

# Universidad de las Ciencias Informáticas

## Facultad 4



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

**Título:**

“Herramienta para el despliegue de *software* bajo la arquitectura Xalix  
del Centro FORTES”

**Autor:**

Yusniel Armando Reynaldo Molina

**Tutores:**

Ing. Jorge Luis Piña González

Ing. Agustín Castillo Cordero

**La Habana, Cuba**

**julio del 2017**

*Pensamiento*

*“Gracias comandante Fidel Castro, su legado ahora es nuestro camino”*

*Piedad Córdoba*



*Declaración de Autoría*

Declaro que soy el único autor de este trabajo y autorizo al Centro de Tecnologías para la Formación (FORTES) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los \_\_\_ días del mes de \_\_\_\_\_ del año\_\_\_\_\_.

\_\_\_\_\_

Yusniel Armando Reynaldo Molina

\_\_\_\_\_

Ing. Jorge Luis Piña González

\_\_\_\_\_

Ing. Agustín Castillo Cordero

### *Dedicatoria*

Le dedico este trabajo a mis padres, a mis hermanos, a mis abuelos, a mis tíos y tías, a mis primos y primas, a mi novia, al resto de la familia, a mis amigos, a mis hermanos de la vida, a mis vecinos, a mis profesores y todas aquellas personas que hicieron posible este sueño.

## *Agradecimientos*

*Agradezco al Universo, a Dios y a todo lo que existe por protegerme durante todo mi camino y darme fuerzas para superar obstáculos y dificultades a lo largo de mi vida.*

*Agradezco a mi madre Martha Molina, que con su demostración de una madre ejemplar me ha enseñado a no desfallecer ni rendirme ante nada y siempre darme sabios consejos.*

*Agradezco a mi ídolo mi padre Armando Miguel Reynaldo Sotolongo, por todos los momentos buenos y malos que hemos vivido. Además, por nunca rendirse y siempre estar a mi lado como mi fiel escudero.*

*A mis hermanos, Yasmany y Luis Ernesto, no sé qué sería de mi mundo sin mis hermanos. Ellos son todo para mí y yo donde quiera que estén los voy a querer con todo mi corazón.*

*A mis abuelas Lola y Mima, esas viejitas no sé cómo lo hacen, pero han hecho de mi vida lo más alegre posible en este mundo.*

*A mis tíos Pachuli y la Eli por darme la posibilidad y el apoyo desde lejos y por siempre confiar en mí. A todos mis tíos y tías por siempre darme consejos sanos para seguir adelante en mi vida. También a mis primos por ser como mis hermanos y siempre cuidarme mucho.*

*A mi novia Yuliet, a todos mis colegas del barrio y de la universidad.*

*A todos los profesores que me han ayudado a lo largo de mi carrera.*

### *Resumen*

El Centro de Tecnologías para la Formación, de la Universidad de las Ciencias Informáticas, se especializa en la producción de aplicaciones y servicios informáticos orientados al sector educacional. En el Centro FORTES se desarrollan varios proyectos como la plataforma educativa ZERA, el Sistema de Gestión de Ingreso a la Educación Superior y la editorial Félix Varela; todos ellos bajo la arquitectura Xalix. Los proyectos bajo esta arquitectura necesitan para el despliegue de sus principales productos, un número elevado de ficheros de configuración e instalación de programas esenciales en los servidores que serán desplegados. La presente investigación describe el desarrollo de la Herramienta para el despliegue de *software* bajo la arquitectura Xalix para dar respuesta al problema que presenta esta institución. Con el objetivo de confeccionar esta herramienta se realizó el estudio de sistemas similares, aportando características esenciales a la investigación. Para el desarrollo de la propuesta de solución se utilizó el lenguaje Python, como IDE de desarrollo Pycharm en su versión 2016.1.4. Para el funcionamiento del IDE de desarrollo se utilizó OpenJDK8 y como herramienta de modelado Visual Paradigm en su versión 8.0. La investigación fue guiada por la metodología AUP en su variación UCI, utilizando el escenario 4 y generando los artefactos de Ingeniería de *Software* durante el desarrollo de la herramienta. Finalmente, se logró desarrollar una herramienta a través de script, facilitando el despliegue de los productos en los diferentes servidores e instituciones.

**Palabras claves:** arquitecturas, despliegue, script, Xalix,

## Índice de contenido

<b>Introducción .....</b>	<b>1</b>
<b>Capítulo I: Fundamentación Teórica.....</b>	<b>7</b>
1.1 <i>Introducción.....</i>	7
1.2 <i>Análisis de homólogos.....</i>	10
1.3 <i>Metodologías de desarrollo de software.....</i>	14
1.4 <i>Lenguajes de desarrollo .....</i>	19
1.5 <i>Entorno de desarrollo integrado .....</i>	24
1.6 <i>Lenguaje para el modelado .....</i>	27
1.7 <i>Herramientas usadas por la arquitectura Xalix.....</i>	30
1.8 <i>Conclusiones del capítulo.....</i>	31
<b>Capítulo II: Propuesta de Solución .....</b>	<b>32</b>
2.1 <i>Introducción.....</i>	32
2.2 <i>Propuesta de solución.....</i>	32
2.3 <i>Modelo de dominio .....</i>	32
2.4 <i>Descripción del sistema propuesto.....</i>	34
2.5 <i>Requisitos funcionales y no funcionales.....</i>	36
2.5.1 <i>Requisitos funcionales.....</i>	36
2.5.2 <i>Requisitos no funcionales.....</i>	38
2.6 <i>Descripción de requisitos .....</i>	38
2.7 <i>Patrones arquitectónicos.....</i>	42
2.8 <i>Patrones de diseño.....</i>	44
2.8.1 <i>Patrones GRASP.....</i>	46

2.9	<i>Modelo de diseño</i> .....	49
2.10	<i>Conclusiones del capítulo</i> .....	50
<b>Capítulo III: Validación de la propuesta de solución</b> .....		<b>51</b>
3.1	<i>Introducción</i> .....	51
3.2	<i>Diagrama de componente</i> .....	51
3.3	<i>Estándar de codificación</i> .....	52
3.4	<i>Pruebas de software</i> .....	53
3.4.1	<i>Métodos de pruebas</i> .....	54
3.4.2	<i>Diseño de casos de prueba</i> .....	55
3.4.3	<i>Resultados obtenidos del método de caja negra</i> .....	59
3.4.4	<i>Resultados obtenidos de las pruebas de aceptación de tipo alfa</i> .....	60
3.5	<i>Conclusiones del capítulo</i> .....	61
<i>Conclusiones generales</i> .....		62
<i>Recomendaciones</i> .....		63
<i>Referencias bibliográficas</i> .....		64
<i>Anexos</i> .....		68



## Índice de figuras

<i>Figura 1: Escenario No1 Metodología AUP-UCI.....</i>	<i>17</i>
<i>Figura 2: Escenario No2 Metodología AUP-UCI.....</i>	<i>18</i>
<i>Figura 3: Escenario No3 Metodología AUP-UCI.....</i>	<i>18</i>
<i>Figura 4: Escenario No4 Metodología AUP-UCI.....</i>	<i>18</i>
<i>Figura 5: Lenguajes de programación .....</i>	<i>23</i>
<i>Figura 6: Modelo de dominio .....</i>	<i>34</i>
<i>Figura 7: Flujo de instalación de los pre-requisitos.....</i>	<i>35</i>
<i>Figura 8: Flujo de instalación de la aplicación .....</i>	<i>36</i>
<i>Figura 9: Arquitectura basada en eventos .....</i>	<i>44</i>
<i>Figura 10: Relación de familias de patrones de diseño .....</i>	<i>46</i>
<i>Figura 11: Ejemplo Patrón Experto.....</i>	<i>47</i>
<i>Figura 12: Ejemplo Patrón Creador.....</i>	<i>48</i>
<i>Figura 13: Ejemplo Patrón Controlador.....</i>	<i>49</i>
<i>Figura 14: Diagrama de clases.....</i>	<i>50</i>
<i>Figura 15: Diagrama de componentes.....</i>	<i>51</i>
<i>Figura 16: Gráfica con los resultados de la complejidad de las pruebas caja negra. ....</i>	<i>60</i>
<i>Figura 17: Gráfica con los resultados de la complejidad de las pruebas de aceptación. ...</i>	<i>60</i>

## Índice de tabla

<i>Tabla 1: Comparación sobre las metodologías de software.....</i>	<i>15</i>
<i>Tabla 2: Descripción de la fase de AUP-UCI .....</i>	<i>17</i>
<i>Tabla 3: HU_ Instalar y configurar los programas del servidor de aplicaciones .....</i>	<i>40</i>
<i>Tabla 4: HU_ Instalar programas adicionales .....</i>	<i>42</i>
<i>Tabla 5: Instalar programas adicionales.....</i>	<i>57</i>
<i>Tabla 6: Instalación del servidor de aplicación.....</i>	<i>58</i>
<i>Tabla 7: Prueba de aceptación. ....</i>	<i>59</i>

## Introducción

La evolución de las Tecnologías de la Información y las Comunicaciones (TIC) en las últimas décadas, ha colocado a los sistemas informáticos en un rol preponderante dentro de las instituciones, teniendo como consecuencia el crecimiento exponencial en la transmisión de información y conocimientos en diferentes esferas como la salud, los servicios, la industria y la educación. Como consecuencia del desarrollo de las TIC ha evolucionado la Internet (Fernández-Montes, 2003). Siendo esta una red integrada por miles de redes y computadoras interconectadas en todo el mundo mediante cables y señales de telecomunicaciones, que utilizan una tecnología común para la transferencia de datos.

Internet permite superar la distancia física como factor limitante convirtiéndose en un nuevo ámbito de desarrollo social. Además, favorece el desarrollo tecnológico y la habitual y creciente dependencia de las sociedades por los productos informáticos, cada día más actualizados y con mayor calidad. Una vez surgida la *World Wide Web* (WWW, por sus siglas en inglés) en el año 1994, se convierten las páginas *web* en la aplicación estándar de la gran red de redes. (Sánchez, 2015)

El uso de las páginas *web* dentro del sector de la informática ha evolucionado enormemente, trayendo consigo nuevas soluciones y retos para las TIC. Las aplicaciones *web* han tenido un gran avance en el mundo de las comunicaciones, de las cuales han surgido famosos buscadores, directorios y portales *web*. Los grandes volúmenes de información (textos, fotografías, gráficos, música y videos) existentes en Internet, se gestionan a través de portales *web*, los cuales constituyen aplicaciones que obtienen de forma uniforme y centralizada contenidos provenientes de diversas fuentes. Estos portales implementan estrategias de navegación sobre los contenidos, además de integrar aplicaciones e incluir mecanismos de colaboración para el conjunto de usuarios a los que sirven de marco de trabajo.

Posicionarse en la *web* se ha convertido en un objetivo estratégico para todas las empresas e instituciones, constituyendo una vía efectiva para hacer llegar contenidos a las diferentes audiencias. Con el objetivo de mantenerse en las primeras líneas de opciones, estas plataformas requieren estrategias que le permitan destacar en las cifras billonarias de información que circulan por la red de redes. A partir del crecimiento sustancial del desarrollo de *software* en el mundo, las principales empresas desarrolladoras necesitan tener un vasto conocimiento en las diferentes esferas del despliegue de sus aplicaciones.

El despliegue de *software* es la infraestructura de distribución escalable para la instalación de *software*, el descubrimiento y la gestión de conformidad. A diferencia de la infraestructura del motor de despliegue, que normalmente se utiliza para automatizar operaciones en conjuntos más pequeños de destinos gestionados, la infraestructura de distribución escalable garantiza una distribución de *software* rápida y fiable en un gran número de sistemas de destino (miles) de diversas topologías.

Para tener un seguro despliegue de *software* existen diversas herramientas que a partir de su funcionamiento y rendimiento logran realizar la tarea de forma rápida y segura. Dentro de las herramientas más usadas se encuentran el Administrador de Configuración de Centro del Sistema (SCCM), se trata de una solución de *software* de administración para grandes grupos de ordenadores en red basados en Windows, también en dispositivos iOS y Androide. Permite administrar de forma centralizada la configuración de todos los sistemas físicos y virtuales de una organización o grupo de organizaciones. Despliega y actualiza servidores, en equipos cliente y dispositivos, ya sean físicos, virtuales, distribuidos y entornos móviles, desde una única consola. Automatiza la gestión de los sistemas, reduciendo costos, ya que el mantenimiento constituye hasta un 70% del costo total de un sistema, y ayuda a los administradores de sistemas a trabajar de forma centralizada. (Williams,2016)

Otra de las herramientas utilizadas para el despliegue de *software* se denomina Automic; se centra en la automatización de pruebas de *software* para DevOps<sup>1</sup>. La herramienta sirve como orquestador para la implementación de flujos de trabajo y canales de lanzamiento. Se puede organizar a través de Jenkins, Chef y Puppet, son herramientas utilizadas para desplegar aplicaciones y sistemas en cualquier ámbito físico, virtual o en la nube con un enfoque en la preparación de instalación y configuración previa del ambiente. Su paquete Docker<sup>2</sup> construye flujos de trabajo visuales y automatiza los contenedores de Docker. (Jiménez,2014)

La Corporación Total de Servicios Administrativos (TASC), que administra los beneficios de los empleados, utiliza Automic para automatizar sus implementaciones de *software*. Esta herramienta

---

<sup>1</sup> DevOps es un acrónimo inglés de development (desarrollo) y operations (operaciones), que se refiere a una cultura o movimiento que se centra en la comunicación, colaboración e integración entre desarrolladores de *software* y los profesionales en las tecnologías de la información (IT).

<sup>2</sup> Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de *software*, proporcionando una capa adicional de abstracción y automatización de Virtualización a nivel de sistema operativo en Linux.

posibilita un despliegue del *software* a partir de una hora determinada, realizando todos los flujos necesarios con el objetivo de terminar la tarea de despliegue, y eso es un gran paso. (Jiménez,2014)

Una de las herramientas más utilizadas en cuestiones de despliegue de *software* es Ansible, es una plataforma de automatización de fuente abierta impulsada por una línea de comando, que sirve para el despliegue de aplicaciones y la reducción de la complejidad. Ansible Tower sirve como un control de misiones; que proporciona control, seguridad y delegación. (Peralta,2014)

A partir del auge que ha existido en el desarrollo de *software* libre y la serie de herramientas utilizadas en los mismos, es necesario saber cuáles de disímiles herramientas son las adecuadas para el funcionamiento correcto de los proyectos. Una manera rápida y efectiva es tener conocimiento previo de todos los programas y herramientas a utilizar por los proyectos, para lograr un objetivo final y tener completa disponibilidad tecnológica adecuada. Una forma de instalación de estas herramientas es a través de script.

Un script es un documento que contiene instrucciones escritas en códigos de programación. Es un lenguaje de programación que ejecuta diversas funciones en el interior de un programa de computadora. Se encarga de combinar componentes, interactuar con el sistema operativo o el usuario, controlar un determinado programa o aplicación de sistemas operacionales. Algunos lenguajes de programación que usan script son: ActionScript, JavaScript, Lua, PHP, Python, Shell Script, Ruby, VBScript. (Facultad de Informática, Universidad Politécnica de Madrid, 2015)

En el marco de la Universidad de las Ciencias Informáticas (UCI), existen varias tareas que se llevan a cabo para el despliegue de las plataformas que serán utilizadas en estos centros educativos. Debido al crecimiento sustancial de aplicaciones y plataformas web en esta institución, se hace necesario contar con una herramienta que brinde facilidad, rapidez y eficacia para el despliegue de estos productos. Según la metodología utilizada en esta institución, para comprobar la calidad de los productos se realizan pruebas en tres niveles. La primera revisión del producto es a nivel de proyecto, denominadas pruebas internas. La segunda está relacionada con las pruebas a nivel de centro. Con el objetivo de certificar la calidad del producto, este pasa por calidad UCI. Dentro de la universidad uno de los centros que evidencia el proceso de desarrollo, pruebas y despliegue de *software* es el Centro de Tecnologías para la Formación (FORTES).

El Centro FORTES desarrolla una serie de proyectos enmarcados en la línea Xauce dedicada a la educación, dentro de ellos se encuentran la plataforma educativa ZERA, y el Sistema de Gestión

de Ingreso a la Educación Superior (SIGIES) y la editorial Félix Varela. Para el desarrollo de estos proyectos fue utilizada la Arquitectura Xalix que es una arquitectura definida por el Centro de producción para sus proyectos utilizando Symfony 2 de conjunto con varios Bundles de terceros y administrativos. Estos se encuentran ya definidos de forma genérica para la implementación de cualquier nuevo proyecto que requiera funcionalidades que ellos brinden. (Manso, 2016)

El Centro FORTES para el despliegue de sus principales proyectos, necesita de un número elevado de ficheros de configuración e instalación de programas, esenciales para la puesta en marcha de sus productos en los servidores que serán montados. Cada vez que se vaya a realizar el despliegue de una aplicación, se debe generar un entorno similar al que realmente será utilizado; esto conlleva que constantemente haya que utilizar personas que no estén capacitadas para realizar esta tarea, debido a la escasez hoy en el Centro de personal evaluado en este tipo de labor, dificultando en gran medida el proceso de despliegue de una aplicación determinada.

A partir de que el Centro FORTES presenta el mismo personal para realizar todos los procesos de despliegue de la institución, este no se encuentra capacitado del todo debido a la variedad de aplicaciones existentes; dando lugar a que se cometan errores simples, pero que se desconozcan por los integrantes del equipo de despliegue de *software*. En la actualidad este proceso se realiza de forma manual, por lo que se hace complicado debido al tiempo en instalación y configuración que se invierte en lograr que el producto tenga la calidad requerida para ser utilizado.

Con este problema vigente en esta institución se desea desarrollar una herramienta que brinde la posibilidad de desplegar estos productos en un entorno determinado, brindando así rapidez, facilidad, seguridad, disponibilidad y eficacia en su uso.

Teniendo en cuenta lo antes planteado, se define como **problema a resolver** en la investigación: ¿Cómo contribuir al despliegue de los componentes de *software* bajo la arquitectura Xalix del Centro de Tecnologías para la Formación?

Se define como **objetivo general**: Desarrollar una herramienta de tipo script para contribuir al despliegue de los componentes de *software* bajo la arquitectura Xalix del Centro de Tecnologías para la Formación.

La presente investigación tiene como **objeto de estudio**: Aplicaciones para el despliegue de componentes de *software*.

El **campo de acción** de la investigación está enfocado en: Proceso de despliegue de componentes de *software* bajo la arquitectura Xalix para el Centro FORTES.

Para darle cumplimiento al objetivo general, se definen los siguientes **objetivos específicos**:

- Analizar los principales conceptos asociados al despliegue de *software*, para obtener la base teórica necesaria para el desarrollo de la solución.
- Caracterizar los sistemas similares que realicen el despliegue de *software*, para un mejor entendimiento de la herramienta a desarrollar.
- Seleccionar la metodología, el lenguaje de programación y las tecnologías que serán utilizadas con el objetivo de realizar la Herramienta para el despliegue de *software* bajo la arquitectura Xalix.
- Realizar el análisis y diseño de la Herramienta para el despliegue de *software* bajo la arquitectura Xalix.
- Desarrollar la Herramienta para el despliegue de *software* bajo la arquitectura Xalix.
- Realizar las pruebas que comprueben el correcto funcionamiento del sistema implementado.

Para la realización de la siguiente investigación se utilizaron los siguientes **métodos**:

#### **Métodos teóricos:**

**Histórico-Lógico:** permitió la comprensión de los scripts y así conocer la evolución de las tendencias alrededor de las tecnologías actuales proporcionando la selección de idónea para desarrollar la propuesta de solución.

**Analítico-sintético:** se realizó un análisis de la teoría y la documentación, posibilitando la extracción de los elementos fundamentales relacionados con el desarrollo de los programas que utilizan los proyectos del Centro FORTES.

#### **Métodos empíricos:**

**Observación:** fue de gran importancia para la obtención de información relacionada con la problemática tratada en la investigación y para identificar las funcionalidades presentes en el desarrollo de la Herramienta para el despliegue de *software* bajo la arquitectura Xalix.

## **Estructura capitular de la tesis**

El documento consta de tres capítulos los cuales se encuentran estructurados de la siguiente forma:

### **Capítulo I: Fundamentación teórica**

En este capítulo se fundamentan los elementos teóricos necesarios que respaldan la investigación. Se realiza un análisis del estado del arte del *software* con soluciones similares, su aporte y las habilidades que permiten desarrollar el mismo. Se fundamenta la utilización de las diferentes herramientas, tecnologías y metodología para dar cumplimiento al objetivo general de la investigación.

### **Capítulo II: Propuesta de solución**

En este capítulo se describen las características fundamentales de los conceptos asociados a la investigación, sus objetivos, en qué consisten, para qué están diseñados, el flujo de pantallas, así como los mecanismos a seguir por los usuarios. Además, se plantea todo lo referente a la propuesta de solución, así como los patrones de diseño utilizados durante el desarrollo del *software*.

### **Capítulo III: Implementación y pruebas.**

En este capítulo se presentan los artefactos relacionados con la implementación y el proceso de pruebas realizados, con el objetivo de verificar el correcto funcionamiento de las funcionalidades de la aplicación en los diferentes entornos y sus características.



## Capítulo I: Fundamentación Teórica

### 1.1 Introducción

En el presente capítulo se expone el marco teórico en el que se desarrolla la investigación, creando una base sólida para el desarrollo del proyecto. Para ello, se analizan los procesos de instalación de *software* de los proyectos, así como las características que presentan cada marco de trabajo y el aporte que brinda su utilización. Además, se especifican las principales características de la metodología, lenguajes y herramientas a utilizar en el desarrollo de la solución.

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 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. Actualmente estos casos son muy raros, o bien, más comúnmente, con una de complejidad intermedia en la que los distintos archivos componentes del *software* (ejecutables, bibliotecas y datos propios) 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). (Silver,2010)

En productos de ventas masivas 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. (Silver,2010)

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 sentido utilizar este tipo de instalación salvo algunas excepciones. (Silver, 2010)
- Típica: instala la mayoría de los archivos que se necesitan para poder ejecutarla 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. (Silver, 2010)
- 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. (Silver, 2010)
- Personalizada: permite al usuario elegir los archivos y paquetes que se instalarán, a diferencia de otras es la mejor opción, ya que permite al usuario elegir las aplicaciones que necesita realmente. (Silver, 2010)

Los procesos de instalación son diferentes en dependencia del sistema operativo que se esté utilizando. A continuación, se muestran ejemplos de algunas de las formas en que se pueden instalar los programas en los sistemas operativos basados en GNU/Linux. Una forma de instalación de estas herramientas y en estos sistemas operativos es a través de script.

