la creación de personalizaciones de Nova Servidores se utilizará el marco de trabajo Django.

Django 1.6 es un marco de desarrollo en Python, de código abierto y muy ligero que permite la creación rápida de páginas y aplicaciones web. Pretende ser sencillo y rápido posibilitando concentrarse principalmente en escribir la aplicación (BBVAOpen4U, 2016).

Las principales características que presenta este marco de trabajo son (BBVAOpen4U, 2016):

- Basado en la filosofía DRY (*Don't Repeat Yourself,* no te repitas). Muchas aplicaciones web y proyectos digitales comparten numerosas líneas de código unos con otros. Django es el marco de desarrollo de refactorización de código casi por excelencia. Permite reutilizar programación de unas aplicaciones a otras sin la obligación de tener que repetir las mismas líneas de código entre distintos proyectos.

- Es de alto nivel basado en el patrón arquitectónico Modelo-Vista-Plantilla. Apuesta por la sencillez, la rapidez y la reutilización de código.
- Viene con SQLite, una base de datos usada por compañías tan importantes como Facebook o Bloomberg.
- Contiene una API (*Application Programming Interface*, Interfaz de Programación de Aplicaciones) propia para el desarrollo de proyectos digitales.

1.6.4 Servidor de aplicaciones web

Los servidores de aplicaciones web (también conocidos como servidores HTTP) son un tipo de servidor utilizados para la distribución de contenido web en redes internas o en Internet. Como parte de una red de ordenadores, un servidor web transfiere documentos a los llamados clientes, es el encargado de proporcionar los datos para la visualización del contenido web. Además un servidor web permite el cifrado de la comunicación entre el servidor web y el cliente, autenticación HTTP para áreas específicas de una aplicación web y almacenamiento en caché de documentos dinámicos para la respuesta eficiente de solicitudes y evitar una sobrecarga del servidor (1&1 Internet España S.L.U., 2016). En el desarrollo de la herramienta para la creación de personalizaciones de Nova Servidores se utilizará como servidor de aplicaciones web Nginx.

Nginx 1.9.15 es un servidor proxy HTTP, proxy inverso y de correo, de código abierto originalmente escrito por Igor Sysoev. Actualmente es usado en el mundo por compañías importantes como Netflix, Wordpress y Dropbox (Nginx.org, 2017).

Características (Nginx.org, 2017):

- El proxy inverso ayuda con el balanceo de carga distribuyendo las peticiones y almacenando en caché parte del contenido.
- Cuenta con caché para solicitudes HTTP.
- Facilita el uso de URL con posibilidad de usar expresiones regulares.
- Función de rastreo y seguimiento de los usuarios.
- Ofrece tolerancia a fallos.

- Permite realizar retransmisión de archivos FLV[6] o MP4[7] de forma nativa.
- Soporte para gestionar y configurar IPv6[8].
- Soporta servidores virtuales.

Además, se utilizó el servidor **Gunicorn**.

Gunicorn 19.6.0 (*Green Unicorn,* Unicornio Verde) es un ligero y rápido servidor WSGI (*Web Server Gateway Interface*, Interfaz de Puerta de enlace del Servidor Web, una especificación para la interfaz simple y universal entre servidores web y aplicaciones web o *frameworks* para Python) (Gunicorn.org, 2017).

Conclusiones del capítulo

El análisis de los principales referentes teóricos acerca del proceso de elaboración de personalizaciones de distribuciones GNU/Linux, así como el estudio de los principales conceptos asociados al problema planteado permitió sentar las bases para el desarrollo de la investigación y conocer las características del objeto de estudio. La comparación de las diferentes herramientas de personalización analizadas permitió demostrar la necesidad de desarrollar una aplicación web ya que las existentes no resuelven las carencias que presenta el proyecto Nova Servidores para la creación de personalizaciones para su sistema operativo. La propuesta de solución se realizará utilizando la metodología de desarrollo de software Variación de AUP para la UCI y herramientas y tecnologías libres.

[6] Flash Video (FLV) es un formato contenedor propietario utilizado para transmitir vídeo por Internet sobre el complemento Adobe Flash.

[7] MP4 es un formato de codificación de audio asociado a la extensión mp4.

[8] IPV6 es el protocolo de Internet de última generación, diseñado para reemplazar al protocolo de Internet actual, IP versión 4.

Capítulo 2: Análisis y diseño de la propuesta de solución

En el presente capítulo, se desarrolla el análisis y diseño de la propuesta de solución tomando como punto de partida las características del proceso de personalización y los elementos planteados por la metodología Variación de AUP para la UCI en el escenario 3. Se definen los requisitos del sistema y la arquitectura del mismo.

2.1 Modelado del negocio

El Modelado del negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito (Sánchez, 2015).

2.1.1 Modelo conceptual

En un modelo conceptual se ofrece una representación de cosas del mundo real, no de componentes de software. Un modelo conceptual muestra: los conceptos, las asociaciones entre conceptos y sus atributos (Ardila y otros, 2014). En la figura 1 se muestra el modelo conceptual del domino del negocio de la propuesta de solución.

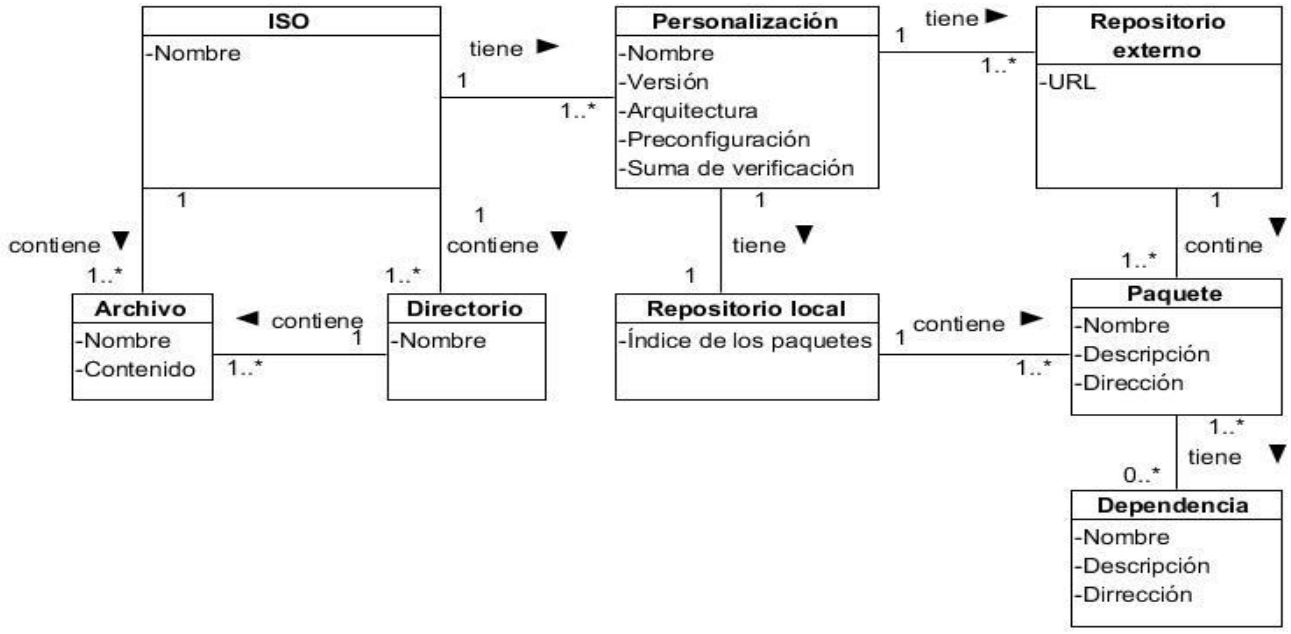


Figura 1 Modelo conceptual

(Fuente: elaboración propia)

Conceptos:

- **Archivo:** Son los ficheros que están almacenados en el ISO y contienen los datos y las configuraciones para la instalación o funcionamiento del sistema operativo.
- **Dependencia:** Es un paquete accesorio requerido por el paquete instalador de las aplicaciones seleccionadas por el usuario, para hacer funcionar correctamente un programa después de instalarlo.
- **Directorio:** Carpeta que almacena una agrupación de archivos de datos y otros subdirectorios.
- **ISO:** Único archivo que almacena todos los componentes necesarios para la instalación del sistema operativo.
- **Paquete:** Es un archivo comprimido que contiene un conjunto de ficheros con instrucciones para la reconstrucción de una aplicación dentro del sistema nuevo.
- **Personalización:** Es la versión de la imagen ISO inicial con una configuración diferente y personalizada a partir de las necesidades del usuario, donde se contestan todas o la mayoría de las preguntas que una instalación normalmente hace. Además, incluye un grupo de software que se instalarán automáticamente antes de que el sistema arranque por primera vez en servicio.
- **Repositorio externo:** Es cualquier servidor o lugar en la red donde se encuentran los programas para el sistema operativo que se está personalizando.
- **Repositorio local:** Es la estructura de directorios que contienen los paquetes específicos para la personalización.

2.1.2 Diccionarios de datos de personalización

Tabla 3. Diccionario de personalización

(Fuente: elaboración propia)

Descripción	Una personalización es la versión de la imagen ISO inicial con una configuración diferente y personalizada a partir de las necesidades del usuario, donde se contestan todas o la mayoría de las preguntas que una instalación normalmente hace. Además incluye un grupo de software que se instalarán automáticamente antes de que el sistema arranque por primera vez en servicio.				
Atributos					
Nombre	**Descripción**	**Tipo**	**¿Puede**	**¿Único?**	**Restricciones de clase**

			ser nulo?		**Válidas**	**No válidas**
Arquitectura	Arquitectura del sistema operativo	Alfanumérico	No	No	i386 ó amd64	No aplica
Nombre	Nombre de la personalización	Alfanumérico	No	No	No aplica	Caracteres extraños
Pre-configuración	Contenido del fichero de auto-matización de la instalación	Alfanumérico	Si	No	No aplica	Caracteres extraños
Suma de verificación	Suma md5 para verificar la integridad del ISO	Alfanumérico	No	Sí	No aplica	Caracteres extraños
Versión	Versión del sistema operativo	Numérico	No	No	2015 o 2017	No aplica

2.1.3 Descripción de proceso de negocio

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente, llevadas a cabo para lograr un resultado de negocio definido, tiene sus entradas, funciones y salidas. Puede ser parte de un proceso mayor que lo abarque o bien puede incluir otros. El enlace entre ellos y generación de valor lleva a algunos practicantes a verlos como los flujos de trabajo que efectúan las tareas de una organización (Villafane, 2017)**.** En la figura 2 se presenta el diagrama del proceso de negocio personalización de Nova Servidores.

2.1.4 Diagrama de procesos del negocio personalización de Nova Servidores

A continuación, se presenta el diagrama de procesos del negocio personalización de Nova Servidores y su descripción.

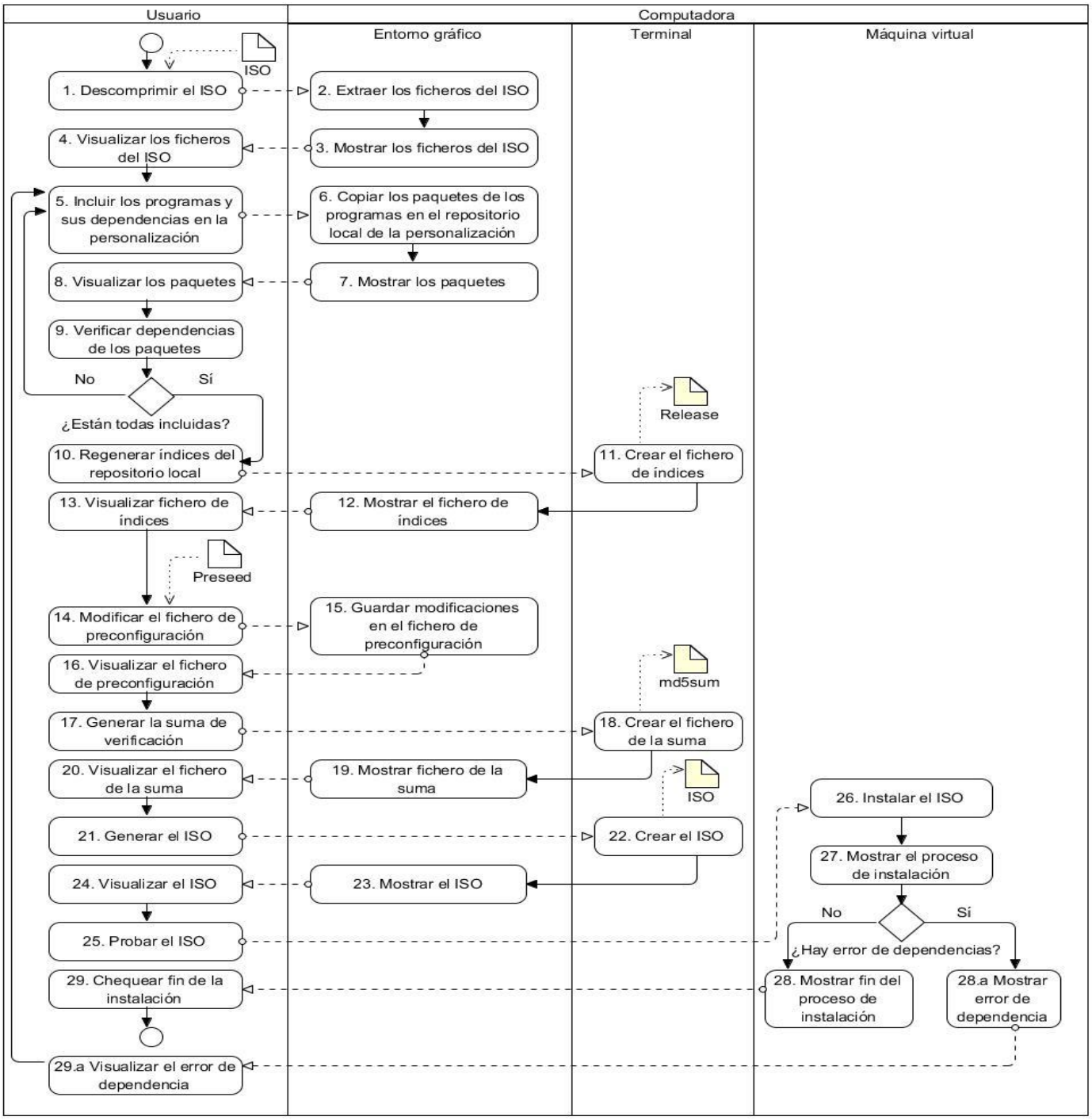


Figura 2. Diagrama de procesos del negocio Personalización de Nova Servidores

(Fuente: elaboración propia)

Tabla 4. Descripción de procesos del negocio Personalización de Nova Servidores

(Fuente: elaboración propia)

Objetivo	Definir el flujo en proceso de personalización.
Evento(s) que lo genera(n)	No aplica.
Pre condiciones	ISO de Nova Servidor, computadora con Linux y una máquina virtual.
Marco legal	No aplica.
Clientes internos	No aplica.
Clientes externos	No aplica.
Entradas	ISO de Nova Servidor.
Flujo de eventos	
Flujo básico	
1.	Descomprimir el ISO. El usuario da la orden a la computadora de descomprimir el ISO.
2.	Extraer los ficheros del ISO. La computadora recibe la orden de descomprimir el ISO y procede a ejecutarla.
3.	Mostrar los ficheros del ISO. Una vez extraídos los ficheros del ISO la computadora procede a mostrarlos.
4.	Visualizar los ficheros del ISO. El usuario visualiza los ficheros extraídos del ISO.
5.	Incluir los programas y sus dependencias en la personalización. El usuario da la orden a la computadora de copiar los paquetes de los programas y sus dependencias en el repositorio local de la personalización
6.	Copiar los paquetes de los programas en el repositorio local de la personalización. La computadora recibe la orden y procede a realizar la copia.
7.	Mostrar los paquetes. Una vez copiados los paquetes de los programas la computadora los muestra.
8.	Visualizar los paquetes. El usuario visualiza los paquetes copiados en el repositorio local de la personalización.
9.	Verificar dependencias de los paquetes. El usuario verifica que todas las dependencias de cada uno los paquetes estén incluidas en el repositorio local. Si todas las dependencias no están incluidas están incluidas, ir al flujo alterno 10.a.
10.	Regenerar índices del repositorio local. El usuario introduce el comando en la terminal para regenerar los índices.
11.	Crear el fichero de índices. La terminal ejecuta el comando y crea el fichero de índices.
12.	Mostrar el fichero de índices. Se muestra el archivo de índices creado.

