Trabajo de diploma para optar por el título:

Ingeniero en Ciencias Informáticas.

Solución ejecutable para la instalación de la herramienta para repositorio institucional en distribuciones de GNU/Linux.

Autor: Daniel Vidaillet Rojas.

Tutor: Ing. Yenisel Valdés Hernández.

Co-tutor(es): Ing. Ramon Ernesto de Avila Leon.

Ing. Luis Carlos Alvarez Fernández.

Junio 2013

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Daniel Vidaillet Rojas

_____

Firma del Autor.

Ing. Yenisel Valdés Hernández

_____

Firma del Tutor.

Ing. Ramon Ernesto de Avila Leon

_____

Firma del Co-tutor.

Ing. Luis Carlos Alvarez Fernández

_____

Firma del Co-tutor.

Nombre y apellido: Yenisel Valdés Hernández
Correo: yvhernandez@uci.cu
Institución: Universidad de las Ciencias Informáticas (UCI).
Dirección: Carretera San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba, Código postal 19370.

Nombre y apellido: Ramón Ernesto de Avila Leon
Correo: redeavila@uci.cu **Situación laboral:**
Institución: Universidad de las Ciencias Informáticas (UCI).
Dirección: Carretera San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba, Código postal 19370.

Nombre y apellido: Luis Carlos Alvarez Fernández
Correo: lcalvarez@uci.cu
Institución: Universidad de las Ciencias Informáticas (UCI).
Dirección: Carretera San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba, Código postal 19370.

Dedico este trabajo, a mi mamá y mi abuela Eloísa por darme su apoyo incondicional, por estar siempre a mi lado, levantarme siempre los ánimos, por su amor, cariño, dedicación y ser todo para mí.

Gracias

Resumen.

En el Centro de Informatización de la Seguridad Ciudadana (ISEC) de la Facultad 2 de la Universidad de las Ciencias Informáticas, se está desarrollando el UCISpace que es una herramienta para repositorios que permite automatizar los procesos que se realizan en torno a la producción científica de las instituciones, facilitando el acceso de las personas de dicha institución a documentos que almacenan todas las investigaciones realizadas en el centro de trabajo. Sin embargo, el sistema cuenta con un proceso de instalación mediante consola, bastante incómodo y bastante complejo para ser ejecutado por un usuario con poco conocimiento sobre GNU/Linux. Por tal motivo se hizo necesario el desarrollo de un instalador con interfaz gráfica, que sea menos complicado para los usuarios, con mensajes sugerentes acerca de lo que ocurre durante el proceso de instalación, y que automatice gran parte de este proceso con el objetivo de hacer de la instalación, un proceso más fácil y cómodo al usuario, ya que es la primera impresión que recibe el cliente sobre el sistema a utilizar, para así poder desplegar satisfactoriamente el sistema. Mediante el desarrollo del instalador para el UCISpace, se logró hacer que el proceso de instalación, anteriormente complejo y difícil, sea una ventaja más del sistema, por su agradable interfaz gráfica y facilidades de uso.

Palabras clave: repositorios, proceso, instalación, instalador, UCISpace.

Índice de contenido.

Índice de tablas.

Índice de figuras.

Introducción.

En la actualidad se trata que las personas tengan acceso a la mayor cantidad de información posible y para leer un libro o consultar cualquier tipo de bibliografía, no sea necesario buscar un documento físico en una biblioteca. Para esto se han creado diversas aplicaciones como bibliotecas virtuales y repositorios digitales que almacenan publicaciones, revistas, o cualquier otro tipo de documentos en formato digital y que poseen o se les pueden integrar además módulos de referencias, recomendaciones, notificaciones, entre otros que facilitan la interacción con los usuarios, por estas razones en el mundo se hace cada vez más importante el uso de estos sistemas, principalmente en las instituciones que poseen bibliotecas o que procesan una gran cantidad de registros bibliográficos.

Cuba está luchando por no mantenerse al margen de esta tendencia y la Universidad de las Ciencias Informáticas desempeña un papel fundamental en este aspecto. En la universidad se maneja una cantidad elevada de registros bibliográficos, ya que la producción científica ha crecido en los últimos años, esto provocó que se hiciera necesario implantar un repositorio digital de tipo institucional, que almacena gran cantidad de las publicaciones e investigaciones realizadas por estudiantes, profesores y trabajadores de la institución.

La herramienta elegida para ser utilizada como repositorio en la universidad fue el DSpace, que es un *software* gratuito de código abierto, esta es una herramienta para repositorio institucional diseñada para almacenar la producción intelectual de una institución en formato digital, permitiendo que cualquier persona perteneciente a dicha institución acceda a estos documentos cuando los necesite. A partir de esta situación surge la idea de desarrollar una herramienta para repositorios en la universidad, con el objetivo de comercializarlo tanto nacional, como internacionalmente, esta herramienta está basada en el DSpace, se encuentra en su primera versión y tiene como nombre UCISpace.

Sin embargo, esta herramienta tiene una desventaja, y es que el proceso de instalación del UCISpace resulta bastante complejo en los sistemas operativos basados en GNU/Linux, pues se realiza a través de la consola requiriendo ejecutar un número significativo de pasos en ella, esto puede resultar complicado para los usuarios, que para poder realizar eficazmente los pasos de la instalación deben poseer al menos conocimientos medios sobre los sistemas operativos GNU/Linux, resultando casi imposible instalar el sistema para personas que no posean estos conocimientos.

Se debe tener en cuenta que una instalación exitosa es una condición necesaria para el correcto funcionamiento de cualquier *software*, además este proceso debe tratar de que sea lo más sencillo posible para el usuario, pues será su primera interacción con la aplicación. Por ejemplo, si la herramienta contiene gran cantidad de archivos, o depende de otros *softwares*, es más probable que ocurra alguna falla durante la instalación, esto será fatal aunque la falla sea solo parcial, ya que el objetivo que persigue la instalación, probablemente no pueda ser alcanzado pues la herramienta no funcionará correctamente y se corre el riesgo de que ocurran diversos errores cuando se trabaje con la misma.

Por estas razones se puede afirmar que la instalación de la herramienta para repositorios institucionales que se está desarrollando en la universidad, es un proceso engorroso debido a la cantidad de procedimientos que son necesarios, y gran parte de los usuarios encuentran difícil este modo de instalación para el sistema. Esto trajo como consecuencia que el equipo de desarrollo, buscara otra manera de instalar la herramienta debido a la importancia de simplificar el proceso de instalación para tratar de reducir la cantidad de errores que se puedan cometer en el mismo y de hacerlo un poco más cómodo, agradable a la vista, y de fácil entendimiento.

A partir de la situación planteada anteriormente, surge el siguiente **Problema de la investigación** ¿Cómo facilitar la instalación de la herramienta para Repositorios Institucionales UCISpace de la Universidad de las Ciencias Informáticas?

Se define como **Objeto de estudio**, los generadores de Instaladores, delimitando como **Campo de acción**, los generadores de Instaladores para herramientas de distribuciones de GNU/Linux basadas en Ubuntu, y como **Objetivo general** desarrollar una solución ejecutable para simplificar la instalación de la herramienta para repositorios institucionales UCISpace de la Universidad de las Ciencias Informáticas (UCI) en distribuciones de sistemas operativos GNU/Linux basados en Ubuntu utilizando tecnologías libres.

Para lograr los objetivos se proponen las siguientes **Tareas de Investigación**:

1. Revisión de bibliografía actualizada referente al tema de instaladores y estudios de los sistemas de instalación del sistema GDA eXcriba y del sistema SIGB de Bibliotex.
2. Realización de un estudio sobre los principales sistemas de diseño de instaladores de aplicaciones.
3. Selección de la metodología y tecnología a usar durante el desarrollo del instalador.
4. Generación de los artefactos propuestos por la metodología durante el análisis para desarrollar el instalador.
5. Realización del diseño de un sistema que realice la instalación de la herramienta para repositorios institucionales UCISpace.
6. Implementación de sistema que realice la instalación de la herramienta para repositorios institucionales UCISpace.
7. Definición de las pruebas a realizarle al instalador.
8. Realización de pruebas al instalador.

Para dar cumplimiento a las tareas planteadas anteriormente se utilizarán diferentes métodos de la investigación.

Métodos Teóricos:

Analítico-Sintético: Este método permite analizar y comprender la teoría y documentación analizando instaladores con el objetivo de extraer elementos importantes que se relacionen con el objeto de estudio y así poder alcanzar los conocimientos necesarios para el desarrollo de este trabajo.

Histórico-Lógico: Se utiliza este método con el fin de conocer los antecedentes, funcionamiento de los instaladores, las necesidades que sustenta, se analiza la lógica interna de su desarrollo y se expresa la esencia del objeto de estudio.

Métodos Empíricos:

Observación: La utilización de este método permite estudiar otros trabajos que están estrechamente relacionados con los instaladores o tienen elementos en común con la investigación.

Justificación de la investigación:

Obtener una solución ejecutable capaz de realizar una instalación de la herramienta para repositorios de la Universidad de las Ciencias Informática de nombre UCISpace, para que el proceso de instalación sea más sencillo.

Estructura capitular:

El presente trabajo está estructurado en introducción, 3 capítulos, conclusiones, recomendaciones, glosario de términos, referencias bibliográficas y bibliografías consultadas, que incluyen toda la investigación sobre temas referentes a los instaladores, el desarrollo del instalador que será utilizado para instalar la herramienta para repositorios institucionales UCISpace y las pruebas realizadas al mismo para comprobar su correcto funcionamiento. A continuación se hace una breve descripción de cada uno de los capítulos.

Capítulo 1: Fundamentación teórica. El objetivo del capítulo es analizar los generadores de instaladores para aplicaciones de GNU/Linux existentes en la actualidad, contiene lo referente a los elementos teóricos que soportan la investigación. Además, describir las tecnologías y la metodología que se utilizan en el desarrollo del presente trabajo.

Capítulo 2: Propuesta de solución del instalador del UCISpace para distribuciones de GNU/Linux. El objetivo de este capítulo es realizar el levantamiento de requisitos y el análisis y diseño del sistema, generando los artefactos correspondientes al Proceso de desarrollo con enfoque ágil, e implementar una propuesta de solución.

Capítulo 3: Validación de la solución propuesta. Este capítulo contiene los aspectos relacionados con las pruebas que se realizaron para validar la solución y se exponen ejemplos de pruebas ejecutadas al sistema que demuestran la integridad y aceptabilidad del mismo.

Capítulo 1: Fundamentación teórica.

1.1 Introducción al capítulo.

En el presente capítulo se hace un estudio referente a los distintos tipos de generadores de instaladores para distribuciones de GNU/Linux existentes en el mundo. De lo cual se deriva un estudio e investigación para definir las ventajas y desventajas de cada uno de ellos y si alguno cumple con las especificaciones que se necesitan para crear un instalador para el UCISpace. Se describen las tecnologías que después se utilizan en el desarrollo enfocándose principalmente en las de desarrollo libre. Se hace un estudio sobre las metodologías existentes para determinar cuál es la más factible para el desarrollo. En fin se hace un resumen de los principales conceptos relacionados con la investigación que se realiza para el desarrollo del instalador.

1.2 Estado del arte.

La instalación de un *software* es el proceso por el cual los programas son transferidos apropiadamente al computador destino, inicializados, y configurados, todo ello con el propósito de ser ya utilizados por el usuario final. Constituye la etapa final en el desarrollo del *software*.

La instalación, dependiendo del sistema desarrollado, puede consistir en una simple copia al disco rígido destino, estos casos son muy raros actualmente, o bien, más comúnmente, con una de complejidad intermedia en la que los distintos archivos componentes del *software* (ejecutables, bibliotecas, datos propios, etc.) son descomprimidos y copiados a lugares específicos preestablecidos del disco. Este último caso, comúnmente es un proceso bastante automático que es guiado por herramientas específicas (instaladores).

En productos de venta masiva tales como sistemas operativos o paquetes de oficina, las instalaciones completas son relativamente simples, y suelen ser realizadas por los propios usuarios finales con herramientas propias de instalación guiada, incluso la configuración suele ser automática. En productos de diseño específico o a medida, la instalación queda restringida normalmente a especialistas involucrados en el desarrollo del *software* en cuestión.

Una vez realizada exitosamente la instalación del *software*, el mismo pasa a la fase de producción, durante la cual cumple las funciones para las que fue realizado, es decir, que el *software* pueda ser utilizado por los usuarios finales.

Entre los tipos de instalaciones de *software* más comunes están:

- Mínima: Instala los archivos mínimos que se necesitan para poder ejecutar la aplicación, su mayor ventaja es que ocupa poco espacio en disco duro (actualmente con la capacidad de los discos duros no tiene mucho sentido utilizar este tipo de instalación salvo algunas excepciones) (1).

- Típica: Instala la mayoría de los archivos que se necesitan para poder ejecutar la aplicación, ocupa más espacio que la anterior pero normalmente no requiere el disco(CD) de instalación salvo que se utilice alguna función que no está instalada por defecto (1).
- Completa: Instala todos o al menos la gran mayoría de archivos y opciones que trae la aplicación aunque todas no son necesarias para poder ejecutar la aplicación, ocupa más espacio que las anteriores pero normalmente no requiere el CD de instalación salvo que se utilice alguna función que no está instalada por defecto (Una excepción suelen ser los juegos de ordenador que aunque se haga una instalación completa suelen requerir el CD/DVD del juego en cuestión) (1).
- Personalizada: Permite al usuario elegir los archivos y paquetes que se instalarán, con diferencia es la mejor opción ya que permite al usuario elegir las aplicaciones que necesita realmente (1).

Los procesos de instalación son diferentes en dependencia del sistema operativo que se esté utilizando, a continuación se mostrarán ejemplos de algunas de las formas en que se pueden instalar los programas en los sistemas operativos basados en GNU/Linux.