### Los scripts

El script o guiones<sup>3</sup> son documentos que contiene instrucciones, escritas en códigos de programación. Es un lenguaje de programación que ejecuta diversas funciones en el interior de un programa de computador. Los scripts se encargan de cumplir las siguientes funciones como combinar componentes, interactuar con el sistema operativo o con el usuario, controlar un determinado programa o aplicación y configurar o instalar sistemas operacionales, especialmente en los juegos se usa para controlar las acciones de los personajes. (Facultad de Informática, Universidad Politécnica de Madrid, 2015)

Son programas, usualmente pequeños o simples, para realizar tareas muy específicas. Son un conjunto de instrucciones generalmente almacenadas en un archivo de texto que deben ser interpretados línea a línea en tiempo real para su ejecución; esto los distingue de los programas (compilados), estos deben ser convertidos a un archivo binario ejecutable para su funcionamiento. Puede haber scripts para ser ejecutados en sistema de línea de comandos como los scripts para UNIX (usualmente archivos de extensión .sh) o los scripts para DOS y la línea de comandos de Windows (estos últimos suelen ser archivos de extensión .bat). Son interpretados por cmd.exe o por el antiguo COMMAND.COM. Los scripts que más utilizan los usuarios hogareños son los que se utilizan al navegar por internet, más específicamente, se ejecutan en los navegadores como

---

<sup>3</sup> Es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

Internet Explorer, Google Chrome o Firefox. Algunos scripts son llamados "Scripts del lado del cliente", dado que son ejecutados en la computadora de quien visita el sitio web. En contraposición están los "Scripts del lado del servidor", que se ejecutan en la computadora servidora del sitio web (el usuario solo ve el resultado del script). Los scripts del lado del cliente (del usuario de hogar) suelen ser escritos en el lenguaje JavaScript o también en VBScript (sólo para Internet Explorer y Chrome). Estos scripts les agregan muchas funcionalidades a los sitios web como validación de datos de un formulario, juegos, botones interactivos, cargar información de forma asíncrona, etc. Es muy común que los usuarios no estén conformes porque los navegadores dan "error en script". Usualmente el error desaparece si el usuario emplea otro navegador web o elimina todo el caché e historial del navegador afectado. (Facultad de Informática, Universidad Politécnica de Madrid, 2015)

**Características específicas de los scripts** (Facultad de Informática, Universidad Politécnica de Madrid, 2015)

Los scripts suelen escribirse más fácilmente, pero con un costo sobre su ejecución.

- Suelen implementarse con intérpretes en lugar de compiladores.
- Tienen fuerte comunicación con componentes escritos en otros lenguajes.
- Los scripts suelen ser almacenados como texto sin formato.
- Los códigos suelen ser más pequeños que el equivalente en un lenguaje de programación compilado.

### **Tipos de scripts y sus características**

#### **En UNIX**

Los archivos guiones suelen ser identificados por el sistema a través de uno de los siguientes encabezamientos en el contenido de archivo (`#!/bin/bash`; `#!/bin/ksh`; `#!/bin/csh`), conocido como shebang. Aunque en entornos UNIX la mayoría de los guiones son identificados por el encabezamiento, también pueden ser identificados a través de la extensión ".sh", siendo esta quizás menos importante que el encabezamiento, ya que casi todos los sistemas no necesitan dicha extensión para ejecutar el guion, por lo tanto, esta suele ser añadida por tradición. Las características de estos elementos siguen siendo útiles para que el usuario pueda identificar los archivos a través de una interfaz de líneas de comandos sin necesidad de abrirlo. Difieren de los programas de aplicación, debido a que los últimos son más complejos; además, los guiones son más bien un programa que le da instrucciones a otros más avanzados. (Cano,2014)

## En Windows y DOS

En el sistema operativo DOS, a los guiones creados para ser interpretados por cmd.exe o el obsoleto COMMAND.COM se les conoce como archivos batch (procesamiento por lotes) y acaban en “.bat”. En el sistema operativo Windows, existen varios lenguajes interpretados como Visual Basic Script, JavaScript, WScript, Batch. (Cano,2014)

## En diseño Web

Los guiones en Internet se pueden clasificar en guiones del lado del cliente y del lado del servidor.

### Guiones del lado del cliente

Los guiones del lado del cliente se deben incluir con la etiqueta <script>, incluyendo el atributo type con el tipo MIME. Generalmente se usa JavaScript, pero se puede usar VBScript (solo Internet Explorer o Google Chrome). Tiene como objetivo, por lo general, AJAX o manipulación del DOM. (Cano,2014)

### Guiones del lado del servidor

No tienen los problemas de accesibilidad que pueden presentar los guiones del lado del cliente. También permiten modificar las cabeceras HTTP, u obtenerlas. Además, permiten acceso a bases de datos y otros archivos internos. (Cano,2014)

### Ejemplo de otros lenguajes de Script: (Cano,2014)

- **VBScript:** Implementación de Visual Basic para crear aplicaciones para Internet Explorer.
- **JScript:** Implementación de Microsoft del lenguaje de scripting basado en Java.
- **ActionScript:** Lenguaje de script de Macromedia para la aplicación Flash.
- **ECMAScript:** Lenguaje de scripting que soporta el estándar ECMA-262 (European Computer Manufacturers Association).

## 1.2 Análisis de homólogos

El proceso de generación de scripts no es algo nuevo en el mundo. compañías productoras de *software* han dedicado esfuerzos en brindar herramientas que suplan esta necesidad. Algunas de estas herramientas son:

**Script de Políticas de Seguridad (Secscript):** este script fue realizado con el objetivo de agrupar todas las políticas de seguridad que debe tener un entorno de trabajo en un proyecto de producción. Dentro de las ventajas que brinda están la instalación completa; Esta incluye la configuración del firewall, instalar Likewise(Pbis), configurar usuarios administradores, instalar Veracrypt, instalar el antivirus, protección del grub con contraseña, instalar actualizaciones pendientes y chequear las políticas de seguridad pertinentes. Todas estas opciones hacen que este script sea importante en cuanto a todo lo relacionado con las políticas de seguridad. Otra de las ventajas que presenta este script es que, se pueden realizar de forma individual, dependiendo del objetivo a alcanzar por el usuario.

**BitRock InstallBuilder:** Tiene la capacidad de generar instaladores para múltiples plataformas como Windows, Mac, Linux, Solaris, entre otros. Esta herramienta está destinada básicamente a los desarrolladores que precisan un instalador para sus proyectos terminados. El gestor de instalación generado tendrá el “look & feel” del sistema operativo para el que habrá sido creado, es decir que se integrará a la perfección en su entorno gráfico y funcional sin necesidad de dependencias. Adicionalmente, aporta soporte para 16 idiomas. Dispone de todas las funciones y opciones necesarias para crear paquetes de instalación, con lo que se pueden crear y distribuir cómodamente los programas para todos los sistemas operativos. Una desventaja del *software* es el pago que se realiza para su utilización plena, pero el mismo posee una versión gratuita de prueba. (Sales, 2016)

En **SQL Server Management Studio:** Se pueden crear scripts de Transact-SQL para varios objetos utilizando el “Asistente Generar y Publicar Scripts” y se puede generar un script para objetos individuales o varios objetos utilizando el menú “Incluir como” del Explorador de objetos. De igual manera, se puede utilizar el mismo menú para generar un script de un solo objeto, de varios objetos o de varias instrucciones para un único objeto. Puede elegir uno de varios tipos de scripts; por ejemplo, crear, modificar o quitar el objeto. Puede guardar un script en una ventana del Editor de consultas, en un archivo o en el Portapapeles. El script también puede ser creado en formato Unicode. El asistente genera un script de todos los objetos de una base de datos o un subconjunto de los objetos que seleccione. Además, dispone de muchas opciones para los scripts, como la posibilidad de incluir permisos, la intercalación, las restricciones, etc. (Studio, 2016)

Es un directorio de scripts listos para usar, ordenados por las tecnologías con las que se han desarrollado y por una clasificación según su contenido. Además, contienen código y datos que proporcionan servicios a programas independientes como Python, Java Script, esto permite que los datos se compartan y puedan modificarse de forma modular. Este tipo de servicios proporcionan a

los programadores un conjunto de instrucciones que simplifica el trabajo y evita duplicar el esfuerzo cada vez que sea necesario realizar una tarea específica. (Valdéz, 2014)

**Docker** es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de *software*, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel del sistema operativo en Linux. Crea contenedores ligeros y portables para aplicaciones que puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues. Permite fijar en un contenedor (“una caja”, algo auto contenido, cerrado) todas aquellas cosas que una aplicación necesita para ser ejecutada (java, Maven, tomcat...) y la propia aplicación. Así puede utilizar el contenedor en cualquier máquina que tenga instalado Docker y ejecutar la aplicación sin tener en cuenta las versiones de *software* instaladas en la máquina. En el caso de los desarrolladores, el uso de Docker hace que puedan centrarse en desarrollar su código sin preocuparse de si funcionará en la máquina en la que se ejecutará. Los conceptos de máquina virtual y Docker son realmente similares, pero un contenedor no es lo mismo que una máquina virtual. Un contenedor es más ligero, mientras que a una máquina virtual necesita instalarse en un sistema operativo para funcionar, un contenedor funciona utilizando el sistema operativo que tiene el ordenador.

El contenedor de Docker toma los recursos más básicos, que no cambian de un ordenador a otro del sistema operativo de la máquina en la que se ejecuta. Los aspectos más específicos del sistema que pueden dar más problemas a la hora de llevar el *software* de un lado a otro, se colocan en el interior del contenedor. El concepto de portabilidad de un contenedor de Docker es algo similar a la máquina virtual de Java. Un contenedor toma los aspectos básicos de funcionamiento del sistema operativo de la máquina en la que se ejecuta y lo vuelve más ligero que una máquina virtual. (Oterino, 2015)

### **Resultados arrojados del estudio de homólogos**

Después de realizar un análisis, se comprobó que estos programas son soluciones similares a la propuesta deseada; no resuelven el problema planteado debido a las desventajas encontradas, pero aportaron características importantes a la presente investigación como:

### **Ventajas obtenidas del estudio de homólogos:**

Script de Políticas de Seguridad (Secscript), Docker y el programa BitRock InstallBuilder: presentan ciertas cualidades que pueden ser tomadas en cuenta como:

- Comprobar la conectividad con los repositorios, posibilitando realizar todas las tareas de comprobación, instalación y configuración.
- Permite comprobar que las instalaciones y configuraciones de programas se realicen con contraseña de superusuario o administrador.
- Posee la habilidad de configurar los repositorios en caso de que estos no estén configurados correctamente.
- BitRock InstallBuilder, esta herramienta está destinada básicamente a los desarrolladores que precisan un instalador para sus proyectos terminados.
- Dispone de todas las funciones y opciones necesarias para crear paquetes de instalación con lo que se puede crear y distribuir los programas para todos los sistemas operativos.

Además de las características antes mencionadas, es importante destacar que cada uno de los scripts brindan la posibilidad de realizar una o varias operaciones de forma sencilla. Otro aspecto fundamental es su capacidad para adaptarse al entorno o sistema en el cual serán ejecutados brindando rapidez, eficiencia y confiabilidad. Se puede confirmar estas características, ya que fueron comprobados en diferentes sistemas operativos arrojando resultados similares.

### **Desventajas obtenidas del estudio de homólogos:**

Docker, Script de Políticas de Seguridad (Secscript) y el programa BitRock InstallBuilder: presentan varias desventajas como:

Sobre Docker:

- Sólo soporta arquitecturas de 64bits.
- Se requiere Kernel<sup>4</sup> 3.8 como mínimo.
- Solo puede ejecutarse en Linux de forma nativa.
- Debido al constante desarrollo, se podría dar el caso que unas versiones den error.

Sobre BitRock InstallBuilder:

- Requiere de un pago para la utilización plena del *software*.

---

<sup>4</sup> Núcleo. Parte esencial del Sistema Operativo que provee los servicios más básicos del sistema. Se encarga de gestionar recursos como el acceso seguro al hardware de la computadora

Script de Políticas de Seguridad (Secscript):

- No presenta la capacidad de configurar los repositorios de nuevas versiones de Linux.

Debido a las desventajas que poseen estas herramientas no se utilizaron para confeccionar la propuesta de solución, pero si se tendrán en cuenta las características y funcionalidades principales en la presente investigación.

### 1.3 Metodologías de desarrollo de *software*

Las metodologías de desarrollo de *software* imponen un proceso de forma disciplinada con el objetivo de hacerlo más predecible y eficiente, definiendo una representación que permite facilitar la manipulación de modelos y el intercambio de información entre todas las partes involucradas en la construcción de un sistema (Cendejas Valdéz, 2014).

Las metodologías se clasifican en dos grupos:

**Metodologías tradicionales:** Están orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán.

**Metodologías ágiles:** Están orientadas a la interacción con el cliente y el desarrollo incremental del *software*, mostrando versiones parcialmente funcionales del *software* al cliente en determinados intervalos de tiempo, para que pueda evaluar y sugerir cambios en el producto.

La tabla siguiente muestra las diferencias entre estos dos grupos de metodologías (Letelier, 2006).



Metodología Ágil	Metodología Tradicional
Pocos Artefactos. El modelado es prescindible, modelos desechables.	Más Artefactos. El modelado es esencial, mantenimiento de modelos.
Pocos Roles, más genéricos y flexibles.	Más Roles, más específicos.
Orientada a proyectos pequeños. Corta duración (o entregas frecuentes) y equipos pequeños (menos de 10 integrantes).	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente usadas en proyectos grandes y con equipos posiblemente dispersos.
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
La arquitectura se va definiendo y mejorando a lo largo del proyecto.	Se promueve que la arquitectura se defina tempranamente en el proyecto.
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo.	Énfasis en la definición del proceso: roles, actividades y artefactos.
Se esperan cambios durante el proyecto.	Se espera que no ocurran cambios de gran impacto durante el proyecto.

Tabla 1: Comparación sobre las metodologías de *software*

A continuación, se hace un breve análisis de dos metodologías robustas con el objetivo de seleccionar la más adecuada, teniendo en cuenta las necesidades se analizan AUP-UCI y RUP.

**Agile Unified Process (AUP)**, Proceso Unificado Ágil según sus siglas en inglés) es una versión simplificada del Proceso Unificado de Rational (RUP) desarrollada por Scott Ambler. AUP aplica técnicas ágiles incluyendo desarrollo orientado a pruebas, modelado ágil, gestión de cambios ágil y refactorización de base de datos para mejorar la productividad (Sánchez, 2015).

Esta metodología pasa por diferentes fases en su ciclo de vida como son:

- **Inicio:** Se encarga de definir el alcance del proyecto, los costes y plazos, los riesgos y la factibilidad del mismo.

- **Elaboración:** Encargada de identificar y validar la arquitectura.
- **Construcción:** Tiene como objetivo la creación de la documentación de apoyo, así como elaboración del *software* de forma incremental.
- **Transición:** El propósito es realizar las pruebas al proyecto para su posterior liberación funcional.

### Variación de AUP para la UCI

Al no existir una metodología de *software* universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de *software* tiene entre sus objetivos aumentar la calidad del *software* que se produce, de ahí la importancia de aplicar buenas prácticas, para ello se tomará como referente el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (UCI, 2015).

### Descripción de las fases de AUP en su variación UCI

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. Para una mejor comprensión.

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o

		no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el <i>software</i> , incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Tabla 2: Descripción de la fase de AUP-UCI

**Escenarios para la disciplina Requisitos**

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos como Casos de Uso del Negocio(CUN), Descripción de Requisitos de Negocio(DPN), o Modelo Conceptual(MC) y existen tres formas de encapsular los requisitos como Casos de Uso del Sistema(CUS), Historias de Usuarios(HU) y Descripción de Requisitos por Proceso(DRP), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

**Escenario No1:** Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.



Figura 1: Escenario No1 Metodología AUP-UCI

**Escenario No2:** Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.



Figura 2: Escenario No2 Metodología AUP-UCI

**Escenario No3:** Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.



Figura 3: Escenario No3 Metodología AUP-UCI

**Escenario No4:** Proyectos que no modelen negocio solo pueden modelar el sistema con HU.



Figura 4: Escenario No4 Metodología AUP-UCI

**Rational Unified Process o Proceso Unificado de Rational** (RUP, por sus siglas en inglés) es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de *software*, para diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y tamaños de proyecto. (Álvarez, 2015)

RUP es una metodología flexible y fácil de adaptarse a las necesidades de cualquier proyecto, esta brinda al equipo de desarrollo guías consistentes y personalizadas del proceso. El mismo, en conjunto con el UML, constituye la metodología estándar utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Utiliza UML para definir los modelos de *software* y puede definirse como un modelo que es dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. (Álvarez, 2015)

#### **Selección de la metodología de *software***

Luego de analizar las metodologías anteriormente mencionadas se ha seleccionado para el desarrollo de la herramienta para el despliegue en el Centro FORTES la metodología AUP en su variación UCI, debido a que describe una manera fácil y ágil de entender la forma de desarrollar

aplicaciones de *software* de negocio usando técnicas ágiles y conceptos que aún se mantiene válidos en RUP. El AUP-UCI aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas.
- Modelado Ágil.
- Gestión de cambios ágil.
- Refactorización de Bases de Datos para mejorar la productividad.
- La metodología AUP-UCI es la adopción de muchas de las técnicas ágiles de XP y otros procesos ágiles que mantiene RUP.
- Dentro de la metodología se seleccionó el escenario número cuatro, ya que este se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información.

Por tanto, esta metodología guiará al desarrollo de la aplicación ya que se adapta a las condiciones existentes y además está avalada por el departamento de Calidad-UCI.

### **1.4 Lenguajes de desarrollo**

Un lenguaje de desarrollo es una técnica estándar de comunicación que puede ser utilizada para controlar el comportamiento de una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen la estructura de un lenguaje informático.

#### **JavaScript**

JavaScript es un lenguaje de programación, al igual que PHP, si bien tiene diferencias importantes con éste. Se utiliza principalmente del lado del cliente (es decir, se ejecuta en nuestro ordenador, no en el servidor) permitiendo crear efectos atractivos y dinámicos en las páginas web. Los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Un usuario escribe una dirección web en su navegador, por ejemplo <http://www.aprenderaprogramar.com>. El servidor recibe la petición y como respuesta a esa petición envía al ordenador del usuario código HTML junto a código JavaScript.

El código HTML se encarga de que en la pantalla se muestre algo, por ejemplo, una imagen, un menú, etc. El código JavaScript se puede encargar de crear efectos dinámicos en respuesta a

acciones del usuario, por ejemplo, que se despliegue un menú tipo acordeón cuando el usuario pasa el ratón por encima de un elemento del menú. La ventaja es que al estar alojado en el ordenador del usuario los efectos son muy rápidos y dinámicos. Al ser un lenguaje de programación permite toda la potencia de la programación como uso de variables, condicionales, bucles, etc. No obstante, hoy día la mayoría de los usuarios navegan por la web con JavaScript activado. Aunque es un lenguaje de programación orientado a objetos no tiene herencia, al contrario del Java que, si la tiene, sino que el JavaScript es más bien un lenguaje orientado a eventos. Otra diferencia entre ambos lenguajes es que mientras con el Java podemos crear aplicaciones autónomas como son los applets (programas que podemos incluir en las páginas web), el JavaScript es un lenguaje que se incorpora dentro de la página web, formando parte del código HTML sin el que no puede existir. (Asensio, 2015)

### **Bash**

El intérprete de mandatos o "*Shell*" es la interfaz principal entre el usuario y el sistema, permitiéndole interactuar con los recursos del ordenador. El usuario introduce sus órdenes, el intérprete las procesa y genera la salida correspondiente. Por lo tanto, un intérprete de mandatos de Unix es tanto una interfaz de ejecución de órdenes y utilidades, como un lenguaje de programación, que admite crear nuevas órdenes – denominadas guiones o "*ShellScript*"–, utilizando combinaciones de mandatos y estructuras lógicas de control, que cuentan con características similares a las del sistema y que permiten que los usuarios y grupos de la máquina cuenten con un entorno personalizado. En Unix existen 2 familias principales de intérpretes de mandatos: los basados en el intérprete de Bourne (BSH, KSH o BASH) y los basados en el intérprete C (CSH o TCSH). Este lenguaje permite comprender, ejecutar y empezar a programar en la *Shell*, haciendo referencia especialmente a BASH (*Bourne Again Shell*) – evolución de BSH, con características de KSH y CSH–, ya que es el intérprete de mandatos más utilizado en Linux e incluye un completo lenguaje para programación estructurada y gran variedad de funciones internas.

### **Características principales**

- Ejecución síncrona de órdenes (una tras otra) o asíncrona (en paralelo).
- Distintos tipos de redirecciones de entradas y salidas para el control y filtrado de la información.
- Control del entorno de los procesos.

- Ejecución de mandatos interactiva y desatendida, aceptando entradas desde teclado o desde ficheros.
- Proporciona una serie de órdenes internas para la manipulación directa del intérprete y su entorno de operación.
- Un lenguaje de programación de alto nivel, que incluye distintos tipos de variables, operadores, matrices, estructuras de control de flujo, entrecomillado, sustitución de valores y funciones.
- Control de trabajos en primer y segundo plano.
- Edición del histórico de mandatos ejecutados.
- Posibilidad de usar una "*Shell*" para controlar el entorno del usuario. (Labrador,2005)

### Ruby

Es un lenguaje de programación interpretado, reflexivo y orientado a objetos. Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos similares a Smalltalk. Comparte también funcionalidad con otros lenguajes de programación como Lisp, Lua, Dylan y CLU. Es un lenguaje de programación interpretado y su implementación oficial es distribuida bajo una licencia de *software* libre. Ruby sigue el "principio de la menor sorpresa", lo que significa que el lenguaje debe comportarse de tal manera que minimice la confusión de los usuarios experimentados. Es orientado a objetos: todos los tipos de datos son un objeto, incluidas las clases y tipos que otros lenguajes definen como primitivas, (como enteros, booleanos, etcétera). Este lenguaje soporta herencia con enlace dinámico, mixins<sup>5</sup> y métodos singleton (pertenecientes y definidos por una sola instancia más que definidos por la clase). A pesar que no soporta herencia múltiple, las clases pueden importar módulos como mixins. (Matz, 2003)

Ruby ha sido descrito como un lenguaje de programación multiparadigma: permite programación procedural y orientada a objetos. Además de soporte para hilos de ejecución gestionados por el intérprete. Este lenguaje tiene tipado dinámico y soporta varios tipos de polimorfismo (permite tratar a subclases utilizando la interfaz de la clase padre). No requiere polimorfismo de funciones al no ser fuertemente tipado. La sintaxis de Ruby es similar a la de Perl y Python. La definición de clases

---

<sup>5</sup> En los lenguajes de programación orientada a objetos, un mixins es una clase que ofrece cierta funcionalidad para ser heredada por una subclase, pero no está ideada para ser autónoma. Heredar de un mixins no es una forma de especialización sino más bien un medio de obtener funcionalidad.

y métodos está definida por palabras claves. Sin embargo, en Perl, las variables no llevan prefijos. La mayor diferencia con C y Perl es que las palabras clave son usadas para definir bloques de código sin llaves. Los saltos de línea son significativos y son interpretados como el final de una sentencia; el punto y coma tiene el mismo uso. De forma diferente que Python, la indentación no es significativa. (Matz, 2003)

### **Python (Castillo, 2008)**

Python es un lenguaje de programación interpretado, su filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Se trata de un lenguaje potente, flexible y con una sintaxis clara y concisa. Además, no requiere dedicar tiempo a su compilación debido a que es interpretado. Python es open source, cualquier usuario puede contribuir a su desarrollo y divulgación. Además, no es necesario pagar ninguna licencia para distribuir *software* desarrollado con este lenguaje. Hasta su intérprete se distribuye de forma gratuita para diferentes plataformas. (Python, 2016)

A continuación, se listan las principales características que este lenguaje posee:

- **Simple:** Es un lenguaje muy simple, por lo que es muy fácil iniciarse en él. El pseudo-código natural es una de sus grandes fortalezas.
- **Propósito General:** Usando el lenguaje Python se puede crear todo tipo de programas; programas de propósito general y también se pueden desarrollar páginas Web.
- **Open Source:** Debido a la naturaleza de Python de ser Open Source; ha sido modificado para que pueda funcionar en diversas plataformas (Linux, Windows, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE y PocketPC). Al ser Open Source es gratuito.
- **Lenguaje Orientado a Objetos:** Al ser un Lenguaje Orientado a Objetos es construido sobre objetos que combinan datos y funcionalidades.
- **Lenguaje de Alto Nivel:** Al programar en Python el usuario no debe preocuparse por detalles de bajo nivel, (como manejar la memoria empleada por el programa).