13.	Visualizar fichero de índices. El usuario visualiza el fichero de índices creado.
14.	Modificar el fichero de preconfiguración. El usuario abre el fichero de preconfiguración y lo modifica.
15.	Guardar configuraciones en el fichero de preconfiguración. La computadora guarda las modificaciones realizadas en el fichero.
16.	Visualizar el fichero de preconfiguración. El usuario visualiza el fichero de configuración en edición.
17.	Generar suma de verificación. El usuario introduce el comando para generar la suma de verificación.
18.	Crear el fichero de la suma. La terminal ejecuta el comando y crea el fichero de la suma de verificación.
19.	Mostrar fichero de la suma. Se muestra el fichero con las suma de verificación generada.
20.	Visualizar fichero de la suma. El usuario visualiza el fichero de la suma de verificación creado.
21.	Generar el ISO. El usuario introduce el comando para generar el ISO.
22.	Crear ISO. La terminal ejecuta el comando y crea el ISO.
23.	Mostrar el ISO. Se muestra el ISO creado.
24.	Visualizar el ISO. El usuario visualiza el ISO creado.
25.	Probar el ISO. El usuario procede a iniciar el proceso de instalación del ISO en la máquina virtual.
26.	Instalar el ISO. La máquina virtual procede a instalar el ISO.
27.	Mostrar el proceso de instalación. La máquina virtual muestra el proceso de instalación. Si hay error de dependencias, ir al flujo alterno 28.a
28.	Mostrar fin del proceso de instalación. Se muestra el fin del proceso de instalación correctamente.
29.	Visualizar fin de la instalación. El usuario visualiza el fin de la instalación sin la ocurrencia de errores.
Pos-condiciones	
1.	Se creó el ISO personalizado.
Salidas	
1.	ISO personalizado.
Flujos paralelos	
No aplica	
Flujos alternos	
10. a Faltan dependencias.	
1.	Incluir los programas y sus dependencias en la personalización. Incluir las dependencias que faltan. Volver al paso 5 del flujo básico.
Pos-condiciones	

1.	Se actualiza el ISO personalizado.
Salidas	
1.	ISO personalizado.
28. a Mostrar error de dependencia.	
1.	Visualizar error de dependencia. El usuario visualiza el error por la falta de dependencias.
2.	Incluir los programas y sus dependencias en la personalización. Incluir las dependencias que faltan. Volver al paso 5 del flujo básico.
Pos-condiciones	
2.	Se actualiza el ISO personalizado.
Salidas	
1.	ISO personalizado.
Asuntos pendientes	
No aplica	

El modelado del negocio le permitió al autor de la investigación conocer las características específicas del entorno del problema, así como las restricciones del dominio de la propuesta de solución que limitan las funcionalidades a desarrollar y sirvió de base para definir los requisitos a implementar, los cuales se presentan en el epígrafe siguiente.

2.2 Requisitos

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto (Sánchez, 2015).

2.2.1 Fuentes para la obtención de requisitos

Las fuentes de obtención de requisitos utilizadas fueron:

- Modelo conceptual (Ver figura 1)
- Diagrama de Proceso de Negocio (Ver figura 2)
- Análisis de las herramientas existentes (Ver epígrafes 1.2 y 1.3)
- Especialistas de CESOL

2.2.2 Técnicas de identificación de requisitos

Las técnicas de identificación de requisitos de software permiten identificar las necesidades de negocio de los clientes y los usuarios. Son mecanismos que se utilizan para recolectar la información necesaria en la obtención de los requisitos de una aplicación, permiten investigar aspectos generales para posteriormente ser especificados con un mayor detalle, requieren ser adecuadamente orientadas para cubrir la información que se requiere capturar (Pressman, 2010).

Entrevista

La entrevista es de gran utilidad para obtener información cualitativa como opiniones o descripciones subjetivas de actividades. Es una técnica muy utilizada y requiere una mayor preparación y experiencia por parte del analista. La entrevista consiste en un "intento sistemático de recoger información de otra persona" a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada donde es muy importante la forma en que se plantea la conversación y la relación que se establece en la entrevista (Guerra, 2017). La entrevista (Ver anexo 4) fue realizada al programador principal de Nova Servidores perteneciente a CESOL lo que permitió la obtención de los primeros requisitos, visibles en la tabla 5.

Observación

Por medio de esta técnica se obtiene información sobre la forma en que se efectúan las actividades. Esta permite observar la manera en que se llevan a cabo los procesos y verificar que realmente se sigan todos los pasos especificados. Los observadores experimentados saben qué buscar y cómo evaluar la relevancia de lo que observan (Guerra, 2017). Después de aplicada la entrevista y con la obtención de los primeros requisitos, se observó el proceso de personalización llevado a cabo por el especialista de CESOL (Ver anexo 7) lo que permitió refinar e incorporar nuevos requisitos a los existentes presentes en la tabla 5.

Desarrollo de prototipos

Los prototipos consisten en versiones reducidas, demos o conjuntos de pantallas (que no son totalmente operativos) de la aplicación pedida. Permiten a los usuarios experimentar para ver cómo este ayuda a su trabajo, fomentan el desarrollo de ideas que desembocan en requerimientos, mejoran las especificaciones de requerimientos, identifican discrepancias entre los desarrolladores y los usuarios y permite formalizar la aceptación previa por parte del cliente de los requisitos del proyecto (Guerra, 2017).

Con los requisitos obtenidos mediante la aplicación de la Entrevista y la Observación se realizaron los prototipos de las interfaces lo que posibilitó verificar la consistencia y completitud de los mismos. Además, permitió la identificación del resto de los requisitos, una vez que el programador, la analista principal y el jefe de proyecto de Nova Servidor 6.0 interactuaron con estos. La aplicación de estas técnicas propició la identificación de los requisitos que se describen a continuación.

2.2.3 Especificación de requisitos de software

El objetivo principal de la Especificación de Requisitos de Software (ERS) es servir como medio de comunicación entre clientes, usuarios, ingenieros de requisitos y desarrolladores. En la ERS deben recogerse tanto las necesidades de clientes y usuarios (necesidades del negocio, también conocidas como requisitos de usuario, requisitos de cliente y necesidades de usuario) como los requisitos que debe cumplir el software a desarrollar para satisfacer todas las necesidades (requisitos del producto, también conocidos como requisitos de sistema o requisitos software) (Junta de Andalucía, 2017). A continuación se relacionan y describen los requisitos funcionales y no funcionales definidos para la implementación de la propuesta de solución.

Requisitos funcionales

Un requisito funcional es una capacidad que debe tener un software para que sea útil en la realización de los procesos de negocio de una organización (Junta de Andalucía, 2017). En la tabla 5 se listan los requisitos funcionales de la propuesta de solución, su descripción y complejidad, determinada mediante el producto de trabajo Evaluación de Requisitos del expediente de proyecto 4.0 disponible para la actividad productiva de la universidad.

Tabla 5. Listado de requisitos funcionales

(Fuente: elaboración propia)

Número	Requisito	Descripción	Complejidad
RF1	Modificar idioma	El sistema debe permitir modificar el idioma del sistema operativo.	Bajo
RF2	Modificar lenguaje de entrada del teclado	El sistema debe permitir modificar el lenguaje de la entrada del teclado en el sistema operativo.	Bajo
RF3	Modificar zona horaria	El sistema debe permitir modificar la zona horaria del sistema operativo.	Bajo

RF4	Configurar red	El sistema debe permitir la configuración la red utilizando DHCP (*Dynamic Host Configuration Protoco*l, Protocolo de Configuración Dinámica de Host) y de manera estática: IP (*Internet Protocol,* Protocolo de Internet), máscara de red, puerta de enlace, servidor DNS (Domain Name System, Sistema de Nombres de Dominio), nombre del equipo y dominio.	Bajo
RF5	Configurar repositorio	El sistema debe permitir configurar el reposito, así como la arquitectura y la versión del sistema.	Bajo
RF6	Buscar aplicación	El sistema debe permitir buscar una aplicación en el repositorio.	Medio
RF7	Listar aplicación	El sistema debe permitir listar los programas adicionados.	Bajo
RF8	Adicionar aplicación	El sistema debe permitir adicionar una aplicación.	Alto
RF9	Eliminar aplicación	El sistema debe permitir adicionar una aplicación adicionada.	Bajo
RF10	Crear usuario	El sistema debe permitir crear un usuario.	Bajo
RF11	Descargar ISO	El sistema debe permitir descargar el ISO creado.	Bajo
RF12	Crear categoría	El sistema debe permitir crear una nueva categoría para los paquetes.	Bajo
RF13	Eliminar categoría	El sistema debe permitir eliminar una categoría.	Bajo
RF14	Modificar categoría	El sistema debe permitir modificar una categoría.	Bajo
RF15	Adicionar paquete a una categoría	El sistema debe permitir adicionar un paquete a una categoría.	Medio
RF16	Eliminar paquete de una categoría	El sistema debe permitir eliminar un paquete de una categoría.	Bajo
RF17	Listar núcleos	El sistema debe permitir listar los núcleos disponibles para la personalización.	Alto

Requisitos no funcionales

Característica de calidad que debe cumplir el software. Existen diferentes clasificaciones para los requisitos no funcionales en dependencia de las características de calidad que se definan entre ellos: de fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad y seguridad (Junta de Andalucía, 2017).

Con objetivo de estandarizar la redacción de los requisitos no funcionales (RNF) de Atributos de calidad se utilizó la Taxonomía basada en la ISO 25010 propuesta en el producto de trabajo Especificación de requisitos de software del expediente de proyecto 4.0 definido por la Dirección de producción de la Universidad de las Ciencias Informáticas.

Tabla 6. Listado de requisitos no funcionales

(Fuente: elaboración propia)

Número	Requisito	Sub-atributo	Descripción
RNF1	Confiabilidad	Madurez	Evitar un fallo total como resultado de producirse un fallo del software.
RNF2	Usabilidad	Aprendibilidad	El idioma de todas las interfaces de la aplicación será español. Los errores cometidos por el usuario les serán notificados. El sistema expondrá el menú general desde cualquiera de sus páginas.
RNF3	Portabilidad	Adaptabilidad	El servidor estará corriendo sobre el sistema operativo Linux.
RNF4	Funcionalidad	Preciso	El sistema realizará las operaciones indicadas en cada momento.
RNF5	Mantenibilidad	Analizabilidad	Se implementará de forma estándar y las funcionalidades serán comentadas.

2.2.4 Descripción de requisitos de software

A continuación se presenta la descripción del requisito funcional 4 configurar red y no funcional 1 confiabilidad

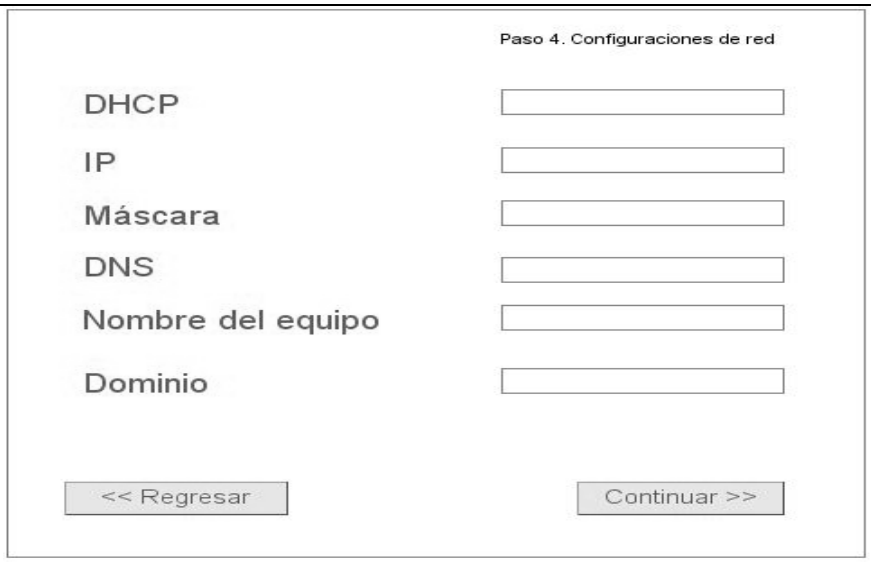
Descripción del RF 4. Configurar red

Tabla 7. RF 4. Configurar red

(Fuente: elaboración propia)

Precondiciones	El archivo con el sistema de ficheros del ISO descomprimido.
Flujo de eventos	
Flujo básico configurar red	
1.	Si el usuario no marca la opción de utilizar la configuración por DHCP, ir al flujo alterno 1a.

2.	El usuario introduce la dirección IP.
3.	El usuario introduce la máscara de red.
4.	El usuario introduce la puerta de enlace.
5.	El usuario introduce el servidor DNS.
6.	El usuario introduce el nombre del equipo.
8.	El usuario introduce el dominio.
7.	El sistema escribe las configuraciones en el fichero de automatización.
Pos-condiciones	
1.	Se configuró la red.
Flujos alternos	
1. a Marca la opción de configuración por DHCP	
1.	El usuario marca la configuración por DHCP. Volver al paso 2 del flujo básico.
Pos-condiciones	
1.	Se configuró la red.
Validaciones	
1	La dirección IP solo puede contener cuatro números ente 0 y 255 separados por un punto.
2	La máscara solo puede contener cuatro números ente 0 y 255 separados por un punto.
3	La dirección IP del servidor DNS solo puede cuatro números ente 0 y 255 separados por un punto.
4	La puerta de enlace solo puede contener cuatro números ente 0 y 255 separados por un punto.
5	El nombre del equipo está no puede contener espacios ni mayúsculas.
6	El dominio no puede contener espacios ni mayúsculas.
Prototipo elemental de interfaz de usuario	

<table>
<tr><td colspan="3"> 
</td></tr>
<tr><td>Conceptos</td><td>Personalización</td><td>Visibles en la interfaz:
IP
DHCP
Máscara
Puerta de enlace
DNS
Nombre del equipo
Dominio
Utilizados internamente:
No aplica</td></tr>
</table>

Descripción del RNF1. Confiabilidad

Tabla 8. RNF 1. Confiabilidad

(Fuente: elaboración propia)

Atributo de Calidad	Confiabilidad
Sub-atributos/Sub-características	Madurez
Objetivo	Evitar un fallo total como resultado de haberse producido un fallo del software.
Origen	Humano
Artefacto	Sistema
Entorno	En tiempo de ejecución
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**

1.a Realizar operación X que requiera entrada de datos.	
Introducir datos incorrectos.	El sistema debe prever en cada interfaz la entrada de datos incorrectos mediante la validación de los datos introducidos y mostrar mensajes de error, en caso de que sean datos inválidos.
Establecer campos con valores predeterminados.	El sistema asumirá que los valores establecidos como predeterminados sean considerados como datos por defecto en el sistema.
Medida de respuesta	
Las interfaces del sistema con campos obligatorios, validación de los campos y valores.	