1.3 Proceso de instalación en Linux.

En los sistemas operativos GNU/Linux los programas que se instalan son un conjunto de paquetes. Esto puede parecer una desventaja, pero el sistema de paquetería le confiere mucha potencia a estos sistemas y hay muchas aplicaciones que simplifican el proceso de instalación, convirtiéndolo en un sistema de instalación de *software* simple y seguro (2).

Las principales formas de instalación en Linux son:

- **Los comandos apt-get o aptitude:** Estos son comandos que se ejecutan mediante la consola y permiten añadir y quitar aplicaciones del sistema. Se puede decir que el comando aptitude es más completo que apt-get porque recuerda las librerías descargadas y las desinstala si están en desuso. Esta es una manera de instalar programas la consola interpreta dichos comando para darle las instrucciones específicas al ordenador e instalar los programas en cuestión (3).
- **Instalando con ".deb"/".rpm":** Los ficheros con extensión ".deb" son paquetes de aplicaciones preparados para instalarse de una forma sencilla en el sistema. Lo que ocurre generalmente es que se descargan, se ejecutan y luego se da clic en instalar. Los paquetes se descargan dependiendo de la distribución de Linux que se utilice. Si se utiliza una versión basada en "Debian" se deben utilizar paquetes ".deb". En cambio si se utiliza "OpenSorce" o "Mandriva" se utilizan los ".rpm" (3).
- ***Agregar o quitar programas:*** Los centros de *software* son herramientas para Añadir/Quitar aplicaciones del sistema operativo, o sea se pueden instalar o desinstalar paquetes del sistema de forma muy sencilla (3). Solo hay que insertar el nombre del programa en el campo de búsqueda y marcarlo para instalar o para desinstalar.

- ***Instalación mediante fichero Bash:*** Bash es un programa informático cuya función consiste en interpretar órdenes. Está basado en la Shell[1] de Unix y es compatible con POSIX[2]. Fue escrito para el proyecto GNU. Su nombre es un acrónimo de Bourne-Again Shell (otro *shell bourne*), haciendo un juego de palabras (*born-again* significa renacimiento) sobre el Bourne Shell, que fue uno de los primeros intérpretes importantes de Unix. Hacia 1978 Bourne era el intérprete distribuido con la versión del sistema operativo Unix Versión 7. Stephen Bourne, investigador de los Laboratorios Bell, escribió la versión original de Bourne. Brian Fox, escribió Bash en 1987. Bash es el intérprete de comandos por defecto en la mayoría de las distribuciones de GNU/Linux, además de Mac OS X Tiger, puede ejecutarse en la mayoría de los sistemas operativos tipo Unix. También se ha llevado a Microsoft Windows por el proyecto Cygwin (4).

Existen una serie de herramientas utilizadas para facilitar todo este proceso de instalación y configuración creadas para cada *software* por separado, haciendo el proceso de instalación prácticamente de forma automática y se denominan instaladores.

1.4 Instalador.

La palabra instalar en el diccionario ABC se define como la acción de colocar, arreglar o disponer determinados elementos para que funcionen o cumplan ciertos objetivos (5).

Otra de las fuentes consultadas fue el libro Beyond Software Architecture del autor Luke Hohmann, plantea que un instalador es una herramienta que le permite al usuario final dar uso del software adicionándolo a la lista de programas que presente el sistema operativo en el cual trabaje este.

En la investigación se define instalador como una aplicación que permite copiar los archivos necesarios de los programas a instalar en el lugar adecuado. Además debe brindar la posibilidad de configurarlos y luego inicializarlos para que la aplicación funcione correctamente.

Muchas veces, un programa está formado por un conjunto de archivos, o depende de otros programas, o sea, el programa en sí mismo es uno solo, pero necesita de esos otros para poder funcionar. En muchas ocasiones, esos archivos tienen que estar colocados en diferentes partes del sistema, a veces relacionados con otros archivos y en otras los programas tienen que estar instalados en el sistema operativo para que la aplicación funcione correctamente.

Todas estas tareas se pueden hacer manualmente, pero en muchas ocasiones resultan complejas y laboriosas. Es por eso por lo que casi siempre se utiliza un instalador, que es un programa especial que realiza todas esas tareas de manera automática, usar un instalador es muy sencillo, solo se van leyendo

[1] Shell es la interfaz entre el usuario y el sistema operativo

[2] POSIX es el acrónimo de Portable Operating System Interface, y se le añade la X para que sepan que se habla de Unix.

las indicaciones que te aparecen en pantalla. Generalmente, tan solo se trata de hacer clic en el botón "Siguiente" para avanzar por los diferentes paneles del instalador. Algunos presentan la opción de instalar solamente algunos elementos, otros nos permiten cambiar la carpeta donde se copiarán los archivos de la instalación.

1.5 Generadores de instaladores.

Para crear estos instaladores existen diversas herramientas, cada una con características específicas, como por ejemplo que los instaladores creados pueden ser libres o privativos, pueden ser multiplataforma, entre otras muchas características, a estas herramientas se les llama generadores de instaladores.

1.5.1 ¿Qué son los generadores de instaladores?

Los generadores de instaladores son herramientas que permiten empaquetar las aplicaciones ya sean Web o de escritorio estableciendo una ruta donde se guarden todos los archivos de la aplicación para luego ser ejecutada por un archivo ejecutable de la misma, la mayoría de estos programas permiten generar paquetes de instalaciones para diversas necesidades por ejemplo instaladores de videos, gráficos o multimedia (6).

1.5.2 Generadores de instaladores estudiados.

Durante todo el proceso de investigación se detectaron los siguientes sistemas que son los que destacan por sus marcadas características para ser evaluados, los mismos son los que más necesidades solventan para el desarrollo del instalador del UCISpace.

IzPack.

Es un generador de instaladores para la plataforma Java. Produce instaladores ligeros que se pueden ejecutar en cualquier sistema operativo donde esté disponible una máquina virtual de Java. Dependiendo del sistema operativo, puede ser lanzado por un doble clic o un simple instalador “java-jar” desde una consola. El uso más común es para distribuir aplicaciones para la plataforma de Java, pero también se puede utilizar para otros tipos de proyectos.

El principal beneficio de IzPack es que proporciona una forma limpia y exclusiva de distribución de un proyecto para los usuarios que utilizan diferentes sistemas operativos. Se ha puesto una gran atención en hacerlo tan pequeño como sea posible a fin de reducir el peso de IzPack en el archivo de instalador final. En consecuencia, se puede especificar cómo debe ser el programa de instalación seleccionando los paneles que deseas mostrar al usuario final. A veces incluso hay varios paneles para el mismo uso, de modo que puedas elegir uno que se adapte mejor a las necesidades del usuario (7).

Estas son algunas de las principales características de IzPack:

- ✓ Integración de Apache Ant
- ✓ Incrustación del archivo de instalación utilizando un elemento de configuración
- ✓ Propiedades del sistema como variables
- ✓ Instaladores automatizados
- ✓ Imagen en el cuadro de diálogo de selección de idioma
- ✓ Modifica el cuadro de diálogo de selección de idioma
- ✓ Los paneles de IzPack
- ✓ Utiliza un panel separado de encabezado
- ✓ Cuadro de diálogo de cancelación alternativo
- ✓ Registro de la instalación
- ✓ Validadores de paquetes
- ✓ Descriptores de instalador basados en XML
- ✓ Utiliza el máximo nivel de compresión de archivos JAR
- ✓ Gran flexibilidad y modularidad, es muy fácil extender IzPack
- ✓ El código nativo no es necesario, pero se puede extender IzPack con código nativo
- ✓ Sistema de creación de accesos directos (Win32 y FreeDesktop.org)
- ✓ Panel de entrada de usuario flexible y potente
- ✓ Creación de desinstalación automática
- ✓ Fácil localización con paquetes de idioma de XML: 16 paquetes de idioma se proporcionan de forma predeterminada
- ✓ Sistema de sustitución de variables
- ✓ Interacción de secuencias de comandos específicas del sistema operativo.

CMake.

CMake es una herramienta multiplataforma de generación o automatización de código. El nombre es una abreviación para "*cross platform make*" (hacer multiplataforma), más allá del uso de "make" en el nombre, CMake es una suite separada y de más alto nivel que el sistema *make* común de Unix.

Creado por la Biblioteca Nacional de Medicina de los Estados Unidos como parte del Proyecto Visible Humano. Fue influenciado por un sistema anterior llamado pcmaker creado por Ken Martin y otros desarrolladores para soportar el grupo de herramientas de visualización, un sistema para gráficos 3D y visualización libre.

CMake es una familia de herramientas diseñada para construir, probar y empaquetar *software*. Se utiliza para controlar el proceso de compilación del *software* usando ficheros de configuración sencillos e independientes de la plataforma. Cmake genera instaladores nativos y espacios de trabajo que pueden usarse en el entorno de desarrollo deseado. Es comparable al GNU *build*

System de Unix en que el proceso es controlado por ficheros de configuración, en el caso de CMake llamados CMakeLists.txt. Al contrario que el Constructor de Sistemas GNU, que está restringido a plataformas Unix, CMake soporta la generación de ficheros para varios sistemas operativos, lo que facilita el mantenimiento y elimina la necesidad de tener varios conjuntos de ficheros para cada plataforma.

El proceso de construcción se controla creando uno o más ficheros CMakeLists.txt en cada directorio (incluyendo subdirectorios). Cada CMakeLists.txt consiste en uno o más comandos. Cada comando tiene la forma COMANDO (argumentos...) donde COMANDO es el nombre del comando, y argumentos es una lista de argumentos separados por espacios (8).

Principales características del CMake:

- Análisis automático de dependencias para C, C++, Fortran, y Java.
- Soporte para varias versiones de Microsoft Visual Studio.
- Genera ficheros para Eclipse CDT.
- Soporte para *builds* multiplataforma.

Install4j.

Install4j es un constructor de instaladores multiplataforma, es una herramienta comercial, que genera instaladores nativos y accesos directos para aplicaciones Java. Install4j es una aplicación que fue diseñada con el fin de proporcionarle un medio fácil de usar para generar sus propios instaladores personalizados, además, sobresale por su facilidad de uso y porque soporta varias plataformas (9).

Características de Install4j:

- Excepcional facilidad de uso: Todas las etapas de configuración son intuitivas y muy explícitas. Construir un instalador es una tarea que consta de pocos minutos.
- Vista enriquecida y sistema de acción: Con Install4j es posible configurar el flujo de vistas para el instalador de la forma que se desee.
- Simple creación de pantallas personalizadas: Install4j incluye un concepto único de vistas de forma que son fáciles de configurar y son de gran aceptación en su aspecto. La creación de una interfaz de usuario no es posible hacerla más fácil.
- Excelente soporte multiplataforma: Se define un instalador común para todo el proyecto y se especifica la información detallada para una plataforma en la vista.
- Amplio soporte de idiomas: Install4j soporta completamente la localización de su instalador en múltiples idiomas.
- Extensibilidad: Es posible agregar acciones propias, vistas y componentes de formularios a los registros de componentes de Install4j. Con esta flexibilidad, es posible integrar

rápidamente el código en el programa de instalación o crear extensiones que puedan reutilizase en múltiples proyectos (10).

BitRock o InstallBuilder.

Es una herramienta multiplataforma que simplifica la implementación y el empaquetado de aplicaciones. Desde un entorno de compilación único, esta herramienta de desarrollo permiten crear rápidamente instaladores multiplataforma, fáciles de usar con un aspecto nativo en todas las plataformas (11).

Estas son algunas de las características claves de BitRock o InstallBuilder:

- Integración RPM: los instaladores BitRock pueden registrar el *software* con una base de datos de paquetes RPM, combinando la facilidad de uso con la potencia de los sistemas de paquetes RPM.
- Soporte para la compilación multiplataforma: la herramienta de creación del instalador se puede ejecutar en Windows, Mac OS X, HP-UX, AIX, IRIX y GNU/Linux y generar instaladores para todas las plataformas de destino en un único archivo de proyecto.
- Facilidad de uso: InstallBuilder es fácil de aprender, de usar en un entorno gráfico de desarrollo. Se puede diseñar, construir y probar instaladores con el clic de un botón.
- Ahorro de tiempo: para usuarios avanzados, un proyecto amistoso en formato XML apoya la integración de los controles y el código, el desarrollo de la colaboración y personalización de proyectos, tanto manual como usando *scripts* externos. Una interfaz de línea de comandos le permite automatizar e integrar el proceso de construcción.
- Instaladores optimizados: los instaladores están optimizados en tamaño y velocidad y no requieren de un proceso de extracción. Esto reduce el tiempo de instalación.
- No hay dependencias externas: los instaladores BitRock multiplataforma son de un solo archivo, ejecutables nativos sin dependencias externas y una sobrecarga mínima.
- Lenguaje y Plataforma Independiente: Los instaladores BitRock pueden instalar las aplicaciones escritas en cualquier lenguaje, incluyendo: Java, PHP, Perl, C / C + + y .NET Mono.
- Integración de escritorio: los instaladores BitRock proporcionan un aspecto nativo y la integración de escritorio para Windows y Gnome.
- Opciones avanzadas de configuración: pida al usuario múltiples entradas en una sola pantalla para agilizar el proceso de instalación.
- Función de desinstalación: un programa de desinstalación se crea como parte de cada instalación, permitiendo a los usuarios desinstalar fácilmente el *software.*

- Soporte de múltiples idiomas: los instaladores BitRock soportan una gran variedad de idiomas de instalación, entre los que se incluyen inglés, alemán, japonés y español. Se puede especificar un idioma predeterminado o dejar que el usuario decida (11).

Install Simple.