- **Incrustable:** Se puede insertar lenguaje Python dentro un programa C/C++ y de esta manera ofrecer las facilidades del scripting<sup>6</sup>.
- **Extensas Librerías:** Contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero. Las librerías pueden ayudar a hacer varias cosas como expresiones regulares, generación de documentos, evaluación de unidades, pruebas, procesos, bases de datos, navegadores web, CGI, ftp, correo electrónico, XML, XML-RPC, HTML, archivos WAV, criptografía, GUI.
- **Sintaxis clara:** Tiene una sintaxis muy visual, gracias a que maneja una sintaxis indentada (con márgenes), que es de carácter obligatorio. Para separar los bloques de código en Python se debe tabular hacia dentro. Esto ayuda a que todos los programadores adopten las mismas notaciones y que los programas hechos en Python tengan un aspecto muy similar.

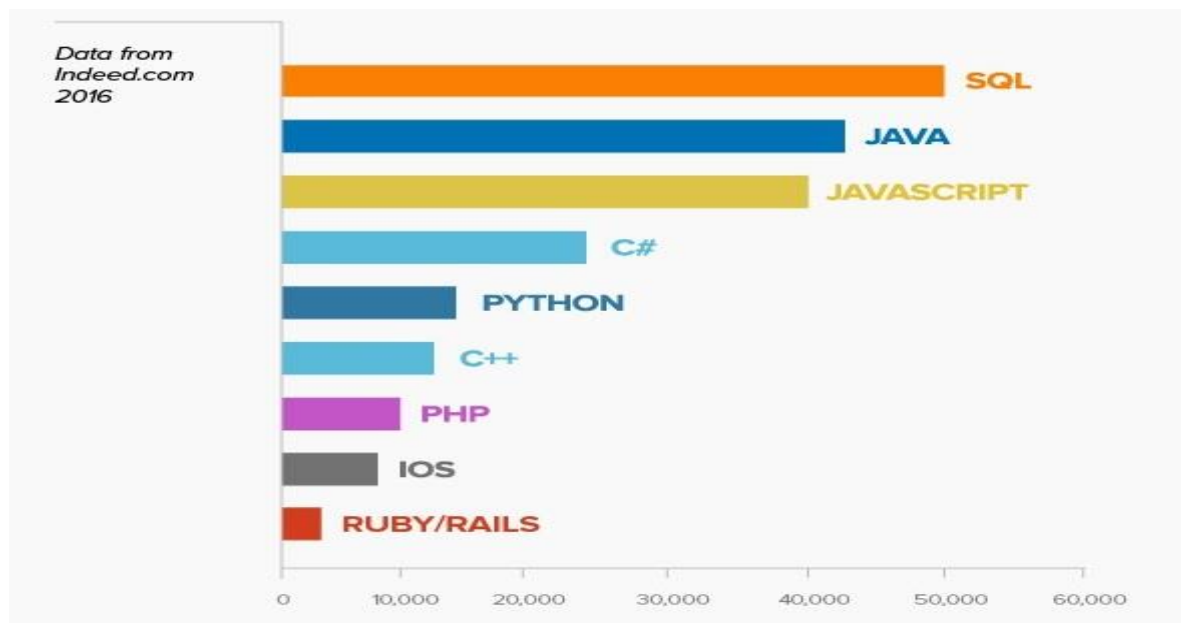


Figura 5: Lenguajes de programación

La figura muestra una gráfica donde se representa una comparación de los lenguajes de programación más utilizados en torno al desarrollo de script en los últimos años. Comprobando la estabilidad y seguridad que brindan estos lenguajes a partir de su curva de aprendizaje, sus

<sup>6</sup> (Scripting language, lenguaje de guión). Un lenguaje scripting es un tipo de lenguaje de programación que es generalmente interpretado.

principales características; evidenciado el criterio de sus comunidades de usuarios y de desarrolladores.

### **Selección del lenguaje de programación**

Al analizar los diferentes lenguajes de programación mencionadas anteriormente, se ha seleccionado para dar respuesta al desarrollo de la herramienta para el despliegue bajo la arquitectura Xalix del Centro FORTES el lenguaje Python, debido a que es un lenguaje muy simple, se puede crear todo tipo de programas. También se seleccionó este lenguaje porque presenta el paradigma Orientado a Objetos ya que está construido sobre objetos que combinan datos y funcionalidades. Al programar en Python, el usuario no debe preocuparse por detalles de bajo nivel. Debido a la naturaleza de Python de ser Open Source; ha sido modificado para que pueda funcionar en diversas plataformas; al ser Open Source es gratuito.

### **1.5 Entorno de desarrollo**

Pycharm IDE fue creado por JetBrains. Este IDE es profesional y viene en dos modalidades: una edición libre y otra muy completa privada que apunta a empresas de desarrollo de *software*. La popularidad de este programa se puede medir a partir de que grandes empresas como Twitter, Groupon, Spotify, eBay y telefónica, han utilizado éste para su trabajo. La mayoría de sus características están disponibles en la versión gratuita, se integra con IPython, soporta Anaconda, así como otros paquetes científicos como matplotlib y NumPy. Características como desarrollo remoto, soporte de bases de datos, soporte de frameworks de desarrollo web; están disponibles solo para la edición profesional. Algo muy útil es su compatibilidad con múltiples marcos de desarrollo web de terceros como Django, Pyramid, web2py, motor de aplicaciones Google y Flask, convirtiéndolo en un completo IDE de desarrollo de aplicaciones rápidas y seguras. (Travel & Umroh, 2016)

#### **Características:** (Travel & Umroh, 2016)

- Apoyo avanzado para el desarrollo de Django, incluyendo finalización de código inteligente, inspecciones, refactorizaciones y navegación de código específico.
- Posee Python Debugger integrado y una unidad de pruebas gráficas en marcha.
- Apoyo para la cadena de herramientas de desarrollo Python modernas, incluyendo virtualenv y buildout.

- Integración de Sistema de Control de Versiones - IU unificada para VCS más popular como Mercurial, Git, SVN, Perforce y CVS.
- Soporte para el Google App Engine, permite ejecutar y desplegar sus aplicaciones para el servidor App Engine, al tiempo que asegura que el código cumple los requisitos del entorno sandbox.
- Consolas REPL y Django con muchas ventajas sobre el estándar: comprobación de sintaxis sobre la marcha con inspecciones; apoyos y citas similares y auto inserción de pares, además de finalización de código.

PyCharm 2016.1 está disponible como una edición profesional completa para Python y desarrollo web, o como una edición de código abierto y de código abierto para Python puro y el desarrollo científico. (Graterol, 2014)

Estos son algunos aspectos destacados de esta versión.

Mejoras relacionadas con Python:

- Inspección de insinuación y compatibilidad de tipo Python 2 y Python 3.
- Ajustes del contenedor Docker Compose y Docker.
- Ayuda Tox.
- Soporte mejorado para los formularios de Django.
- Mejoras significativas en el depurador.
- Soporte mejorado para IPython Notebook.

Mejoras en la plataforma:

- Git Rebase y cambio de nombre.
- Soporte de Git Worktree.
- Cambios por palabra que se destacan en Diff Viewer.
- Herramientas de base de datos mejoradas.
- Mejoras en terminales locales.

PyDev IDE es libre de costo y está lleno de características poderosas para programar en Python. Es un plugin de código abierto y se ejecuta en Eclipse. Presenta integración con Django, completa el código de manera automática, soporta varios lenguajes, plantillas de código, análisis de código, marcado de errores. Se mantiene siempre actualizado y contiene una gran comunidad de usuarios y empresas de patrocinio como Licolipse, Squish, TraceTronic. Aunque PyDev califica como uno de los mejores IDE de Python de código abierto, también viene empaquetado junto con otro producto llamado Licolipse, un producto comercial construido sobre Eclipse que proporciona mejoras en la usabilidad y temas adicionales.

Wing IDE es comercial y apunta a desarrolladores profesionales. Fue lanzado hace 15 años atrás y es un producto muy maduro, con una cantidad de herramientas y características para programar en Python. Es soportado por Windows, OS X Linux. Presenta al igual que Pycharm una versión básica gratuita, una edición personal y una profesional muy potente. En el Debugging es donde más brilla e incluye funciones como depuración de procesos múltiples, depuración de subprocesos, depuración automática de procesos secundarios, puntos de interrupción, inspección de código, etc. También ofrece funciones para depurar remotamente el código que se ejecuta en Raspberry PI. También soporta una gran cantidad de frameworks Python como: Maya, MotionBbuilder, Zope, PyQt, PySide, pyGTK, Django, matplotlib.

Vim IDE es uno de los editores más avanzados y populares dentro de la comunidad de desarrolladores Python. Es de código abierto y se encuentra disponible gratuitamente bajo licencia GPL. Sin embargo, es más conocido como editor, aunque nos ofrece un entorno completo de desarrollo para Python cuando está configurado correctamente. Es ligero, modular y el más adecuado para los amantes del teclado, para los que no utilizan el mouse mientras se escribe código. La configuración inicial puede llevar un poco de tiempo ya que es necesario utilizar varios complementos VIM para que funcione de la manera que queramos.

### **Selección del IDE de desarrollo**

Luego de analizar los diferentes IDEs, se ha seleccionado para el desarrollo de la herramienta de despliegue bajo la arquitectura Xalix del Centro FORTES el IDE Pycharm, debido a sus principales características como: es soportado por los sistemas operativos Windows, OS y Linux. PyCharm es un IDE completo con un editor altamente personalizable y potente heredado de la plataforma IntelliJ. Presenta una versión básica gratuita, una edición personal y una profesional muy potente. Además, presenta una herramienta de bases de datos mejorada. Este IDE de desarrollo brinda también la posibilidad de comprobación de sintaxis sobre la marcha con inspecciones; apoyos y

citas similares y auto inserción de pares, además de finalización de código. Está dirigido en apoyar la cadena de herramientas de desarrollo Python modernas, soportando mejoras para IPython Notebook. Es compatible con múltiples marcos de desarrollo web de terceros como Django, Pyramid, web2py, motor de aplicaciones Google y Flask, lo que lo convierte en un competo IDE de desarrollo de aplicaciones rápidas. Otra de las características por la cual se selecciona este IDE es por su gran popularidad que se puede medir a partir de que grandes empresas como Twitter, Groupon, Spotify, eBay y telefónica.

### 1.6 Lenguaje para el modelado

#### Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas, más conocido y utilizado en la actualidad; está respaldado por el OMG (*Object Management Group*). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML es un lenguaje que proporciona un vocabulario y reglas para permitir una comunicación, este se centra en la representación gráfica de un sistema. Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones: (FOWLER,2003)

- Visualizar: permite expresar de una forma gráfica un sistema de forma que otro lo pueda entender.
- Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura visión.

#### Utilidad de UML

UML es un lenguaje para modelamiento de propósito general evolutivo, ampliamente aplicable. Se aplica a una multitud de diferentes tipos de sistemas, dominios y métodos o procesos.

- Como lenguaje de propósito general, se enfoca en un conjunto de conceptos para la adquisición, compartición y utilización de conocimientos emparejados con mecanismos de extensión.

- Como un lenguaje para modelamiento ampliamente aplicable, puede ser montado en diferentes tipos de sistemas, dominios (negocios, *software*) y métodos o procesos. Como un lenguaje para modelamiento soportable por herramientas, estas están disponibles para soportar la aplicación del lenguaje para especificar, visualizar, construir y documentar sistemas.
- Es utilizado para el modelamiento industrialmente estandarizado, es un lenguaje abierto y totalmente extensible reconocido por la industria. (FOWLER,2003)

### Visual Paradigm para UML 8.0

**CASE** *Computer Aided software Engineering (computadora auxilio ingeniería del software)* es el conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software*. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de *software*. (Bittner, 2003)

Visual Paradigm para UML es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, implementación y pruebas, además es considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos UML. Teniendo en cuenta sus características y los beneficios que brinda para la construcción de *software*, especialmente referente al modelado, se decidió utilizar Visual Paradigm para UML para el modelado de la aplicación. (González y otros, 2012)

### Características

- Soporta aplicaciones Web.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

## Ventajas

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Genera modelos VP-UML instantáneamente a partir de código binario .Net.
- Generación de documentación en formatos HTML y PDF.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X y Solaris.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo *Visio* y *Rational Rose*.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.

Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas. (Visual Paradigm for UML, 2016)

## Rational Rose

Es una solución de gestión del ciclo de vida del *software* que permite colaboración contextual en tiempo real para equipos distribuidos. Proporciona una configuración del proceso, un marco de guía y obligatoriedad que da soporte a todo el entorno de entrega de *software*.

- Mejora la colaboración del equipo con características integradas, que incluye la gestión de la configuración de *software*, la creación y los elementos de trabajo.
- Proporciona una gran visibilidad de actividades de proyectos y progreso del equipo con paneles de instrumentos de varios niveles y características de creación de informes.
- Facilita la planificación y ejecución de proyectos ágiles o formales con plantillas y herramientas de planificación. Las existencias de procesos coherentes mejoran la calidad del *software*.

- Ayuda a mejorar la productividad con control de origen avanzado para equipos distribuidos geográficamente.
- Mejora la colaboración del equipo con características integradas.
- Proporciona una gran visibilidad de actividades de proyectos y progreso del equipo.
- Ayuda a mejorar la productividad con control de origen avanzado.

DIA es un programa para la creación de diagramas GTK+ para GNU/ Linux, Unix y Windows liberado bajo la licencia GPL. Está inspirado en el programa comercial Windows Visio, aunque más orientado hacia diagramas sencillos para el uso ocasional. Puede ser utilizado para dibujar diferentes tipos de diagramas. Actualmente posee objetos especiales para ayudar a dibujar diagramas entidad relación, diagramas UML, organigramas, diagramas de red, y muchos otros diagramas. También es posible añadir soporte para nuevas formas de escritura mediante archivos XML, usando un subconjunto de SVG para dibujar las formas. (González, 2016)

### **Selección de la herramienta de modelado**

Al analizar las diferentes herramientas de modelado, se ha seleccionado Visual Paradigm con el objetivo de realizar el ciclo de vida completo de la Herramienta para el despliegue de *software* bajo la arquitectura Xalix. También ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite construir al usuario diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación. Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes. Presenta un apoyo adicional en cuanto a generación de artefactos.

### **1.7 Herramientas usadas por la arquitectura Xalix**

El Centro FORTES desarrolla una serie de proyectos enmarcados en la línea Xauce dedicada a la educación, dentro de ellos se encuentran la plataforma educativa ZERA, el Sistema de Gestión de ingreso a la Educación Superior y Félix Varela. Para el desarrollo de estos proyectos fue utilizada la Arquitectura Xalix, es una arquitectura definida para los proyectos de producción. Para que esta arquitectura funcione correctamente necesita de ciertos programas como:

#### **Nginx**

- Versión 1.9.5 o mayor. Utilizado como servidor web.



### PHP

- Versión 7.0.13 o mayor. Utilizado como lenguaje de programación del lado del servidor.
- Se necesitan además ciertas librerías como: php7-cli, php7-common, php7-curl, php7-fpm, php7-gd, php7-intl, php7-json, php7-pgsql.

### PostgreSQL

- Versión 9.5.0 o mayor.
- Utilizado como servidor de bases de datos.

### Openjdk

- Versión 8 o mayor.

### Mozilla Firefox

- Versión 50 o mayor
- Utilizado como navegador web.

## 1.8 Conclusiones del capítulo

En el presente capítulo se analizaron los conceptos fundamentales asociados a la presente investigación como los tipos de instalación existentes. Además, se abordó la definición de script, sus principales características y funciones. Se identificaron las tecnologías y herramientas que serán usadas en el desarrollo de la solución. A partir de los aspectos antes mencionados se arriba a las siguientes conclusiones:

- El estudio de los diferentes sistemas homólogos existentes en el ámbito internacional, nacional e institucional, así como de sus principales tendencias en la actualidad permitió precisar las características fundamentales que se incorporarían durante el desarrollo de la Herramienta para el despliegue de *software* en el Centro FORTES. El análisis y comparación de los principales conceptos y bases teóricas relacionados con aquellos factores determinantes en la construcción de un script permitieron definir a AUP-UCI como metodología de desarrollo de *software*. El empleo de herramientas como PyCharm 2016.1.4, Visual Paradigm, posibilitaron el desarrollo de la propuesta de solución. La utilización del lenguaje de programación Python 2.7, permitió desarrollar la herramienta para solucionar los problemas detectados.

## Capítulo II: Propuesta de Solución

### 2.1 Introducción

En el presente capítulo se ejemplifican las características del sistema a desarrollar, definiendo el modelo de dominio según el objeto de estudio haciendo uso de la metodología AUP-UCI. Se identifican los requisitos con los que debe cumplir la herramienta para el despliegue de *software*. Se explican además la arquitectura y el diseño del sistema a desarrollar.

### 2.2 Propuesta de solución

El presente trabajo va destinado a la implementación y prueba de la herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES, garantizando de esta forma que el proceso de despliegue del *software* se realice de manera rápida y efectiva. Para realizar esta tarea se llevará a cabo la gestión de los programas definidos por la arquitectura, la configuración del servidor web Nginx y las formas de configurar estos programas y servidores definidos por el usuario. El contenido de la herramienta se encuentra estructurado de la siguiente forma:

El usuario para usar la aplicación debe tener permiso de administrador en el ordenador donde se encuentra. Para hacer eficiente esta herramienta será necesario comprobar la conexión con el repositorio, si está configurado y la versión del Sistema Operativo; de no estar correctamente configurados la herramienta posibilitará realizar estas operaciones. La aplicación mostrará un menú, en el cual se encontrarán todas las opciones; el usuario debe seguir cada instrucción para garantizar que el *software* pueda ser desplegado correctamente. Permitirá realizar la instalación completa, esta acción se puede realizar de forma organizada todas las opciones del menú. También una característica especial es la selección de la instalación de forma personalizada, posibilitará la instalación de los programas que contribuirán al despliegue de las aplicaciones que estén bajo la arquitectura Xalix realizando esta operación con pasos ordenados. Otro de los objetivos de esta herramienta será que el usuario pueda agregar programas definidos por él, para su entorno de trabajo. Con el objetivo de que los servidores y programas instalados funcionen correctamente, la herramienta para el despliegue bajo la arquitectura Xalix solicitará al usuario los valores necesarios para dejar configurado el entorno de trabajo.

### 2.3 Modelo de dominio

El modelo del dominio captura, comprende y describe los objetos más importantes dentro del contexto de un sistema. Se describe mediante diagramas de UML, especialmente mediante

diagramas de clases. Los objetos del dominio o clases, pueden obtenerse a partir de una especificación de requisitos o mediante la entrevista con los expertos del tema. Las clases que lo componen suelen aparecer en tres formas típicas:

- Captura los tipos más importantes de objetos en el contexto de sistema.
- Objetos del negocio que representan elementos que se manipulan en el negocio.
- Objetos del mundo real y conceptos de los que el sistema debe hacer seguimiento.
- Sucesos que ocurren o han ocurrido.

### Definición de las clases

**Centro FORTES:** centro especializado a la realización y elaboración de *software* en la universidad.

**Administrador:** persona encargada de instalar la aplicación mediante su contraseña de root<sup>7</sup> o de administrador.

**Proyectos:** es una planificación que consiste en un conjunto de actividades que se encuentran interrelacionadas y coordinadas.

**Arquitectura Xalix:** arquitectura definida para los proyectos del Centro FORTES como SIGIES, ZERA y Félix Varela, utilizada para desarrollar *software*.

**Información:** información suministrada por la aplicación para contribuir a su utilización.

**Programas:** son los programas que están enmarcados bajo la arquitectura Xalix.

**Repositorio:** es un medio para gestionar, almacenar, preservar, difundir y facilitar el acceso a los objetos digitales que alberga.

---

<sup>7</sup> Es la contraseña utilizada por el usuario para acceder a un ordenador. Se utiliza en Linux con el objetivo de realizar cambios en el sistema.