2.2.5 Validación de requisitos de software

La validación de los requisitos, tiene como objetivo comprobar que estos son correctos. Esta fase debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación, con el costo que eso conlleva (Pressman, 2010).

Técnicas de validación

Prototipado de interfaz de usuario

El prototipado de interfaz de usuario es una técnica de representación aproximada de la interfaz de usuario de un software que permite a clientes y usuarios entender más fácilmente la propuesta de los requisitos para resolver sus problemas de negocio. Está definido como la creación u obtención de modelos concretos o la aproximación de ideas mediante una serie de tecnologías para demostrar conceptos y probar opciones de diseño, posibilitando la validación y optimización de los mismos en un tiempo e inversión mínima, pudiendo así obtener más conocimiento sobre el problema y sus posibles soluciones (Pacheco, 2014). Los prototipos de interfaz de usuario realizados con los requisitos obtenidos fueron analizados con el programador, la analista principal y el jefe de proyecto de Nova Servidores 6.0

Revisiones técnicas formales

La revisión técnica formal (RTF) es una actividad de control de la calidad del software que llevan a cabo por los ingenieros de software. Es un tipo de revisión que incluye recorridos, inspecciones y otro pequeño grupo de evaluaciones de software. Sus objetivos son: descubrir errores en la función, lógica o implementación en cualquier representación del software, verificar que el software en revisión satisface los requisitos, garantizar que el software se ha representado de acuerdo con los estándares predefinidos, lograr un software desarrollado de manera uniforme y hacer proyectos más manejables (Pressman, 2010).

Los productos de trabajo generados en las disciplinas Modelado del Negocio, Requisitos y Análisis y diseño fueron revisados por la administradora de la calidad del proyecto Nova Servidores 6.0.

Diseño de casos de pruebas

Es una parte de las pruebas de componentes y sistemas en las se diseñan los casos de prueba (entradas y salidas esperadas) para probar el sistema. Su objetivo es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos. Para su diseño, se selecciona una característica del sistema o componente que se está probando, un conjunto de entradas que ejecutan dicha característica, se documentan las salidas esperadas o rasgos de salida y donde sea posible se diseña una prueba automatizada que demuestre que las salidas reales y las esperadas son las mismas (Sommerville, 2011). A continuación se presenta el diseño de casos de prueba para el requisito funcional 4 Configurar red.

Caso de prueba configurar red

Descripción general: Permite configurar la red utilizando DHCP y de manera estática.

Precondiciones: El archivo con el sistema de ficheros del ISO descomprimido.

Escenario	Descripción	DHCP	IP	Máscara	Puerta de enlace	DNS	Nobre de equipo	Dominio	Respuesta del sistema	Flujo central
EC 1.1 Configurar red insertando valores válidos	El sistema perimite configurar la red en caso de que no exista ningun error	V	V	V	V	V	V	V	Continua al siguiente paso de la personalización .	1. Seleccionar la opción crear ISO del menú. 2. Avanzar en la configuración hasta el paso 4 configurar red.
		si	10.8.112.22	255.255.255.	10.8.112.254	10.0.0.3	javier-pc	uci.cu		
		V	V	V	V	V	V	V		
		no	10.8.20.222	255.255.0.0	10.8.20.254	10.0.0.4	carlos	cub.cu		
EC 1.2 Configurar red insertando valores inválidos	El sistema permite verificar si existen valores inválidos y no configura la red	I	I	I	I	I	I	I	Marca los campos incorrectos de color rojo	1. Seleccionar la opción crear ISO del menú. 2. Avanzar en la configuración hasta el paso 4 configurar red.
		NA	102893	255255255	10.8.112.3.4	10.0.0.0.1	javier pc	uci cu		
		I	I	I	I	I	I	I		
		NA	10.0.0.6.5	222.255.255	10..0..5.3.	10.9.5	Carlos	cuba.		
EC 1.3 Configurar red insertando dejando campos vacios	El sistema permite verificar si existen campos vacios y configura la red	V	V	V	V	V	V	V	Continua al siguiente paso de la personalización .	1. Seleccionar la opción crear ISO del menú. 2. Avanzar en la configuración hasta el paso 4 configurar red.
		NA	vacío	vacío	vacío	vacío	vacío	vacío		
		V	V	V	V	V	V	V		
		Sí	10.8.112.234	vacío	10.8.112.254	vacío	javier-pc	uci.cu		
EC 1.4 Regresar	El sistema perimite regresar al paso anterior de la personalización	V	V	V	V	V	V	V	Regresa al paso 3 de la personalización .	1. Seleccionar la opción crear ISO del menú. 2. Avanzar en la configuración hasta el paso 4 configurar red.
		NA	NA	NA	NA	NA	NA	NA		

I: Incorrecto, V: Válido, NA: No Aplica

Figura 3. Casos de prueba. Configurar red

(Fuente: elaboración propia)

Descripción de las variables

No	Nombre de	Clasificación	Valor Nulo	Descripción
1	**DHCP**	Campo de verificación	Sí	NA
2	**IP**	Campo de texto	Sí	Este campo solo puede contener cuatro números ente 0 y 255 separados por un punto.
3	**Máscara**	Campo de texto	Sí	Este campo solo puede contener cuatro números ente 0 y 255 separados por un punto.
4	**Puerta de enlase**	Campo de texto	Sí	Este campo solo puede contener cuatro números ente 0 y 255 separados por un punto.
5	**DNS**	Campo de texto	Sí	Este campo solo puede contener cuatro números ente 0 y 255 separados por un punto.
6	**Nombre del equipo**	Campo de texto	Sí	Este campo solo puede contener una cadena de letras minusculas sin espacios.
7	**Dominio**	Campo de texto	Sí	Este campo solo puede contener cadenas de letras minusculas separadas por un punto..
NA: No Aplica				

Figura 4. Descripción de las variables

(Fuente: elaboración propia)

2.2.6 Administración de requisitos de software

La administración de requerimientos es el proceso de comprender y controlar los cambios en los requisitos del software. Es necesario mantenerse al tanto de los requisitos particulares y mantener vínculos entre los dependientes de forma que se pueda evaluar el impacto de los cambios en los requisitos. Se debe establecer un proceso formal para implementar las propuestas de cambios y vincular estos a los requisitos del software (Sommerville, 2011).

En la propuesta de solución se utilizaron las matrices de trazabilidad para la administración de los requisitos de la herramienta a desarrollar, permitiendo seguir la evolución de los requisitos durante el ciclo de vida del proyecto. Para ello se definieron los siguientes elementos de seguimiento para la trazabilidad de requisitos en correspondencia con el documento Guía de Trazabilidad establecido por la Dirección de producción de la Universidad de las Ciencias Informáticas para la actividad productiva de software:

- Modelo conceptual
- Descripción de procesos de negocio
- Especificación de requisitos de software
- Requisitos
- Descripción de requisitos por procesos
- Diagrama de clases del diseño
- Diseños de casos de prueba
- Paquetes de implementación

En el desarrollo de la solución se realizaron las siguientes matrices de trazabilidad: requisitos-descripción de requisitos por procesos, requisitos-modelo conceptual, requisitos-descripción de procesos de negocio, requisitos-especificación de requisitos de software, requisitos-diseño de casos de prueba y requisitos-paquetes de implementación. En la figura 5 se presenta la matriz Requisitos-Descripción de requisitos por procesos.

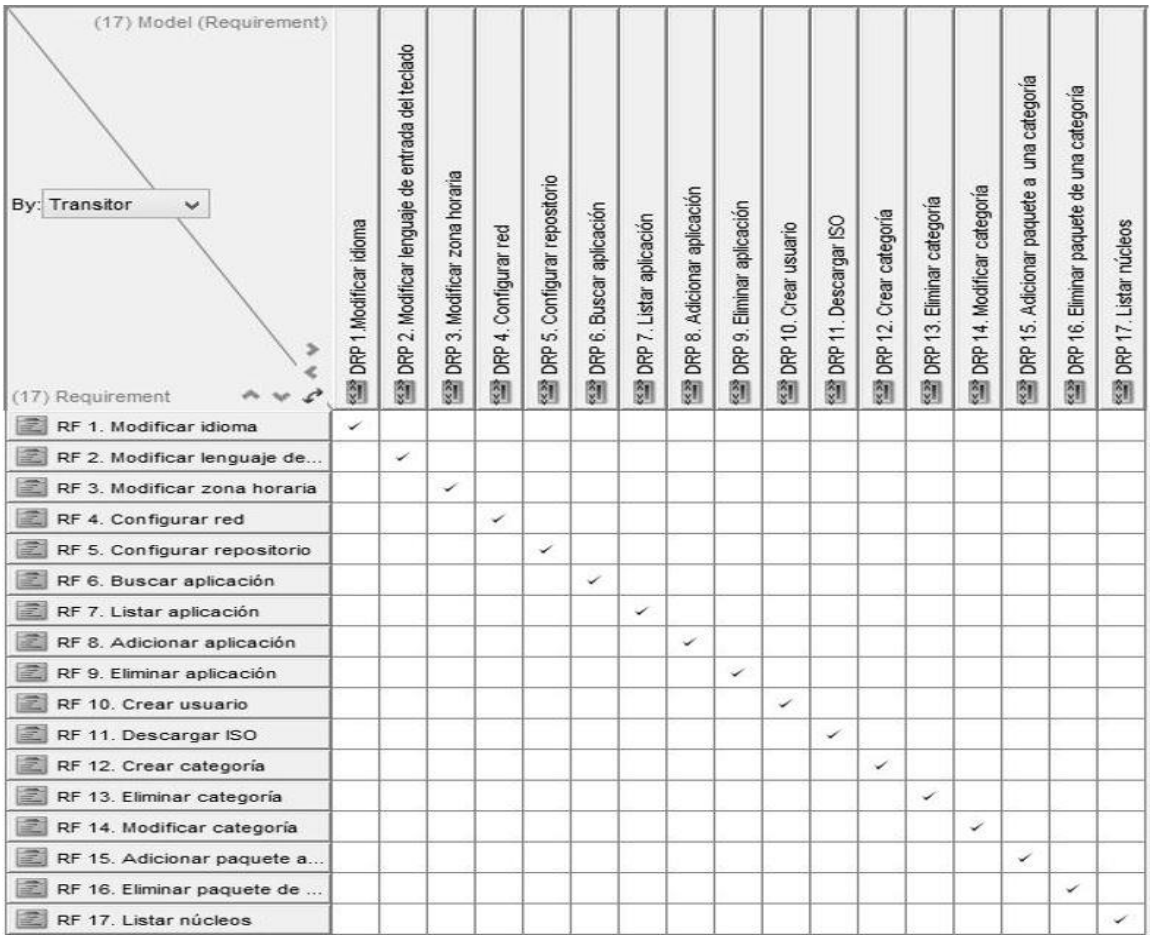


Figura 5. Matriz de trazabilidad Requisitos-Descripción de requisitos por proceso

(Fuente: elaboración propia)

La disciplina Requisitos permitió definir la propuesta de solución generando especificaciones correctas que describen con claridad, en forma consistente y compacta, las necesidades existentes y sirviendo como una base para el análisis y diseño de la propuesta de solución que se realiza a continuación.

2.3 Análisis y diseño

En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además en esta disciplina se modela el sistema y su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos (Sánchez, 2015).

2.3.1 Diseño arquitectónico

En el desarrollo de la solución se utiliza el patrón arquitectónico MVT (*Model-View-Template*, Modelo-Vista-Plantilla) utilizado por el marco de trabajo Django. En este marco, el Modelo es la capa de acceso a la base de datos y contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene y las relaciones entre los datos. La Vista es la capa de la lógica de negocios incluye la lógica que accede al modelo y la delega a la plantilla apropiada. Es una conexión entre los modelos y las plantillas. La Plantilla es la capa de presentación y comprende las decisiones relacionadas a la presentación: como son mostradas sobre una página web u otro tipo de documento (LibrosWeb, 2017).

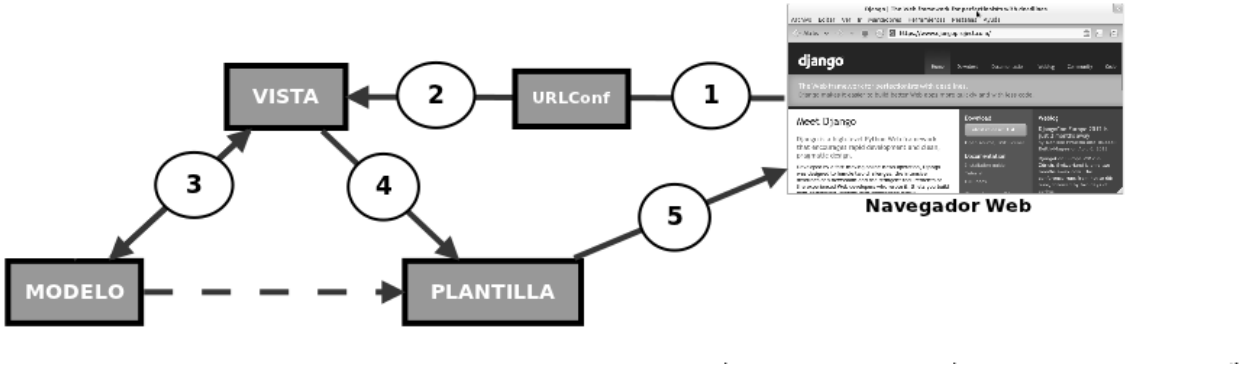


Figura 6. Modelo Vista Plantilla

(Fuente: maestrosdelweb.com)

La Figura 7 representa la arquitectura de la aplicación en el marco de trabajo según el patón arquitectónico MVT.