Install Simple 1.45 es un programa gratuito diseñado para construir programas de instalación de forma rápida y sencilla, sin complicaciones a través de todo el proceso de creación del paquete de instalación, durante el cual solo se tiene la necesidad de ir escogiendo qué elementos se desea incluir en ese paquete, como mensaje de bienvenida, sistemas operativos necesarios, imágenes de muestra para visualizar durante el proceso de instalación, licencia del programa, creación de iconos de acceso directo y obviamente, los archivos del programa que se van a distribuir a través de Install Simple.

Una vez indicados todos los componentes del paquete, Install Simple crea el programa de instalación automáticamente, y lo ofrece listo para ser distribuido, como un paquete de instalación plenamente funcional, y de aspecto totalmente profesional.

Utilizando un asistente muy práctico e intuitivo, se podrá ir estableciendo los parámetros para el proceso de instalación de un producto para cualquier plataforma de Windows (12).

Las características principales de Install Simple son:

- Crea de forma automática accesos directos.
- Programas de desinstalación.
- Actualización de entradas del registro.
- Acuerdos de licencia.
- Mensajes a los usuarios.
- Los archivos que se incluyen en el fichero de instalación se empaquetan en alta densidad.
- Soporta cualquier versión de Windows.

InstallJammer.

InstallJammer es un generador de instaladores multiplataforma muy bueno, el cual admite temas múltiples y alto nivel de configuración, los instaladores se construyen como archivos ejecutables individuales para cada plataforma, para facilitar su distribución a través de Internet y manejar la instalación de todo lo necesario para su aplicación de una forma sencilla. Permite conocer si se encuentran instaladas o no las dependencias que necesita el *software* para un correcto funcionamiento, da información de suma importancia para tratar las dependencias necesarias para el correcto funcionamiento del sistema durante el proceso de instalación y después del mismo, ya que se pueden manipular y modificar mensajes alertas e informaciones de errores durante todo el

proceso de instalación. Los instaladores son construidos como un simple fichero ejecutable para una fácil distribución sobre la Web, instalando todo lo que se necesita para la aplicación en cualquier plataforma que se ejecute (13).

Características de InstallJammer:

- Desarrollo rápido de instaladores.
- Alto nivel de configuración, disponible en todas las etapas de construcción del instalador.
- Soporte multiplataforma.
- Creación de un instalador disponible en varios lenguajes.
- Los instaladores realizan instalaciones rápidas.
- Se crea un instalador ligero.
- Posee edición de diálogos.
- Control por línea de comandos.
- Renderizado nativo de las ventanas.
- Posibilita la creación de un desinstalador.
- Ejecución de Programas externos (eje: *Scripts Bash*)
- Detección de paquetes instalados o no en el sistema operativo usado.

Permite la realización de una aplicación con una agradable interfaz visual, con una gran facilidad de uso, entendible, ligera en cuanto al consumo de recursos del ordenador, posibilita una instalación rápida, permite ejecutar el instalador con diferentes lenguajes durante el proceso de instalación, permite detectar las dependencias instaladas o no en el sistema entre otras disímiles características. Contienen todo lo necesario para la instalación de la aplicación, con total independencia del *software* que se tenga instalado (14).

1.5.3 Generador de Instaladores a utilizar en la solución.

Durante la investigación sobre los diferentes generadores de instaladores existentes en la actualidad, se identificaron varios que por sus características fueron los que se tomaron en cuenta para ser utilizados en el desarrollo del instalador del UCISpace, estos generadores fueron descritos anteriormente en el trabajo, y se le realizó una selección rigurosa hasta encontrar el más adecuado según un grupo de parámetros seleccionados de antemano dentro de los que se encuentran:

1. *Software* libre.
2. Rápido desarrollo
3. Usabilidad.
4. Alto nivel de Configuración.
5. Uso de dependencias.

6. Detección de paquetes instalados o no en el sistema operativo usado.
7. Ejecución de *scripts bash*.
8. Creación de un desinstalador.
9. Soporte distribuciones de GNU/Linux.
10. Soporte para varios lenguajes.

Se realizó la siguiente tabla comparativa para medir los generadores estudiados y determinar el que sería utilizado para dar solución a la problemática del trabajo, teniendo en cuenta de los parámetros mencionados anteriormente los que se consideran de mayor importancia.

Instalador	**Plataforma que soporta**	**Software libre**	**Nivel de configuración**	**Soporta varios lenguajes**	**Uso de dependencias**	**Detección de paquetes**
IzPacK	Multiplataforma	Sí	Alto	Sí	Sí	No
Cmake	Multiplataforma	Sí	Bajo	Sí	No	No
Install Simple	Windows	Sí	Medio	Sí	No	No
Install4j	Multiplataforma	No	Alto	Sí	Sí	No
Installbuilder	Multiplataforma	No	Alto	Sí	No	No
InstallJammer	Multiplataforma	Sí	Alto	Sí	No	Sí

Tabla 1: Tabla comparativa.

Se obtuvieron los siguientes resultados:

Se descarta la utilización de Install4j y BitRock o Installbuilder, por ser *software* privativos, y se hace necesario para el desarrollo de la solución propuesta, un generador de instaladores de *software* libre, dado que la Universidad de las Ciencias Informáticas es una de las principales instituciones que promueve la migración del *software* propietario al *software* libre en el país.

El no uso de dependencias en la medida de lo posible, es un aspecto importante a tener en cuenta para el uso o no, de una determinada aplicación, cuantas menos dependencias esta utilice, más factible será el uso del *software* en cuestión. Atendiendo a este parámetro se define como no recomendable el uso de la aplicación IzPacK e Install4j (que ya fue descartado por ser un software privativo) dado que dependen de tener la máquina virtual de Java instalada en el ordenador.

En cuanto a usabilidad, no se toma en cuenta Cmake debido a que es una aplicación poco entendible para el usuario lo que conlleva a un desarrollo más lento del resultado esperado. Analizando el nivel de configuración permitido por las aplicaciones restantes se determina que Install Simple no cuenta

con los elementos necesarios para darle solución a la problemática planteada debido a que solo genera instaladores para Windows y la problemática de este trabajo plantea que el instalador creado debe ser para GNU/Linux.

Por todo lo anteriormente expuesto se decide que el InstallJammer es el generador de instaladores que más condiciones tiene para dar solución a la problemática de la presente investigación, dado que permite la realización de una aplicación con una agradable interfaz visual, amigable, fácil de usar, entendible, ligera para el sistema, posibilitando una instalación rápida, el conociendo dependencias instaladas o no en el sistema entre otras disímiles características, además se sobresale por su facilidad de uso, el ahorro de tiempo en la implementación, no posee dependencias externas, cuenta con múltiples modos de instalación, brinda lenguaje y plataforma independiente, integración de escritorio, posibilita opciones avanzadas de configuración, función de desinstalación y soporte de múltiples idiomas. InstallJammer se caracteriza también por ser utilizado para la construcción del instalador del ECM Alfresco y del SIGB Bibliothex.

1.6 Lenguaje de programación.

El lenguaje de programación a utilizar para la implementación del instalador a desarrollar en este trabajo es Tcl/Tk, la utilización de este lenguaje está condicionada por el sistema generador de instaladores escogido ya que este es el lenguaje predefinido para el InstallJammer, este lenguaje ofrece todas las características necesarias, aportando además las herramientas para lograr este propósito: Tcl/Tk. TCL originado del acrónimo en inglés "*Tool Command Language*" o lenguaje de herramientas de comando, actualmente se escribe como "Tcl" en lugar de "TCL", es un lenguaje de *script* creado por John Ousterhout, que ha sido concebido con una sintaxis sencilla para facilitarse su aprendizaje, sin detrimento de la funcionalidad y expresividad. Se utiliza principalmente para el desarrollo rápido de prototipos, aplicaciones "script", interfaces gráficas y pruebas, se combina con el lenguaje Tk del inglés "*Tool Kit*" para la creación de interfaces gráficas (15). Las interfaces creadas con el lenguaje Tk tienden a ser mucho más personalizables y dinámicas que las construidos con herramientas basadas en lenguajes como C o C + + y le da una buena apariencia a las herramientas creadas (16).

1.7 Metodología de investigación y desarrollo.

Inicialmente para poder entender la definición de metodología de desarrollo de *software* se hace necesario mencionar un conjunto de conceptos que ayudarán a comprender esta definición.

- Metodología: Conjunto de métodos que rigen una investigación o exposición sistemática.
- Tarea: actividad que tiene un tiempo límite de realización.
- Procedimiento: modo de ejecutar alguna actividad.
- Técnica: Conjunto de procedimientos y recursos de que se sirve una ciencia o arte. Habilidad para

ejecutar alguna actividad.

- Herramienta: instrumento con el que trabaja una persona.

Las metodologías de desarrollo de *software* surgieron por la necesidad de la industria del *software* de agilizarse y robustecerse al mismo tiempo utilizando una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un *software*. La evolución de estas metodologías siempre ha estado unida al crecimiento del desarrollo de *software* en sí mismo, estas metodologías guían el proceso de desarrollo y aportan diversas herramientas, las cuales facilitan y agilizan el proceso de la documentación y otras el proceso de desarrollo (17).

A lo largo de los años han surgido una gran variedad de metodologías. Entre las más utilizadas están la Programación Extrema (XP) que pertenece a las llamadas metodologías ágiles y el Proceso Unificado de Software (RUP) que pertenece a las llamadas metodologías tradicionales (17).

1.7.1 Metodología a utilizar.

Actualmente no existe una metodología de desarrollo de *software* que sea global, o sea, que tenga las características para que se pueda aplicar en cualquier tipo de proyecto. Las características de cada proyecto conjuntamente con su equipo de desarrollo, recursos y requisitos exigen que se escoja una que se adapte en la mayor medida posible a estas características, y después de estudiar varias de las principales metodologías de desarrollo existentes en la actualidad se descartaron para utilizar en el presente trabajo RUP ya que es de las denominadas tradicionales, porque en ella se genera una gran cantidad de artefactos, y es más recomendable para proyectos grandes que posean gran equipo de trabajo, además de requisitos y roles bien definidos.

El Centro de Informatización Universitaria (CENIA) ha definido para la realización de sus proyectos de *software*, un Proceso de desarrollo con enfoque ágil orientado al nivel 2 de CMMI el cual utiliza las buenas prácticas de las metodologías XP y SCRUM, y por tanto este proceso será utilizado para guiar el desarrollo de este trabajo.

1.7.2 Proceso de desarrollo.

Un proceso define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. En la Ingeniería del Software el objetivo es construir un producto de software o mejorar uno existente, o sea es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de *software*. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad y en consecuencia, reduce el riesgo y hace el proyecto más predecible (18).

Enfoque ágil.

El enfoque ágil surge como alternativa a las metodologías tradicionales a las que se consideran excesivamente pesadas y rígidas. Este enfoque plantea los proyectos desde el cumplimiento de un objetivo más amplio: entregar software con el mayor valor posible (19).

La Universidad de las Ciencias Informáticas no se encuentra ajena a los problemas que enfrenta la industria del *software,* por ese motivo esta institución actualmente está desplegando un proceso de mejora basado en el modelo CMMI, buscando incrementar la madurez y capacidad de la institución en la producción de *software*. El programa de mejora tiene como objetivo cumplir con las áreas de procesos del nivel 2 de CMMI, garantizando así la reducción de diversos problemas que presenta la producción de *software* en la institución (20).

Este proceso está compuesto por las metodologías XP y SCRUM, que ofrece una estrategia tecnológica a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de *software* para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo.

XP (Programación Extrema).

Programación Extrema es una metodología ágil de desarrollo de software que establece entregas frecuentes con posibilidad de reestructuración continua, permitiendo mejorar el diseño. Esta metodología está basada en un proceso simple de desarrollo, así como en la comunicación entre los clientes y los desarrolladores, permitiendo mejorar el diseño cada vez que se añade una funcionalidad. Cuenta con cuatro áreas fundamentales: planificación, diseño, desarrollo y pruebas (21).

SCRUM.

Scrum es una metodología ágil enfocada a la gestión de proyectos. Tiene como características el desarrollo de iteraciones, así como las reuniones a lo largo del desarrollo. La evolución del proyecto se define a través de reuniones diarias donde el trabajo del día anterior es revisado por el equipo, previendo además la labor a realizar el día siguiente(20).

Con la utilización de SCRUM para la gestión, se logra una buena planificación y organización, mientras que XP respalda con sus prácticas todo el proceso de desarrollo, obteniéndose de esta forma un proceso de *software* completo (20).

Este proceso está especialmente indicado para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ayuda a que trabajen todos

juntos, en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes pueden ver día a día como progresa el trabajo.

CMMI.

CMMI (*Capability Maturity Model Integration*) es un modelo de mejora de los procesos para el desarrollo de productos y de servicios. Este modelo tiene como propósito ayudar a las organizaciones a mejorar sus procesos de desarrollo y de mantenimiento, tanto para los productos como para los servicios. Además, es un modelo descriptivo que detalla los atributos esenciales que deberían caracterizar a una organización en un nivel de maduración determinado(22). CMMI tiene 5 niveles de madurez para clasificar a las organizaciones, ellos son: Inicial, Gestionado, Definido, Gestionado Cuantitativamente y Optimizado, para poder alcanzar un nivel superior, se debe cumplir con todas las áreas del nivel inferior (21).

Para alcanzar el nivel 2 de CMMI es necesario realizar cambios en la forma de trabajar de las empresas, los proyectos son controlados y gestionados durante su desarrollo. La disciplina del proceso reflejada por el nivel de madurez 2 ayuda a asegurar que existen prácticas, y los proyectos son realizados y manejados de acuerdo con los planes documentados. En el nivel 2 el estado de los artefactos y la entrega de los servicios siguen planes definidos, además satisfacen sus descripciones especificadas, estándares y procedimientos(21).