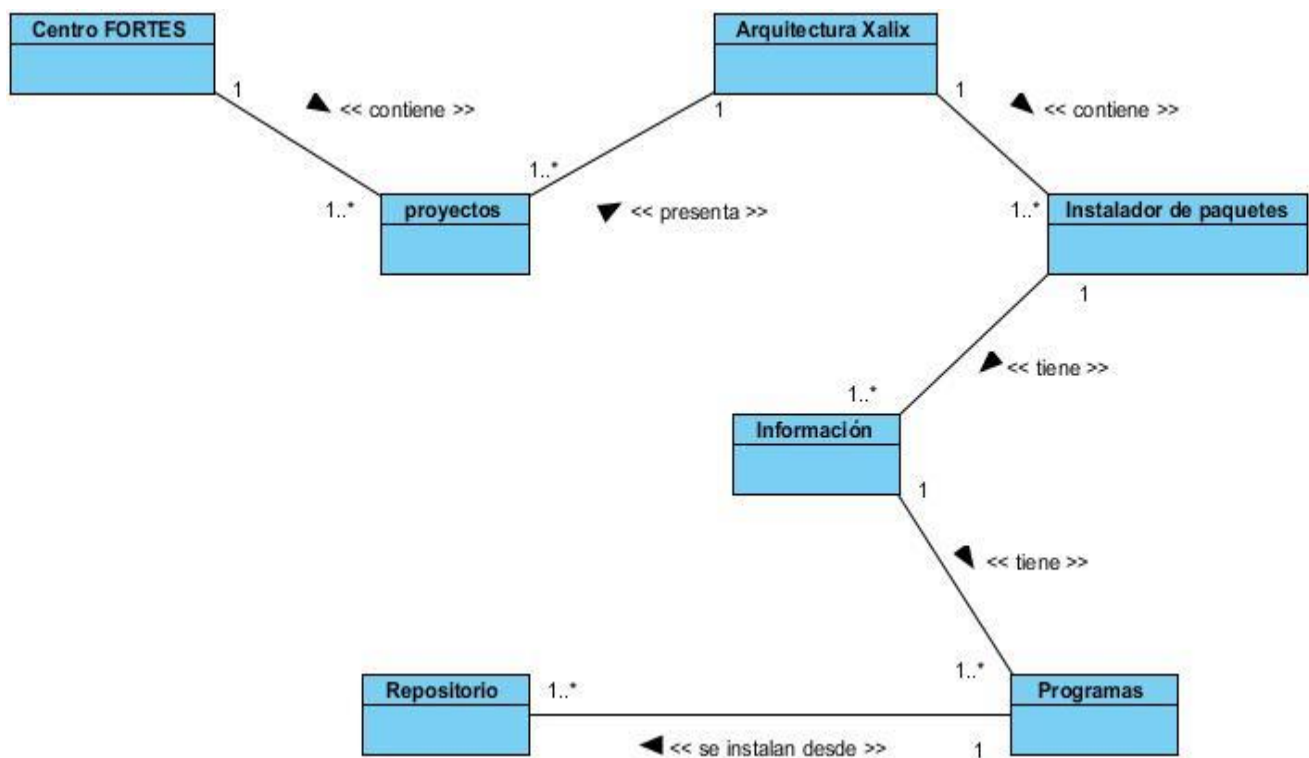


Figura 6: Modelo de dominio

## 2.4 Descripción del sistema propuesto

A continuación, en la figura se muestra cómo quedaría conformado el flujo de las acciones que debe chequear el *script* para su correcto funcionamiento. Los pre-requisitos que tiene el script son los siguientes:

- Verificar los permisos de administrador o de root para comenzar la instalación.
- Verificar la versión del repositorio, actualizarlo e instalarlos dependiendo del sistema operativo instalado.
- Instalar las librerías necesarias de Python.

De esta manera quedarían instalados los pre-requisitos de la Herramienta para el despliegue de *software*. Una vez que estén en funcionamiento cada uno de estos, comenzaría la instalación.

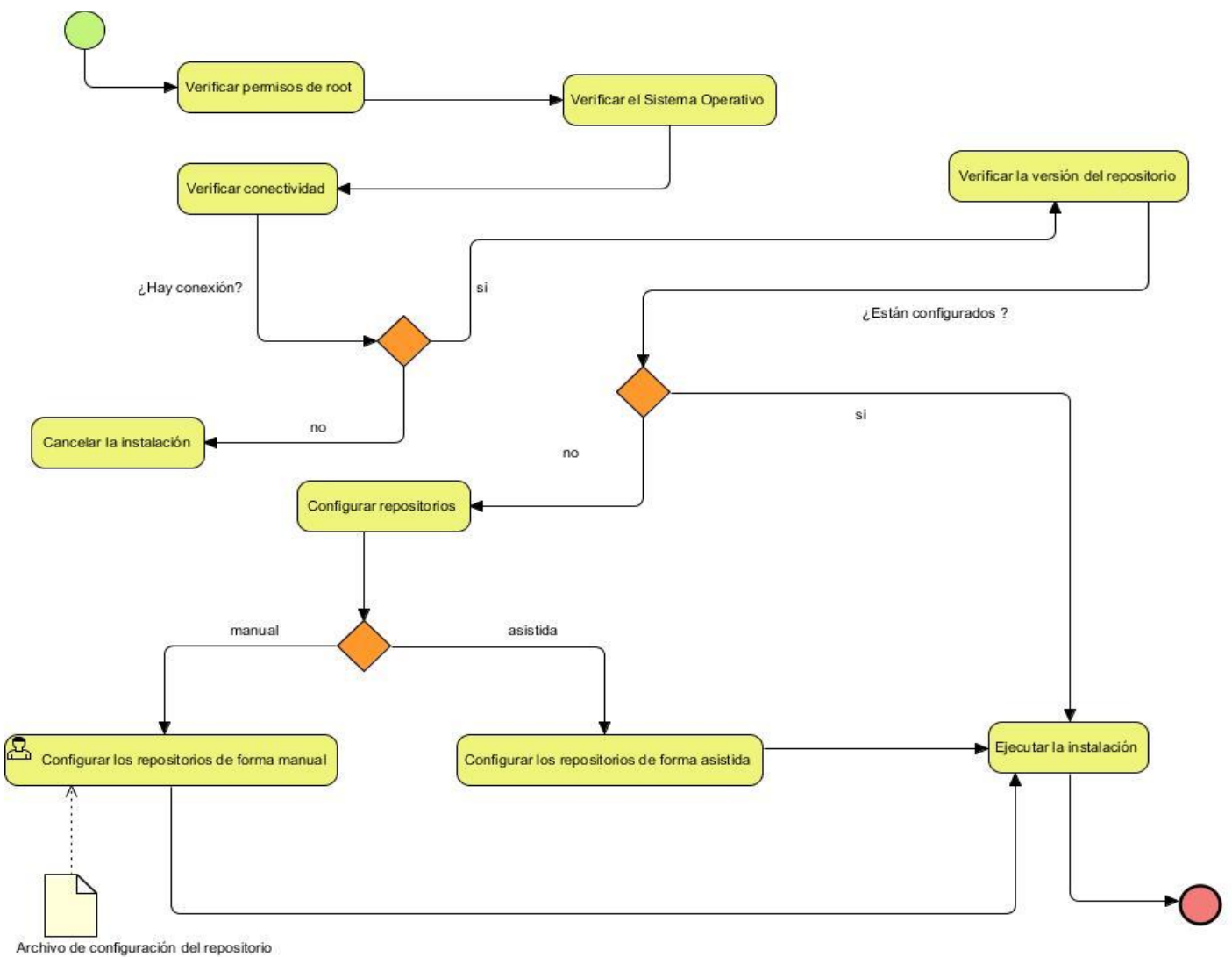


Figura 7: Flujo de instalación de los pre-requisitos

La figura muestra cómo quedaría conformado el flujo de acciones que comprende la plataforma de instalación una vez ejecutado el instalador. Al ejecutar dicho instalador lo primero que se hace es chequear que los pre-requisitos mencionados, si al verificar cada uno de estos no se cumplen, la aplicación no pasará a la siguiente fase de la instalación y deberá instalarlos a través de la opción que brinda el script. Una vez verificada, se procede a seleccionar el tipo de instalación deseada por el usuario. Si selecciona la opción instalación completa, la aplicación accederá a instalar las opciones que brinda el menú. La aplicación también posee una opción en la cual al seleccionarla se instalarán los programas que necesita la plataforma Xalix. El usuario también tiene como opción agregar los programas adicionales, donde tendrá que escribirlos en un archivo, con el fin de que la aplicación los pueda instalar satisfactoriamente. Luego de tener definido cada acción se procederá a instalar los programas desde el repositorio. Una vez instalados estos programas el script

procederá a configurar los programas necesarios que se encuentran bajo la arquitectura Xalix. Y así el entorno de trabajo estará listo para utilizarlo.

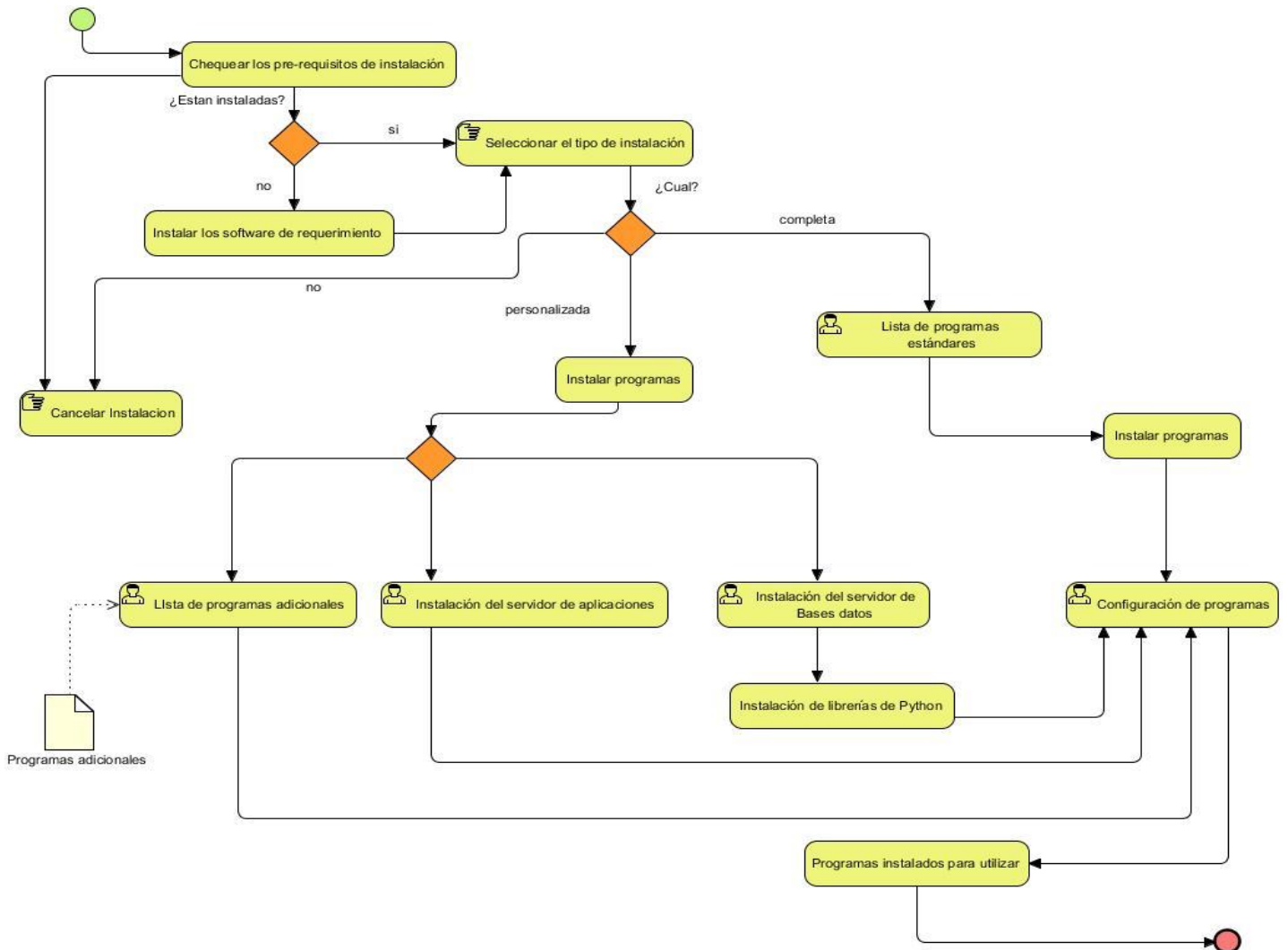


Figura 8: Flujo de instalación de la aplicación

## 2.5 Requisitos funcionales y no funcionales

El flujo de trabajo Requerimientos, constituye uno de los flujos más importante que propone la metodología AUP-UCI, pues en el mismo se capturan los requisitos del sistema. Los mismos tienen como objetivo fundamental guiar el desarrollo hacia un sistema correcto.

### 2.5.1 Requisitos funcionales

Son declaraciones de servicios que el sistema debe brindar; cómo el sistema debe reaccionar en determinadas entradas y comportarse ante situaciones particulares:

- **RF1. Verificar Permisos de root.** Permite que la aplicación solo se ejecute con permisos de administrador.
- **RF2. Verificar el Sistema Operativo.** Permite que la aplicación identifique el sistema operativo para luego configurar los repositorios.
- **RF3. Verificar conectividad.** La aplicación permite comprobar la conectividad que existe con el repositorio para realizar las acciones de instalación y configuración.
- **FR4. Verificar la configuración archivo PIP:** La aplicación realizará una comprobación si el archivo pip esta creador y se encuentra configurado correctamente, de no estarlo pasaría a gestionar su configuración.
- **RF5. Ver los repositorios.** La aplicación permite mostrar al usuario los repositorios que serán colocado en el fichero de configuración dependiendo del Sistema Operativo.
- **RF6. Crear los repositorios.** La aplicación permite crear los repositorios y colocarlos en el archivo de configuración dependiendo del Sistema Operativo.
- **RF7. Modificar los repositorios.** Permite a la aplicación que compruebe si los repositorios están configurados correctamente, en caso de no ser así; los configura en dependencia de la versión del Sistema Operativo correspondiente.
- **RF8. Instalar las librerías dependientes de Python.** Permite a la aplicación instalar las librerías necesarias de Python para su correcto funcionamiento.
- **RF9. Mostrar Menú.** La aplicación mostrará un menú que le permitirá al usuario escoger la opción deseada.
- **RF10. Instalar y configurar los programas del servidor de aplicaciones.** Permite que el usuario seleccione esta opción para instalar los programas que están bajo la arquitectura Xalix y que se encuentran catalogados dentro del servidor de aplicaciones.
- **RF11. Instalar y configurar los programas del servidor de Bases datos.** Permite que el usuario seleccione esta opción para instalar los programas que están bajo la arquitectura Xalix y que se encuentran catalogados dentro del servidor de Bases datos.
- **RF12. Instalar programas adicionales.** El usuario después de seleccionar esta opción deberá colocar en un documento todos los programas que desee instalar para su entorno de trabajo.
- **RF13. Finalizar proceso de instalación.** Permite que el usuario pueda cancelar la instalación seleccionando la opción de salir.

## 2.5.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que no se refieren a las funcionalidades que proporciona el sistema, sino a las propiedades como restricciones del entorno o de la implementación, rendimiento, dependencias y soportes entre otras.

- **Usabilidad**

**RnF1.** La aplicación será distribuida en el idioma español.

**RnF2.** La aplicación poseerá una estructura y diseño homogéneo, que facilite su utilización.

**RnF3.** Se debe lograr una correcta estructura de la información, con el empleo de menús, que proporcionen que la aplicación sea entendible.

**RnF4.** Estructura simple y lo más intuitiva posible de la arquitectura de la información.

**RnF5.** El tiempo de respuesta de la aplicación no debe exceder de los 5 segundos para una petición realizada.

- **Soporte**

**RnF6.** El estándar de codificación es el definido en el marco de trabajo Xalix.

**RnF7.** La aplicación se puede utilizar en Ubuntu y sus derivados.

- **Restricciones de diseño**

**RnF8.** Lenguaje de Programación utilizado Python 2.7

**RnF9.** Entorno de desarrollo integrado Pycharm 2016.1.4

## 2.6 Descripción de requisitos

A continuación, se presentan la descripción de requisitos a través de las historias de usuarios.

<b>Número:</b> HU_10	<b>Nombre del requisito:</b> Instalar y configurar los programas del servidor de aplicaciones
<b>Programador:</b> Yusniel Armando Reynaldo	<b>Iteración Asignada:</b> 10ma



Molina	
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si el usuario no selecciona esta opción puede provocar que su entorno de trabajo no esté disponible para desplegar los <i>software</i> que estén bajo la arquitectura Xalix	<b>Tiempo Real:</b> 3 días
<p><b>Descripción:</b></p> <p><b>1- Objetivo:</b> Permitir instalar los programas para desplegar la arquitectura Xalix.</p> <p><b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> - Debe seleccionar esta opción para poder proceder a la instalación.</p> <p><b>3- Flujo de la acción a realizar:</b> Después de seleccionar la opción instalar y configurar los programas del servidor de aplicaciones, la aplicación mostrará un mensaje indicando los programas que comenzarán a instalarse y a configurarse respectivamente. Saldrá un mensaje donde podrá seleccionar si desea instalar mediante la opción(s/n). Si se selecciona la opción “n”, la aplicación mostrará un mensaje de cancelación y luego aparecerá un menú para seleccionar más opciones. Si el usuario selecciona la opción “s”, la aplicación accederá a instalar los programas mostrados anteriormente de forma organizada. Luego de realizar la instalación del primer programa mostrará un mensaje que se instaló correctamente. Después mostrar el mensaje que se instaló correctamente, la aplicación mostrará un mensaje diciendo si desea configurar la instalación mediante la opción(s/n); en caso de seleccionar la opción “n”, aparecerá un mensaje diciendo que la configuración del programa instalado se ha cancelado y tendrá la opción de instalar los otros programas pasando por el mismo ciclo. Después de realizar los pasos correspondientes aparecerá la opción del menú.</p>	
<b>Observaciones:</b> N/A	
<b>Prototipo de interfaz:</b>	

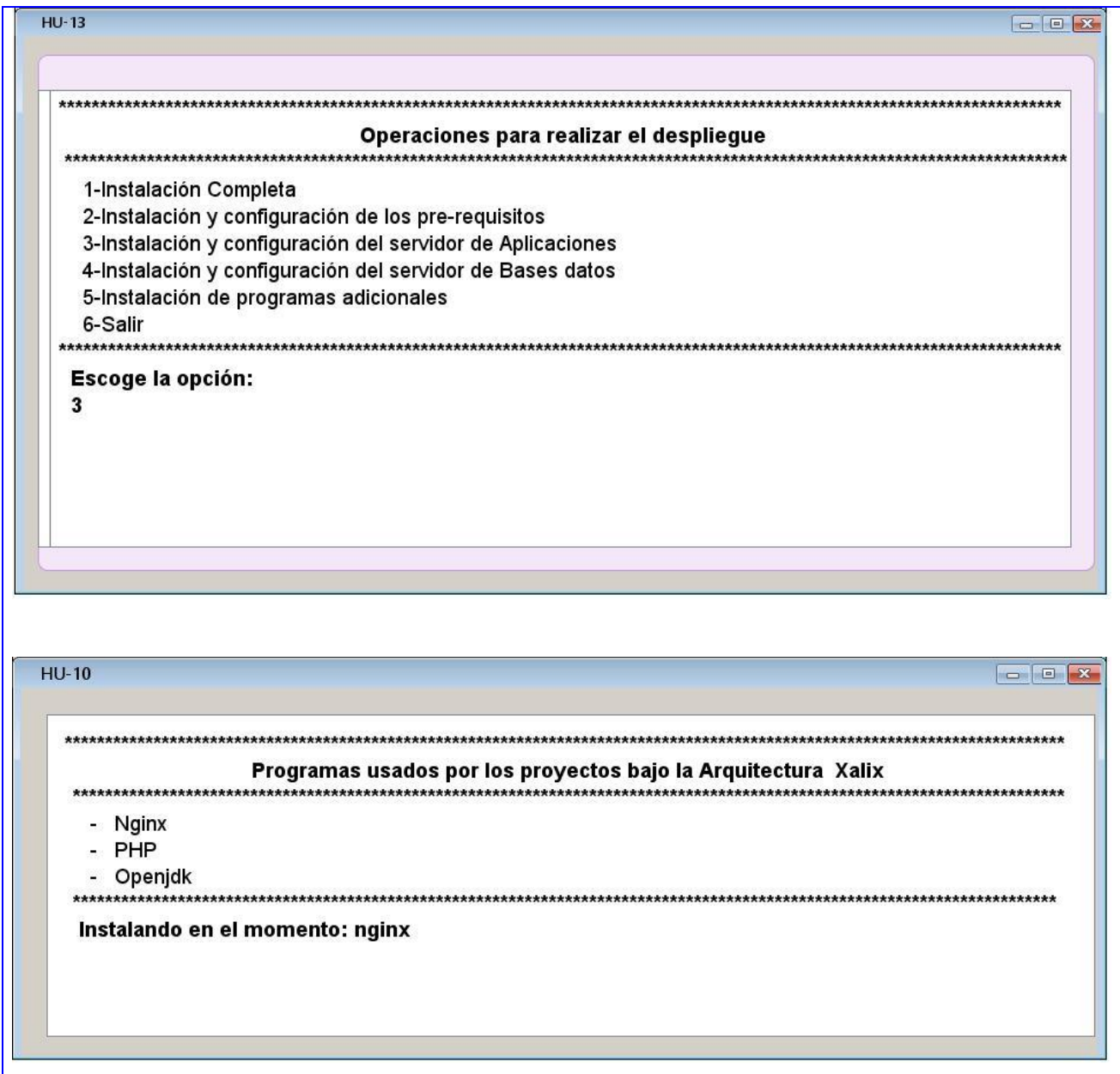


Tabla 3: HU\_ Instalar y configurar los programas del servidor de aplicaciones

<b>Número:</b> HU_13	<b>Nombre del requisito:</b> Instalar programas adicionales
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 13
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si el usuario no selecciona esta opción puede provocar que su entorno de trabajo no esté funcionando correctamente.	<b>Tiempo Real:</b> 3 días
<p><b>Descripción:</b></p> <p><b>1-Objetivo:</b> Permitir instalar programas adicionales para el entorno de trabajo</p> <p><b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> - Debe seleccionar esta opción para poder proceder a la instalación.</p> <p><b>3- Flujo de la acción a realizar:</b> La aplicación después de seleccionar esta opción, mostrará un mensaje indicando si desea instalar los programas colocados por usted en un archivo txt, mediante la opción(s/n). Después de seleccionar la opción “s” se le pedirá al usuario que entre la dirección donde se encuentran los programas a instalar. Luego mostrará la lista de programas que desea instalar. Saldrá un mensaje donde podrá seleccionar la opción (s/n). Si el usuario selecciona la opción si, la aplicación accederá a instalar todos los programas necesarios por usted. La aplicación mostrará un mensaje de confirmación de que los programas fueron instalados correctamente. Si se selecciona la opción “n”, la aplicación mostrara un mensaje de cancelación. La aplicación mostrará el menú para seleccionar más opciones.</p>	
<p><b>Observaciones:</b> Debe colocar los nombres de los programas en un archivo txt definido por la aplicación; estos no serán programas deben ser configurados por usted.</p>	
<p><b>Prototipo de interfaz:</b></p>	