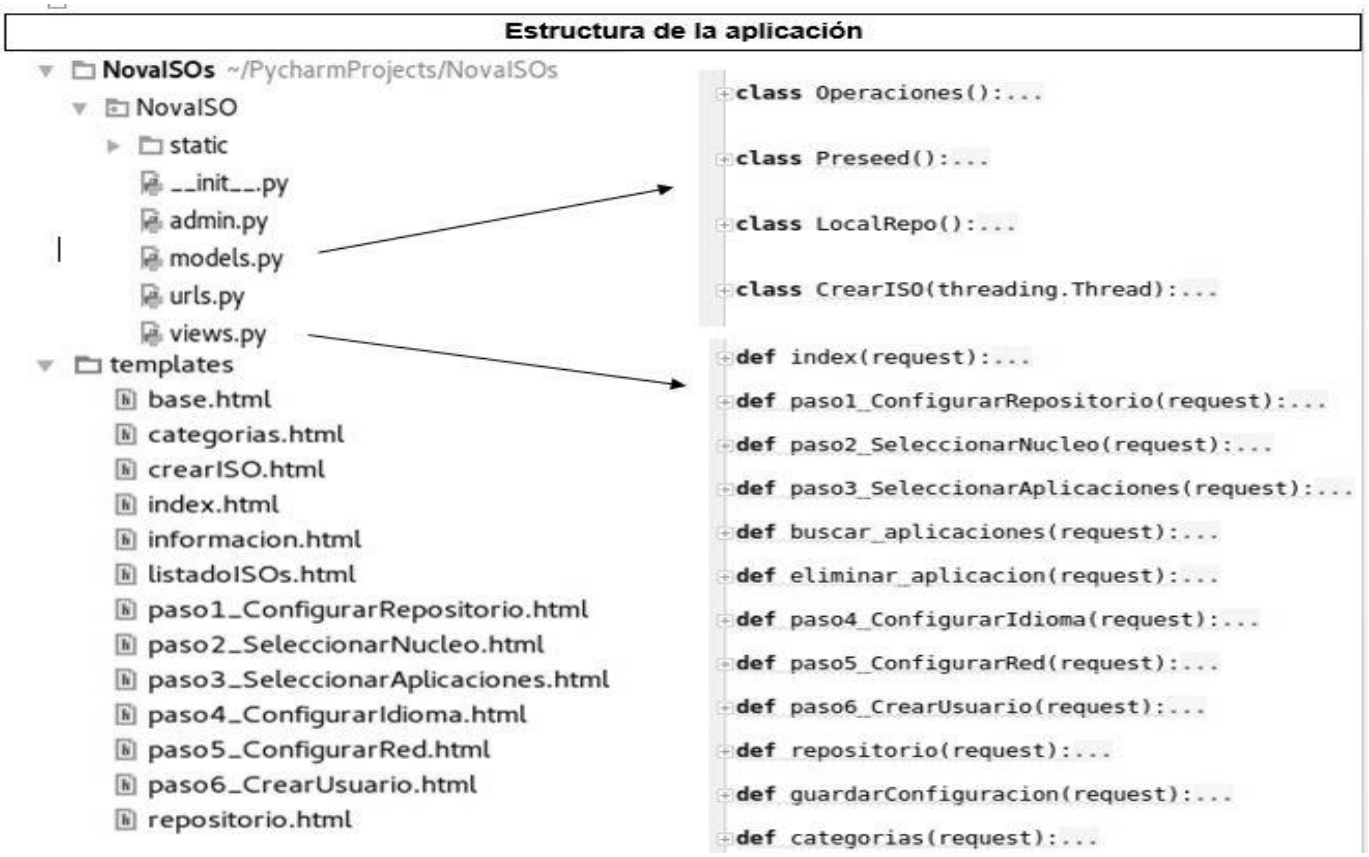


Figura 7. Arquitectura de la solución

(Fuente: elaboración propia)

Estructura de la aplicación

models.py: Contiene cuatro clases:

- Operaciones, es la clase responsable del trabajo con los ficheros (crearlos, eliminarlos, modificarlos).
- Preseed, es la clase responsable del trabajo con el fichero de preconfiguración (escribir configuraciones para la instalación).
- LocalRepo, es la clase responsable de crear el repositorio local (descargar los paquetes y sus dependencias desde el repositorio externo).
- CrearISO, clase que extiende de *Thread* (Hilo) encargada de hacer la llamada a los métodos para la creación del ISO.

views.py: Contiene todas la views encargadas de acceder a las clases del models.py y llamar a las plantillas contenidas en la carpeta templates.

Templates: Es la carpeta que contiene todas las plantillas utilizadas en la aplicación y llamadas desde las views.

2.3.2 Modelo de datos

Un modelo de datos es un conjunto de conceptos y reglas que permiten estructurar los datos resultantes de la observación de la realidad, de forma que pueden ser representadas todas sus propiedades, tanto estáticas como dinámicas. Los modelos de datos de alto nivel o conceptuales utilizan conceptos muy cercanos a la forma en que los usuarios perciben los datos, mientras que los de bajo nivel o físicos describen detalles de cómo se almacenan los datos en la computadora (Gómez, 2013). En la tabla 9 se presentan los atributos que persistirán en el fichero de preconfiguración del ISO.

Tabla 9. Persistencia de los datos

(Fuente: elaboración propia)

Atributo	Tipo de dato	Estructura de persistencia
Idioma	string	*d-i debian-installer/locale string es_ES* *d-i debian-installer/locale select es_ES.UTF-8*
Entrada del teclado	string	*d-i console-keymaps-at/keymap select us*
Zona horaria	string	*d-i time/zone string US/Pacific*
DHCP	boolean	*d-i netcfg/disable_dhcp boolean false*
IP	string	*d-i netcfg/get_ipaddress string 10.8.112.223*
Máscara	string	*d-i netcfg/get_netmask string 255.255.255.0*
Puerta de enlace	string	*d-i netcfg/get_gateway string 10.8.112.254*
DNS	string	*d-i netcfg/get_nameservers string 10.0.0.3*
Nombre del equipo	string	*d-i netcfg/get_hostname string javier-pc*
Dominio	string	*d-i netcfg/get_domain string uci.cu*
Nombre del usuario	String	*d-i passwd/user-fullname Javier*
Usuario	string	*d-i passwd/username string javier*
Contraseña	string	*d-i passwd/user-password-crypted passwd 3c9c03d6008a5adf42c2a55dd4a1a9f2*
Repositorio	String	*d-i mirror/http/hostname string http://nova.f10.uci.cu/nova/2017*

2.3.3 Modelado del diseño

A continuación, se presenta el modelado del diseño de la propuesta de solución.

Diagrama de clases del diseño

Un diagrama de clases es una presentación gráfica de la vista estática, que muestra una colección de elementos declarativos (estáticos) del modelo, como clases, tipos, contenidos y relaciones. Contiene ciertos elementos materializados de comportamiento, como operaciones, pero cuya dinámica se puede representar en otros diagramas, como diagramas de estados o diagramas de colaboración (Ferrer, 2016).

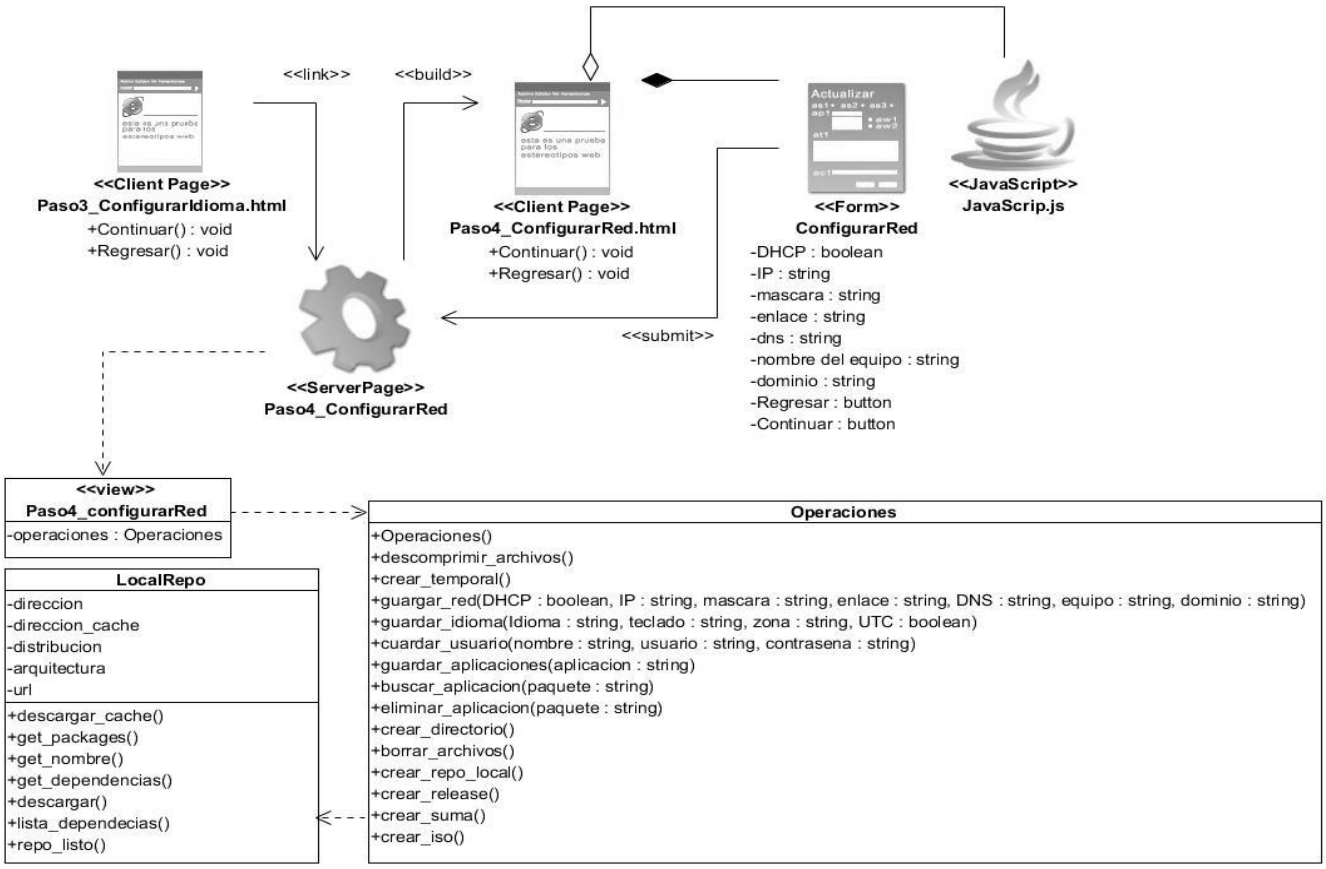


Figura 8. Diagrama de clases del diseño

(Fuente: elaboración propia)

El diagrama de clases del diseño permitió definir estructura estática de las clases en el sistema, su contenido y las relaciones que se establecen entre ellas.

Patrones del diseño de software

Un patrón es un problema que tiene una solución asociada que se puede adaptar a nuevos contextos y que proporciona consejos acerca de cómo puede aplicarse a nuevas situaciones. Los patrones de diseño son una herramienta de apoyo en la búsqueda de soluciones a problemas comunes durante el desarrollo de software. Un patrón de diseño puede dar como resultado una solución a un problema de diseño, una solución puede considerarse un patrón si es efectiva resolviendo problemas similares y si puede ser reutilizable (Peréz, 2013). En el diseño de la propuesta de solución se utilizaron los siguientes patrones.

GRASP

Los GRASP (patrones generales de software para asignar responsabilidades) son patrones generales de software para asignación de responsabilidades. Estos patrones fomentan una serie de buenas prácticas para el diseño de software y su objetivo es describir los principios fundamentales del diseño de objetos y la asignación de responsabilidades. Cada patrón describe un problema, una solución y los beneficios de implementarla (Peréz, 2013).

Experto: Es el patrón que define bajo qué principio se deben asignar responsabilidades a los objetos. Asignar una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con la responsabilidad (Peréz, 2013). Se utiliza en la clase LocalRepo ya que es la que contiene toda la información y las funcionalidades necesarias para crear el repositorio local.

Creador: Es el patrón que define quién debe ser el responsable de crear una nueva instancia de la clase. Asignar a la clase B la responsabilidad de crear una instancia de la clase A (Peréz, 2013). El patrón se evidencia en la clase Operaciones la cual tiene responsabilidad de crear el repositorio local.

Bajo Acoplamiento: Es el patrón que define cómo mantener bajas dependencias, bajo impacto al cambio e incrementar la reutilización. Asignar una responsabilidad de manera que el acoplamiento sea bajo. Un componente con bajo acoplamiento no mantiene fuertes dependencias con otros componentes (Peréz, 2013). Este patrón se utiliza en la clase LocalRepo la cual para cumplir con sus funciones solo necesita relacionarse con la clase operaciones.

En la figura 9 se muestra la aplicación de los patrones GRAPS en la propuesta de solución.

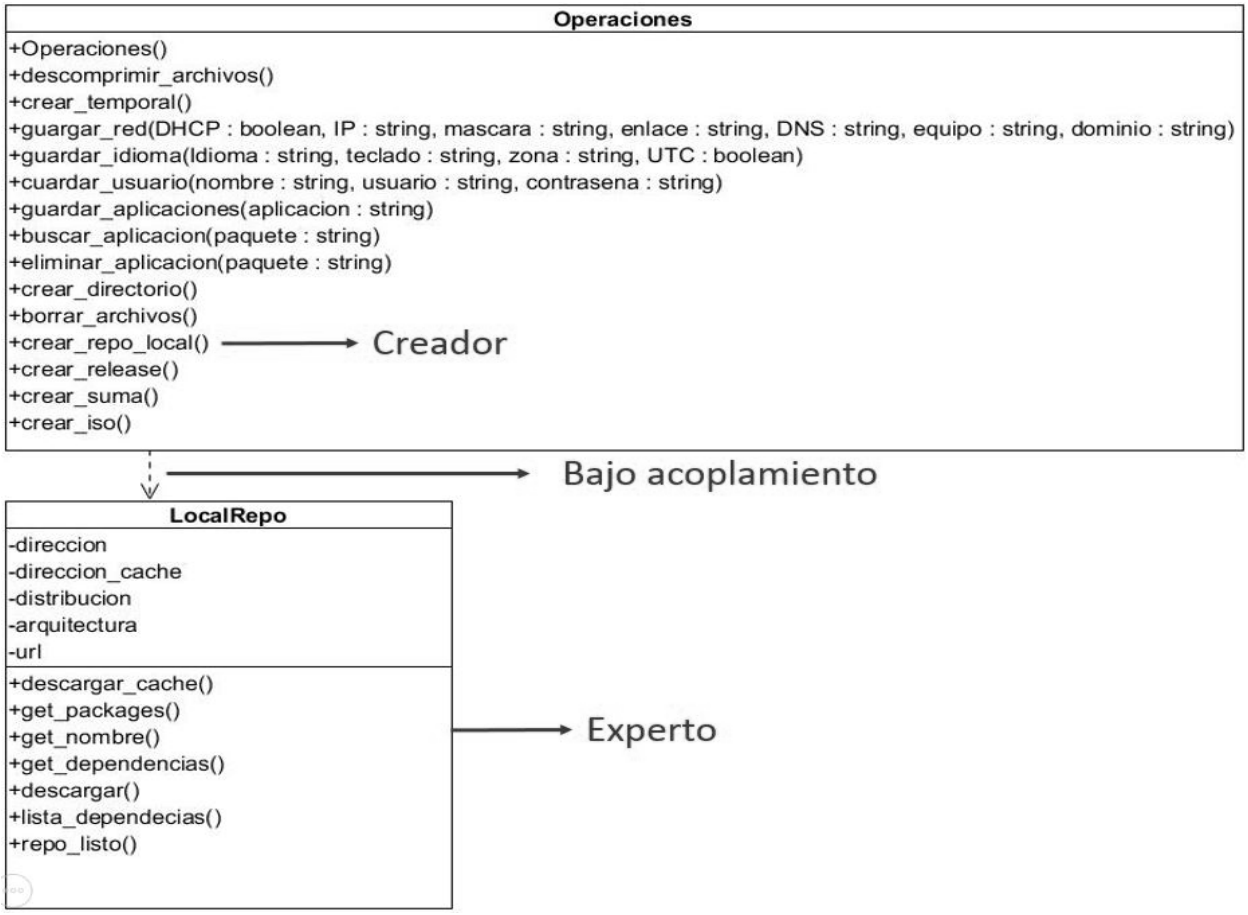


Figura 9. Aplicación de los patrones GRAPS

(Fuente: elaboración propia)

GOF (The Gang of Four, Patrulla de los Cuatro)

Los GOF (The Gang of Four, Patrulla de los Cuatro) son patrones que describen soluciones simples a problemas específicos en el diseño de software orientado a objetos, definen los patrones de diseño como combinaciones de componentes, casi siempre clases y objetos que por experiencia se sabe que resuelven ciertos problemas de diseño comunes *(Guerrero y otros, 2013)*.

Decorador: se utilizó el patrón Decorador el cual permitió añadir responsabilidades adicionales a la plantilla base que almacena el código HTML común a todas las páginas de la aplicación. Las demás plantillas pueden heredar el código que esta posee y redefinir su contenido basada en esa estructura heredada.