1.8 Herramientas CASE.

La Ingeniería de Software Asistida por Computadora, conocidas sus herramientas como CASE, permite diseñar completamente el *software* de forma que provea ahorro de tiempo y recursos humanos en lugar de proceder a implementar el *software* directamente, utilizando esta ingeniería se evita, inclusive, realizar regresiones propias por concepto de diseño, de estas herramientas se presentan las de mayor impacto en el medio informático por su calidad y utilidad. Estas herramientas tienen como objetivo principal computarizar y apoyar una o más fases del ciclo de vida del perfeccionamiento de sistemas, acelerando el proceso, siendo además la aplicación de conocimientos informáticos a las actividades, las técnicas y las metodologías de desarrollo(23).

Ventajas de las herramientas CASE:

- Aumento considerable en la velocidad de construcción de los sistemas.
- Los analistas se benefician de más tiempo para el análisis y diseño, recortan además el tiempo para codificar y probar.
- Automatiza el esbozo de diagramas.
- Auxilian en la documentación del sistema.
- Ayudan en la creación de relaciones en la base de datos.

- Aprueba generar estructuras de código

Hoy en día estas herramientas, acompañadas por una metodología y algún lenguaje de modelado, son utilizadas como punto clave en el desarrollo de cualquier *software*, pues admiten la creación y modificación de diagramas con gran facilidad y sencillez, aumentando en gran escala la calidad de los diseños de *software*(23).

Estas son las que se tuvieron en cuenta para utilizar en el desarrollo del presente trabajo de diploma:

1.8.1 Visual Paradigm 8.0.

Es un *software* de modelado que utiliza UML como lenguaje para el modelado. Soporta el ciclo completo de vida del *software*, análisis y diseño orientados a objetos, implementación, pruebas y despliegue, permitiendo en cada una de sus etapas generar los diagramas necesarios sin ningún tipo de problema, está orientado para posibilitar tanto ingeniería directa como inversa, pues posee varios lenguajes de programación que aprueban la generación de código. El *software* de modelado UML ayuda a una rápida construcción de aplicaciones de calidad a menor costo(24).

Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y también la documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML(24).

Esta herramienta CASE soporta la importación y exportación de varias versiones de XML. Facilita la modelación de diversos tipos de diagramas, transformando códigos de estos modelos, concibiendo de esta manera, los códigos fuentes de los diagramas(24).

Algunas características de Visual Paradigm:

- Posee generación de código para Java y la exportación de todos los diagramas a formato ".html "y ".jpg."
- Posee un medio de creación de diagramas para UML 2.0.
- Posibilita la integración con los principales Entornos de Desarrollo Integrado.
- Cuenta con un diseño enmarcado en casos de uso y dirigido al negocio.
- Contiene facilidades para representar especificaciones de casos de uso del sistema.
- Este se emplea como herramienta para el modelado de diagramas debido a que es multiplataforma. Posibilita la transformación de diagramas de entidad-relación en tablas de base de datos. Posee además una distribución automática de diagramas, ya que cuenta con una reorganización de las figuras y conectores de los diagramas UML. También permite exportar los diagramas a imágenes y páginas ".html. " Facilita además la conversión de diagramas de colaboración a secuencia y viceversa.

1.8.2 Rational Rose 7.0.

Rational Rose es una herramienta de diseño orientada a objetos, que da soporte al modelado visual, es decir, que permite representar gráficamente el sistema, permitiendo hacer énfasis en los detalles más importantes, centrándose en los casos de uso y enfocándose hacia un *software* de mayor calidad, empleando un lenguaje estándar común que facilita la comunicación. Proporciona mecanismos para realizar la ingeniería inversa, es decir, que a partir del código se pueda obtener información sobre su diseño, adicionalmente permite generar código en diferentes lenguajes a partir de un diseño en UML, brinda la posibilidad de que varias personas trabajen a la vez, permitiendo que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y permite que tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo. El desarrollo es un proceso iterativo, que comienza con una aproximación del análisis, diseño e implementación para identificar los riesgos y probar el sistema, cuando la implementación pasa todas las pruebas que se determinan, se añaden los elementos modificados al modelo y una vez modificado el modelo se realiza la siguiente iteración. Rational además, soporta los diagramas de UML, excepto los Diagramas de Implementación (25).

Entre las características principales de Rational se pueden destacar:

- Admite como notaciones: UML, OMT y Booch.
- Permite desarrollo multiusuario.
- Genera documentación del sistema.
- Disponible en múltiples plataformas.

1.8.3 Herramienta CASE que se utilizará para el diseño del instalador.

Sin lugar a dudas las herramientas CASE han venido a revolucionar la forma de automatizar los aspectos claves en el desarrollo de los sistemas de información, debido a la gran plataforma de seguridad que ofrecen a los sistemas que las usan y es que estas, brindan toda una gama de componentes que incluyen todas o la mayoría de los requisitos necesarios para el desarrollo de los sistemas. La escogida para utilizarse en ente trabajo es Visual Paradigm, el cual posee características que lo hacen destacar sobre sus competidores, da soporte para la creación de proyectos con trabajos realmente grandes, además de su potente integración con el lenguaje de modelado UML, el cual será utilizado en el proceso de modelación, análisis y diseño del *software* instalador.

1.9 Conclusiones del Capítulo.

Presentada la fundamentación teórica, que guiará el proceso de desarrollo del instalador para distribuciones de GNU/Linux basadas en Ubuntu de la herramienta para repositorios *UCISpace*. Se seleccionaron las diferentes herramientas y tecnologías a utilizar para el desarrollo del instalador, justificando su selección, se utilizará el InstallJammer para crear un instalador con interfaz gráfica que facilite la copia del paquete de instalación de la herramienta, y configura archivos específicos de dicho paquete para el correcto funcionamiento del sistema, además quedó definido que el desarrollo del *software* estará guiado por el Proceso de desarrollo con enfoque ágil orientado al nivel 2 de CMMI, el cual utiliza las buenas prácticas de las metodologías XP y SCRUM.

Capítulo 2: Propuesta de solución del instalador del UCISpace para distribuciones de GNU/Linux.

2.1 Introducción al capítulo.

En el presente capítulo se aborda el desarrollo de la solución propuesta, guiado por el Proceso de desarrollo con enfoque ágil orientado al nivel 2 de CMMI, el cual utiliza las buenas prácticas de las metodologías XP y SCRUM. Se presentaron los principales artefactos generados en las primeras fases de dicha metodología, donde se definen un conjunto de actividades de las que se obtienen los requisitos, el diseño, las tareas a realizar durante la implementación y la obtención del *software*.

Se ofrece una valoración del diseño propuesto para los cambios necesarios en la transición del diseño a la implementación, teniendo en cuenta los requerimientos funcionales que especifican las condiciones que el sistema debe cumplir, representadas a través de los requisitos funcionales y los no funcionales que describen las características y restricciones que debe cumplir el sistema para su correcto funcionamiento.

2.2 Concepción del sistema.

Todo proyecto al comienzo y para un correcto desempeño necesita una buena planificación, la cual se logra mediante encuentros entre el cliente y el equipo de desarrollo. De estos encuentros se obtiene una visión general del producto a desarrollar y un conjunto de informaciones muy importantes, que dan paso al inicio de la solución propuesta.

2.2.1 Descripción de la propuesta de solución.

Como se ha expuesto hasta este punto de la investigación, para dar solución al problema tratado en la presente investigación se propone el desarrollo de una aplicación que permita realizar una instalación del UCISpace fácil de usar, diseñada completamente por las tecnologías y herramientas expuestas en el capítulo 1, permitiendo una instalación cómoda, agradable a la vista y de fácil entendimiento.

El instalador debe poder ser utilizado por cualquier persona incluso aunque no posea grandes conocimientos sobre informática, ajustando el sistema a los requerimientos del administrador en cuestión, dándole una información del estado de la instalación del sistema durante toda la duración de la misma.

Entre las principales acciones que debe permitir el instalador se encuentran: escoger, qué idioma se va a utilizar durante el proceso de instalación (español e inglés), seleccionar la dirección donde se van a copiar las carpetas y demás archivos de la instalación, crear el usuario de PostgreSQL que utilizará el sistema para administrar la base de datos, también crear dicha base de datos que será utilizada para almacenar los datos del UCISpace y creación del usuario de Unix y hacerlo administrador de la computadora, editar los archivos del *Tomcat* y de los paquetes de instalación del UCISpace, compilar el sistema y crear una administrador del UCISpace.

2.3 Modelo de dominio.

En dependencia de la situación o escenario se determina si es posible realizar un modelo del negocio completo o de lo contrario se procede a definir el modelo conceptual o modelo de dominio. Debido a la poca estructuración de los procesos del negocio que tienen que ver con el objeto de estudio, se describe el funcionamiento de la aplicación mediante un modelo de dominio.

En este trabajo el proceso de negocio no está bien definido por lo que se define un modelo de dominio, el mismo representa las clases conceptuales u objetos del mundo real en un dominio de interés, siendo su función principal ayudar a entender el problema a tratar. Este se crea con el objetivo de ayudar a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la solución.

Seguidamente se muestra el modelo de dominio de la propuesta:

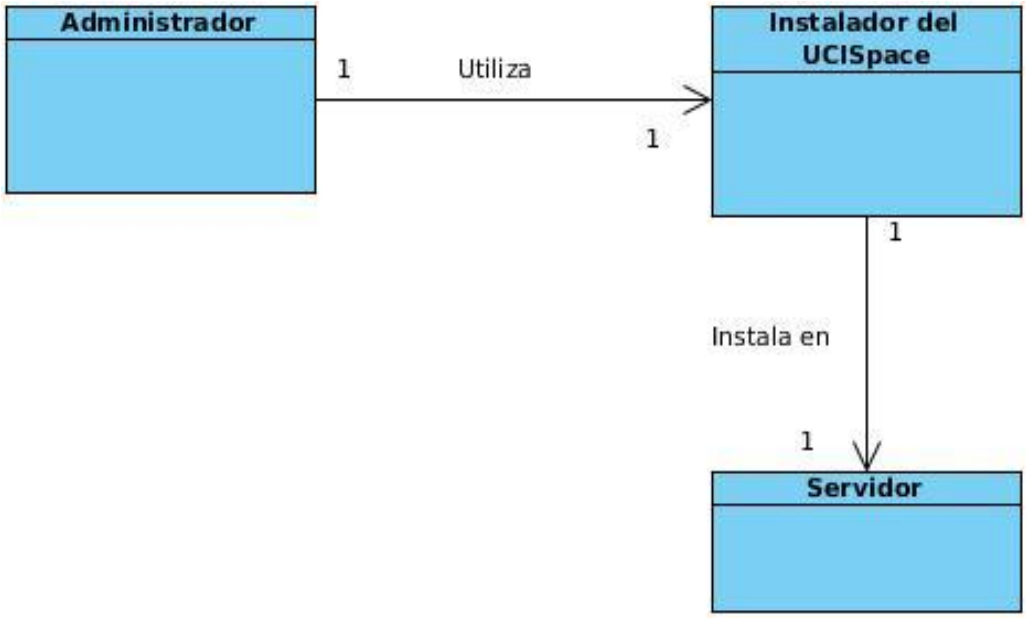

Ilustración 1: Modelo de dominio de la propuesta de solución.

2.3.1 Conceptos del modelo del dominio.

Administrador: Es el encargado de instalar la aplicación del UCISpace en un servidor apto para dicha operación, configurando el instalador correctamente para lograr una operación satisfactoria. Luego de instalado el sistema procederá a realizar las operaciones pertinentes para dejar al UCISpace ejecutándose con todas sus funcionalidades en estado óptimo.

Instalador UCISpace: Es el *software* que va a instalar el UCISpace en el servidor seleccionado, posibilita una configuración básica utilizada por el sistema tales como: base de datos a utilizar por el

sistema, contraseña del usuario de administración del PostgreSQL y puertos a utilizar para los diferentes servicios que brinda el UCISpace.

Servidor: Es la computadora que reune las condiciones de *hardware* y *software* óptimas o al menos las necesarias para soportar al UCISpace ejecutándose y brindando los servicios para los que está creado.

2.4 Captura de Requisitos.

Para el Análisis y Diseño de un sistema se toma como punto de partida la captura de requisitos, la definición de estos es un paso muy importante para el desarrollo de las siguientes etapas, pues un error en esta fase inicial puede traer consigo la implementación de un sistema que no cumpla las expectativas y en el peor de los casos no aporte valor agregado al negocio para el que debe ser concebido. Para esta operación se parte de un previo encuentro con los clientes para establecer los requerimientos funcionales, por los cuales se rige el equipo de desarrollo, en la confección de las historias de usuarios para cada uno de los requisitos, según su prioridad.

2.4.1 Requisitos funcionales.

Luego de una investigación sobre el objeto de estudio, se analiza qué debe hacer el sistema para darle cumplimiento a los objetivos planteados. Para ello se enumeran a través de requisitos funcionales que son capacidades o condiciones que el sistema debe cumplir. De acuerdo con los objetivos planteados, los requerimientos funcionales más importantes del sistema propuesto son:

RF1: Seleccionar el idioma del proceso de instalación.

RF2: Detección de los paquetes necesarios para la instalación del sistema.

RF3: Seleccionar la ruta de la instalación.

RF4: Configuración del Tomcat para que pueda ser utilizado por el UCISpace.

RF5: Configurar archivos del UCISpace.

RF6: Compilar el sistema.

RF7: Crear un usuario administrador del UCISpace.