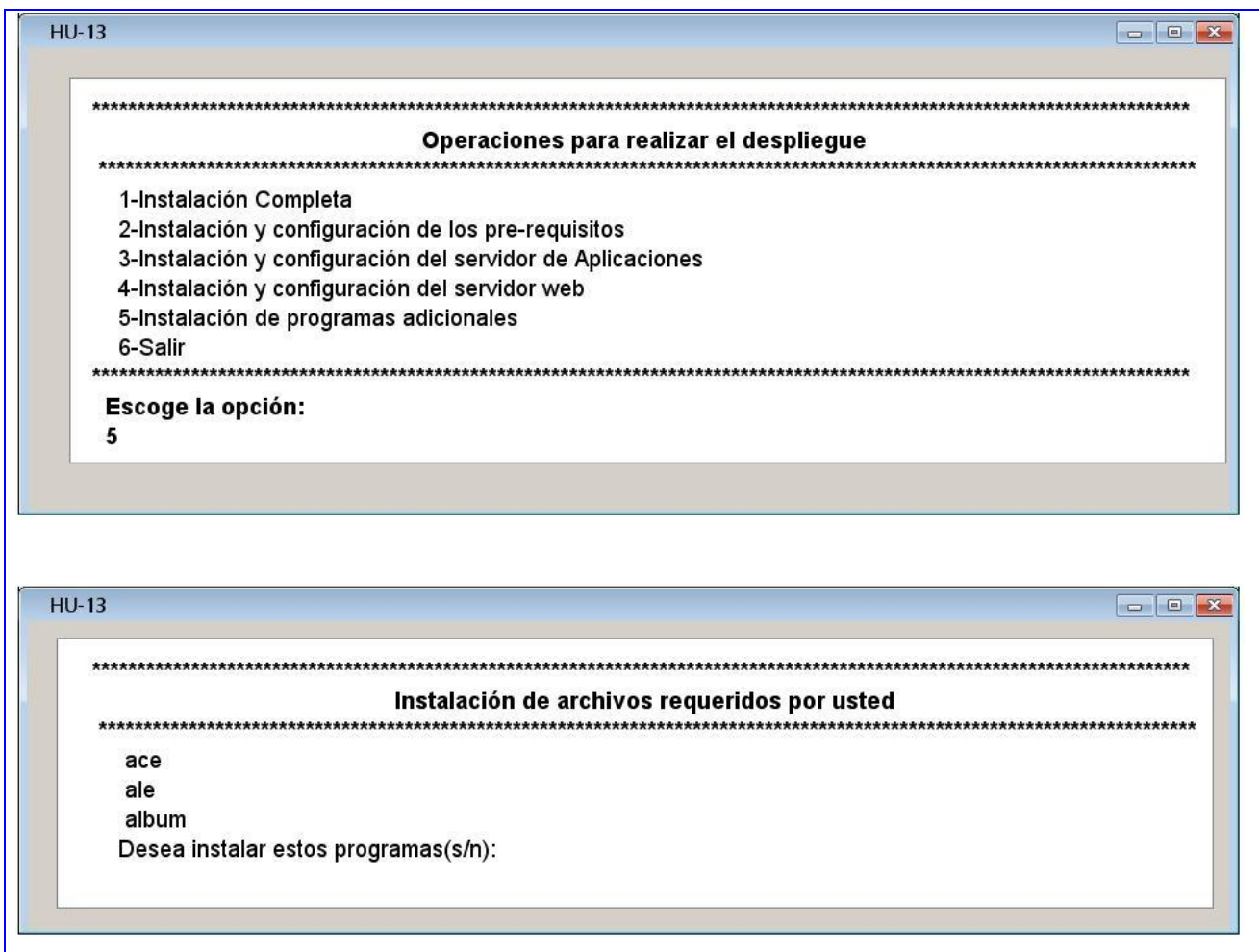


Tabla 4: HU\_ Instalar programas adicionales

## 2.7 Patrones arquitectónicos

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. Con el objetivo de guiar el proceso de desarrollo de *software* se seleccionó el patrón arquitectónico: Basada en eventos o componentes (Peláez, 2009)

La arquitectura basada en eventos o componentes consiste en una rama de la ingeniería de *software* en la cual se trata con énfasis la descomposición del *software* en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en piezas más

grandes de *software*. Este proceso de construcción de una pieza de *software* con componentes ya existentes, da origen al principio de reutilización del *software*, mediante el cual se promueve que los componentes sean implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro. Se debe entonces, para terminar de definir la arquitectura basada en componentes, saber que es un componente de *software*. Un componente de *software* se define típicamente como algo que puede ser utilizado como una caja negra, en donde se tiene de manera externa una especificación general, la cual es independiente de la especificación interna.

El interior del componente es una pieza de *software* que cumple con un conjunto de propiedades y que se encuentra conformada como un artefacto del cual se espera que sea reutilizable.

La relación interior-exterior es la que define el proceso de relación entre el interior y exterior del componente, a través de conceptos como especificación, implementación y encapsulación. De forma evidente se puede determinar que, el principal elemento de *software* dentro de un Arquitectura basada en componentes son precisamente los componentes de *software*. Existen 5 principios definidos por Clemens Szyperski and David Messerschmitt.

Los componentes interactúan por medio de invocaciones explícitas de procedimientos o funciones. Además, se exponen datos que son compartidos con su entorno y pueden registrarse a una clase de datos de interés. Existe un manejador de mensajes que coordina la comunicación entre componentes, invocando al componente cuando un mensaje que llega es para ese determinado componente.

Las arquitecturas basadas en eventos o componentes se vinculan históricamente con sistemas basados en actores, *daemons* y redes de conmutación de paquetes (publicación-suscripción). Los conectores de estos sistemas incluyen procedimientos de llamada tradicionales y vínculos entre anuncios de eventos e invocación de procedimientos. La idea dominante en la invocación implícita es que, en lugar de invocar un procedimiento en forma directa (como se haría en un estilo orientado a objetos) un componente puede anunciar mediante difusión uno o más eventos. Un componente de un sistema puede anunciar su interés en un evento determinado asociando un procedimiento con la manifestación de dicho evento. Los principios fundamentales cuando se diseña un componente es que estos deben ser:

- Reusable: Los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas. (Veryard, 2001)

- Sin contexto específico: Los componentes son diseñados para operar en diferentes ambientes y contextos. Información específica como el estado de los datos deben ser pasadas al componente en vez de incluirlos o permitir al componente acceder a ellos.
- Extensible: Un componente puede ser extendido desde un componente existente para crear un nuevo comportamiento. (Veryard, 2001)
- Encapsulado: Los componentes exponen interfaces que permiten al programa usar su funcionalidad. Sin revelar detalles internos, detalles del proceso o estado. (Veryard, 2001)
- Independiente: Los componentes están diseñados para tener una dependencia mínima de otros componentes. Por lo tanto, los componentes pueden ser instalados en el ambiente adecuado sin afectar otros componentes o sistemas. (Veryard, 2001)

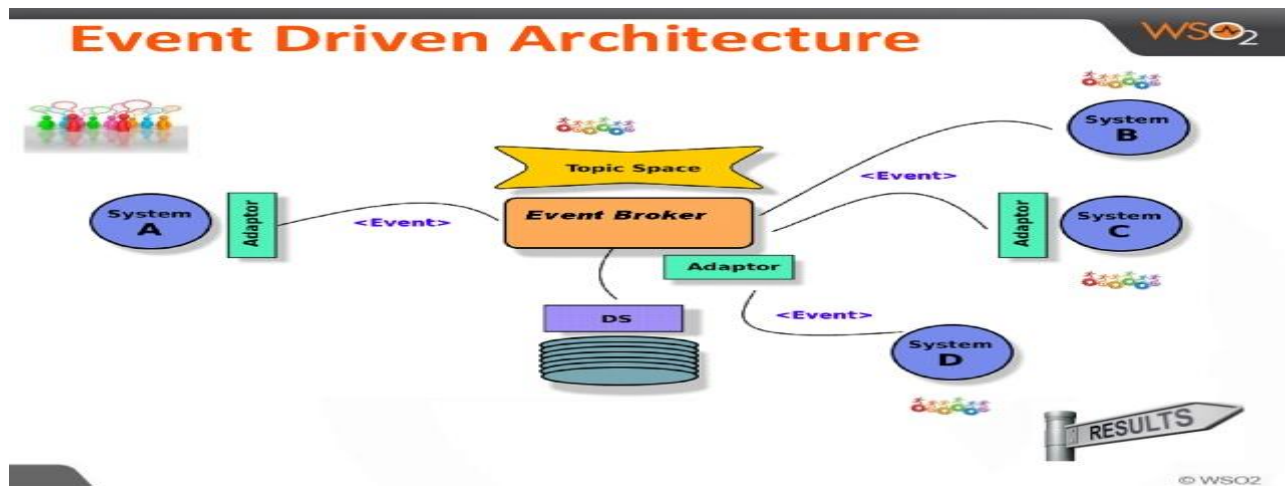


Figura 9: Arquitectura basada en eventos

## 2.8 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de *software*. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de *software* que están sujetos a contextos similares. Se debe tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de

diseño es una solución a un problema de diseño, para que una solución sea considerada un patrón debe poseer ciertas características, una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores, otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (Tedeschi,2012)

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas de *software*.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente. (Johnson y otros,2004)

### **Categorías de patrones**

Los patrones de diseño se dividen en categorías según la escala o nivel de abstracción:

Patrones de arquitectura: Aquellos que expresan un esquema organizativo estructural fundamental para sistemas de *software*.

Patrones de diseño: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de *software*.

Dialectos: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

A continuación, se muestra una imagen con la relación entre los dos tipos de familia de patrones de diseño, las más usadas:

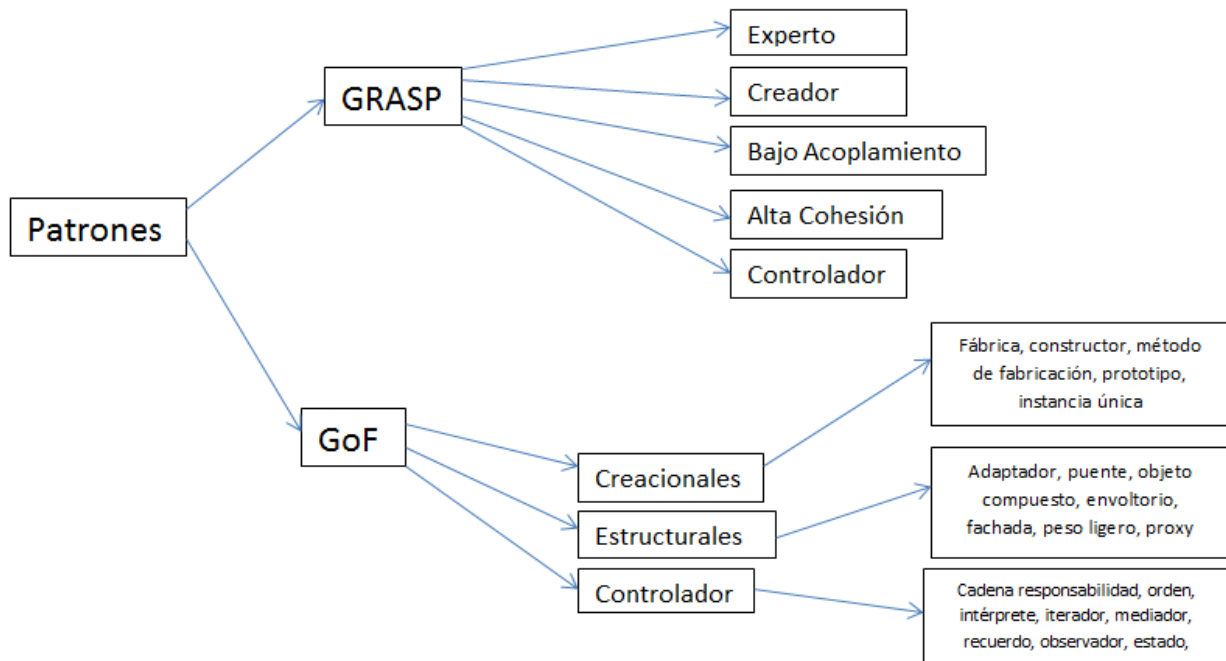


Figura 10: Relación de familias de patrones de diseño

### 2.8.1 Patrones GRASP

En términos generales, un patrón es un conjunto de información que proporciona respuesta a un conjunto de problemas similares, es decir, un patrón es una solución a un problema en un contexto, donde:

- Contexto son las situaciones recurrentes a las que es posible aplicar el patrón.
- Problema es el conjunto de metas y restricciones que se dan en ese contexto.
- Solución es el diseño a aplicar para conseguir las metas dentro de las restricciones.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. GRASP es un acrónimo que significa *General Responsibility Assignment software Patterns*. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el *software* orientado a objetos. (Larman, 2004)



## Experto

Asigna responsabilidades a la clase que tiene la información necesaria para llevar a cabo las operaciones en el sistema.

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con este no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. (Larman,2004)

```

class Todo:
    cero = 0

    lugar = os.path.expanduser('~')
    conectado = Conectado()
    def salir(self):...
    def menu(self):...

    # ----- Pre-Requisitos del sistema -----
    cpre = Pre_Requisitos()

    # ----- Servidor de Aplicaciones -----
    serverapps = Servidor_Aplicaciones()

    # ----- Servidor Web -----
    serverweb = Servidor_Web()

    # ----- Instalacion de archivos -----
    insta_ar = Instalar_Archivos()
    
```

Figura 11: Ejemplo Patrón Experto.

## Creador

El patrón creador utilizado en la solución presenta varias características como:

- Tiene la información necesaria para realizar la creación del objeto.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.
- Contiene o agrega la clase.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón

es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. En un diagrama de clases se registran las relaciones muy frecuentes entre las clases. El patrón Creador indica que la clase incluyente del contenedor o registro es idónea para asumir la responsabilidad de crear la directriz contenida o registrada. (Larman,2004)

En la siguiente figura se evidencia el uso de este patrón en la solución.

```
# ----- Pre-Requisitos del sistema -----
cpre = Pre_Requisitos()

# ----- Servidor de Aplicaciones -----
serverapps = Servidor_Aplicaciones()

# ----- Servidor Web -----
serverweb = Servidor_Web()

# ----- Instalacion de archivos -----
insta_ar = Instalar_Archivos()
```

Figura 12: Ejemplo Patrón Creador.

### Controlador

Asignar la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente una de estas dos opciones:

1. Representa el sistema global, dispositivo o subsistema (controlador de fachada).
2. Representa un escenario de caso de uso en el que tiene lugar el evento del sistema (controlador de caso de uso o de sesión).

En todos los casos, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Desde el punto de vista técnico, las responsabilidades del controlador podrían cumplirse en un objeto de interfaz, pero esto supone que el código del programa y la lógica relacionada con la realización de los procesos del dominio puro quedarían incrustados en los objetos interfaz o ventana. Un diseño de interfaz como controlador reduce la posibilidad de reutilizar la lógica de los procesos del dominio en aplicaciones futuras, por estar ligada a una interfaz determinada (por ejemplo, un objeto similar a una ventana) que rara vez puede utilizarse en otras aplicaciones. En cambio, el hecho de delegar a

un controlador la responsabilidad de la operación de un sistema entre las clases del dominio soporta la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras. (Larman,2004)

```
def menu(self):
    print(chr(27) + "[0;31m" + "*****" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "Opciones para la puesta en marcha de las operaciones" + chr(27))
    print(chr(27) + "[0;31m" + "*****" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "----- Leyenda de acciones -----" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "----- Selección opción: De 0 a 5 -----" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "----- Aceptar opción:s ---" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "----- Cancelar opción:n ---" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "1.-Instalación Completa" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "2.-Instalación y configuración de los pre-requisitos del sistema" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "3.-Instalación y configuración del servidor de Aplicaciones" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "4.-Instalación y configuración del servidor Web" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "5.-Instalar programas adicionales" + chr(27))
    print(chr(27) + "[0;31m" + "*" + chr(27) + "[0m" + "0.-Salir" + chr(27))
    print(chr(27) + "[0;31m" + "*****" + chr(27))

    seleccion = raw_input('Escribe una opción:\n')

    if (seleccion == '1'):
        self.cpre.prerequisitos()
        self.serverapps.servidor_aplicaciones()
        self.serverweb.servidor_web()
        self.insta_ar.instalararchivos()
        self.menu()
    elif (seleccion == '2'):
        self.cpre.prerequisitos()
        self.menu()
```

Figura 13: Ejemplo Patrón Controlador.

## 2.9 Modelo de diseño

Representa todas las clases del diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos, constituyendo la entrada principal a las actividades de la fase de implementación.

A continuación, se describe de manera general el significado de los principales elementos presentes en el diagrama de clases.

**Clase Principal:** muestra la información con la que el usuario debe interactuar.

**Clase Controladora:** contiene las clases que realizan el tratamiento de eventos.

**Clase Pre-Requisitos:** realiza las operaciones relacionadas con las pre-condiciones que debe tener un ordenador.

**Clase Instalar\_Archivos:** contiene la clase que realiza la instalación de programas adicionales.

**Clase Servidor de Bases datos:** contiene todas las clases y los métodos relacionados con la configuración e instalación del servidor de Bases datos.

**Clase Servidor Aplicaciones:** contiene todas las clases y los métodos relacionados con la configuración e instalación del servidor aplicaciones.

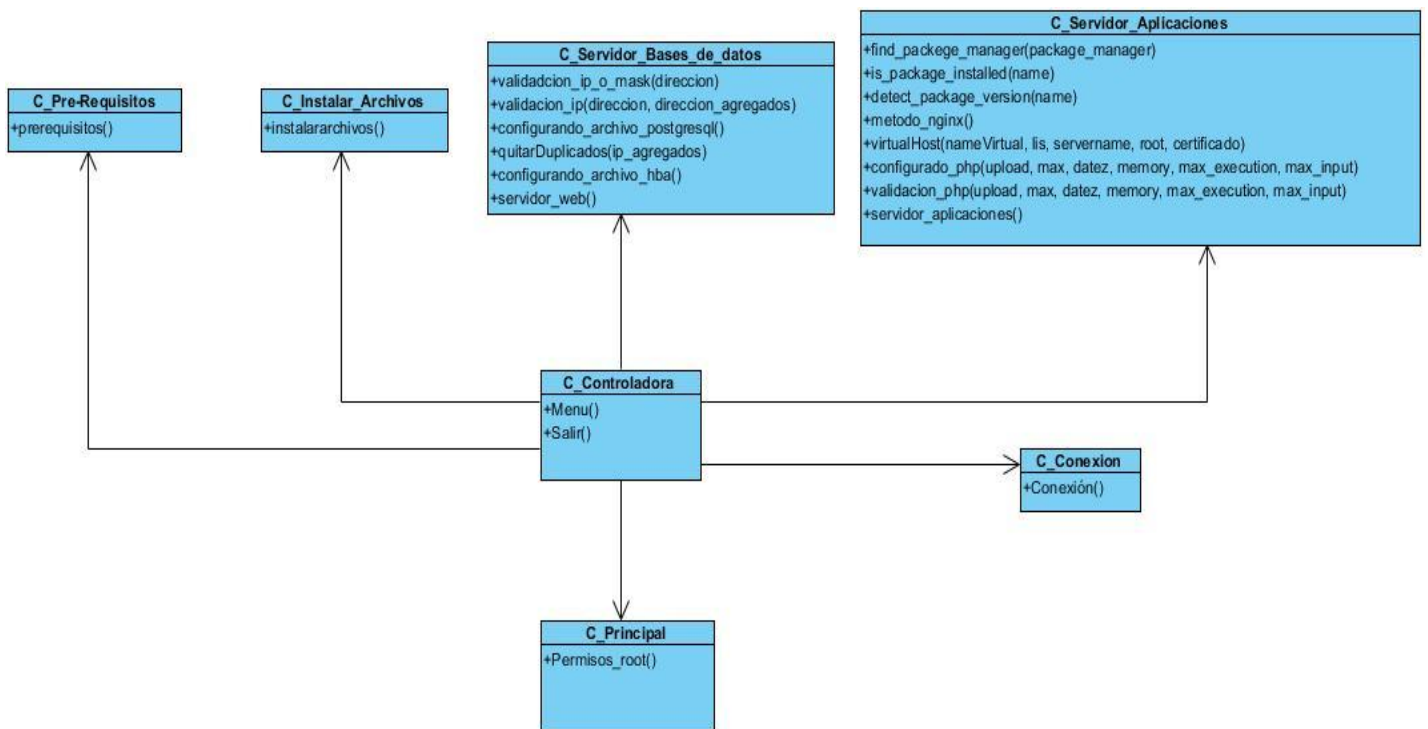


Figura 14: Diagrama de clases.

## 2.10 Conclusiones del capítulo

En este capítulo se realizó la descripción del modelo de dominio; así como la planificación de las diferentes etapas que corresponden a desarrollar según la metodología utilizada, para de esta manera obtener una visión más precisa de la solución. También se presentó la propuesta de solución, explicó el flujo de instalación de la herramienta, mencionándose además todas las funcionalidades que esta brindará, los requisitos de funcionales y no funcionales; y la especificación de cada uno de estos escenarios para lograr una perspectiva final del sistema. Las descripciones de las HU posibilitaron conocer el funcionamiento general del sistema, estableciendo un conjunto de pasos lógicos, flujos de información y respuestas del sistema al usuario. Se describieron los pre-requisitos de instalación, siendo parte fundamental de la solución; ya que permitió conocer todas las condiciones que debe cumplir el ordenador antes de realizar el proceso de despliegue. A través de la realización de este capítulo se ha profundizado en la propuesta de solución, con el propósito de conocer con exactitud todas las características que debe tener la Herramienta para el despliegue de *software* bajo la arquitectura Xalix y llevar a cabo su implementación.

## Capítulo III: Validación de la propuesta de solución

### 3.1 Introducción

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 de la herramienta para el despliegue de *software* bajo la arquitectura Xalix.

### 3.2 Diagrama de componente

El diagrama de componentes muestra la organización y las dependencias entre componentes de código fuente, componentes del código binario, y componentes ejecutables. Según (Rumbaugh, y otros, 2000) los componentes bien diseñados no dependen directamente de otros componentes sino de las interfaces que ofrecen los componentes. Estos diagramas pueden incluir paquetes que permiten organizar la construcción del sistema en subsistemas y que recogen aspectos prácticos relacionados con la secuencia de compilación entre componentes, la agrupación de elementos en librerías, etc. (Cillero, 2016)

A continuación, se muestra el diagrama de componentes correspondiente a la Herramienta para el despliegue de *software* bajo la Arquitectura Xalix.