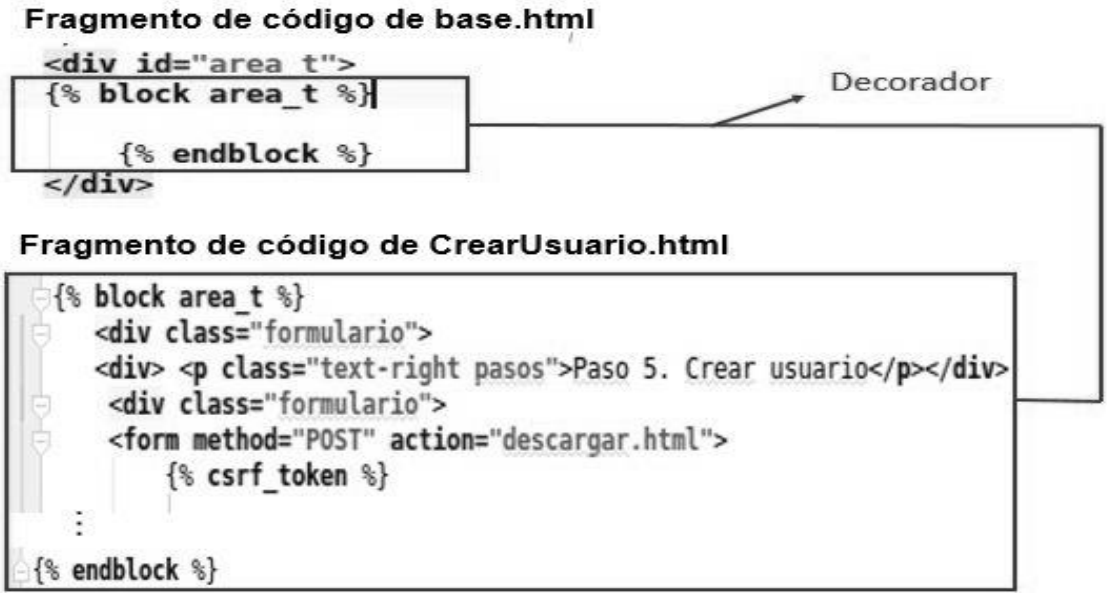


Figura 10. Uso del patrón Decorador

(Fuente: elaboración propia)

2.4 Validación del diseño

En la validación de la propuesta se utilizaron las métricas Tamaño operacional de clase (TOC), para conocer el estado de la responsabilidad, complejidad de implementación y reutilización que poseen las clases definidas y Relaciones entre clases (RC), para medir el acoplamiento, la complejidad de mantenimiento y la reutilización que presentan estas clases, así como la cantidad de pruebas necesarias para probar una clase. A continuación, se explican los pasos seguidos para la aplicación de estas métricas y los resultados obtenidos.

2.4.1 Métrica tamaño operacional de clase (TOC)

Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

- Responsabilidad: mide la responsabilidad que tiene una clase determinada. Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- Complejidad de implementación: evalúa la complejidad de implementación que tiene una clase del diseño. Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- Reutilización: indica el grado de reutilización que tiene una clase. Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Pasos para la aplicación de la métrica TOC

1. Se determinó la cantidad de procedimientos que tienen las clases del sistema.
2. Se calcula el promedio de procedimientos:
   Promedio=cantidad total de procedimientos /cantidad de clases
   Promedio= 33/4=8.25
3. Se le asigna una categoría (Baja, Media, Alta) a cada atributo de calidad (Responsabilidad, Complejidad de implementación y Reutilización) por cada una de las clases siguiendo los siguientes criterios:

Responsabilidad y Complejidad de implementación:

- Baja: si la cantidad de procedimientos es <= Promedio.
- Media: si la cantidad de procedimientos está entre Promedio y 2* Promedio.
- Alta: si la cantidad de procedimientos > 2* Promedio.

Reutilización:

- Baja: si la cantidad de procedimientos es > 2* Promedio.
- Media: si la cantidad de procedimientos está entre Promedio y 2* Promedio.
- Alta: si la cantidad de procedimientos <=Promedio

Resultados obtenidos

En la tabla 10 se presentan los resultados obtenidos de la aplicación de la métrica TOC.

Tabla 10. Resultados de la aplicación de la métrica TOC

(Fuente: elaboración propia)

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
Operaciones	15	Media	Media	Media
Preseed	8	Baja	Baja	Alta
LocalRepo	8	Baja	Baja	Alta
CrearISO	2	Baja	Baja	Alta

En la figura 11 se presentan los resultados obtenidos de la aplicación de la métrica TOC.

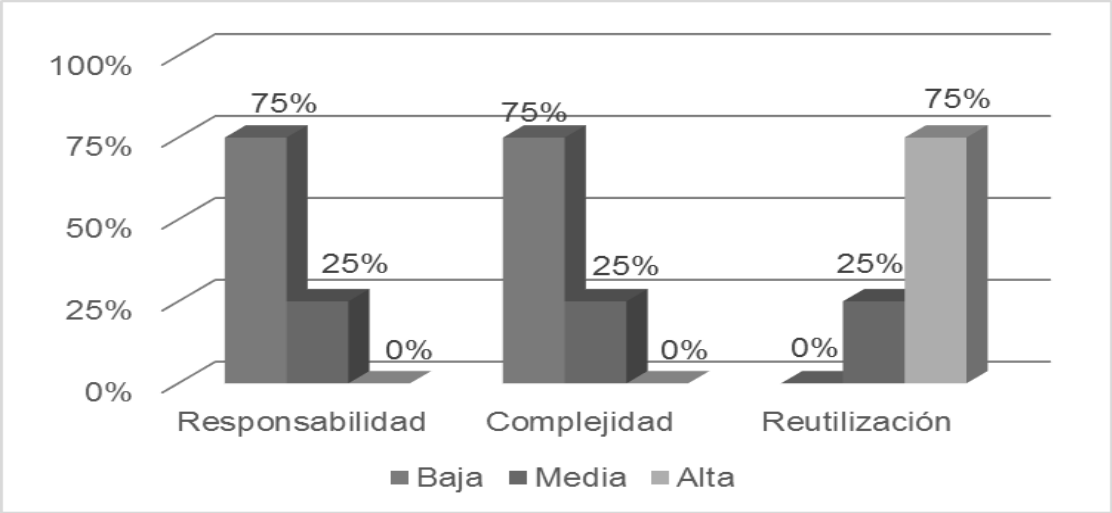


Figura 11. Resultados de la aplicación de la métrica TOC

(Fuente: elaboración propia)

La aplicación de la métrica TOC arrojó como resultados que las clases del sistema poseen un 75% de baja responsabilidad, un 75% de baja complejidad de implementación y un 75% de alta reutilización demostrando que el diseño de las clases en cuanto a cantidad de funcionalidades o procedimientos es bueno.

2.4.2 Métrica Relaciones entre Clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

- Acoplamiento: representa las conexiones físicas (colaboraciones) entre las clases del diseño. Un aumento del RC implica un aumento del Acoplamiento de la clase.
- Cantidad de pruebas: evalúa la cantidad de pruebas necesarias para probar una clase. Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar la clase.
- Complejidad de mantenimiento: mide la complejidad de mantenimiento que tiene una clase determinada. Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- Reutilización: proporciona una medida de la reutilización de una clase. Un aumento del RC implica una disminución en el grado de reutilización de la clase.

Pasos seguidos para la aplicación de la métrica

1. Se determinó la cantidad de relaciones de uso que tienen las clases del sistema.
2. Se calcula el promedio de relaciones de uso:
   Promedio = cantidad total de relaciones de uso/cantidad de clases
   Promedio = 6/4= 1.5
3. Se le asigna una categoría (Ninguno, Bajo, Medio, Alto) al atributo de calidad Acoplamiento y una categoría (Bajo, Medio, Alto) a los siguientes atributos de calidad (Complejidad de mantenimiento, Reutilización y Cantidad de pruebas) por cada una de las clases siguiendo los siguientes criterios:

Acoplamiento:

- Ninguno: si las relaciones de uso es 0
- Bajo: si las relaciones de uso es 1
- Medio: si las relaciones de uso son 2
- Alto: si las relaciones de uso son > 2

Complejidad de mantenimiento y Cantidad de pruebas:

- Baja: si las relaciones de uso son <= Promedio.
- Media: si las relaciones de uso está entre Promedio y 2* Promedio.
- Alta: si las relaciones de uso > 2* Promedio.

Reutilización:

- Baja: si las relaciones de uso son > 2* Promedio.
- Media: si las relaciones de uso están entre Promedio y 2* Promedio.
- Alta: si las relaciones de uso <= Promedio.

Resultados obtenidos

En la tabla 11 se presentan los resultados obtenidos de la aplicación de la métrica RC.

Tabla 11. Resultados obtenidos de la aplicación de la métrica RC

(Fuente: elaboración propia)

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mantenimiento	Reutilización	Cantidad de Pruebas
Operaciones	2	Medio	Medio	Medio	Medio
Preseed	1	Baja	Baja	Alta	Baja
LocalRepo	1	Baja	Baja	Alta	Baja
CrearISO	2	Medio	Medio	Medio	Medio

En la figura 12 se presentan los resultados obtenidos de la aplicación de la métrica RC.

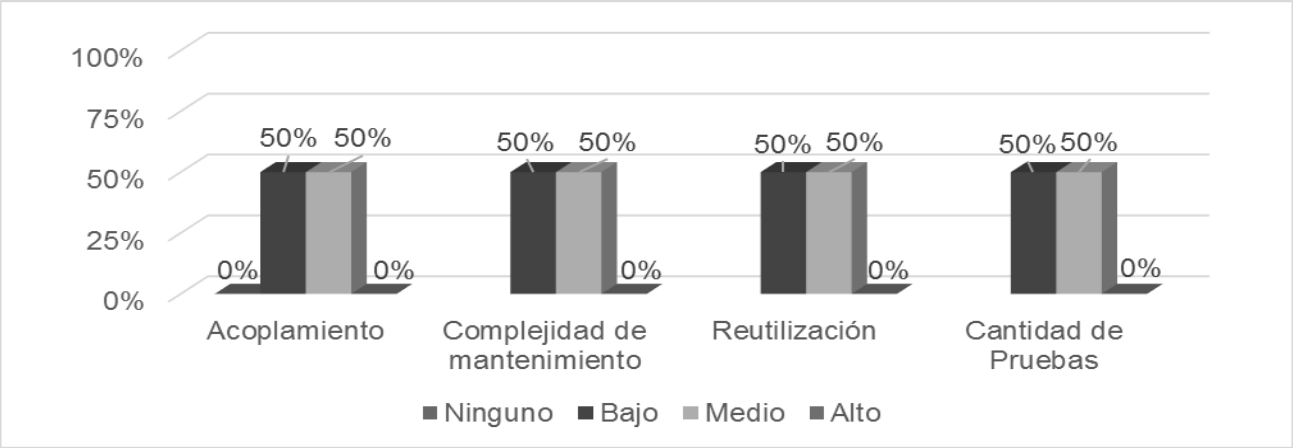


Figura 12. Resultado de la aplicación de la métrica RC

(Fuente: elaboración propia)

El resultado de aplicar la métrica RC evidencia un 50% de bajo acoplamiento, un 50 % de la baja complejidad de mantenimiento, un 50% de la baja cantidad de pruebas y un 50% de alta reutilización que poseen las clases diseñadas para el sistema.

La aplicación de las métricas TOC y RC permitió tener una medida cuantitativa del grado en el que la aplicación cumple con los indicadores de calidad anteriormente descritos. Proporciona una visión del diseño realizado en el proceso de desarrollo de la propuesta de solución y demuestra que las clases del sistema poseen una baja carga de responsabilidades, un bajo acoplamiento y una alta reutilización permitiendo una fácil implementación, una baja complejidad de mantenimiento y baja la cantidad pruebas de unidad necesarias para probar las clases, pudiéndose afirmar que los resultados obtenidos por la aplicación y evaluación de las métricas son positivos.

Conclusiones del capítulo

En el desarrollo del capítulo se transitó por las disciplinas, Modelado del negocio, Requisitos y Análisis y diseño. A través de un modelo conceptual se relacionaron los conceptos fundamentales asociados a la creación de personalizaciones, se identificaron 17 requisitos funcionales y 5 no funcionales cumpliendo con las necesidades del cliente. Se elaboraron los diagramas de clases del diseño utilizando el patrón arquitectónico MVT y los patrones de diseño GRASP y GOF quedando definida la arquitectura a seguir en la implementación de la herramienta web para la creación de personalizaciones de Nova Servidores.

Capítulo 3: Implementación, pruebas y validación de la propuesta de solución

El presente capítulo se enfoca en la construcción del sistema a partir de los resultados del Análisis y diseño. Se elaboran los diagramas de componentes y de despliegue y se define los estándares de codificación a utilizar en la implementación de la solución. Además, se realizan pruebas de software con el objetivo de descubrir y corregir errores y se evalúa la propuesta de solución.

3.1 Implementación

En la disciplina implementación, a partir de los resultados del Análisis y diseño se construye el sistema (Sánchez, 2015).

3.1.1 Modelo de implementación

El Modelo de implementación es el proceso de convertir una especificación del sistema en un sistema ejecutable. Describe cómo los elementos de diseño se implementan en componentes, estos componentes incluyen: ficheros ejecutables, ficheros de código fuente y otros tipos de ficheros necesarios para la implementación y el despliegue del sistema. En este modelo se describen las relaciones entre los paquetes y las clases el diseño con los diferentes subsistemas y componentes físicos (Cardoso, 2014).

Diagrama de componentes

Un diagrama de componentes muestra la organización, relaciones y dependencias entre los componentes de un sistema (partes modulares, desplegables y reemplazables de un sistema que encapsula implementación y expone un conjunto de interfaces) (Cardoso, 2014). A continuación se presenta el diagrama de componentes para la propuesta de solución.

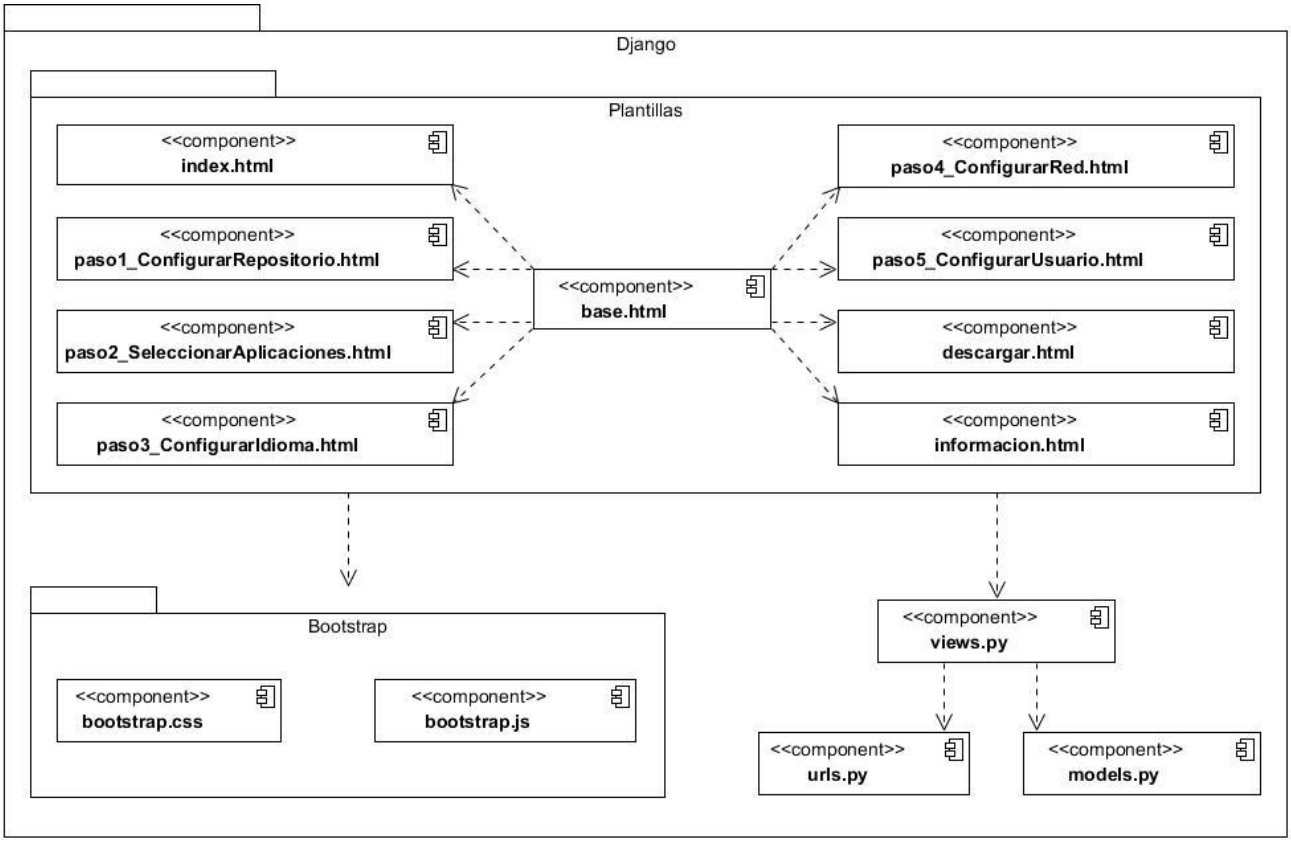


Figura 13. Diagrama de componentes

(Fuente: elaboración propia)

Descripción de los componentes

Plantillas: Contiene las plantillas o interfaces con las que interactúa el usuario.