2.4.2 Historias de usuario.

Para el desarrollo del sistema se hace uso del Proceso de desarrollo con enfoque ágil orientado al nivel 2 de CMMI, integrando las metodologías XP y SCRUM, el cual, para describir los casos de uso como correspondería en RUP, lo hace mediante historias de usuarios, descriptoras de las tareas que el sistema debe hacer, cuestión que depende en gran medida de las especificaciones realizadas por el cliente. Se escriben con un lenguaje natural y con palabras concisas para no exceder su tamaño en unas pocas líneas de texto. Estas van a ser la guía para la posterior construcción de las pruebas de

aceptación, comprobando de esta manera, la correcta implementación de las historias de usuario. Cada una de ellas debe ser comprensible y delimitada hasta el punto de permitirle al programador o programadores, su desarrollo en unas pocas semanas. Se descomponen en tareas que son asignadas al personal correspondiente dependiendo del tipo ya sea una tarea de análisis o de desarrollo, esto se hace con el objetivo, de que su duración no pase de una iteración. A continuación se exponen las historias de usuario correspondientes a la propuesta del sistema a desarrollar, las cuales se arrojaron de una planificación inicial que pueden ir cambiando para adecuarse a las necesidades del cliente.

Historias de usuario:

Historia de Usuario	
Número: HU_1	**Nombre Historia de Usuario:** Selección del idioma a utilizar durante el proceso de instalación.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Daniel Vidaillet	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Se le da al usuario la posibilidad de seleccionar un idioma de los que tiene disponible el instalador para que sea utilizado durante el proceso de instalación, toda la información que reciba el usuario a partir de la próxima interfaz debe estar escrita en ese idioma.	
Observaciones: Ninguna	

Tabla 2: Historia de usuario: Selección del idioma a utilizar durante el proceso de instalación.

Historia de Usuario	
Número: HU_2	**Nombre Historia de Usuario:** Detección de los paquetes necesarios para la instalación del sistema.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Daniel Vidaillet	**Iteración Asignada:** 1

Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El sistema detectará si los *software* de los que depende el sistema y que son necesarios para el funcionamiento correcto del mismo (PostgreSQL, Tomcat6 y el Ant) están instalados o no.	
Observaciones: Ninguna	

Tabla 3: Historia de usuario: *Detección de los paquetes necesarios para la instalación del sistema.*

Historia de Usuario	
Número: HU_3	**Nombre Historia de Usuario:** Seleccionar la Ruta de instalación.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Daniel Vidaillet	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El sistema dará la posibilidad de especificar la ruta donde serán copiados los archivos de instalación del sistema, esto se hará con la ayuda de un formulario.	
Observaciones: Ninguna	

Tabla 4: Historia de usuario: Selección de la ruta de la instalación.

Historia de Usuario	
Número: HU_4	**Nombre Historia de Usuario:** Configuración de archivos del Tomcat.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Daniel Vidaillet	**Iteración Asignada:** 1

Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: La aplicación modificará varios archivos del Tomcat además de hacer al usuario dspace propietario de estos archivos para que puedan ser utilizados por el sistema.	
Observaciones: Ninguna	

Tabla 5: Historia de usuario: *Configuración de archivos del* Tomcat.

Historia de Usuario	
Número: HU_5	**Nombre Historia de Usuario:** Configurar archivos del UCISpace.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Daniel Vidaillet	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El usuario deberá llenar los campos de texto en las interfaces pertenecientes al proceso de configuración del archivo “dspace.cfg” para configurar dicho archivo con los valores insertados en estos campos.	
Observaciones: Ninguna	

Tabla 6: Historia de usuario: *Configuración archivos del UCISpace.*

Historia de Usuario	
Número: HU_4	**Nombre Historia de Usuario:** Compilar el sistema.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Daniel Vidaillet	**Iteración Asignada:** 2

Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Se ejecutará el archivo "INSTALL.sh" el cual mediante el comando "ant fresh_install" inicializará la base de datos y compilará el sistema.	
Observaciones: Ninguna	

Tabla 7: Historia de usuario: *Compilar sistema.*

Historia de Usuario	
Número: HU_7	**Nombre Historia de Usuario:** Crear un usuario administrador del UCISpace.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Daniel Vidaillet	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El Luego de compilar el sistema el archivo "INSTALL.sh" procederá a crear un administrador del UCISpace mediante el comando "create-administrator".	
Observaciones: Ninguna	

Tabla 8: Historia de usuario: *Creación un usuario administrador del UCISpace.*

2.4.3 Requisitos no funcionales.

Los requerimientos no funcionales son cualidades o propiedades que el producto debe tener. No son más que restricciones impuestas al producto que está siendo desarrollado. Estos requerimientos no se dedican a describir la funcionalidad del software, sino como la ejecutará, de ahí su importancia, ya que permite al cliente evaluar y valorar las características no funcionales del producto como usabilidad, seguridad, rapidez, confiabilidad entre otros, funcionalidades que pueden establecer la diferencia entre un producto bien aceptado y uno con poca aceptación.

Seguridad

RNF1 - El instalador solo podrá ser ejecutado por un usuario administrador del ordenador.

Restricciones de diseño

RNF2 – El lenguaje de programación a utilizar es TcL/Tk.

RNF3 – El generador de instaladores a utilizar es InstallJamer.

Software

RNF4 – Debe estar instalado el servicio Tomcat.

RNF5 – Debe estar instalado el servicio Ant.

RNF6 – Debe estar instalado el gestor de base de datos PostgreSQL.

RNF7 - Disponer del sistema operativo Linux Ubuntu en cualquiera de sus versiones.

Hardware

RNF8 - El sistema necesitara 1 GB de memoria RAM.

RNF9 - 2 GB de espacio libre en la partición del disco duro en la que será instalada la aplicación.

2.6 Arquitectura del sistema.

La arquitectura es el esqueleto o base de una aplicación, en esta se analiza el sistema desde varios puntos de vista, brindando una clara perspectiva del mismo, necesaria para controlar el desarrollo. Esta abarca decisiones importantes sobre la organización del sistema, elementos estructurales que lo compondrán, sus interfaces y comportamiento.

Habitualmente en los sistemas se emplean un conjunto de arquitecturas tales como: arquitectura basada en servicios (SOA), arquitectura basada en objetos, arquitectura basada en capas, modelo vista controlador (MVC), entre otros. Seleccionar una arquitectura que guíe el proceso de desarrollo del *software* es imprescindible, por lo que en la solución propuesta se hace necesario la utilización del patrón arquitectónico: N Capas que contara con dos capas:

- Capa de interfaz: Esta se encarga que el sistema interactúe con el usuario y viceversa, muestra el sistema al usuario, le presenta la información y obtiene la información del usuario. Dentro de este nivel han de crearse todas las interfaces, con las que el usuario va a interactuar. Esta capa se comunica con la capa de negocio.

- Capa de lógica de negocios: Esta capa contiene las funciones que se ejecutan, se reciben las peticiones del usuario, se procesa la información y se envían las respuestas de ser necesario tras el proceso. Se denomina capa de negocio o capa de lógica del negocio, porque es aquí donde se establecen todas las reglas que deben cumplirse y ha de garantizar todo lo referente a la existencia de los archivos que contiene el UCISpace dentro de su instalador. Aquí se crean todas las acciones que son imprescindibles para el funcionamiento del instalador. Esta capa se comunica con la de interfaz.

2.5 Diagrama de componentes.

Para lograr una organización lógica de la implementación de un sistema se hace necesario del Diagrama de Componentes el cual se conforma de componentes y dependencias entre ellos. Representa todos los tipos de elementos software que entran en la fabricación de la aplicación informática, desde simples archivos y paquetes hasta bibliotecas cargadas dinámicamente. Este diagrama describe los elementos físicos del sistema y sus relaciones.(26)

Su objetivo es proporcionar a todo el equipo una misma visión del fin del sistema y de su arquitectura en general. Con ello se facilita que todos los desarrolladores hablen un mismo idioma y que nuevos desarrolladores lo adquieran más rápido para poder integrarse en el proyecto sin dificultades.

A continuación se muestra el diagrama de componentes correspondiente al instalador del UCISpace:

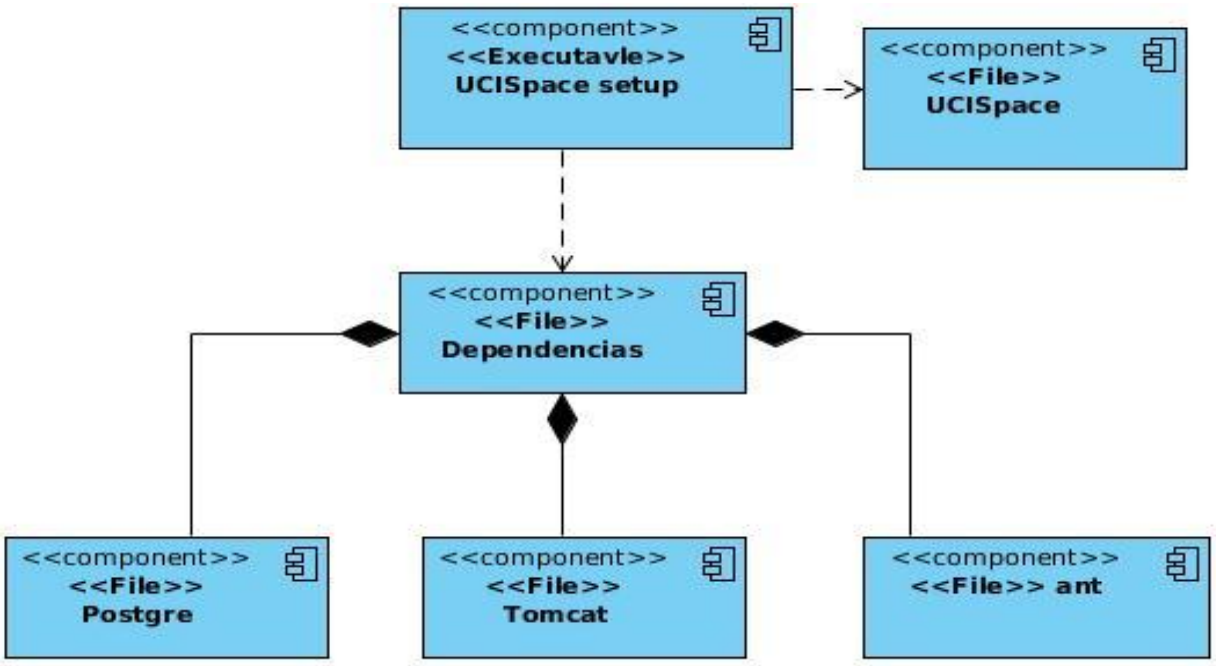

Ilustración 2: Diagrama de componentes para el instalador del UCISpace.

2.7 Tareas de ingeniería.

A continuación se definen cada una de las tareas que se encuentran asociadas a las historias de usuario. En las que se dan a conocer una serie de datos entre los que se encuentran el tiempo y el desarrollador asignado a cada una de las historia de usuario, lo que se refleja en una mejor estimación de los recursos y el tiempo que consumirá cada historia de usuario de acuerdo con su complejidad.

Tareas de ingeniería:

Tarea de ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** HU_1
Nombre Tarea: Selección del idioma a utilizar durante el proceso de instalación.	
Tipo de Tarea : Desarrollo	
Fecha Inicio: 1/02/2013	**Fecha Fin:** 8/02/2013
Programador Responsable: Daniel Vidaillet.	
Descripción: Configuración de los idiomas con que va a contar la aplicación durante el proceso de instalación, para que toda la información mostrada este en correspondencia con el idioma seleccionado.	

Tabla 9: Tarea de ingeniería: Selección del idioma a utilizar durante el proceso de instalación.

Tarea de ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** HU_2
Nombre Tarea: Detección de los paquetes necesarios para la instalación del sistema.	
Tipo de Tarea : Desarrollo	
Fecha Inicio: 8/02/2013	**Fecha Fin:** 14/02/2013
Programador Responsable: Daniel Vidaillet.	
Descripción: El instalador deberá detectar si todos los paquetes de los que depende el sistema están instalados, y en caso del que no esté instalado mostrar un mensaje de error informándoselo al usuario y dándole instrucciones de lo que debe hacer.	

Tabla 10: Tarea de ingeniería: Detección de los paquetes necesarios para la instalación del sistema.

Tarea de ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** HU_3
Nombre Tarea: Seleccionar la Ruta de instalación.	
Tipo de Tarea: Desarrollo	
Fecha Inicio: 14/02/2013	**Fecha Fin:** 22/02/2013
Programador Responsable: Daniel Vidaillet.	
Descripción: Un formulario dará la posibilidad al usuario de elegir el directorio donde se copiarán las carpetas y archivos de instalación del sistema.	

Tabla 11: Tarea de ingeniería: *Seleccionar la Ruta de instalación* 3.

Tarea de ingeniería	
Número Tarea: 4	Número Historia de Usuario: HU_4
Nombre Tarea: Configuración de archivos del *Tomcat*.	
Tipo de Tarea: Desarrollo	
Fecha Inicio: 9/03/2013	**Fecha Fin:** 14/03/2013
Programador Responsable: Daniel Vidaillet.	
Descripción: La aplicación configurará varios archivos del T*omcat*, modificando algunas líneas dentro de estos archivos, además de hacer al usuario "*dspace*" propietario de los mismos para que puedan ser utilizados por el sistema.	

Tabla 12: Tarea de ingeniería: *Configuración de archivos del* Tomcat.

Tarea de ingeniería

Número Tarea: 5	**Número Historia de Usuario:** HU_5
Nombre Tarea: Configuración archivos del UCISpace.	
Tipo de Tarea: Desarrollo	
Fecha Inicio: 16/03/2013	**Fecha Fin:** 21/03/2013
Programador Responsable: Daniel Vidaillet.	
Descripción: El sistema mediante dos paneles le pedirá al usuario que llene los campos de texto con los datos necesarios para modificar el archivo "dspace.cfg" para que la aplicación funcione correctamente. Entre los datos que deberá especificar el usuario están la dirección donde se instaló el UCISpace, la contraseña del usuario de PostgreSQL que utiliza la aplicación y el puerto a utilizar por el sistema. Luego el instalador accederá al archivo modificando los campos especificados.	