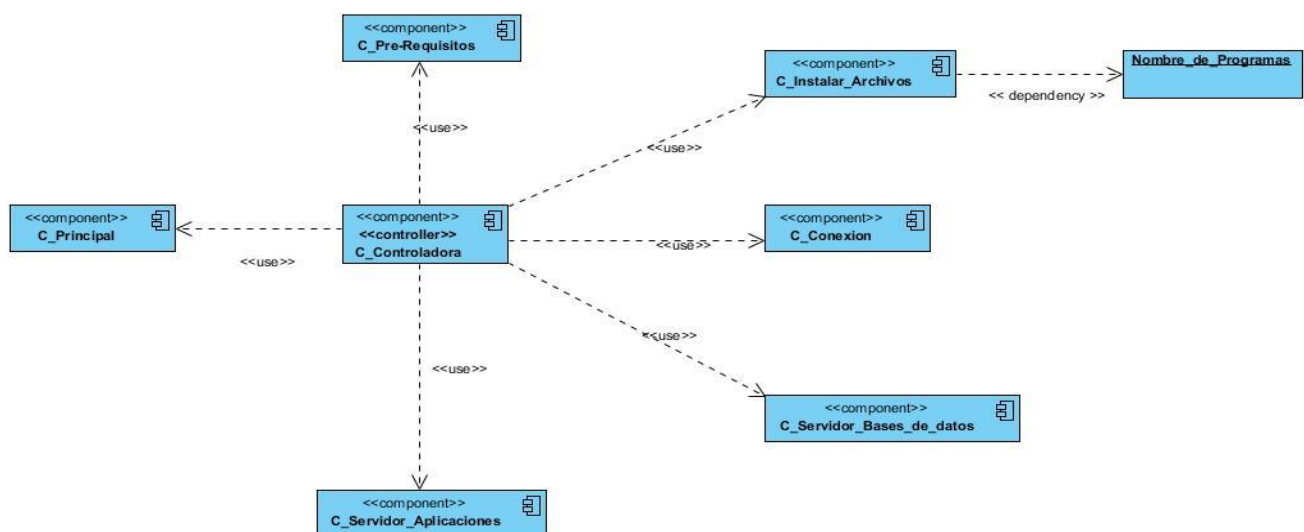


Figura 15: Diagrama de componentes.

### 3.3 Estándar de codificación

Los imports deben colocarse en distintas líneas y al comienzo del código, por ejemplo:

```
import os

import sys
```

Cuando se importa una clase de un módulo, se realiza de la siguiente manera:

```
from myclass import MyClass
from foo.bar.yourclass import YourClass
```

**Excepciones:** Cuando se capturan excepciones, se especifica cuanto sea posible.

Por ejemplo:

```
try:
    import platform_specific_module
except ImportError:
    platform_specific_module = None
```

**Indentación:** La indentación significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores. Se usa cuatro espacios por cada nivel de indentación.

```
if (int(arreglo1[0]) > int (arreglo [0])):
    return False //Indentación y comienzo del bloque
```

**Declaraciones de clases e interfaces:** al codificar clases de Python, se siguen las siguientes reglas de formato:

- Se coloca la palabra clave “class”.
- Después de la palabra “class” se coloca en nombre de la clase.
- Por último, se colocan seguido del nombre de la clase “:”
- Dentro de las clases existen métodos pasándole parámetros y vacíos, pero siempre deben tener la palabra “self”. Esta palabra sirve para hacer referencias a los atributos de clase y si es necesario en cada método de la misma.

```
class Instalar_Archivos:
    cero = 0

    def Ejemplo(self, n, l):
        name = n
        lista = l

    def MetodoVacio(self):
```

**Sentencias if, if-else, if else-if else:** la clase de sentencias if-else debe tener la siguiente forma:

```
if (condición):
    sentencias

else:
    sentencias

if (condición):
    sentencias

elif (condición):
    sentencias
```

**Sentencias for:** una sentencia for debe tener la siguiente forma:

```
for i in archivos:
    sentencia
```

**Nota:** Todas las clases comienzan con la primera letra mayúscula; si son palabras compuestas se separan con guión bajo “\_”. Todos los métodos y variables se escriben en minúscula.

### 3.4 Pruebas de *software*

Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de la implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo. Las pruebas de *software* es un elemento importante para la garantía del correcto funcionamiento de una herramienta. Entre sus objetivos están:

- Detectar defectos del *software*.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el *software* al cliente.
- Diseñar casos de prueba que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. (Sommerville, 2005)

## **Tipos de pruebas según AUP en su variación UCI**

### **Pruebas internas**

En esta disciplina se verifica el resultado de implementación la probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas.

### **Pruebas Aceptación**

Las pruebas de aceptación verifican que el sistema funcione correctamente de acuerdo con las especificaciones. Se realizan de acuerdo con protocolos específicos del producto en nuestra fábrica o en terreno. Al participar en un proceso de prueba de aceptación, puede obtener un conocimiento más profundo acerca de cómo funciona el equipo. (Pressman, 2010). Estas pruebas se dividen en dos partes como:

- Alfa: se lleva a cabo por el cliente, en el lugar de desarrollo. Se usa el *software* de forma natural con el desarrollador como observador del usuario y registrando los errores y problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado. (Valdez Huaraca, y otros, 2013)
- Beta: se realizan con posterioridad a las pruebas alfa, y se desarrollan en el entorno del cliente. En este caso, el cliente se queda a solas con el producto y trata de encontrarle fallos de los que informa al desarrollador. (Valdez Huaraca, y otros, 2013)

### **Pruebas Liberación**

Son las pruebas diseñadas y ejecutadas por una entidad certificadora externa, a todos los proyectos antes de ser entregado al cliente para su aceptación.

#### **3.4.1 Métodos de pruebas**

Un método de prueba es un procedimiento definitivo que produce un resultado. Se puede probar cualquier producto de ingeniería en dos formas: conociendo la función específica para la cual fue diseñado el mismo y conociendo el funcionamiento del producto. Al primer enfoque se le denomina prueba de caja negra y al segundo, prueba de caja blanca.



**Pruebas de caja blanca:** Permite desarrollar pruebas de forma que se asegure que la operación interna se ajusta a las especificaciones, y que todos los componentes internos se han probado de forma adecuada. (Pressman, 2010)

**Pruebas de caja negra:** 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. (Almerña, 2015)

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. (Almerña, 2015)

Para realizar la validación de la propuesta solución se empleó el método de caja negra utilizando clases o partición de equivalencia, con el objetivo de comprobar las respuestas de las funcionalidades.

### 3.4.2 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. (El Proceso Unificado de Desarrollo de *Software*, 2000)

A continuación, se muestran los casos de prueba realizados al sistema.

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
<b>EC 1.1</b> Seleccionar la opción "s"	Permite optar por la opción de instalar los programas adicionales en un archivo.	N/A	N/A	N/A	Muestra al usuario la información de que debe adicionar la dirección donde se encuentra el archivo.txt	Después de seleccionar la opción si, aparecerá un mensaje en pantalla pidiendo al usuario la dirección donde se encuentra el archivo donde estarán los programas que desea instalar. La aplicación comprobará si la dirección existe, en caso contrario pedirá al usuario que entre una dirección real.
<b>EC 1.2</b> Seleccionar la opción "n"	Permite optar por la opción de no instalar los programas adicionales en un archivo.	N/A	N/A	N/A	Muestra al usuario la información de que ha cancelado la opción de instalar los programas adicionales.	Al seleccionar la opción de "no", la aplicación mostrará un mensaje de que ha cancelado la instalación de los programas adicionales. Después de este mensaje, aparecerá en pantalla la opción del menú.
<b>EC 1.3</b> Seleccionar la opción de instalar los programas en pantalla "s"	El usuario deberá seleccionar la opción "sí" para pasar a instalar los programas	N/A	N/A	N/A	La aplicación mostrará todos los programas adicionales por usted en el archivo de configuración.	La aplicación luego de mostrar todos los programas en pantalla, pasará a instalarlos en el sistema. Después de realizar esta operación si no hubo problemas, mostrará un mensaje diciendo que la instalación fue un éxito.

## Capítulo III: Validación de la propuesta de solución

<b>EC 1.4</b> Seleccionar la opción de instalar los programas en pantalla "n"	Permite optar por la opción de no instalar los programas adicionales en un archivo.	N/A	N/A	N/A	Muestra al usuario la información de que ha cancelado la opción de instalar los programas adicionales.	Al seleccionar la opción de "no", la aplicación mostrará un mensaje de que ha cancelado la instalación de los programas adicionales. Después de este mensaje, aparecerá en pantalla la opción del menú.
--	---	-----	-----	-----	--	---

Tabla 5: Instalar programas adicionales.

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
<b>EC 1.1</b> Seleccionar la opción "s"	Permite optar por la opción de instalar los programas del servidor web	N/A	N/A	N/A	La aplicación mostrará la instalación de los programas asociados a la aplicación web y también instalará y configurará algunas dependencias de Python que se necesitan para desarrollar este método.	Instalar y configurar los programas para el funcionamiento del servidor web.
<b>EC 1.2</b> Seleccionar la opción "s" para configurar el servidor web	La aplicación pedirá al usuario si desea configurar el servidor web.	N/A	N/A	N/A	Luego de seleccionar la opción de "si" la aplicación accederá a pedir al usuario valores necesarios para configurar el servidor web.	Comenzar a configurar el servidor web.
<b>EC 1.3</b> Configurar la contraseña del servidor de bases de	La aplicación brindará la posibilidad de	N/A	N/A	N/A	Luego de acceder a configurar el servidor de bases de datos, la	Configurar la contraseña del gestor de bases de datos PostgreSQL.

### Capítulo III: Validación de la propuesta de solución

datos PostgreSQL	configurar el servidor de bases de datos				aplicación mostrará un mensaje guiando al usuario para configurar la contraseña del gestor de bases de datos. Para configurar la contraseña el usuario deberá escribir el comando <code>\password</code> . Después de escribir este comando, el usuario deberá redactar la contraseña que será utilizada en el gestor. Para salir de la configuración del PostgreSQL, el usuario deberá escribir el comando <code>/q</code> .	
<b>EC 1.4</b> Configurar las direcciones ip.	La aplicación pedirá al usuario que escriba las direcciones ip que tendrán acceso al gestor de bases de datos.	N/A	N/A	N/A	La aplicación pedirá al usuario que escriba las direcciones ip que tendrán acceso al gestor de bases de datos. En caso de que el usuario escriba una dirección ip incorrecta o que ya esté en los archivos de configuración, la aplicación mostrará un mensaje describiendo que la dirección ip no es válida.	Escribir en los archivos de configuración del PostgreSQL las direcciones ip que pueden tener acceso al gestor de bases de datos.

Tabla 6: Instalación del servidor de aplicación.

### 3.4.3 Resultados obtenidos del método de caja negra

Se realizaron cuatro iteraciones a cada uno de los escenarios del sistema. En la primera iteración se encontraron 11 no conformidades, de estas 7 fueron resueltas y 4 quedaron pendientes. En la primera iteración los resultados obtenidos fueron evaluados en un rango de: Alta (6), Media (3), Baja (2). En la segunda iteración se encontraron 5 no conformidades, de ellas 3 fueron resueltas, quedando pendiente 2; estos resultados fueron evaluados como: Alta (2), Media (1), Baja (2). En la tercera iteración se encontraron 2 no conformidades, de ellas resueltas 2, quedando el sistema libre de errores; estos resultados fueron evaluados como: Alta (1), Media (1), Baja (0). A continuación, se muestra una tabla con los resultados obtenidos:

- Alta: errores en el código o errores de funcionalidad.
- Media: errores de ortografía o de validación.
- Baja: errores de interfaz.

No conformidades	1era Iteración	2era Iteración	3era Iteración	4ta Iteración
Alta	6	2	1	0
Media	3	1	1	0
Baja	2	2	0	0

Tabla 7: Rango de evaluación de las pruebas caja negra.

Para una mejor comprensión de los resultados, se muestra en el siguiente gráfico la magnitud de las no conformidades encontradas en cada una de las iteraciones realizadas.

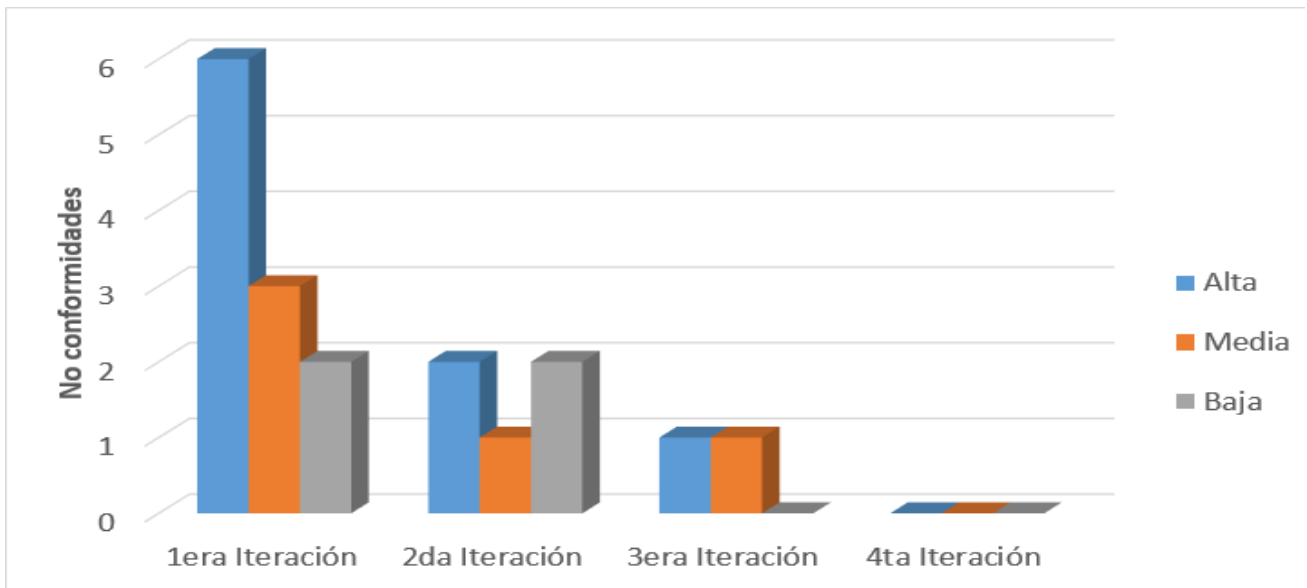


Figura 16: Gráfica con los resultados de la complejidad de las pruebas caja negra.

### 3.4.4 Resultados obtenidos de las pruebas de aceptación de tipo alfa.

Las pruebas de aceptación de tipo alfa fueron realizadas a todas las funcionalidades que conforman la propuesta de solución. Para ello se efectuaron tres iteraciones. En la primera de ellas se detectaron 4 no conformidades; de estas 0 se clasificaron de complejidad alta, 2 de complejidad media y 2 de complejidad baja. En la segunda y tercera iteración el número de no conformidades detectadas se redujo considerablemente. A continuación, se expone un gráfico que muestra la relación entre las no conformidades detectadas de complejidad alta, media y baja.

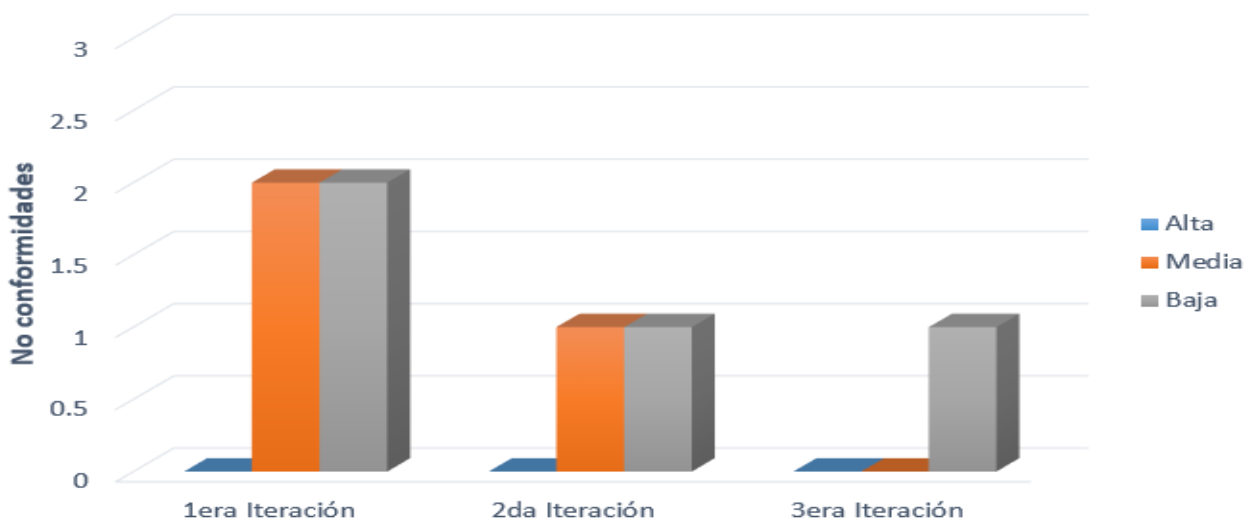


Figura 17: Gráfica con los resultados de la complejidad de las pruebas de aceptación.

Dichos resultados son satisfactorios ya que contribuyen al funcionamiento óptimo de la herramienta; con el objetivo de que a medida que se detectan los problemas se irán solucionando, obteniendo una aplicación cumpla con los requisitos que se plantean para la misma.

### **3.5 Conclusiones del capítulo**

El presente capítulo abordó a cerca de la implementación del sistema y la descripción del flujo de pruebas utilizadas por herramienta para el despliegue de *software* bajo la arquitectura Xalix de las funcionalidades descritas en las historias de usuarios. Se describieron y realizaron un conjunto de casos de pruebas de Caja Negra, con el fin de comprobar cada uno de los métodos del sistema. A partir de las pruebas se comprobó el código fuente de la aplicación para asegurar que cada una de las partes que la integran funcione correctamente por separado.

Los casos de prueba fueron utilizados con el objetivo de asegurar el correcto funcionamiento de los requerimientos funcionales, asegurando así la calidad del *software*. De manera general las pruebas fueron satisfactorias, contribuyendo a obtener un producto de mayor calidad.

## Conclusiones generales

La presente investigación tiene un papel fundamental en el análisis, diseño e implementación de la Herramienta para el despliegue de *software* bajo la arquitectura Xalix de la Universidad de las Ciencias Informáticas, permitiendo concluir que:

- El análisis de los principales conceptos asociados al despliegue de *software*, permitió obtener la base teórica necesaria con el objetivo de desarrollar la Herramienta para el despliegue de *software* bajo la arquitectura Xalix.
- La realización de un estudio a las soluciones similares existentes, permitió identificar las características principales para darle solución al problema planteado al inicio de la investigación.
- La utilización de la metodología AUP-UCI en su escenario cuatro y el empleo del lenguaje de programación Python, contribuyeron al éxito en el desarrollo de la Herramienta para el despliegue de *software* bajo la arquitectura Xalix, obteniendo la documentación y los artefactos necesarios.
- El análisis y diseño de la herramienta para el despliegue de *software*, permitió comprender las principales funcionalidades del sistema.
- El desarrollo de la herramienta para el despliegue de *software*, automatiza en gran parte el proceso de despliegue de los productos del Centro FORTES. Su utilización aporta ventajas como la disminución del tiempo destinado a realizar este proceso, así como la eliminación de errores que se pueden cometer durante la instalación.
- Se ejecutaron los casos de pruebas para cada historia de usuario, con el objetivo de comprobar el correcto funcionamiento de la Herramienta para el despliegue de *software*. 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.

El desarrollo propuesto en el presente trabajo de diploma, mejora gran parte del proceso de instalación y configuración del entorno de trabajo para el despliegue de los principales programas bajo la arquitectura Xalix. Su utilización aporta ventajas como la disminución del tiempo destinado a la instalación y configuración de los programas, así como la disminución de los errores que se puedan cometer durante este proceso. Además de contar con una guía para realizar el proceso satisfactoriamente.



## Recomendaciones

Teniendo en cuenta que se desea brindar un mejor servicio y en aras de agregar nuevas funcionalidades al proceso de instalación se ofrece la siguiente recomendación:

- Realizar modificaciones a la solución para que funcione en otras distribuciones de GNU/Linux como CentOS.

## Referencias bibliográficas

- alegsa. 2014.** Definición de script. [En línea] junio 6, 2014. <http://www.alegsa.com.ar/Dic/script.php>.
- Almerña. 2015.** Técnicas de prueba. [En línea] diciembre 7, 2015. <http://indalog.ual.es/mtorres/LP/Prueba.pdf>.
- Asensio, Rafael Menéndez-Barzanallana. 2015.** JAVASCRIPT. [En línea] 2015. <http://www.um.es/docencia/barzana/DAWEB/Lenguaje-de-programacion-JavaScript-1.pdf>.
- Cano, Jose Antonio. 2014.** Tipos de scripts en cuanto a su ejecución. [En línea] junio 27, 2014. <https://webcache.googleusercontent.com/search?q=cache:4RJlimgZyOAJ:https://canodelacuadra.wordpress.com/temas/tema-2-1-el-lenguaje-de-guion/tema-2-1-4-tipos-de-scripts-en-cuanto-a-su-ejecucion/+&cd=5&hl=es-419&ct=clnk&gl=cu>.
- Castellar, Puig. 2015.** Guiones del intérprete de comandos (Shell scripts o archivos por lotes). [En línea] 5 20, 2015. <http://elpuig.xeill.net/Members/vcarceler/c1/didactica/apuntes/ud3/na6>.
- Castillo, Celia Clemente. 2008.** Python. [En línea] julio 27, 2008. <http://www.it.uc3m.es/spickin/docencia/comsoft/presentations/spanish/doc/Python.pdf>.
- Díaz, ALEXANDER. 2010.** Desplegando *software* de manera segura. [En línea] septiembre 27, 2010. <https://highscalability.wordpress.com/2010/09/27/desplegando-software-de-manera-segura/#more-648>.
- El Proceso Unificado de Desarrollo de Software.* **JACOBSON. 2000.** ISBN 84-7829-036-2, s.l. : Wesley, 2000.
- Faculta de Informática, Universidad Politécnica de Madrid. 2015.** Programación con lenguajes de guión en páginas web. [En línea] 10 15, 2015. <http://www.launiversidadlaboral.com/verCurso.php?id=4217>.
- Graterol, Yohan. 2014.** PyCharm: El mejor IDE para tus proyectos en Python. [En línea] septiembre 16, 2014. [http://www.cristalab.com/tutoriales/pycharm-el-mejor-ide-para-tus-proyectos-en-python-c114084/...](http://www.cristalab.com/tutoriales/pycharm-el-mejor-ide-para-tus-proyectos-en-python-c114084/)
- Informática, Instituto Tecnológico de. 2014.** Testeo de funcionalidad. [En línea] mayo 2014. [Citado el: abril 20, 2017.] <http://www.iti.es/servicios/servicio/resource/7234/index.html>.
- Informática, Revista. 2015.** LENGUAJE DE PROGRAMACIÓN JAVASCRIPT. [En línea] 2015. <http://www.larevistainformatica.com/JavaScript.htm>.

**infostudio. 2011.** Principales Características del Lenguaje Python. [En línea] agosto 31, 2011. [http://www.cuatorrios.org/index.php?option=com\\_content&view=article&id=161:principales-caracteristicas-del-lenguaje-python&catid=39:blogsfeeds](http://www.cuatorrios.org/index.php?option=com_content&view=article&id=161:principales-caracteristicas-del-lenguaje-python&catid=39:blogsfeeds).