Boostrap: Contiene los archivos .css (extensión de los archivos CSS, *Cascading Style Sheets* , Hojas de Estilo en Cascada) y .js (extensión de los archivos *JavaScript*) utilizados en las plantillas.

Views: Archivo que contiene todas las views.

Urls*:* Archivo que contiene todas las url (*Uniform Resource Locator*, Localizador Uniforme de Recursos)*.*

Models: Archivo que contiene todas las clases del modelo.

Diagrama de despliegue

El diagrama de despliegue permite modelar la disposición física de un sistema. Muestra el hardware usado y los componentes instalados en el hardware, además de las conexiones físicas entre este y las

relaciones entre componentes. Es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. El mismo está compuesto por: nodos, dispositivos y conectores (Acosta y otros, 2013). A continuación se presenta el diagrama de despliegue para la propuesta de solución.

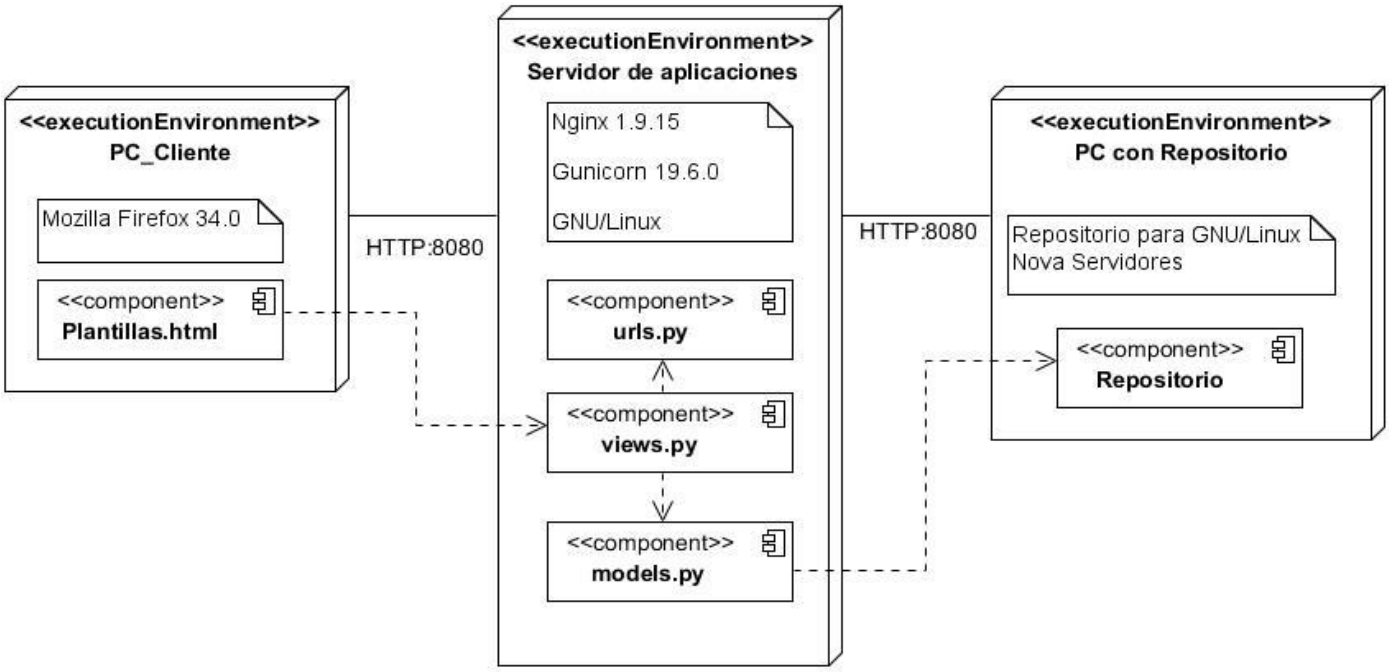


Figura 14. Diagrama de despliegue

(Fuente: elaboración propia)

Descripción de los nodos

PC_Cliente: contendrá el navegador web Mozilla Firefox 34.0 o superior mediante el cual los usuarios tendrán acceso al sistema y harán uso del mismo.

Servidor de aplicaciones: proporciona los servicios de la aplicación a las computadoras cliente y gestiona las funciones de la lógica de negocio y de acceso a los datos necesarios, los servidores de aplicaciones serán Nginx 1.9.15 y Gunicorn 19.6.0 y el sistema operativo una distribución de GNU/Linux.

PC_Repositorio: Repositorio de Nova Servidores y que proporciona los paquetes necesarios en el modelo.

3.1.2 Estándares de implementación

Para la codificación de la propuesta de solución se utiliza el estándar de codificación para Python utilizado en el Universidad de las Ciencias Informáticas. En este documento se listan distintas convenciones utilizadas en el código Python comprendido en la librería estándar de la distribución principal de Python (Dirección Técnica de la Producción, 2010).

Indentación

- Su utilizaron 4 espacios por cada nivel de indentación.

Tabuladores o espacios

- No se mezclaron tabuladores y espacios en la codificación. Las formas más populares de indentar en Python son utilizando sólo espacios o sólo tabuladores, el código indentado con una mezcla de tabuladores y espacios se reformateó y se usaron espacios exclusivamente.

Tamaño máximo de línea

- Se limitaron todas las líneas a un máximo de 79 caracteres.

Convenciones de nombres

- No se utilizaron los caracteres 'l' (letra ele minúscula), 'O' (letra o mayúscula) o 'I' (letra i mayúscula) como nombres de variables de un solo caracter debido a que en algunas fuentes, estos caracteres son indistinguibles de los numerales uno y cero.
- Se utilizó CapWords (palabras que comienzan con mayúsculas) para los nombres de las clases.
- Los nombres de funciones están en letras minúsculas, con palabras separadas mediante guiones bajos según sea necesario para mejorar la legibilidad.

Otras consideraciones

- Se rodearon siempre los siguientes operadores binarios con un espacio en cada lado: asignación (=), asignación aumentada (+=, -=,*=,/=), comparación (==, <, >, !=, <>, <=, >=, in, not in, is, is not), booleanos (and, or, not).
- Se utilizaron espacios alrededor de los operadores aritméticos.

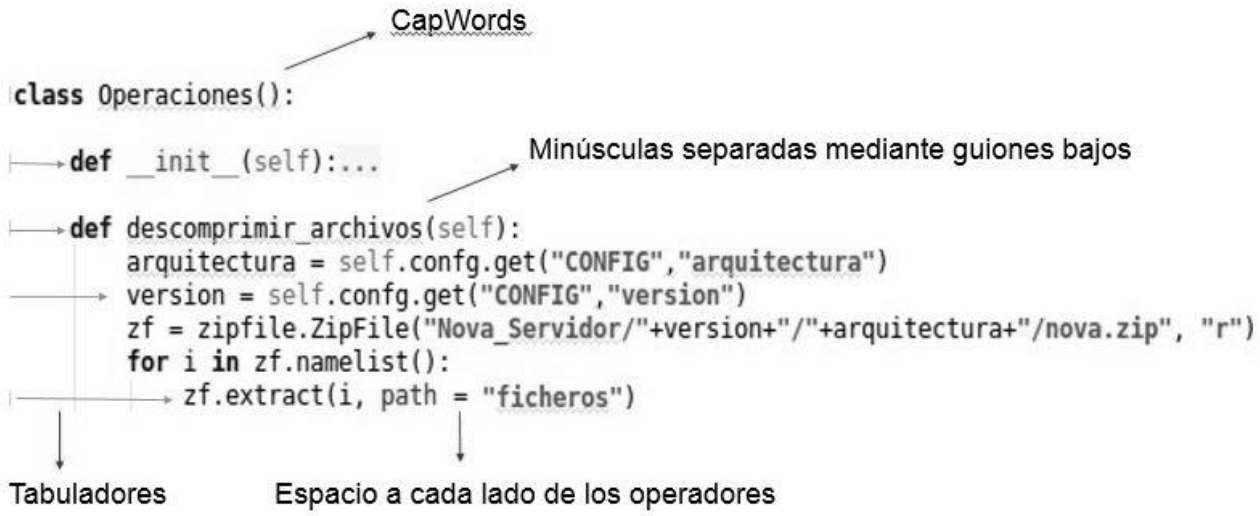


Figura 15. Aplicación de los estándares de codificación

(Fuente: elaboración propia)

3.1.3 Interfaz gráfica de usuario

La interfaz gráfica de usuario permite representar las acciones y la información disponible utilizando un conjunto de imágenes y objetos gráficos. Su función principal consiste en facilitar un entorno visual sencillo que permita la comunicación con el ordenador (Softwaredoit, 2017). Para el diseño de las interfaces gráficas de usuario de la propuesta de solución se tuvieron en cuentas las siguientes reglas.

Reglas de oro (Pressman, 2010)**:**

- Dar control al usuario: Definir los modos de interacción de forma que el usuario no realice acciones innecesarias o indeseables, proporcionar una interfaz flexible, incluir opciones de interrumpir y deshacer la interacción del usuario, ocultar al usuario ocasional los elementos técnicos internos, diseñar interacción directa con los objetos que aparecen en pantalla.
- Reducir la carga en memoria del usuario: Reducir la demanda de memoria a corto plazo, definir valores por defecto que tengan significado, definir accesos directos intuitivos, el formato visual de la interfaz debe basarse en la metáfora tomada de la realidad, desglosar la información de manera progresiva.
- Lograr que la interfaz sea consistente: Permitir que el usuario incluya la tarea actual en el contexto que tenga algún significado.

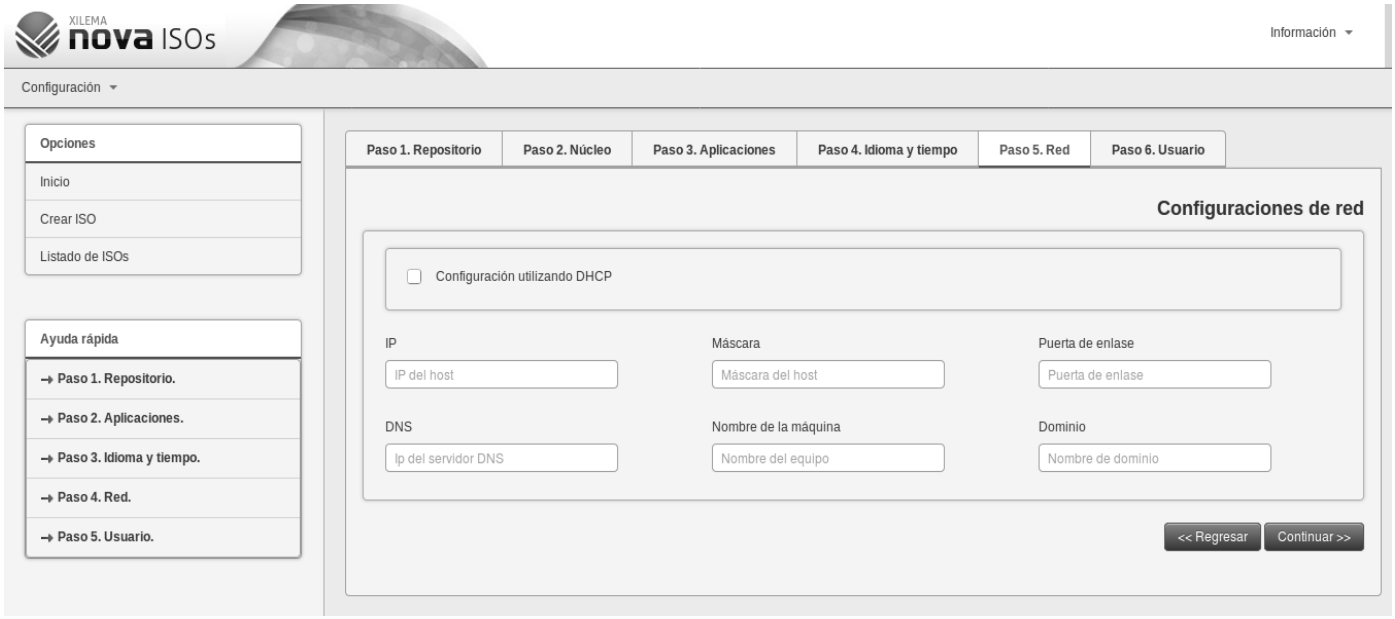


Figura 16. Interfaz de usuario para configuración de red

(Fuente: elaboración propia)

3.2 Pruebas de software

Las pruebas de software son un conjunto de actividades que se realizan con el objetivo de detectar fallos y evaluar las características del software, regulan la ejecución de los proyectos y garantizan la calidad del producto desarrollado. Incluyen técnicas y métodos específicos del diseño de casos de prueba (Pressman, 2010). Consisten en la verificación dinámica del comportamiento de un programa en un conjunto finito de casos de prueba, adecuadamente seleccionado de los posibles escenarios del sistema, para asegurarse que arroja el resultado definido en la especificación de requisitos (Granda, 2013).

3.2.1 Pruebas a realizar

Pruebas unitarias

Las pruebas unitarias se concentran en el esfuerzo de verificación de la unidad más pequeña del software, el componente o módulo de software. Tomando como guía de partida la descripción del diseño a nivel de componentes, se prueban importantes caminos de control para descubrir errores dentro de los límites del módulo. Centran su actividad en verificar la funcionalidad y la estructura (lógica interna) de cada elemento individualmente, una vez que ha sido codificado (Pressman, 2010).

Pruebas funcionales

Las pruebas funcionales se concentran en las acciones visibles para el usuario y en la salida del sistema que este puede reconocer. La validación se alcanza cuando el software funciona de tal manera que satisface las expectativas razonables del cliente (Pressman, 2010).

3.2.2 Métodos de prueba

Caja negra se enfoca en probar el sistema sin tomar en cuenta la estructura interna del mismo, su objetivo es validar que las salidas sean las esperadas. Se centra en encontrar las circunstancias en las que el sistema no se comporta conforme a las especificaciones establecidas (Campos, 2015).

Caja blanca el objetivo es analizar el objeto de prueba, ejecutando el mismo. Se llevan a cabo por medio de revisiones y herramientas (Campos, 2015).

3.2.3 Técnicas de prueba

Partición de equivalencia (posibles valores divididos en clases, valores de entrada y valores de salida). Se agrupan todos los valores para los cuales se espera que el programa tenga un comportamiento común (rango de valores) y esa es una clase de equivalencia, existen clases de equivalencia válidas y clases de equivalencias inválidas (Campos, 2015).