Tabla 13: Tarea de ingeniería: *Configuración archivos del UCISpace.*

Tarea de ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** HU_6
Nombre Tarea: Compilar el sistema.	
Tipo de Tarea: Desarrollo	
Fecha Inicio: 16/03/2013	**Fecha Fin:** 20/03/2013
Programador Responsable: Daniel Vidaillet.	
Descripción: Después de finalizar el instalador ejecutará el archivo "INSTAL.sh" y este procederá a dar las indicaciones para realizar los cambios necesarios en el sistema operativo, asegurando el correcto funcionamiento del UCISpace.	

Tabla 14: Tarea de ingeniería: *Compilar sistema.*

Tarea de ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** HU_7
Nombre Tarea: Creación un usuario que sea administrador del UCISpace.	
Tipo de Tarea: Desarrollo	
Fecha Inicio: 25/03/2013	**Fecha Fin:** 3/04/2013
Programador Responsable: Daniel Vidaillet.	
Descripción: El archivo "INSTALL.sh" después de compilar el sistema creará un usuario administrador del UCISpace, el usuario deberá pasarle la dirección de correo del administrador, el nombre, el segundo nombre y la contraseña.	

Tabla 15: Tarea de ingeniería: *Creación un usuario que sea administrador del UCISpace.*

2.8 Plan de iteraciones.

La planificación en todo momento y lugar es de suma importancia, ya que crea compromiso y organización. Una planificación acertada del proyecto da una idea de tiempo necesario para su desarrollo. En este trabajo se elabora un plan de iteraciones donde se decide que historias de usuario deben ser incluidas en un lanzamiento o iteración. Primeramente se lanzan aquellas historias de usuario de mayor riesgo y mayor prioridad. Este artefacto proporciona varias ventajas como son:

- Divide el proceso de desarrollo de *software* en iteraciones, proyectando el trabajo a realizar en cada una de ellas.
- Determina las historias de usuario más importantes, y las ubican en las primeras iteraciones según su prioridad.
- Define las entregas intermedias y final del *software* a desarrollar.

Quedando un plan de iteraciones que se muestra a continuación, el cual guiará el desarrollo del producto:

Iteración	**Descripción de la iteración**	**Orden de la HU a implementar**	**Duración total**
Iteración 1	En esta iteración se desarrollarán las primeras historias de usuarios, las que tiene prioridad muy alta y son de menor complejidad.	HI_1, HI_2, HI_3, HI_4	3 semanas

Iteración 2	En esta iteración se desarrollarán las historias de usuarios a partir de la HI_4, y se integran a las ya realizadas.	HI_4, HI_5, HI_6, HI_7	5 semanas
Iteración 3	En esta iteración se terminaran de desarrollarán las historias de usuarios a partir de la HI_6 hasta la última para así terminar la aplicación.	HI_6, HI_7	7 semanas

Tabla 16: Plan de liberación.

2.9 Conclusiones del capítulo.

En el capítulo se realizó la descripción de la propuesta de solución como elemento fundamental para la obtención de un producto que cumpla con las expectativas de clientes y desarrolladores. Primero se ejecutará un archivo que instale las dependencias y cree las condiciones para la correcta instalación y posterior funcionamiento del sistema. También se llevó a cabo la planificación y diseño, que permitirá la construcción del *software*. Determinando los requerimientos no funcionales que se deben cumplir, como propiedades de la computadora donde se instalarán el sistema, entre otros y se capturaron siete requerimientos funcionales, que fueron implementados para crear un producto que cumpla con las expectativas del cliente.

Capitulo 3: Validación de la solución propuesta.

3.1 Introducción al capítulo.

Una parte fundamental en el ciclo de desarrollo del *software* es la etapa de pruebas, proceso que permite verificar y revelar la calidad de un producto. La realización de pruebas, proporcionan grandes ventajas, permitiendo a los programadores principalmente medir la calidad de su trabajo y garantizar la entrega de un producto con calidad y en correspondencia con las necesidades del cliente. En el presente capítulo se mostrarán las pruebas funcionales realizadas a los requisitos correspondientes a cada una de las historias de usuario en cada iteración del desarrollo del instalador del UCISpace.

3.2 Pruebas de funcionalidad.

Las pruebas son un grupo de actividades planeadas con anticipación y que se realizan de forma sistemática. Además, son el último bastión para la evaluación de la calidad del *software* y para la detección de errores en el mismo(27).

Las pruebas de funcionalidad examinan si una aplicación cumple con los requisitos funcionales propuestos por el cliente, también tienen como objetivo revelar los problemas y errores que poseen las funcionalidades. Además, se debe verificar la validación de los datos y es importante realizarlas en cualquier fase de desarrollo del *software*(28).

Para llevar a cabo las pruebas de funcionalidades se utilizan las pruebas de caja negra.

3.3 Pruebas de caja negra.

El método de pruebas de caja negra, se llevan a cabo sobre la interfaz del *software*. Esta prueba se centra principalmente en los requisitos funcionales y examina algunos aspectos del modelo del sistema sin tener mucho en cuenta la estructura interna del *software*(29).

Las pruebas de caja negra consisten en la realización de pruebas con el objetivo de comprobar que cada función es operativa y que funcione adecuadamente. Permite demostrar que la entrada es aceptada, que se produce una salida de forma correcta de acuerdo con las especificaciones del cliente(29).

Al realizar estas pruebas se pretenden encontrar diferentes errores o problemas que puede presentar las funcionalidades desarrolladas, entre los que se pueden mencionar: funciones ausentes o incorrectas, errores de interfaz, de rendimiento, entre otros(29).

Diseño de casos de prueba.

Los casos de prueba especifican una forma de comprobar el sistema, incluye un conjunto de datos de entrada, el resultado del sistema y las condiciones bajo las que ha de probarse(18).

A continuación se muestran los casos de prueba realizados al instalador.

Condiciones de ejecución.

El usuario debe estar autenticado como administrador.

SC Elegir lenguaje

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC.1.1 Seleccionar idioma.	El instalador muestra una interfaz para seleccionar el lenguaje a utilizar durante el proceso de instalación.	El instalador muestra una lista con los lenguajes disponibles para el proceso de instalación, a partir de la siguiente interfaz todos los mensajes e información que reciba el usuario deberán estar escritos en dicho lenguaje.	El usuario ejecuta el instalador y se autentica como administrador, luego se muestra una interfaz que tiene como lenguaje predefinido el español, pero el usuario puede elegir también el inglés. Después el usuario debe presionar el botón “OK”, para comenzar con la instalación.

Tabla 17: Caso de prueba: Elegir lenguaje.

Condiciones de ejecución.

El usuario debe estar autenticado como administrador.

SC Detectar paquetes instalados

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC.1.1 Todos los paquetes están instalados.	El instalador en la interfaz de bienvenida, detecta si los paquetes de los que depende el UCISpace están instalados, al estar todos instaladores permite continuar con la instalación.	El instalador en la interfaz de bienvenida detecta que los paquetes de los que depende el UCISpace (PostgreSQL, Tomcat y Ant) todos están instalados y habilita el botón “Siguiente” permitiéndole al	El instalador muestra la interfaz de bienvenida diciéndole al usuario que se instalará el UCISpace en su ordenador y habilita el botón “Siguiente” para comenzar la instalación.

		usuario continuar con la instalación.	
EC.1.2 Falta el Ant por instalar.	El instalador detecta que falta el Ant por instalar, por lo que muestra un mensaje informándoselo al usuario, le da indicaciones de cómo instalarlo y deshabilita el botón “Siguiente” para garantizar que el usuario cumpla con las indicaciones.	El instalador, al detectar que el Ant no está instalado, muestra un mensaje de error informando que dicho paquete no está instalado, le da indicaciones para instalarlo y se deshabilita el botón “Siguiente”.	El instalador muestra un mensaje informando que falta por instalar el Ant, le da indicaciones de que debe hacer para instalarlo y deshabilita el botón “Siguiente” para que no pueda comenzar la instalación, y le informa que cierre el instalador.
EC.1.3 Falta el PostgreSQL por instalar.	El instalador detecta que falta el PostgreSQL por instalar, por lo que muestra un mensaje informándoselo al usuario, le da indicaciones de cómo instalarlo y deshabilita el botón “Siguiente” para garantizar que el usuario cumpla con las indicaciones.	El instalador, al detectar que el PostgreSQL no está instalado, muestra un mensaje de error informando que dicho paquete no está instalado, le da indicaciones para instalarlo y se deshabilita el botón “Siguiente”.	El instalador muestra un mensaje informando que falta por instalar el PostgreSQL, le da indicaciones de que debe hacer para instalarlo y deshabilita el botón “Siguiente” para que no pueda comenzar la instalación, y le informa que cierre el instalador.
EC.1.4 Falta el Tomcat6 por instalar.	El instalador detecta que falta el Tomcat por instalar, por lo que muestra un mensaje informándoselo al usuario, le da indicaciones de cómo instalarlo y deshabilita el botón “Siguiente” para garantizar que el usuario cumpla con las	El instalador, al detectar que el Tomcat6 no está instalado, muestra un mensaje de error informando que dicho paquete no está instalado, le da indicaciones para instalarlo y se deshabilita el botón	El instalador muestra un mensaje informando que falta por instalar el Tomcat6, le da indicaciones de que debe hacer para instalarlo y deshabilita el botón “Siguiente” para que no pueda comenzar la instalación, y le informa que cierre el instalador.

	indicaciones.	“Siguiente”.	

Tabla 18: Caso de prueba: Detectar paquetes instalados.

Condiciones de ejecución. El usuario debe estar autenticado como administrador. **SC Seleccionar ruta de la instalación**			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC.1.1 Seleccionar ruta de instalación correctamente.	El instalador muestra la interfaz para seleccionar la ruta de la instalación, verifica que sea válida y permite continuar con la instalación.	El instalador después de establecer la ruta de la instalación permite continuar con el proceso de instalación copiando los archivos del UCISpace para esa dirección.	El instalador muestra la interfaz para seleccionar la ruta de la instalación, el usuario selecciona el directorio donde copiar los archivos y presiona el botón “siguiente” para que el instalador copie los archivos del UCISpace en esa dirección.
EC.1.2 Seleccionar ruta de instalación no válida.	El instalador en la interfaz para seleccionar la ruta de la instalación, determina si la dirección es incorrecta, si el directorio escogido no es válido mostrará un mensaje de error informando que el directorio escogido no es correcto.	Muestra un mensaje de error informando al usuario que la dirección no es válida y no permite avanzar al siguiente panel hasta que el usuario elija una dirección válida.	El instalador muestra la interfaz para seleccionar el directorio en el cual se copiarán los archivos de la aplicación, si el usuario selecciona una dirección que no es correcta y presionar el botón “Siguiente”, se mostrará un mensaje informando que la dirección no es correcta para que se seleccione otra dirección.

Tabla 19: Caso de prueba: Seleccionar ruta de la instalación.

Condiciones de ejecución. El usuario debe estar autenticado como administrador.

SC Configurar Tomcat			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC.1.1 Configurar Tomcat.	El instalador deberá configurar los archivos “tomcat6” y “server.xml” para que puedan ser utilizados por el UCISpace.	El instalador configurará el archivo tomcat6 cambiando el usuario del Tomcat (es tomcat6 por defecto) por el del UCISpace (que es dspace) y quitando el comentario de la línea de seguridad. En el archivo “server.xml” cambia el puerto a utilizar por el puerto seleccionado por el usuario y cambia la dirección de la carpeta “webapps” por la dirección actual.	En la interfaz, para configurar el archivo ¨dspace.cfg¨, el instalador configura el archivo tomcat6 ubicado en la dirección “/etc/default/tomcat6” y el archivo server.xml ubicado en la dirección “/etc/tomcat6/server.xml” con los valores correspondientes.

Tabla 20: Caso de prueba: Configurar Tomcat.

Condiciones de ejecución.

El usuario debe estar autenticado como administrador.