**Jiménez, Xavier. 2014.** Atomic Development for Modern WebApps. [En línea] octubre 26, 2014. <https://github.com/tapquo/atoms>.

**Leandro. 2009.** DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. [En línea] 2009. <http://www.alegsa.com.ar/Dic/uml.php>.

**Manso, Yerandy. 2016.** *Que es la arquitectura Xalix*. Habana, Cuba, diciembre 2016.

**Matz. 2003.** Ruby EL MEJOR AMIGO DE UN DESARROLLADOR. [En línea] mayo 12, 2003. <https://www.ruby-lang.org/es/about/>.

*Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. **Letelier, Patricio. 2006.** noviembre 15, 2006.

**Peláez, Juan Carlos. 2009.** Arquitectura basada en Componentes. [En línea] abril 8, 2009. <https://geeks.ms/jkpelaez/2009/04/18/arquitectura-basada-en-componentes/>.

**peralta, Xavier. 2014.** Ansible-automatización-de-tareas-y-despliegues-de-forma-simple. [En línea] abril 4, 2014. <http://www.cloudadmins.org/2014/04/ansible-automatizacion-de-tareas-y-despliegues-de-forma-simple/>.

**PHP.net. 2012.** 6 Buenos Motivos Para Trabajar Con PHP. [En línea] diciembre 10, 2012. <http://php.net/manual/es/intro-whatcando.php>.

**Picajoso. 2010.** ¿Cuáles son los lenguajes de programación de script más interesantes para encontrar trabajo? [En línea] agosto 9, 2010. <http://www.muylinux.com/2010/08/19/%C2%BFcuales-son-los-lenguajes-de-programacion-de-script-mas-interesantes-para-encontrar-trabajo>.

**Pressman, Roger S. 2005.** Ingeniería de *Software*. Un enfoque práctico. [aut. libro] PRESSMAN. s.l.: 6ta ed, 2005.

**Rondón, Yoandri Quintana. 2011.** *Visual Paradigm como herramienta CASE*. 2011.

**Sales. 2016.** InstallBuilder 17. [En línea] mayo 24, 2016. <https://installbuilder.bitrock.com/>.

**Silver, Félix. 2010.** Tipos de instalaciones de *software* (Programas). [En línea] febrero 11, 2010. <https://silverfenix7.wordpress.com/2010/02/11/>.

**Studio, SQL Server Management. 2016.** Usar SQL Server Management Studio. [En línea] mayo 10, 2016. <https://msdn.microsoft.com/es-es/library/ms174173.aspx>.

**Valdéz, José Luis Cendejas. 2014.** IMPLEMENTACIÓN DEL MODELO INTEGRAL COLABORATIVO (MDSIC) COMO FUENTE DE INNOVACIÓN PARA EL DESARROLLO ÁGIL DE SOFTWARE EN LAS EMPRESAS DE LA ZONA CENTRO. [En línea] diciembre 9, 2014. <http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm>.

**Veryard, Richard. 2001.** component-based development. [En línea] octubre 25, 2001. <http://www.users.globalnet.co.uk/~rxv/CBDmain/cbdfaq.htm#Component-Based%20Development>.

**Williams, Elise. 2016.** Best 5 *Software* Deployment Tools for Enterprise. [En línea] Business Tips, junio 17, 2016. <https://pdf.wondershare.com/pdf-business-tips/software-deployment-tools>.

*Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información.* **Lianet Cabrera González, Enrique Roberto Pompa Torres. 2012.** La Habana : ISSN | RNPS, 2012.

*Design Patterns: Elements of Reusable Object-Oriented Software.* **Ralph Johnson, Erich Gamma, John Vlissides, Richard Helm. 2004.** s.l. : Addison Wesley, 2004.

**Larman, Craig.** UML y Patrones Introducción al análisis y diseño orientado a objetos. [En línea] [Citado el: 3 23, 2017.] <http://fmonje.com/UTN/ADES%20-%20208/UML%20y%20Patrones%20%202da%20Edicion.pdf>.

**Tedeschi, Nicolás.** ¿Qué es un Patrón de Diseño? [En línea] Microsoft. [Citado el: 11 15, 2016.] <http://msdn.microsoft.com/es-es/library/bb972240.aspx>.

*UML gota a gota.* **MARTIN FOWLER, KENDALL SCOTT. 2003.** Mexico : Editorial Mexicana, 2003.

*Use Case Modeling.* **Bittner, Kurt. 2003.** Boston : Pearson Education,INC, 2003.

Visual Paradigm for UML. [En línea] [Citado el: 2 12, 2017.] <http://www.visual-paradigm.com/product/vpuml>.

**Labrador, Ramón M. Gómez. 2005.** PROGRAMACIÓN AVANZADA EN SHELL. [En línea] 4 2005. [Citado el: 1 20, 2017.] <http://www.informatica.us.es/~ramon/articulos/Programacion-BASH.pdf>.

**Oterino, Ana M. del Carmen García. 2015.** ¿Qué es Docker? ¿Para qué se utiliza? Explicado de forma sencilla. [En línea] 7 24, 2015. [Citado el: 3 5, 2017.] <http://www.javiergarzas.com/2015/07/que-es-docker-sencillo.html>.

Get Started, Part 1: Orientation and Setup. [En línea] [Citado el: 3 5, 2017.]  
<https://docs.docker.com/get-started/>.

**González, Israel Chava. 2016.** Rational rose. [En línea] 8 27, 2016.  
<https://es.slideshare.net/IsraelChavaGonzales/rational-rose-65411778>.

**Pressman, Roger S. 2010.** *Software Engineering and practitioners approach. Séptima.* New York : Hair Education, 2010.

**Sommerville, Ian. 2005.** *Ingeniería del software Séptima edición.* Madrid : s.n., 2005.

**Cillero, Manuel. 2016.** Diagrama de Componentes. [En línea] 2016. [Citado el: 4 6, 2017.]  
<https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/>.

**Rumbaugh, Jacobson. 2000.** El Lenguaje Unificado del Modelado. Manual de Referencia. UML. [En línea] 2000. [Citado el: 4 3, 2017.]  
<https://ingenieriasoftware2011.files.wordpress.com/2011/07/el-lenguaje-unificado-de-modelado-manual-de-referencia.pdf>. ISBN: 84-7829-037-0.

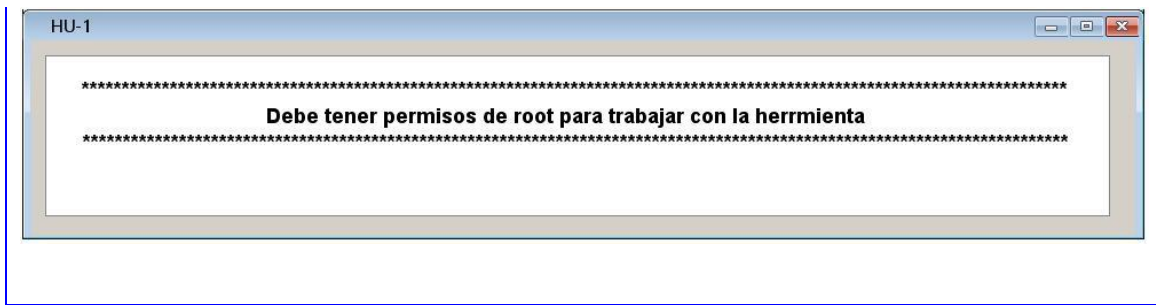
**Valdez, Huaraca. 2013.** Pruebas de sistema y Pruebas de aceptación. [En línea] 4 2, 2013. [Citado el: 5 15, 2017.] <https://es.slideshare.net/abnergerardo/pruebas-de-sistemas-y-aceptacion-23663195>.

**Álvarez, Jorge Cortés. 2015.** [En línea] 2015. [Citado el: 5 3, 2017.]  
<http://es.slideshare.net/cortesalvarez/metodologa-rup>.

## 1 Anexos

### Anexos I: Historias de usuarios

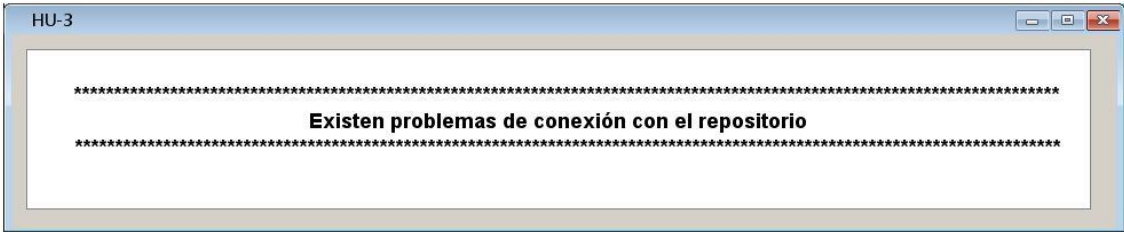
<b>Número:</b> HU_1	<b>Nombre del requisito:</b> Verificar permisos de root
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 1era
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si el usuario no coloca la contraseña de superusuario o administrador, la aplicación no accederá a comenzar con el proceso de instalación.	<b>Tiempo Real:</b> 3 días
<p><b>Descripción:</b></p> <p><b>1-Objetivo:</b> Permitir que la aplicación realice todos los cambios con la contraseña de superusuario o administrador.</p> <p><b>2-Acciones para lograr el objetivo (precondiciones y datos):</b> Colocar la contraseña de superusuario o administrador.</p> <p><b>3-Flujo de la acción a realizar:</b> Antes de escribir el nombre de la aplicación el usuario debe colocar la contraseña de superusuario o administrador para que tenga acceso a realizar las diferentes acciones que provee la solución. En caso de no realizar este procedimiento la aplicación mostrará un mensaje diciendo que no tiene permisos de administrador para realizar todos los cambios en la solución.</p>	
<b>Observaciones:</b> N/A	
<b>Prototipo de interfaz:</b>	



HU-1\_ Verificar permisos de root

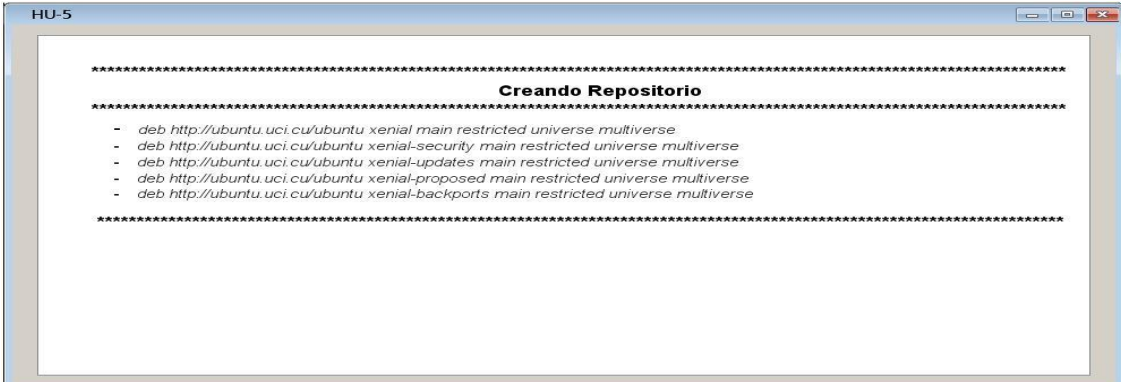
<b>Número:</b> HU_2	<b>Nombre del requisito:</b> Verificar el Sistema Operativo.
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 2da
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si la aplicación no comprueba el SO donde se está desplegando, no tendrá acceso posteriormente a configurar los repositorios.	<b>Tiempo Real:</b> 3 días
<b>Descripción:</b> <b>1-Objetivo:</b> Verificar el SO del hardware <b>2- Flujo de la acción a realizar:</b> La aplicación accederá a verificar el SO, con el objetivo de configurar este sistema con los repositorios adecuados.	
<b>Observaciones:</b> N/A	

HU-2\_ Verificar el Sistema Operativo

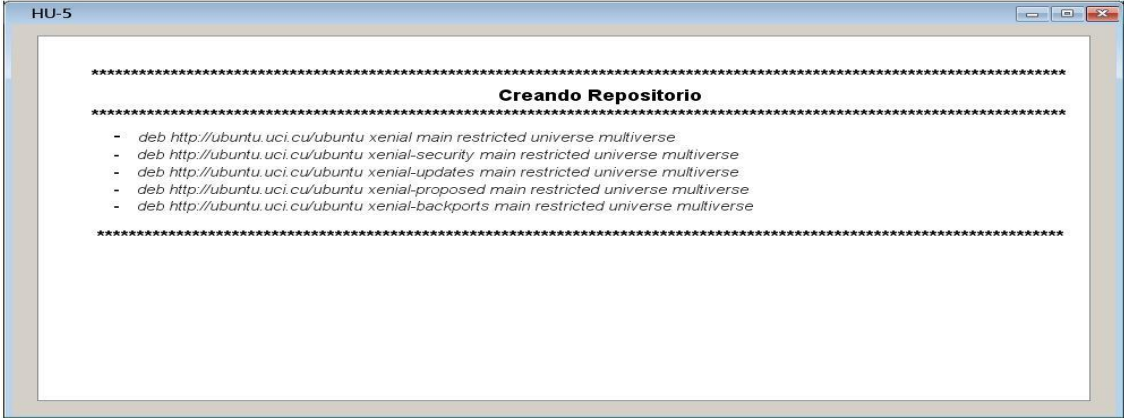
<b>Número:</b> HU_3	<b>Nombre del requisito:</b> Verificar la conectividad con el repositorio.
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 3era
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si la aplicación no verifica la conexión con los repositorios, no podrá a acceder a instalar los programas necesario para el despliegue de los <i>software</i> .	<b>Tiempo Real:</b> 3 días
<p><b>Descripción:</b></p> <p><b>1-Objetivo:</b> Permitir comprobar la conexión con los repositorios</p> <p><b>2- Flujo de la acción a realizar:</b> La aplicación después hacer sido iniciada, comprueba que exista conexión con el repositorio. En caso de no hacer conexión, la aplicación mostrará un mensaje diciendo que existe problemas de conectividad y no podrá acceder a menú.</p>	
<b>Observaciones:</b> N/A	
<b>Prototipo de interfaz:</b>	
 <p>The screenshot shows a window titled 'HU-3' with a standard Windows-style title bar. The main content area of the window displays a message centered between two lines of asterisks: 'Existen problemas de conexión con el repositorio'.</p>	

HU-3\_ Verificar la conectividad con el repositorio



<b>Número:</b> HU_4	<b>Nombre del requisito:</b> Mostrar repositorios
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 4ta
<b>Prioridad:</b> Media	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si el repositorio no está configurado, la aplicación no podrá ejecutar la instalación correctamente.	<b>Tiempo Real:</b> 3 días
<p><b>Descripción:</b></p> <p><b>1- Objetivo:</b> Permitir mostrar el repositorio adecuados en los archivos del sistema.</p> <p><b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> - Debe seleccionar esta opción para poder proceder a la creación de los repositorios.</p> <p><b>3- Flujo de la acción a realizar:</b> Si los repositorios no están creados, muestra en pantalla los repositorio dependiendo del sistema operativo.</p>	
<b>Observaciones:</b> N/A	
<p><b>Prototipo de interfaz:</b></p>  <pre> HU-5 *****                           Creando Repositorio ***** - deb http://ubuntu.uci.cu/ubuntu xenial main restricted universe multiverse - deb http://ubuntu.uci.cu/ubuntu xenial-security main restricted universe multiverse - deb http://ubuntu.uci.cu/ubuntu xenial-updates main restricted universe multiverse - deb http://ubuntu.uci.cu/ubuntu xenial-proposed main restricted universe multiverse - deb http://ubuntu.uci.cu/ubuntu xenial-backports main restricted universe multiverse ***** </pre>	

HU-4\_Mostrar repositorios

<b>Número:</b> HU_5	<b>Nombre del requisito:</b> Crear los repositorios
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 5ta
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si el repositorio no está configurado, la aplicación no podrá ejecutar la instalación correctamente.	<b>Tiempo Real:</b> 3 días
<b>Descripción:</b>  <b>1- Objetivo:</b> Permitir crear los repositorios adecuados en los archivos del sistema.  <b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> Debe seleccionar esta opción para poder proceder a la creación de los repositorios.  <b>3- Flujo de la acción a realizar:</b> Si los repositorios no están creados, la aplicación pasará a crear y colocar los archivos correspondientes. En caso de que no se realice esta operación y los repositorios no están creados, no se podrá ejecutar correctamente la instalación de los programas.	
<b>Observaciones:</b> N/A	
<b>Prototipo de interfaz:</b> 	

<b>Número:</b> HU_6	<b>Nombre del requisito:</b> Modificar repositorios
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 6ta
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si los repositorios no están correctamente configurados, la aplicación no instalará los programas correctamente.	<b>Tiempo Real:</b> 3 días
<p><b>Descripción:</b></p> <p><b>1-Objetivo:</b> Permitir modificar los repositorios del sistema</p> <p><b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> Debe seleccionar la opción de los pre-requisitos del sistema.</p> <p><b>3- Flujo de la acción a realizar:</b> Después de seleccionar esta opción la aplicación se encargará de verificar si los repositorios están configurados correctamente. En el caso de que estén mal configurados, ella se encargará de realizar el proceso de configuración correctamente.</p>	
<b>Observaciones:</b> N/A	
<b>Prototipo de interfaz:</b>	

HU-6\_ Modificar repositorios

<b>Número:</b> HU_7	<b>Nombre del requisito:</b> Mostrar menú
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 7ma
<b>Prioridad:</b> Baja	<b>Tiempo Estimado:</b> 2 días

**Riesgo en Desarrollo:** Si no se selecciona una opción del menú, este no accederá a realizar las operaciones que desee.

**Tiempo Real:** 1 días

**Descripción:**

**1- Objetivo:**

*Permitir mostrar todas las opciones y operaciones que puede realizar la aplicación*

**2- Acciones para lograr el objetivo (precondiciones y datos):**

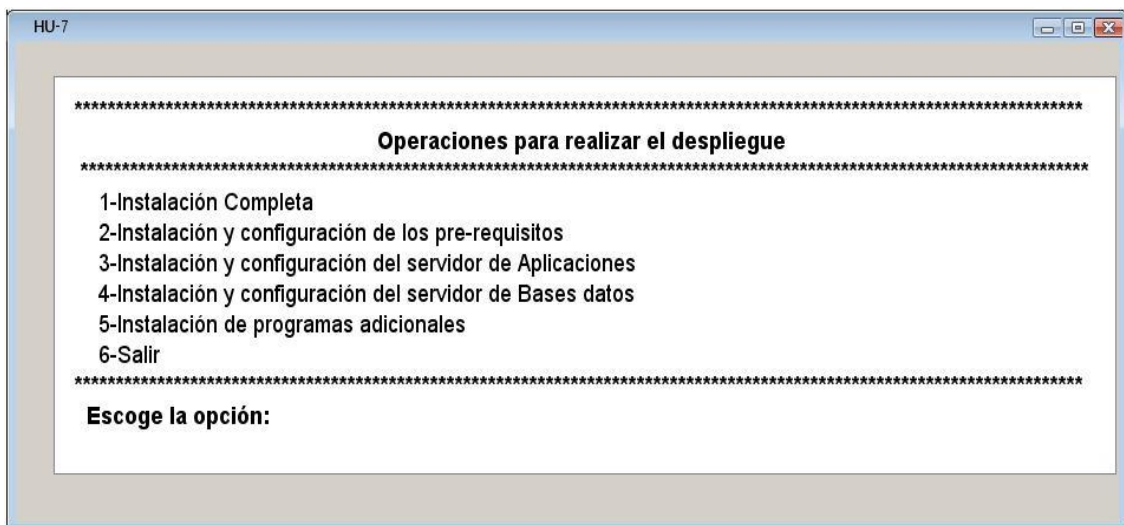
Debe tener contraseña de administrador.  
Debe existir conectividad con el repositorio.

**3- Flujo de la acción a realizar:**

Después de verificar la contraseña de administrador, y comprobar la conexión con el repositorio; la aplicación accederá a mostrar el menú donde contiene todas las operaciones que se pueden realizar para realizar el despliegue de *software*.