Camino básico permite obtener una medida de la complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de pruebas derivados para ejercitar el conjunto básico deben garantizar que se ejecuta cada instrucción por lo menos una vez durante la prueba (Pressman, 2010).

3.3 Aplicación de las pruebas de software

3.3.1 Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando los productos de trabajo implementados, a través de instrumentos de prueba como: diseños de casos de prueba y listas de chequeo (Sánchez, 2015). A continuación se presentan las pruebas realizadas en la disciplina.

Pruebas unitarias

Las pruebas unitarias se desarrollaron utilizando la técnica del camino básico del método de prueba caja blanca. En esta técnica se utilizó la métrica del software complejidad ciclomática que proporciona una medida cuantitativa de la complejidad lógica de un programa o procedimiento. Cuando se utiliza en el contexto de prueba el valor calculado mediante la complejidad ciclomática define el número de caminos independientes en el conjunto básico de un programa o procedimiento y proporciona un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman, 2010). En la figura 17 se presenta el procedimiento Guardar aplicación del RF8 Adicionar aplicación.

Figura 17. Procedimiento guardar aplicación del RF8 Adicionar aplicación

(Fuente: elaboración propia)

1. **Dibujar el grafo de flujo de la funcionalidad o procedimiento a analizar**

   En la figura 18 se utilizó la notación de grafo de flujo para representar en un grafo de flujo el código del procedimiento a probar.

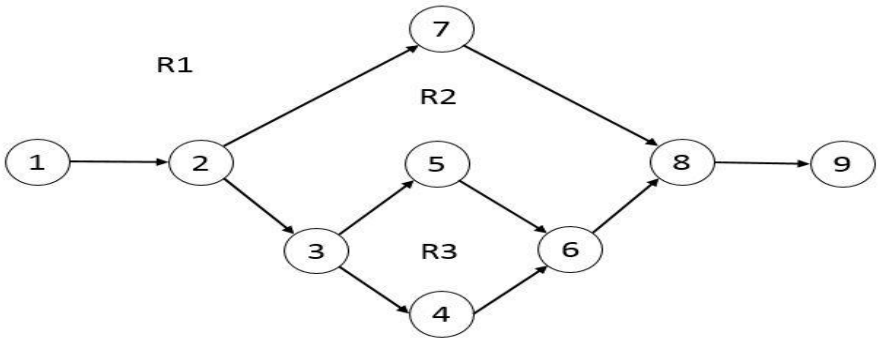


Figura 18. Grafo de flujo

(Fuente: elaboración propia)

2. Determinar la complejidad ciclomática del grafo

La complejidad ciclomática del grafo V (G) se puede calcular de las tres formas siguientes:

V(G) = A - N + 2 V(G) = 10 – 9 + 2 = 3.	Dónde: A es el número de aristas del grafo de flujo y N es el número de nodos del grafo.
V(G) = P + 1 V(G) = 2 + 1 = 3	Dónde: P es el número de nodos predicados (nodos con más de una arista de salida) contenidos en el grafo.
V(G) = R V(G) = 3	Dónde: R son las regiones, áreas delimitadas por nodos y aristas en el grafo.

3. Determinar los caminos linealmente independientes

- Camino básico 1: 1, 2, 3, 4, 6, 8, 9
- Camino básico 2: 1, 2, 3, 5, 6, 8, 9
- Camino básico 3: 1, 2, 7, 8, 9

4. Definir los casos de prueba para comprobar la ejecución de cada camino

En el diseño de los casos de prueba se debe especificar los siguientes elementos:

- Descripción: contiene una descripción sobre las restricciones de los datos de entrada que debe tener el caso de prueba.

- Condición de ejecución: se especifican los parámetros que debe poseer el caso de prueba para que se cumpla una condición deseada como respuesta del funcionamiento del procedimiento.
- Entrada: se muestran los parámetros de entrada al procedimiento.
- Resultados esperados: se explica el resultado esperado de la ejecución del procedimiento.

La Tabla 12 muestra el diseño de caso de prueba para el camino 3 del conjunto básico de caminos linealmente independientes, correspondiente a la funcionalidad Guardar aplicación.

Tabla 12. Caso de prueba para el camino 3 del procedimiento Guardar aplicación

(Fuente: elaboración propia)

Diseño de caso de pruebas para el camino 3	
Descripción	Los datos de entrada no nulos.
Condición de ejecución	No se ha adicionado ninguna aplicación con anterioridad y por tanto la sección “APP” no ha sido creada en el fichero temporal.
Entrada	app: postgresql
Resultado esperado	Se crea la sección “APP” y se guarda satisfactoriamente la aplicación en el fichero temporal.

En este camino se prueba el guardado de la primera aplicación en el fichero temporal de forma satisfactoria.

Pruebas funcionales

Se realizaron pruebas funcionales, a través del método de caja negra y la técnica de partición de equivalencia. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Se retomaron los casos de prueba diseñados en el epígrafe 2.2.5 Validación de requisitos de software, que contienen clases de equivalencias válidas y no válidas, para cada campo de entrada del sistema.

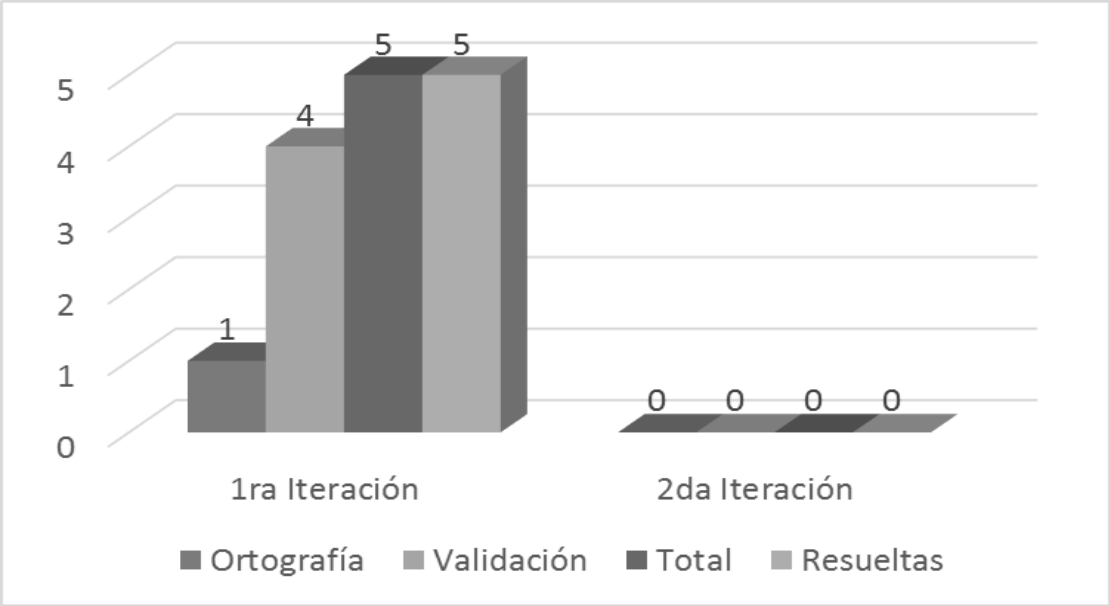


Figura 19. Resultados de las pruebas funcionales

(Fuente: elaboración propia)

Las pruebas funcionales se realizaron en dos iteraciones, en la primera se encontraron 5 no conformidades, de ellas 4 de validación y 1 de ortografía, en la segunda iteración no se encontraron no conformidades. En la Figura 19 se presenta un gráfico con los resultados de las pruebas.

3.3.2 Pruebas de aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Sánchez, 2015). El cliente realizó pruebas funcionales a través del método de caja negra y la técnica de partición de equivalencia, en las cuales no se detectaron no conformidades y emitió un acta de aceptación de productos de trabajo (ver anexo 6) en total conformidad con la herramienta desarrollada.

3.4 Evaluación del objetivo de la investigación

Cuando se realiza una propuesta, es recomendable retroalimentarse con la opinión de los usuarios potenciales. Esta información es útil para conocer las debilidades de la propuesta y profundizar en sus fortalezas. En ese sentido, **la técnica de Iadov** es un instrumento que ayuda a conocer el grado de satisfacción de los potenciales usuarios (Silega, 2014).

Iadov es una técnica efectiva para el estudio del nivel de satisfacción de los participantes a través de la consulta a un panel de experto. Este método calcula el Índice de Satisfacción Grupal (ISG) se implementa mediante un cuestionario (Ver anexo 7) en el cual se le incluyen tres preguntas cerradas que se intercalan dentro de un cuestionario de cinco preguntas y cuya relación el encuestado desconoce (Ruiz, 2016).

Estas tres preguntas se relacionan a través del "Cuadro Lógico de Iadov" el cual permite ubicar a cada encuestado, según el cuadro lógico en una escala de satisfacción, para luego calcular el ISG. La escala de satisfacción la cual toma valores de 1 a 6 es la siguiente 1-Clara satisfacción, 2-Más satisfecho que insatisfecho, 3-No definida, 4-Más insatisfecho que satisfecho, 5-Clara insatisfacción y 6-Contradictoria (Ruiz, 2016).

Tabla 13. Cuadro Lógico de Iadov

(Fuente: elaboración propia)

5. Luego de haber visto la herramienta para la creación de personalizaciones de Nova Servidores refleje en qué medida le gusta la solución desarrollada.	2. ¿La forma en que se realizan estas personalizaciones en la universidad permite satisfacer todas las necesidades existentes?								
	No			**No sé**			**Sí**		
	3. ¿Considera usted que sería factible contar con una herramienta web permita agilizar la creación de personalizaciones de Nova Servidores?								
	Sí	**No sé**	**No**	**Sí**	**No sé**	**No**	**Sí**	**No sé**	**No**
Me gusta mucho	1	2	6	2	2	6	6	6	6
Me gusta más de lo que me disgusta	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	3	4
No me gusta nada	6	6	6	6	4	4	6	6	5
No sé decir	2	3	6	3	3	3	6	6	4

El número resultante de la interrelación de las tres preguntas indica la posición en la escala de satisfacción siguiente: clara satisfacción (A), más satisfecho que insatisfecho (B), no definida (C), más insatisfecho que satisfecho(D), clara insatisfacción (E) y contradictoria (C).

A partir de la cantidad de respuestas por categoría es posible calcular el Índice de Satisfacción Grupal (ISG) siguiendo la siguiente fórmula:

$$ISG = \frac{A(+1) + B(+0.5) + C(0) + D(-0.5) + E(-1)}{N}$$

Donde N es la cantidad total de respuestas

El valor del ISG permite identificar las siguientes categorías grupales:

- Máxima insatisfacción: desde -1 hasta -0,49
- Más insatisfecho que satisfecho: desde -0,5 hasta -0,1
- No definido y contradictorio: 0
- Más satisfecho que insatisfecho: desde 0,1 hasta 0,49
- Máximo de satisfacción: desde 0,5 hasta 1

El índice general arroja valores entre + 1 y - 1. Los valores que se encuentran comprendidos entre -1 y -0,5 indican insatisfacción; los comprendidos entre - 0,49 y + 0,49 evidencian contradicción y los que están entre 0,5 y 1 indican que existe satisfacción.

Resultados obtenidos

1. Los resultados obtenidos de la aplicación de la encuesta se presentan en la Tabla 14.

Tabla 14. Resultados de la escala de satisfacción

(Fuente: elaboración propia)

Categorías grupales de satisfacción	N=5	Escala
Clara satisfacción	4	A
Más insatisfecho que satisfecho	1	B
No definido	0	C
Más satisfecho que insatisfecho	0	D
Máximo de satisfacción	0	E
Contradictorio	0	C

2. Cálculo del Índice de Satisfacción Grupal

ISG=A(+1)+B(+0.5)/N

ISG=(4(+1)+1(+0.5))/5 = 0.9

3. Interpretación del resultado del ISG.

El valor obtenido del ISG fue 0.9 lo que indica máxima satisfacción de los usuarios con respecto a la herramienta web para la creación de personalizaciones de Nova Servidores. Se puede afirmar que se cumplió el objetivo de la investigación. Las respuestas a las preguntas abiertas brindadas por los encuestados reafirman los beneficios que traerá la utilización del sistema propuesto.

Conclusiones del capítulo

En el desarrollo del capítulo se transitó por las disciplinas, Implementación, Pruebas internas y Pruebas de aceptación. La implementación permitió obtener una herramienta web que permite agilizar la creación de personalizaciones para Nova Servidores cumpliendo con los estándares de codificación definidos. Las pruebas unitarias comprobaron que el flujo de trabajo de las funcionalidades es correcto y las pruebas funcionales contribuyeron a identificar y corregir 5 no conformidades. Se evidenció la satisfacción del cliente a través de las pruebas de aceptación y la aplicación de la técnica de Iadov la cual propició la evaluación satisfactoria de la herramienta web para la creación de personalizaciones de Nova Servidores.

Conclusiones

La investigación realizada cumple con los objetivos planteados mediante el desarrollo de la herramienta web para la creación de personalizaciones de Nova Servidores y se arriba a las siguientes conclusiones:

- El análisis de los referentes teóricos y de los sistemas informáticos estudiados evidenció la necesidad de desarrollar una herramienta para agilizar el proceso de construcción de personalizaciones de Nova Servidores.
- Se obtuvo una herramienta web que permite la construcción de personalizaciones de Nova Servidores cumpliendo con los requisitos definidos por el cliente.
- La evaluación de la investigación se realizó a partir de la aplicación de técnicas, métricas y pruebas que garantizan el correcto funcionamiento de la aplicación y demostraron la satisfacción del cliente hacia el sistema desarrollado.

Recomendaciones

- Se recomienda incluir a la herramienta la personalización para otras variantes, versiones y arquitecturas de Nova.
- La creación de la personalización utilizando varios repositorios.

Referencias bibliográficas

1&1 Internet España S.L.U. 2016. Servidor web: definición, historia y programas. [En línea] 25 de 10 de 2016. [Citado el: 28 de 2 de 2017.] https://www.1and1.es/digitalguide/servidores/know-how/servidor-web-definicion-historia-y-programas/.

Acosta, Tomás y Hernández Rodríguez, Nodelvis. 2013. Módulo de reportes webmétricos para el motor de búsqueda Orión. Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas. [En línea] 2013. [Citado el: 22 de 4 de 2017.] https://repositorio_institucional.uci.cu/ jspui/handle/ident/8296.

Alvarez, Miguel Angel. 2014. DesarrolloWeb. Qué es MVC. [En línea] 2 de 1 de 2014. [Citado el: 5 de 12 de 2016.] http://www.desarrolloweb.com/articulos/que-es-mvc.html.

Analítica. 2014. Sistema de Gestión de Procesos. Manual de diagramación de procesos bajo estándar BPMN. [En línea] 2014. [Citado el: 13 de 10 de 2017.] http://www.analitica.com.co/ website/images/stories/documentosTecnicos_SGP/Manual de Diagramacion de Procesos Bajo.

Aprenderaprogramar. 2016. Programación web-app JavaScript. Librerías, frameworks. Jquery, AngularJS. Ventajas. Diferencias. [En línea] 2016. [Citado el: 8 de 12 de 2016.] http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=884:programacion-web-app-javascript-librerias-frameworks-jquery-angularjs-ventajas-diferencias-cu01194e&catid=78:tutorial-basico-programador-web-javascript-desde-&Itemid=206.

Ardila, Carlos Arberto, Pino, Francisco José y Pardo, César Jesús. 2014. Modelado del dominio. [En línea] 2014. [Citado el: 12 de 2 de 2017.] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwirl_Dy7ubTAhUl24MKHXcqATwQFggnMAE&url=https%3A%2F%2Fdialnet.unirioja.es%2Fdescarga%2Farticulo%2F5017550.pdf&usg=AFQjCNHm1xcgMCBkTtjSin2SREiFfrkCrA&sig2=OXAod44GgW-19.