SC Configurar archivo dspace.cfg 1

Escenario	**Descripción**	**Puerto**	**Correo del administrador**	**Contraseña del usuario(dspace) de Postgres**	**Carpeta de assestore**	**Respuesta del sistema**	**Flujo central**

EC.1.1 Configurar archivo “dspace.cfg ” correctamente.	El instalador muestra la primera interfaz donde se llenarán los campos de texto que son necesarios para la configuración del archivo “despace.cfg ”. El usuario procederá a llenar los campos y oprimirá el botón “Configurar” para modificar el archivo y habilitar el botón “Siguiente”.	V 8082	V admin@uci.cu	V dspace	V dirección de la instalación/ assestore	El instalador modificará los parámetros del archivo “dspace.cfg ” con los valores introducidos por el usuario.	Al mostrarse la interfaz para llenar los campos de texto, el usuario los llenará con los valores correspondientes y con el mismo formato especificado en cada campo de texto, luego presionará el botón “configurar” para modificar el archivo y habilitar el botón “Siguiente”.
EC.1.2 Configurar archivo “dspace.cfg ” incorrectamente.	El instalador muestra la primera interfaz donde se llenarán los campos que son necesarios para la configuración del archivo	I Vacío	V admin@uci.cu	V dspace	V (dirección de la instalación) /assestore	Cuando el usuario oprime el botón “Configurar ”, el instalador muestra un mensaje indicando que el campo de	El usuario llenará los campos requeridos en la interfaz, pero el campo de texto “Puerto” se quedará vacío, cuando el usuario presione el botón “Configurar”, el instalador mostrará un

	"despace.cfg". El usuario procederá a llenar los campos pero alguno se quedará vacío, luego oprimirá el botón "Configurar" para modificar el archivo y habilitar el botón "Siguiente", pero el instalador mostrará un mensaje de error informándole que campo de texto está vacío.					texto" Puerto" está vacío, y mantiene deshabilitado el botón "Siguiente".	mensaje de error y mantendrá deshabilitado el botón "Siguiente" hasta que el usuario cumpla con las indicaciones.
		V 8082	I Vacío	V dspace	V dirección de la instalación/ assestore	Cuando el usuario oprime el botón "Configurar", el instalador mostrará un mensaje indicando que el campo de texto "Correo del administrador" está vacío, y mantiene deshabilitado el botón "Siguiente".	El usuario llenará los campos requeridos en la interfaz, pero el campo de texto "Correo del administrador" se quedará vacío, cuando el usuario presione el botón "Configurar", el instalador mostrará un mensaje de error y mantendrá deshabilitado el botón "Siguiente" hasta que el usuario cumpla con las indicaciones.
		V 8082	V admin@uci.cu	I Vacío	V dirección de la instalación/ assestore	Cuando el usuario oprime el botón "Configurar", el instalador mostrará	El usuario llenará los campos requeridos en la interfaz, pero el campo de texto "Contraseña del usuario de PostgreSQL" se

						un mensaje indicando que el campo de texto "Contraseña del usuario de Postgres" está vacío, y mantiene deshabilitado el botón "Siguiente".	quedará vacío, cuando el usuario presione el botón "Configurar", el instalador mostrará un mensaje de error y mantendrá deshabilitado el botón "Siguiente" hasta que el usuario cumpla con las indicaciones.
		V 8082	V admin@uci.cu	V dspace	I No válida	Cuando el usuario oprime el botón "Siguiente", el instalador mostrará un mensaje indicando que la dirección de la carpeta "assestore" es incorrecta, y no permitirá avanzar al siguiente panel.	El usuario seleccionará la dirección de la carpeta "assestore" del sistema, cuando el usuario presione el botón "Siguiente", el instalador mostrará un mensaje de error indicando que la dirección no es correcta y no permitirá avanzar hacia el siguiente panel hasta que el usuario no cambie la dirección elegida por una válida.

Descripción de las variables.				
No.	**Nombre del campo**	**Clasificación**	**Valor nulo**	**Descripción**
1	Puerto	Lista desplegable	No	El campo permite al usuario introducir o elegir de los valores disponibles el puerto a utilizar por el UCISpace, el valor que tiene por defecto el campo es el “8082”.
2	Correo del administrador	Campo de texto	No	El campo permite introducir la dirección de correo del administrador del UCISpace.
3	Contraseña del usuario (dspace) de Postgres	Campo de texto	No	El campo permite introducir la contraseña del usuario de PostgreSQL con el que se administrará la base de datos, el valor por defecto del campo es “dspace”.
4	Carpeta de assestore	Navegación	No	Este campo es para elegir la dirección de la carpeta de almacenamiento “assestore”. La dirección que tiene por defecto es “(dirección de la instalación)/assestore”.

Tabla 21: Caso de prueba: Configurar archivo dspace.cfg número 1.

Condiciones de ejecución.

El usuario debe estar autenticado como administrador.

SC Configurar archivo dspace.cfg 2

Escenario	**Descripción**	**Servidor de correo**	**Puerto del servidor de correo**	**Dirección de correo del sistema**	**Dirección de correo para responder al sistema**	**Respuesta del sistema**	**Flujo central**
EC.1.1 Configurar archivo “dspace.cfg”	El instalador muestra la segunda interfaz donde se llenarán los campos de	V server	V 25	V ucispace@uci.cu	V dspace@uci.cu	Cuando el usuario oprima el botón “Configurar”, el instalador	Al mostrarse la interfaz para llenar los campos de texto, el usuario el usuario introducirá los valores correspondientes a cada campo, utilizando el

correctamente.	texto que son necesarios para la configuración del archivo "despace.cfg". El usuario procederá a llenar los campos de texto y presionará el botón "Configurar" para modificar el archivo y habilitar el botón "Siguiente".					modificará los parámetros del archivo "dspace.cfg" con los valores introducidos por el usuario y habilitará el botón "Siguiente".	mismo formato que tenga el texto que está en el campo de texto por defecto, luego presionará el botón "Configurar" para modificar el archivo "dspace.cfg" y habilitar el botón "Siguiente" para avanzar al siguiente panel.
EC.1.2 Configurar archivo "dspace.cfg" incorrectamente.	El instalador muestra la segunda interfaz donde se llenarán los campos de texto que son necesarios para la configuración del archivo "despace.cfg". El usuario después de llenar los campos no se percata de que uno de	I Vacío	V 25	V ucispace@uci.cu	V dspace@uci.cu	Cuando el usuario oprime el botón "Configurar", el instalador mostrará un mensaje indicando que el campo de texto "Puerto del servidor de correo" está vacío, y mantiene deshabilitado el botón	El usuario llenará los campos requeridos en la interfaz, pero el campo de texto "Puerto del servidor de correo" se quedará vacío, cuando el usuario presione el botón "Configurar", el instalador mostrará un mensaje de error y mantendrá deshabilitado el botón "Siguiente" hasta que el usuario cumpla con las indicaciones.

	ellos está vacío y oprime el botón “Configurar”, entonces el instalador le muestra un mensaje de error y mantiene deshabilitado el botón “Siguiente”.					“Siguiente”.	
		V server	I Vacío	V ucispace@uci.cu	V dspace@uci.cu	Cuando el usuario presione el botón “Configurar”, el instalador mostrará un mensaje indicando que el campo de texto “Servidor de correo” está vacío, y mantiene deshabilitado el botón “Siguiente”.	El usuario llenará los campos requeridos en la interfaz, pero el campo de texto “Servidor de correo” se quedará vacío, cuando el usuario presione el botón “Configurar”, el instalador mostrará un mensaje de error y mantendrá deshabilitado el botón “Siguiente” hasta que el usuario cumpla con las indicaciones.
		V server	V 25	I Vacío	V dspace@uci.cu	Cuando el usuario presione el botón “Configurar”, el instalador mostrará un mensaje indicando que el campo de texto “Dirección de correo del	El usuario llenará los campos requeridos en la interfaz, pero el campo de texto “Dirección de correo del sistema” se quedará vacío, cuando el usuario presione el botón “Configurar”, el instalador mostrará un mensaje de error y mantendrá deshabilitado el botón “Siguiente” hasta que el usuario cumpla con las

						sistema" está vacío, y mantiene deshabilitado el botón "Siguiente".	indicaciones.
		V server	V 25	V ucispace@uci.cu	I Vacío	Cuando el usuario oprime el botón "Configurar", el instalador mostrará un mensaje indicando que el campo de texto "Dirección de correo para responder al sistema" está vacío, y mantiene deshabilitado el botón "Siguiente".	El usuario llenará los campos requeridos en la interfaz, pero el campo de texto "Dirección de correo para responder al sistema" se quedará vacío, cuando el usuario presione el botón "Configurar", el instalador mostrará un mensaje de error y mantendrá deshabilitado el botón "Siguiente" hasta que el usuario cumpla con las indicaciones.

Descripción de las variables.

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Servidor de correo	Campo de texto	No	El campo permite introducir el servidor de correo que utilizará el sistema.
2	Puerto del servidor de correo	Campo de texto	No	El campo permite introducir el puerto que utilizará el servidor de correo del sistema. El valor por defecto del campo es 25.

3	Dirección de correo del sistema	Campo de texto	No	El campo permite introducir la dirección de correo con la que el UCISpace enviará los correos.
4	Dirección de correo para responder al sistema	Campo de texto	No	El campo permite introducir la dirección de correo a la que los usuarios podrán escribir para informar cualquier inquietud o sugerencias que tengan sobre el sistema.

Tabla 22: Caso de prueba: Configurar archivo dspace.cfg número 2.

Condiciones de ejecución.

El usuario debe estar autenticado como administrador.

SC Compilar sistema

Escenario	Descripción	Respuesta del sistema	Flujo central
EC.1.1 Compilar sistema.	El instalador después de configurar el archivo "dspace.cfg" mediante los paneles de instalación, ejecuta el archivo "INSTALL.sh", el cual se encargará de realizar todos los cambios necesarios en el sistema operativo, entre estos cambios está compilar el sistema.	El instalador compila el UCISpace mediante el comando "ant fresh_install".	El instalador después de configurar el archivo "dspace.cfg" y cerrar los paneles de instalación gráficos, ejecuta el archivo "INSTALL.sh", este le dice al usuario como ejecutar el archivo "ant.sh" el cual mediante el comando "ant fresh_install" compila el sistema.

Tabla 23: Caso de prueba: Compilar sistema.

Condiciones de ejecución.

El usuario debe estar autenticado como administrador.

SC Crear administrador

Escenario	Descripción	Respuesta del sistema	Flujo central
EC.1.1	Después de configurar el archivo "dspace.cfg"	El sistema crea el administrador del	El instalador mediante el archivo "ant.sh" después de

Crear administrador.	mediante los paneles de instalación, se ejecuta un archivo “INSTALL.sh”, este archivo le informa al usuario como ejecutar el archivo “ant.sh” el cual tiene como último paso crear un administrador del UCISpace.	UCISpace mediante el comando “create-administrator”.	compilar el sistema procede a crear un usuario administrador ejecutando el comando “create-administrator”.

Tabla 24: Caso de prueba: Crear administrador.

3.4 Resultados de las pruebas de caja negra.

Las pruebas se realizaron en tres iteraciones, dos revisiones parciales y una final, las cuales se realizaron siguiendo los diseños de casos de pruebas elaborados por cada funcionalidad del sistema. En la primera iteración llevada a cabo se hallaron seis no conformidades que fueron solucionadas completamente, en la segunda iteración se encontraron nueve no conformidades de las cuales se solucionaron siete y en la última iteración se detectaron cuatro no conformidades incluyendo las dos no resueltas en la iteración anterior y todas fueron solucionadas. Este proceso se puede apreciar mejor con la siguiente gráfica.

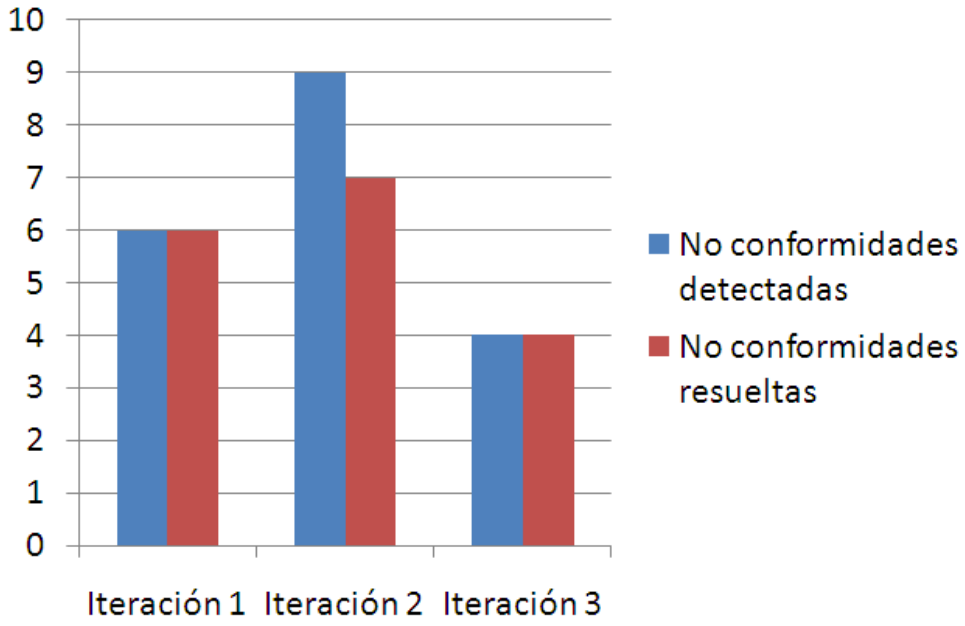

Ilustración 3: Resultados de las pruebas.

3.5 Conclusiones del capítulo.

En el capítulo se elaboraron y ejecutaron los casos de prueba funcionales para cada historia de usuario, con el objetivo de validar el correcto funcionamiento del instalador. La puesta en práctica de estas pruebas permitió arribar a la conclusión de que el *software* cuenta con todas las cualidades y condiciones necesarias para ser utilizado en la comercialización y despliegue del UCISpace.

Conclusiones generales.

La presente investigación tiene un papel fundamental en análisis, diseño e implementación del instalador de la herramienta para repositorios institucionales de la Universidad de las Ciencias Informáticas UCISpace, permitiendo concluir que:

- El análisis de las características de los sistemas que permiten la creación de instaladores, posibilitó comparar varios de los generadores de instaladores existentes en la actualidad. Eligiendo el InstallJammer como la herramienta para crear el instalador del UCISpace.
- Con el análisis y diseño realizado se logró un mejor entendimiento de las principales funcionalidades que fueron implementadas para la creación del instalador.
- Aplicadas las técnicas de validación a la propuesta de solución, se comprobó el buen funcionamiento del sistema de acuerdo a los requisitos planteados y que el producto cumple con las exigencias del cliente.

El desarrollo del instalador propuesto en el presente trabajo de diploma, automatiza gran parte del proceso de instalación de la herramienta para repositorios institucionales que se está desarrollando en la Universidad de las Ciencias Informáticas de nombre UCISpace. Su utilización aporta ventajas como la disminución del tiempo destinado a la instalación del sistema y la disminución de los errores que se puedan cometer durante este proceso, además de contar con una agradable interfaz gráfica, comentarios e indicaciones claras, por lo que se puede decir que se cumplió con el objetivo principal del trabajo, obteniéndose los resultados esperados.