**Observaciones:** N/A

**Prototipo de interfaz:**



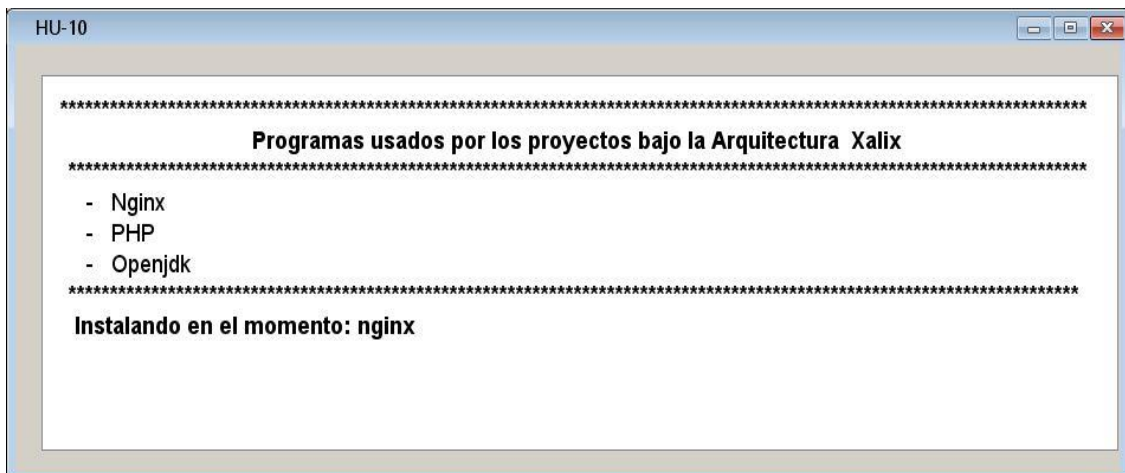
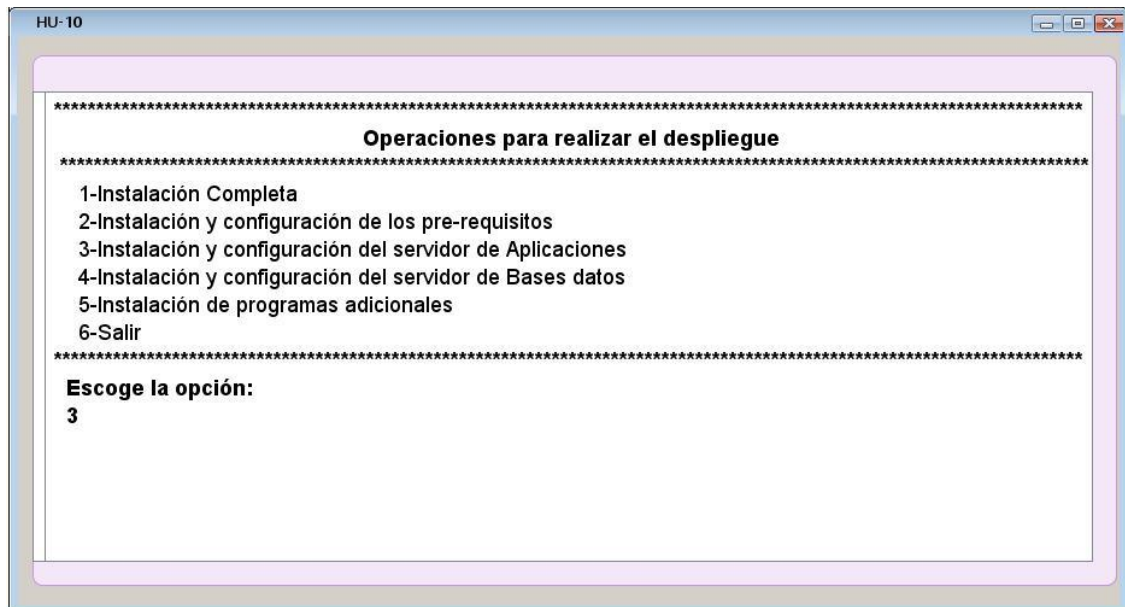
HU-7\_ Modificar repositorios

<b>Número:</b> HU_8	<b>Nombre del requisito:</b> Instalar y configurar los programas del servidor de aplicaciones.
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 8va
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si el usuario no selecciona esta opción puede provocar que su entorno de trabajo no esté disponible para desplegar los software que estén bajo la arquitectura Xalix	<b>Tiempo Real:</b> 3 días
<p><b>Descripción:</b></p> <p><b>1- Objetivo:</b> <i>Permitir instalar los programas para desplegar la arquitectura Xalix.</i></p> <p><b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> - Debe seleccionar esta opción para poder proceder a la instalación.</p> <p><b>3- Flujo de la acción a realizar:</b> Después de seleccionar la opción instalar y configurar los programas del servidor de aplicaciones, la aplicación mostrará un mensaje indicando los programas que se comenzaran a instalar y configurar respectivamente. Saldrá un mensaje donde podrá seleccionar si desea instalar mediante la opción(s/n). Si se selecciona la opción “n”, la aplicación mostrará un mensaje de cancelación y luego aparecerá un menú para seleccionar más opciones. Si el usuario selecciona la opción “s”, la aplicación accederá a instalar los programas mostrados anteriormente de forma organizada. Luego de realizar la instalación del primer programa mostrará un mensaje que se instaló correctamente. Después mostrar el mensaje que se instaló correctamente, la aplicación mostrará un mensaje diciendo si desea configurar la instalación mediante la opción(s/n); en caso de seleccionar la opción “n”, aparecerá un mensaje diciendo que la configuración del programa instalado se ha cancelado y tendrá la opción de instalar</p>	

los otros programas pasando por el mismo ciclo. Después de realizar los pasos correspondientes aparecerá la opción del menú.

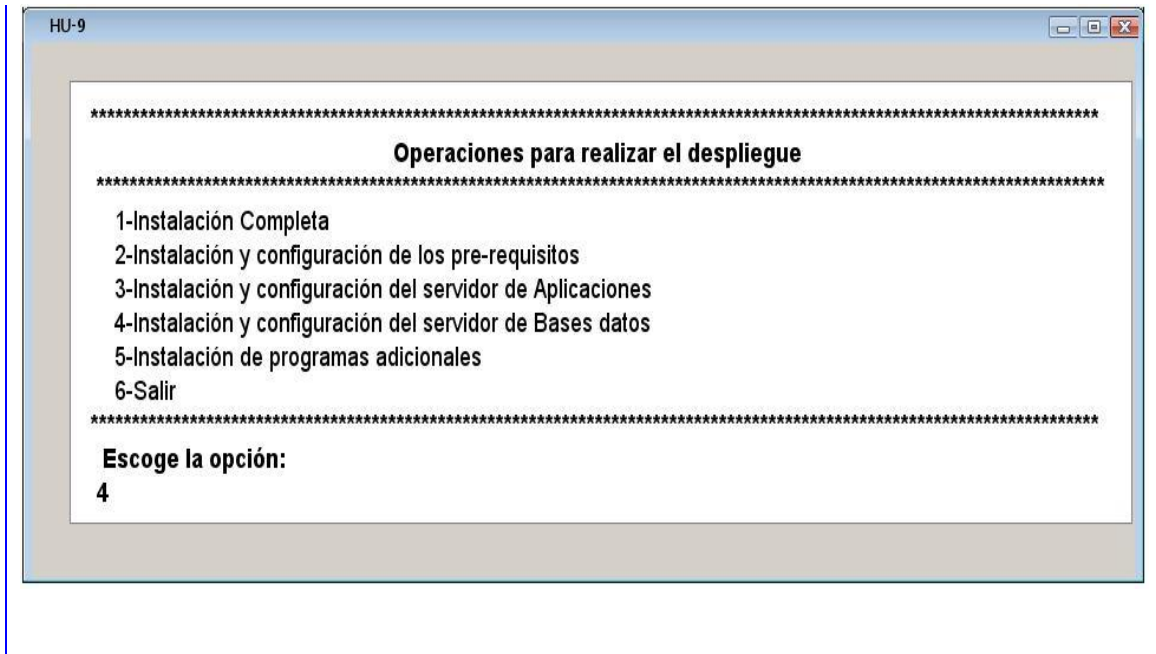
**Observaciones:** N/A

**Prototipo de interfaz:**



HU-8\_ Instalar y configurar los programas del servidor de aplicaciones.

<b>Número:</b> HU_9	<b>Nombre del requisito:</b> Instalar y configurar los programas del servidor de Bases datos.
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 9na
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 5 días
<b>Riesgo en Desarrollo:</b> En caso de que no se instalen o configuren los programas del servidor web, no se realizará un despliegue correcto.	<b>Tiempo Real:</b> 6 días
<p><b>Descripción:</b></p> <p><b>1-Objetivo:</b> <i>Permitir instalar y configurar los programas del servidor web</i></p> <p><b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> Debe seleccionar esta opción para poder proceder a la instalación y configuración.</p> <p><b>3- Flujo de la acción a realizar:</b> La aplicación después de seleccionar esta opción, procederá a instalar ciertas dependencias de Python y también el servidor de bases de datos PostgreSQL. Luego de realizar esta instalación, aparecerá un mensaje pidiendo al usuario si desea configurar la instalación (s/n). Si el usuario selecciona la opción "s" aparecerá un mensaje pidiendo la contraseña a colocar para el usuario postgre en el servidor de bases de datos. Después se pedirán las direcciones ip o máscaras de red para colocar en el archivo de configuración. Luego se pedirá la dirección ip solamente que se colocaran el archivo Postgresql.conf y después que se realice todo este procedimiento bien aparecerá el menú. Si el usuario selecciona la opción "n", aparecerá un mensaje diciendo que se ha cancelado la configuración.</p>	
<b>Observaciones:</b> N/A	
<b>Prototipo de interfaz:</b>	



HU-9\_ Instalar y configurar los programas del servidor de Bases datos.

<b>Número:</b> HU_10	<b>Nombre del requisito:</b> Verificar la configuración del archivo pip
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 10ma
<b>Prioridad:</b> Media	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b>	<b>Tiempo Real:</b> 3 días
<b>Descripción:</b> <b>1-Objetivo:</b> Permitir que se puedan utilizar librerías de Python a la hora de seleccionar el componente donde se encuentra.  <b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> - Debe seleccionar esta opción para poder proceder a la instalación.  <b>3- Flujo de la acción a realizar:</b> La aplicación después de seleccionar la opción de instalar y configurar el servidor web, procederá a verificar si el archivo de configuración PIP de Python está en el	



sistema. En caso de que no esté configurado la aplicación accederá a configurar y posteriormente instalar las librerías necesarias para poder ejecutar el componente correctamente.

**Observaciones:** N/A

**Prototipo de interfaz:**

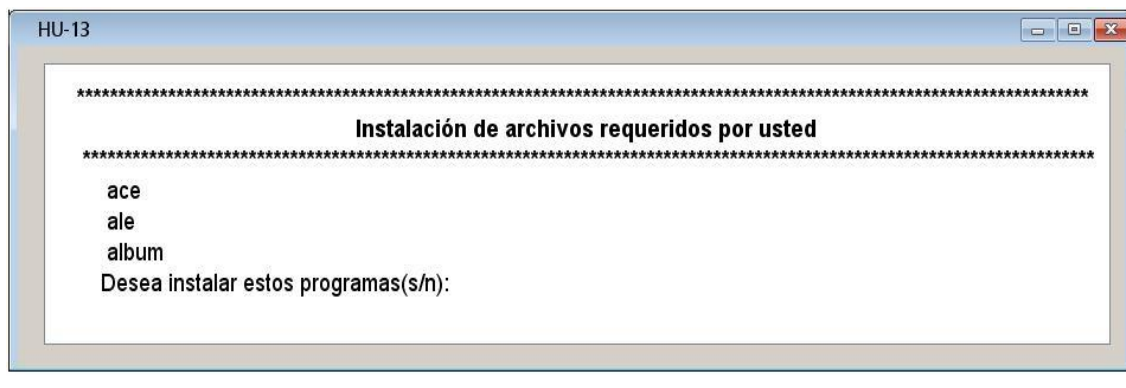
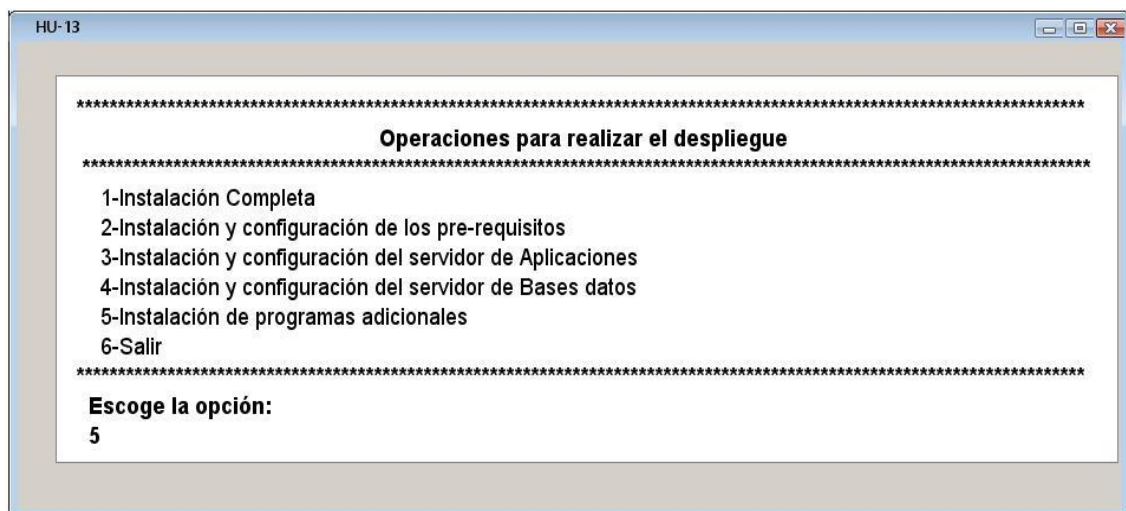
HU-10\_ Verificar la configuración del archivo pip.

<b>Número:</b> HU_11	<b>Nombre del requisito:</b> Instalar archivos necesarios
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 11na
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si el usuario no selecciona esta opción puede provocar que su entorno de trabajo no se instalen los programas adicionales que desee.	<b>Tiempo Real:</b> 3 días
<p><b>Descripción:</b></p> <p><b>1-Objetivo:</b> Permitir instalar programas adicionales para el entorno de trabajo</p> <p><b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> - Debe seleccionar esta opción para poder proceder a la instalación.</p> <p><b>3- Flujo de la acción a realizar:</b> La aplicación después de seleccionar esta opción, mostrará un mensaje indicando si desea instalar los programas colocados por usted en un archivo txt, mediante la opción(s/n). Después de seleccionar la opción "s" se le pedirá al usuario que entre la dirección donde se encuentran los programas a instalar. Luego mostrará la lista de programas que desea instalar. Saldrá un mensaje donde podrá seleccionar la opción</p>	

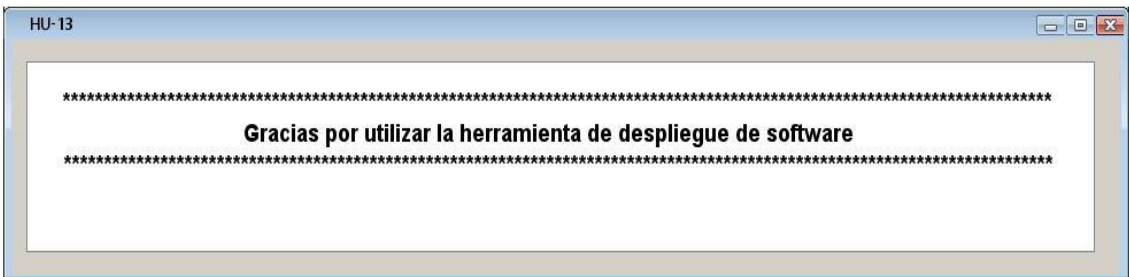
(s/n). Si el usuario selecciona la opción si, la aplicación accederá a instalar todos los programas necesarios por usted. La aplicación mostrará un mensaje de confirmación de que los programas fueron instalados correctamente. Si se selecciona la opción “n”, la aplicación mostrara un mensaje de cancelación. La aplicación mostrará el menú para seleccionar más opciones.

**Observaciones:** N/A

**Prototipo de interfaz:**



HU-11\_ Instalar archivos necesarios.

<b>Número:</b> HU_12	<b>Nombre del requisito:</b> Finalizar el proceso de instalación.
<b>Programador:</b> Yusniel Armando Reynaldo Molina	<b>Iteración Asignada:</b> 12ma
<b>Prioridad:</b> Media	<b>Tiempo Estimado:</b> 4 días
<b>Riesgo en Desarrollo:</b> Si el usuario no	<b>Tiempo Real:</b> 3 días
<p><b>Descripción:</b></p> <p><b>1-Objetivo:</b> Permitir finalizar el proceso de instalación</p> <p><b>2- Acciones para lograr el objetivo (precondiciones y datos):</b> Debe seleccionar esta opción para poder proceder a finalizar la instalación.</p> <p><b>3- Flujo de la acción a realizar:</b> El usuario puede seleccionar la opción salir, para terminar el proceso de instalación. Luego de seleccionar esta opción aparecerá un mensaje especificando que ha terminado el proceso.</p>	
<b>Observaciones:</b> N/A	
<p><b>Prototipo de interfaz:</b></p>  <p>The screenshot shows a window titled 'HU-13' with a standard Windows-style title bar. The main content area contains a message centered between two lines of asterisks: 'Gracias por utilizar la herramienta de despliegue de software'.</p>	

HU-12\_ Finalizar el proceso de instalación.

**Anexos II: Casos de pruebas**

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Verificar permisos de root	Permite verificar si todos los cambios se realizarán con permisos de root	N/A	N/A	N/A	El usuario para entrar a la aplicación deberá tener permisos de administrador o de root, con el objetivo de que pueda realizar todos los cambios en el sistema.  En caso de que no entre con la contraseña de administrador, no tendrá acceso a la aplicación y mostrará un cartel diciendo que no tiene permisos de root.	Verificar permisos de administrador.

CP-1\_ Verificar permisos de root

## Herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES

---

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Verificar SO	Permite verificar datos importantes del SO	N/A	N/A	N/A	La aplicación obtendrá datos como el nombre, la versión y la distribución; con el objetivo de configurar correctamente los repositorios.	Obtener datos del Sistema Operativo.

### CP-2\_ Verificar SO

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Verificar conexión con el repositorio.	Permite verificar si existe conexión con el repositorio.	N/A	N/A	N/A	La aplicación antes de comenzar cualquier acción que implique realizar algún trabajo con el repositorio, lo primero que realiza una comprobación para ver si existe conexión con el mismo. En caso contrario se mostrará un cartel diciendo que existen problemas con la conexión y saldrá rápidamente.	Verificar conexión.

### CP-3\_ Verificar conexión con el repositorio

## Herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES

---

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Mostrar Menú	Muestra el menú donde se podrán seleccionar las opciones para realizar el despliegue.	N/A	N/A	N/A	Luego de que la aplicación compruebe que hay conexión y que el usuario entro con permisos de administrador, se mostrará el menú donde el usuario podrá seleccionar las opciones.	Mostrar el menú.

### CP-4\_ Mostrar Menú

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Crear repositorios	La aplicación creará el repositorio dependiendo de la versión del sistema operativo	N/A	N/A	N/A	El sistema debe mostrar el nombre del repositorio y las direcciones que debe tener. En caso de no encontrar la versión del repositorio, la aplicación mostrará un mensaje diciéndole al usuario que debe insertar los repositorio de forma manual.	Crear repositorios

### CP-5\_ Crear repositorios

## Herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES

---

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Modificar repositorios	Permite modificar los repositorios del sistema.	N/A	N/A	N/A	La aplicación en caso de que los repositorios no se encuentren configurados correctamente, accederá a modificar y así restaurar los repositorios del sistema.	Modificar los repositorios.

### CP-6\_ Modificar repositorios

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar la opción "3".	Permite optar por la opción de instalar los programas del servidor aplicaciones	N/A	N/A	N/A	La aplicación muestra un mensaje indicando los programas que desea instalar.	Instalar y configurar los programas para el funcionamiento del servidor aplicaciones.

## Herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES

EC 1.2	La aplicación después de seleccionar la opción "s" para instalar los programas.	N/A	N/A	N/A	Luego de seleccionar la opción "s", la aplicación instalará el 1er programa que aparecen en un listado que se muestra en pantalla de los programas para montar un servidor.	Comenzar con la instalación de los programas.
EC 1.3	La aplicación luego de instalar el programa, pedirá al usuario si desea configurar el programa.	N/A	N/A	N/A	Luego de que se seleccione la opción de "s", la aplicación pedirá ciertos valores, que el usuario debe escribir correctamente. En caso de que no escriba los parámetros correctos la aplicación lanzará un mensaje pidiendo valores válidos para el sistema.	Configurar los programas.
EC 1.4	La aplicación después de seleccionar la opción "3", pedirá al usuario si desea instalar los programas mostrados en pantalla.	N/A	N/A	N/A	La aplicación pedirá al usuario seleccionar si desea instalar los programas mostrados en pantalla.  Si el usuario selecciona la opción de "n", la aplicación mostrará un mensaje diciendo que se ha cancelado la instalación.	Cancelar la instalación.
EC 1.5	Si el usuario	N/A	N/A	N/A	Si el usuario selecciona	Cancelar la



## Herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES

Seleccionar la opción "n".	selecciona la opción "n", la aplicación cancela el proceso.		A		la opción "n", la aplicación cancela el proceso de configuración del programa que se instaló anteriormente.	configuración del programa.
----------------------------	---	--	---	--	---	-----------------------------

### CP-7\_ Instalar y configurar el servidor de aplicaciones

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar la opción "4".	Permite optar por la opción de instalar los programas del servidor de Bases datos	N/A	N/A	N/A	La aplicación muestra un mensaje indicando el servidor de Bases datos.	Instalar y configurar los programas para el funcionamiento del servidor de Bases datos.
EC 1.2 Seleccionar la opción "s" para instalar los programas.	La aplicación después de seleccionar la opción s, comenzará a instalar el servidor de Bases de datos	N/A	N/A	N/A	Luego de seleccionar la opción "s", la aplicación instalará el servidor de Bases datos PostgreSQL	Comenzar con la instalación de los programas.
EC 1.3 Seleccionar la opción "s" para	La aplicación luego de instalar el programa,	N/A	N/A	N/A	Luego de que se seleccione la opción de "s", la aplicación pedirá ciertos valores, que el	Configurar el programa.

## Herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES

configurar el programa instalado	pedirá al usuario si desea configurar el programa.				usuario debe escribir correctamente. En caso de que no escriba los parámetros correctos la aplicación lanzará un mensaje pidiendo valores válidos para el sistema.	
EC 1.4 Seleccionar la opción "n".	La aplicación después de seleccionar la opción "4", pedirá al usuario si desea instalar los programa mostrados en pantalla.	N/A	N/A	N/A	La aplicación pedirá al usuario seleccionar si desea instalar los programas mostrados en pantalla.  Si el usuario selecciona la opción de "n", la aplicación mostrará un mensaje diciendo que se ha cancelado la instalación.	Cancelar la instalación.
EC 1.5 Seleccionar la opción "n".	Si el usuario selecciona la opción "n", la aplicación cancela el proceso.	N/A	N/A	N/A	Si el usuario selecciona la opción "n", la aplicación cancela el proceso de configuración del programa que se instaló anteriormente.	Cancelar la configuración del programa.

CP-8\_Instalar y configurar el servidor de Bases datos

## Herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES

---

Escenario	Descripción	Nombre	País	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Verificar y configurar el archivo PIP de Python.	Permite a la aplicación configurar el archivo PIP de Python.	N/A	N/A	N/A	La aplicación permite configurar el archivo PIP de Python con el objetivo de utilizar las ventajas de las librerías y su repositorio en la UCI.	Instalar y configurar librerías en Python.

CP-9\_ Verificar y configurar el archivo PIP

## Herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES

---

Escenario	Descripción	Nombre	Países	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar la opción "5"	Permite optar por la opción de instalar los programas adicionales por el usuario	N/A	N/A	N/A	La aplicación mostrará un mensaje diciendo si desea acceder al contenido de esta opción, luego debe ingresar la dirección donde se encuentra el archivo con los nombres. Después la aplicación mostrará un mensaje con los nombres de todos los programas escritos en el fichero de configuración, que debe llevar por nombre "programas", en caso de que no se escriba correctamente la dirección donde está el archivo con el nombre de los programas; la aplicación mostrará un error y pidiendo que se coloque una dirección válida. El usuario debe seleccionar "s" o "n" para instalar los programas, en caso de que exista algún error, la aplicación mostrará los errores en pantalla.	Instalar programas desde un archivo llamado "programas"

## Herramienta para el despliegue de *software* bajo la arquitectura Xalix del Centro FORTES

---

EC 1.2 Seleccionar la opción "n"	La aplicación pedirá al usuario si desea instalar los programas mostrados en pantalla.	N/A	N/A	N/A	Luego de que el usuario seleccione la opción "n", la aplicación mostrará un mensaje diciendo que se ha cancelado la instalación y aparecerá el menú de las opciones.	Cancelar la opción de instalar.
-------------------------------------	--	-----	-----	-----	--	---------------------------------

CP-10\_Instalar archivos adicionales