Areatecnologia. 2016. Tecnologia. Lenguages de programación. [En línea] 2016. [Citado el: 8 de 12 de 2016.] http://www.areatecnologia.com/informatica/lenguajes-de-programacion.html.

Aviztar. 2017. Curso práctico de Modelado de Negocios con BPMN y UML. [En línea] 2017. [Citado el: 2017 de 2 de 9.] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm.

BBVAOpen4U. 2016. BBVAOpen4U. Django: guía rápida para desarrollar páginas web con este framework. [En línea] 2016. [Citado el: 29 de 10 de 2016.] https://bbvaopen4u.com/ es/actualidad/django-guia-rapida-para-desarrollar-paginas-web-con-este-framework.

Benguría, Sara, Belén, Martín Alarcon y Maria Victoria, Valdés López. 2010. Observación. [En línea] 14 de 12 de 2010. [Citado el: 3 de 12 de 2016.] https://www.uam.es/personal_pdi/stmaria/jmurillo/InvestigacionEE/Presentaciones/Curso_10/Observacion_trabajo.pdf.

Cabrera, Lianet. 2012. Extensión de Visual Paradigm for UML para el Desarrollo Dirigido por Modelos de aplicaiones de gestión de información. [En línea] 2012. [Citado el: 2 de 10 de 2016.] https://www.redib.org/recursos/Record/oai_articulo983403-extension-visual-paradigm-uml-desarrollo-dirigido-modelos-aplicaciones-gestion-informacion.

Campos, Cindy. 2015. Las pruebas en el desarrollo de software. Trabajo de Diploma para optar por el Título de Ingeniero en Computación. [En línea] 2015. [Citado el: 16 de 2 de 2017.] http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/7627/Las%20pruebas%20en%20el%20desarrollo%20de%20software.pdf?sequence=1.

Cardoso, Dayana y González Iznaga, Jorge Luis. 2014. Componente para la Digitalización de Contornos de Estructuras Geológicas. [En línea] 2014. [Citado el: 15 de 2 de 2017.] https://repositorio_institucional.uci.cu/jspui/handle/ident/9119.

Carmona, Armando. 2012. Desarrollo de un sistema de información web para la automatización de los procesos de control académico, gestión administrativa, emisión de reportes y pagos. Caso: unidad educativa “Juan Bautista Arismendi”. Tesis de grado presentado. [En línea] 1 de 2012. [Citado el: 20 de 11 de 2016.] http://www.miunespace.une.edu.ve/jspui/bitstream/123456789/ 1220/1/TG4700.pdf.

Chávez, Hugo. 2004. Decreto N° 3.390. [En línea] 28 de 12 de 2004. [Citado el: 29 de 5 de 2017.] http://www.wipo.int/edocs/lexdocs/laws/es/ve/ve052es.pdf.

CodeGamers. 2014. Sitio web CodeGamers | Linux: conceptos básicos. [En línea] 2014. [Citado el: 23 de 10 de 2016.] http://programandogamers.com/linux-conceptos-basicos/.

Codejobs. 2016. ¿Qué es Python? - Aprende a Programar - Codejobs. [En línea] 2016. [Citado el: 29 de 10 de 2016.] https://www.codejobs.biz/es/blog/2013/03/02/que-es-python.

Correa, Rafael. 2008. Decreto Ejecutivo Nº 1014. [En línea] 10 de 4 de 2008. https://softwarelibre.conocimiento.gob.ec/wpcontent/uploads/2016/04/Decreto_1014_software_libre_Ecuador_c2d0b.pdf.

Debian. 2016. ¿Qué es GNU/Linux? [En línea] 2016. [Citado el: 12 de 15 de 2016.] https://www.debian.org/releases/stable/mips/ch01s02.html.es.

Dirección Técnica de la Producción. 2010. Programa de mejora. Estándar de codificación para Python. Univercidad de las Ciencias Informáticas. La Habana. Cuba : s.n., 2010.

El Senado y la Cámara de Representantes de la República Oriental del Uruguay. 2014. Ley Nº 19.179. Software Libre y Formatos Abiertos en el Estado. [En línea] 8 de 1 de 2014. [Citado el: 29 de 5 de 2017.] http://antel.com.uy/wps/wcm/connect/67df2d0045ea0b53a081e68320768a44/03--ley19179.pdf?MOD=AJPERES. 28879.

Ferrer, Lautaro. 2016. Propuesta de conceptualización de requisitos para proyectos software basados en formalismo de ingeniería de conocimiento. [En línea] 2016. [Citado el: 2 de 3 de 2017.] http://revistas.unla.edu.ar /software/article/view/1279/1114.

Foreui.com. 2017. ForeUI User Manual. [En línea] 2017. [Citado el: 27 de 2 de 2017.] http://www.foreui.com/doc/html/index.html.

Gavasa, Juan. 2014. Panamericanworld. [En línea] 2014. [Citado el: 6 de 11 de 2016.] http://www.panamericanworld.com/es/articulo/software-libre-cada-vez-mas-utilizado-latino-america.

Gesoft. 2015. Personalización de software. [En línea] 2015. [Citado el: 8 de 12 de 2016.] http://www.gesoft.com.br/vista/desarrollo/personalizacion-de-software.jsp.

Gómez, Maria del Carmen. 2013. Notas del curso Base de Datos. MÉXICO, D.F : Casa abierta al tiempo, 2013.

Granda, Ailec. 2013. Modelo didáctico para el uso de comunidades virtuales en el proceso de enseñanza aprendizaje de la Disciplina Ingeniería y Gestión de Software en la Universidad de las Ciencias Informáticas. Tesis para obtener el grado de doctora en tecnología educativa. [En línea] 2013. [Citado el: 3 de 4 de 2017.] http://www.tesisenred.net/bitstream/handle/10803/127226/ tagd1de1.pdf?sequence=1.

Guerra, César Arturo. 2017. SG Buzz. Obtención de Requerimientos. Técnicas y Estrategia. [En línea] 2017. [Citado el: 13 de 02 de 2017.] https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

Guerrero, C.A, Suárez, Johanna M y Gutiérrez, Luz E. 2013. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. Colombia : s.n., 2013. ISSN 0718-0764.

Gunicorn.org. 2017. Gunicorn. [En línea] 2017. [Citado el: 7 de 5 de 2017.] http://gunicorn.org.

Infonucleo. 2017. Crear CDs de Ubuntu personalizados con Reconstructor. [En línea] 2017. [Citado el: 21 de 1 de 2017.] http://www.infonucleo.com/2010/05/13/crear-cds-de-ubuntu-personalizados-con-reconstructor/.

LibrosWeb. 2017. El patrón de diseño MTV. [En línea] 2017. [Citado el: 17 de 02 de 2017.] http://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.

Linux en español. 2017. Cómo crear tu propia distribución Linux. [En línea] 2017. [Citado el: 21 de 1 de 2017.] http://linuxespanol.arrayzone.com/2015/08/como-crear-tu-propia-distribucion-linux/.

LinuxCOE. 2014. Instalinux. Instalinux. [En línea] 2014. [Citado el: 21 de 1 de 1017.] http://www.instalinux.com/howto.php.

Linux-es. 2015. Sobre Linux | El rincón de Linux. [En línea] 2015. [Citado el: 5 de 12 de 2016.] http://www.linux-es.org/sobre_linux.

Maikel Enrique Pernía Matos, Delio Gabriel Orozco González, Alberto Miguel Nuevo Rojo. 2016. Misox: Personalización de debian GNU/LINUX. [En línea] 2016. [Citado el: 4 de 12 de 2016.] http://www.informaticahabana.cu/sites/default/files/ponencias/SWL15.pdf.

Market Share Reports. 2016. Market Share Reports. [En línea] 11 de 2016. http://netmarketshare.com/.

Maya, Esther. 2014. Métodos y técnicas de investigación. [En línea] 2014. [Citado el: 3 de 12 de 2016.] http://arquitectura.unam.mx/uploads/8/1/1/0/8110907/metodos_y_tecnicas.pdf.

Nginx.org. 2017. nginx. [En línea] 2017. [Citado el: 4 de 4 de 2017.] https://nginx.org/en/.

Pacheco, Lucy Gabriela. 2014. Desarrollo de una aplicación móvil en andorid de soporte para la prevención de recaídas en pacientes en proceso de recuperación del hospital psiquiátrico Humberto Uglade Camacho. Tesis previa a la obtención del título de Ingeniero de Sistemas. [En línea] 2014. [Citado el: 2017 de 5 de 2017.] http://dspace.ups.edu.ec/bitstream/123456789/ 6294/1/UPS-CT002853.pdf.

Peréz, S.L Peréz. 2013. Fundamentos de Ingeniería de Software. Patrones de diseño. Universidad Autónoma de México. [En línea] 2013. [Citado el: 5 de 4 de 2017.] http://computacion.cs.cinvestav.mx/~sperez/cursos/fis/PatronesDiseno.pdf.

Piña , Olga Lidia y Bringas Linare, José Antonio. 2006. La Modelación Teórica como método de la investigación científica. [En línea] 2006. [Citado el: 4 de 12 de 2016.] http://www.redalyc.org/ pdf/3606/360635561003.pdf.

Pressman, R.S. 2010. Software engineering: a practitioner's approach. 7th ed. New York : EUA, 2010. ISBN:978-0-07-337597.

Pythonízame. 2014. PyCharm Community Edition. [En línea] 2014. [Citado el: 5 de 12 de 2016.] http://pythoniza.me/pycharm-community-edition.

Reventós, Laia. 2012. El País. Linux, el triunfo silencioso. [En línea] 22 de 4 de 2012. [Citado el: 10 de 12 de 2016.] http://tecnologia.elpais.com/tecnologia/2012/04/22/actualidad/ 1335117737_156353.html.

Ruiz, María de los Ángeles. 2016. MOPIGD: Modelo para la implementación de la gestión de documentos en el sistema empresarial cubano. [En línea] 2016. [Citado el: 12 de 3 de 2017.] http://www.congreso-info.cu/index.php/info/2016/paper/viewFile/287/40.

Saldaño, Osmar Horacio. 2014. Metodología de la la investigación. [En línea] 2014. [Citado el: 22 de 1 de 2017.] https://es.slideshare.net/israelcarvajal58/tesis-gradometodologiainvestigacion23736completo-14352115.

Sánchez, Manuel Alejandro. 2015. Personalización de GNU/Linux Nova para el Sistema de Educación Primaria en Cuba. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas. [En línea] 2015. [Citado el: 2017 de 5 de 23.] https://repositorio_institucional.uci.cu/jspui/handle /123456789/7101.

Sánchez, Tamara Rodríguez. 2015. Metodología de desarrollo para la actividad productiva de la UCI. Universidad de las Ciencias Informáticas. La Habana. Cuba : s.n., 2015.

Silega, Nemury. 2014. Método para la trasformación automatizada del modelo de procesos de negocio a modelo de componentes para sistemas de gestión empresarial. Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas. [En línea] 2014. [Citado el: 20 de 1 de 2017.] https://repositorio_institucional.uci.cu/jspui/handle/ident/8584.

Softwaredoit. 2017. ¿Qué es la Interfaz Gráfica de Usuario? [En línea] 2017. [Citado el: 22 de 03 de 2017.] https://www.softwaredoit.es/definicion/definicion-interfaz-grafica-de-usuario.html.

Sommerville, I. 2011. Ingeniería de software. Novena ed. España : Pearson, 2011. ISBN:9786073206044.

Sparxsystems. 2017. Sparxsystems. Diagrama de Componentes UML 2. [En línea] 2017. [Citado el: 22 de 03 de 2017.] http://www.sparxsystems.com.ar/resources/tutorial/ uml2_componentdiagram.html.

Suárez, Yuniel. 2015. DePrograмación. Qué es un IDE? [En línea] 2015. [Citado el: 8 de 12 de 2016.] http://deprogramacion.cubava.cu/2016/02/01/que-es-un-ide/.

Targetware. Software.com.ar. Visual Paradigm para UML. [En línea] [Citado el: 29 de 10 de 2016.] http://www.software.com.ar/p/visual-paradigm-para-uml.

Ubuntu-es. 2016. Ubuntu-es. Sobre Ubuntu. [En línea] 2016. [Citado el: 8 de 12 de 2016.] http://www.ubuntu-es.org/sobre_ubuntu.

UniNotas. 2014. Revisiones Tecnicas Formales. [En línea] 2014. [Citado el: 17 de 02 de 2017.] http://www.uninotas.net/revisiones-tecnicas-formales/.

Universidad de Granada. 2016. Python | Alhambra. [En línea] 2016. [Citado el: 5 de 12 de 2016.] https://alhambra.ugr.es/index.php?page=aplicaciones/aplicaciones_detalle&area=Desarrollo&id_area=5&id_app=46&nombre=Python&ver_mas.

Universidad Politécnica de Puebla. 2016. Diseñodesistemas_iads. Caracteristicas de UML. [En línea] 2016. [Citado el: 15 de 12 de 2016.] https://sites.google.com/site/disenodesistemasiads/home/caracteristicas.

Universidad Politécnica de Valencia. 2014. Introducción a Herramientas CASE y System Architect. [En línea] 2014. [Citado el: 22 de 2 de 2017.] http://users.dsic.upv.es/asignaturas/eui/ mtp/doc-practicas/intro_case_SA.pdf.

Usemoslinux. Cómo personalizar una imagen de Ubuntu según tus necesidades | Desde Linux. [En línea] [Citado el: 29 de 10 de 2016.] http://blog.desdelinux.net/como-personalizar-una-imagen-de-ubuntu-segun-tus-necesidades/.

Villafane, Agustin. 2017. Modelo de procesos de negocio. [En línea] 2017. [Citado el: 11 de 2 de 2017.] http://strellis.com.ar/articulos/soa/modelo/.

Villafuerte, Deymor B. Centty. 2006. Manual metodológico para el investigador científico. [En línea] 2006. http://www.eumed.net/libros-gratis/2010e/816/METODO%20DEL%20ANALISIS% 20SINTESIS.htm.

Wordpress.com. 2017. Ingienería de requerimientos. El proceso de administración de requerimientos. [En línea] 2017. [Citado el: 17 de 02 de 2017.] https://laingenieriaderequerimientos. wordpress.com/el-proceso-de-administracion-de-requerimientos/.

Anexos

Anexo 1: Entrevista realizada al programador principal de Nova Servidores en el centro CESOL.

1. ¿Cree usted que son importantes las personalizaciones en Nova Servidores? ¿Por qué?
2. ¿Cuáles son los aspectos fundamentales a tener en cuenta cuando se realiza una personalización?
3. ¿Cuáles son los pasos a seguir para realizar una personalización?
4. ¿Cree necesario la creación de una herramienta que permita realizar estas personalizaciones? ¿Por qué?
5. ¿Cuáles serían las características que tendría esta herramienta?

Anexo 2: Guía de observación para el proceso de creación de personalizaciones de Nova Servidores.

Observador: Javier Piñeiro Cárdenas

Lugar: Laboratorio del proyecto Nova Servidores

Objetivo: Identificar los requisitos fundamentales para la realización del proceso de construcción de personalizaciones en Nova Servidores.

1- Datos de identificación del proceso
    - Nombre del proceso
    - Especialista que ejecuta el proceso
2- Características del espacio donde se desarrolla el proceso
    - ¿Dónde se debe realizar?
    - ¿Bajo qué condiciones se debe realizar?
    - ¿Qué se necesita para iniciar el proceso?
3- Características del proceso.
    - ¿Cómo es el proceso?
    - ¿Qué herramientas se emplean?
    - ¿Cuáles son los pasos a seguir?
    - ¿Cuál es el resultado?