Recomendaciones.

Teniendo en cuenta que se desea brindar un mejor servicio y en aras de lograr un mejor acabado al proceso de instalación del UCISpace se ofrecen las siguientes recomendaciones:

- Mejorar el instalador para que funcione en otras distribuciones de GNU/Linux como Nova o CentOS.
- Realizar un estudio mayor del lenguaje TcL, para mejorar las validaciones de las entradas de datos en los campos de texto.

Glosario de términos.

- **Biblioteca:** Cualquier tipo de colección organizada, ya sea de libros o publicaciones en serie, o bien de documentos gráficos o audiovisuales, y que se encuentran disponibles para ser consultados o tomados en préstamo.
- **Consola:** Método que permite a las personas dar instrucciones a algún programa informático por medio de una línea de texto simple. Debe notarse que los conceptos de CLI, *Shell* y Emulador de Terminal no son lo mismo, aunque suelen utilizarse como sinónimos.
- **Archivo ejecutable:** Fichero que tenga asignado el permiso de ejecución.
- **Interfaz gráfica**: La interfaz gráfica de usuario, conocida también como GUI (del inglés *graphical user interface*) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.
- **Ant**: Apache Ant, herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (*build*). Es similar a *Make* pero desarrollado en lenguaje Java y requiere la plataforma Java. Esta herramienta, hecha en el lenguaje de programación Java, tiene la ventaja de no depender de las órdenes del Shell de cada sistema operativo, sino que se basa en archivos de configuración XML y clases Java para la realización de las distintas tareas, siendo idónea como solución multiplataforma.
- **Lenguaje de programación:** Idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras.
- **Java:** lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++.
- **Shell:** En informática, el término *Shell* se emplea para referirse a programas que proveen una interfaz de usuario para acceder a los servicios del Sistema Operativo.
- **Sistema Operativo:** Un Sistema operativo es un software que actúa de interfaz entre los dispositivos de hardware y los programas de usuario, o el usuario mismo para utilizar un ordenador.
- ***Software*:** Equipamiento lógico o soporte lógico de una computadora digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de tareas específicas. Término informático que define a los programas que se utilizan en una computadora o hacen funcionar completamente a esta.
- **Instalador multiplataforma:** Instaladores de programas que funcionan nativamente en varias plataformas en GNU/Linux y Microsoft Windows por ejemplo.

- **Código Abierto:** Es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas, las cuales destacan en el llamado *software* libre.
- **Instalación:** Llevar a cabo las acciones y disposiciones previas adecuadas para que un elemento informático pueda funcionar correctamente.
- **Instalador:** Con el fin de instalar nuevo software en su ordenador, a menudo necesitan ejecutar un programa de instalación. Este programa descomprime los datos comprimidos incluido en el y escribe la nueva información en su disco duro.
- **Make:** Utilidad que permite definir reglas de dependencia entre ficheros. Aunque puede utilizarse para diferentes fines, está especialmente orientado a la compilación de código.
- **Programas de Shell:** Llamados scripts o guiones, son ficheros de texto que contienen órdenes de Shell y son interpretados por una Shell.
- **SSH:** *secure shell*, en español: (intérprete de órdenes segura) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos, y también puede redirigir el tráfico de X para poder ejecutar programas gráficos si se tiene un servidor X (en sistemas Unix y Windows) corriendo.

Referencias bibliográficas.

1. *Tipos de instalación*. [Consultado el: 3 de diciembre de de 2012]. Disponible en: http://silverfenix7.wordpress.com/2010/02/11/tiposdeinstalacionesdesoftwareprogramas.

2. *Instalar Aplicaciones.* Disponible en: http://doc.ubuntu-es.org/InstalarAplicaciones.

3. *Manual de Instalación de aplicaciones en Ubuntu*. [Consultado el: 19 de enero de 2013]. Disponible en: http://ayudalinux.wordpress.com/2007/02/04/manual-de-instalacion-de-aplicaciones-en-ubuntu-synaptic-agregarquitar-programas-aptitude-compilar-archivos-deb-rpm-run-y-bin.

4. *Bash Interiorismo, Seguridad y Distribución.* Empresas BASH, Disponible en: http://www.bash.cl/.

5. *Diccionario ABC*. [Consultado el: 20 de enero de de 2013]. Disponible en: www.definicionabc.com/general/instalacion.php.

6. VEGA, A. D. [Consultado el: 7 de diciembre de 2012]. Disponible en: http://www.albertodevega.es/index.php/generacion-de-instaladores?blog=1.

7. *IzPack*. [Consultado el: 10 de diciembre de de 2012]. Disponible en: http://www.softpedia.es/programa-IzPack-58733.html.

8. *CMake*. [Consultado el: 14 de diciembre de de 2012]. Disponible en: http://www.ecured.cu/index.php/CMake.

9. *Install4j*. [Consultado el: 13 de diciembre de de 2012]. Disponible en: http://es.softpicks.net/software/Desarrollo/Instalacion-Configuracion/install4j-x64_es-223335.html.

10. *10 razones para escoger Install4j*. [Consultado el: 15 de diciermbre de de 2012]. Disponible en: http://www.ejtechnologies.com/products/install4j/top10.html.

11. *BitRock*. [Consultado el: 16 de diciembre de de 2012]. Disponible en: http://www.softpedia.es/programa-BitRock-InstallBuilder-Professional-122351.html.

12. *Install Simple*. [Consultado el: 18 de diciembre de de 2012]. Disponible en: http://install-simple.programas-gratis.net.

13. *InstallJammer*. [Consultado el: 17 de diciembre de de 2012]. Disponible en: http://es.download.cnet.com/InstallJammer/3000-2216_4-75451502.html.

14. *Installjammer*. [Consultado el: 9 de diciembre de de 2012]. Disponible en: http://www.installjammer.com/.

15. *TcL/Tk*. [Consultado el: 6 de enero de de 2013]. Disponible en: http://gmartinezs.blogspot.com/2010/08/lenguaje-de-programacion-tcl-tk.html.

16. *Tclsh/Tk*. [Consultado el: 9 de enero de de 2013]. Disponible en: http://linux.about.com/cs/linux101/g/tclslshtk.htm.

17. *Metodologías de desarrollo de software*. [Consultado el: 10 de enero de de 2013]. Disponible en: http://www.ecured.cu/index.php/Metodolog%C3%ADas_de_desarrollo_de_software.

18. JACOBSON, I. y BOOCH, G. *El Proceso Unificado de Desarrollo de Software.* Editado por: Wesley, A. 2000, ISBN 84-7829-036-2.

19. TORRES, L. y SÁNCHEZ, E. A. *Metodologías ágiles en el desarrollo de software.* 2003, Disponible en: http://tg-tatiana-oquendo.googlecode.com/svn/trunk/actas.pdf.

20. *METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE.* 2009, Disponible en: http://usbvirtual.usbcali.edu.co/ijpm/images/stories/documentos/v1n2/009.pdf.

21. CASTRO, E. L. *PROPUESTA PARA LA INTEGRACIÓN DE PRÁCTICAS DE LAS INTEGRACIÓN DE PRÁCTICAS DE LAS METODOLOGÍAS XP Y SCRUM CON NIVEL 2 DE CMMI, Ciudad de la* Habana: 2011.

22. *CMMI® for Development,Version 1.2.* Editado por: Institute, S. E. Pittsburgh: 2006,

23. *Herramientas CASE.* 2005, Disponible en: www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf.

24. *Visual Parading*. [Consultado el: 20 de enero de 2013]. Disponible en: http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.

25. *Rational Rose*. [Consultado el: 21 de enero de de 2013]. Disponible en: http://curso_sin2.blogia.com/2005/060401-herramientas-case-rational-rose.-por-maria-de-nobrega.php.

26. J. RUMBAUGH y JACOBSON, I. *UML, El Lenguaje Unificado de Modelado.* Fernando Berzal, 2004,

27. *PRESSMAN, R.S. Ingeniería de Software. Un enfoque práctico.* 6ta ed. 2005,

28. *INFORMÁTICA, I. T. D. Testeo de funcionalidad* [Consultado el: 21 de abril de 2013]. Disponible en: http://www.iti.es/servicios/servicio/resource/7234/index.html.

29. *Técnicas de prueba. .* Editado por: Almerña, U. D. España: Disponible en: http://indalog.ual.es/mtorres/LP/Prueba.pdf.

Bibliografía consultada.

1. Bitrock-Installbuilder. [Consultado el: 16 de diciembre de 2012]. Disponible en: http://bitrock-installbuilder.softonic.com/linux.

2. CASTRO, E. L. *PROPUESTA PARA LA INTEGRACIÓN DE PRÁCTICAS DE LAS INTEGRACIÓN DE PRÁCTICAS DE LAS METODOLOGÍAS XP Y SCRUM CON NIVEL 2 DE CMMI.* La Habana: 2011.

3. *CMMI® for Development,Version 1.2.* Editado por: Institute, S. E. Pittsburgh: 2006.

4. COMPANIONI, M. G. G. y PUENTE, J. C. P. *Instalador Multiplataforma para el Gestor de Documentos Administrativos eXcriba.* 2011.

5. *INFORMÁTICA, I. T. D. Testeo de funcionalidad* [Consultado el: 21 de abril de 2013]. Disponible en: http://www.iti.es/servicios/servicio/resource/7234/index.html.

6. MAZANARES, C. *Install Builder*. [Consultado el: 19 de diciembre de 2012]. Disponible en: http://gratis.portalprogramas.com/BitRock.html.

7. Instalación de programas en Linux. [Consultado el: 30 de noviembre de 2012]. Disponible en: http://www.taringa.net/posts/linux/1926259/Instalacion-de-Programas-en-Linux.html.

8. *Installjammer*. [Consultado el: 9 de diciembre de 2012]. Disponible en: http://www.installjammer.com/.

9. COURTNEY, D. Install-Jammer. [Consultado el: 15 de diciembre de 2012]. Disponible en: http://es.download.cnet.com/InstallJammer/3000-2216_4-75451502.html.

10. *Install Simple*. [Consultado el: 18 de diciembre de 2012]. Disponible en: http://install-simple.programas-gratis.net.

11. *IzPack*. [Consultado el: 10 de diciembre de 2012]. Disponible en: http://www.softpedia.es/programa-IzPack-58733.html.

12. Izpak. [Consultado el: 18 de diciembre de 2012]. Disponible en: http://izpack.wprogramas.com/.

13. *Install4j*. [Consultado el: 13 de diciembre de 2012]. Disponible en: http://es.softpicks.net/software/Desarrollo/Instalacion-Configuracion/install4j-x64_es-223335.html.

14. JACOBSON, I. y BOOCH, G. *El Proceso Unificado de Desarrollo de Software.* Editado por: Wesley, A. 2000, ISBN 84-7829-036-2.

15. J. RUMBAUGH y JACOBSON, I. *UML, El Lenguaje Unificado de Modelado.* Fernando Berzal, 2004.

16. MORALES, L. C. *Instalador del Sistema Integrado de Gestión Bibliotecaria Bibliothex para distribuciones de GNU/Linux.* Universidad de las Ciencias Informáticas, Ciudad de la Habana, 2011.

17. *PRESSMAN, R.S. Ingeniería de Software. Un enfoque práctico.* 6ta ed. 2005.

18. NOBREGA, M. D. *Rational Rose*. [Consultado el: 21 de enero de 2013]. Disponible en: http://curso_sin2.blogia.com/2005/060401-herramientas-case-rational-rose.-por-maria-de-nobrega.php.

19. LEÓN, U. A. D. N. *TcL/Tk*. 2010 [Consultado el: 6 de enero de 2013]. Disponible en: http://gmartinezs.blogspot.com/2010/08/lenguaje-de-programacion-tcl-tk.html.

20. HAAS, J. *Tclsh/Tk*. 2010 [Consultado el: 9 de enero de 2013]. Disponible en: http://linux.about.com/cs/linux101/g/tclslshtk.htm.

21. MALFARÁ, D.; CUKERMAN, D.*, et al. Testing en eXtreme Programming.* 2006, Disponible en: http://www.fing.edu.uy/inco/cursos/gestsoft/Presentaciones/Testing%20en%20XP%20-%20G6/Testing%20en%20XP.doc.

22. GIL, M. T. *Técnicas de prueba .* Editado por: Almerña, U. D. España: Disponible en: http://indalog.ual.es/mtorres/LP/Prueba.pdf.

23. *Tipos de instalación*. [Consultado el: 3 de diciembre de 2012]. Disponible en: http://silverfenix7.wordpress.com/2010/02/11/tiposdeinstalacionesdesoftwareprogramas.

24. Principales características de Install4j. [Consultado el: 15 de diciembre de 2012]. Disponible en: http://www.ejtechnologies.com/products/install4j/top10.html.

25. *Programa de Mejora en la Universidad de las Ciencias Informáticas.* Editado por: Informáticas, U. D. L. C.

26. PEÑALVER, G.; MENESES, A. *Metodología ágil para el desarrollo de software.* Ciudad de la Habana: 2010. Universidad de las Ciencias Informáticas, Disponible en: http://usbvirtual.usbcali.edu.co/ijpm/images/stories/documentos/v1n2/009.pdf.

27. NUÑEZ, C. M. C. *Instalador gráfico avanzado de PostgreSQL con facilidad para la personalización del gestor.* Universidad de las Ciencias Informáticas, Ciudad de la Habana, 2011.

28. VEGA, A. D. *Generadores de instaladores.* [Consultado el: 7 de diciembre de 2012]. Disponible en: http://www.albertodevega.es/index.php/generacion-de-instaladores?blog=1.

29. ZÚÑIGA, V. *INTRODUCCIÓN A LA PROGRAMACIÓN DE SCRIPT EN BASH.* 2006.