Universidad de las Ciencias Informáticas

Facultad 3

Título: Solución para automatizar el proceso de instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía.

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor: Gustavo Yunier Leyte-Vidal Chacón

Tutores: Ing. Marieta Peña Abreu

Ing. Manuel Enrique Gutiérrez

La Habana, Cuba

"Año 53 de la Revolución"

Declaración de autoría.

Declaro ser el único autor de este trabajo y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Autor: Gustavo Yunier Leyte-Vidal Chacón

_____ _____

Tutor: Ing. Marieta Peña Abreu Tutor: Ing. Manuel Enrique Gutiérrez

Dedicatoria.

A mis padres principalmente y tíos por todo el apoyo, la confianza, ayuda y el amor incondicional que siempre me han dado.

A mis hermanos por estar conmigo cada día y hacerme saber que puedo contar con ellos para cualquier cosa.

A mi novia Maylin por ser una de las personas más importantes en mi vida y estar en todo momento conmigo.

A mi prima Daylin para que sepa que es posible conseguir sus metas.

A toda mi familia que tanto se preocupa por mí.

Agradecimientos.

A mis padres, con su guía he podido llegar a ser alguien mejor en la vida.

A mis tíos por su ayuda en todo momento.

A la decana Ivonne por brindarme su ayuda cuando más lo necesité.

A mi familia por confiar siempre en mí.

A todos, los que de una forma u otra, han colaborado con la realización de este trabajo.

Resumen.

La rápida evolución de las Tecnologías de la Información y las Comunicaciones (TIC), ha adquirido una creciente importancia para el mejoramiento de la gestión de información de las organizaciones en todo el mundo. Cuba ha obtenido progresos con la creación de varios centros destinados al desarrollo de software, como lo es la Universidad de las Ciencias Informáticas (UCI).

La UCI en convenio con la Fiscalía General de la República de Cuba, realiza el proyecto Sistema de Informatización de la Gestión de la Fiscalía (SIGEF) para automatizar los procesos que se realizan de manera manual en las fiscalías de todo el país, al finalizarlo, se detectan varias deficiencias en el proceso de despliegue realizando la instalación y actualización del sistema, las cuales no permiten la facilidad ni agilidad del proceso.

El presente trabajo de diploma propone el desarrollo de una herramienta que implemente las funcionalidades necesarias para garantizar la instalación y actualización de SIGEF, facilitándole el proceso al personal del despliegue. Para ello se utiliza el entorno de desarrollo integrado Eclipse, el lenguaje de programación Python, y la metodología XP para guiar todo el proceso de desarrollo.

Como resultado se obtuvo una herramienta con la cual es posible llevar a cabo la instalación y actualización de SIGEF reduciendo considerablemente el tiempo de realizar dicho proceso y los errores, al realizarlo sin necesidad de tener un alto nivel de conocimiento sobre el proceso.

Índice

Introducción.

En la actualidad la mayoría de los procesos que se llevan a cabo en las diferentes esferas de la sociedad en el mundo están relacionados con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), la cual contribuye al desarrollo social y económico de cualquier país. En Cuba se han obtenido progresos en este sentido, como es el desarrollo del Sistema de Informatización de la Gestión de la Fiscalía (SIGEF) en el año 2008.

SIGEF surge como un proyecto de la Universidad de las Ciencias Informáticas con la Fiscalía General de la República de Cuba, el mismo está dividido en varios subsistemas los cuales están estrechamente vinculados a la estructura organizativa que se emplea actualmente en las fiscalías de todo el país, permitiendo agilizar los procesos que se realizan de manera manual y la comunicación entre ellas para la toma de decisiones. Concluye su primera fase en el 2012 con éxito y se procede a realizar el despliegue en algunas de las fiscalías de La Habana.

Al realizar el proceso de instalación y actualización del sistema de manera manual, se han detectado varias deficiencias que no permiten la facilidad ni agilidad durante el desarrollo de dicho proceso. Es obligatorio realizar numerosos pasos los cuales incluyen operaciones con archivos y con la base de datos, además de configuraciones en el servidor web e instalaciones de los programas necesarios.

Para actualizar SIGEF inicialmente se deben obtener los archivos que han sido creados o modificados desde la última instalación, y luego efectuar los procedimientos necesarios para que continúe funcionando correctamente el sistema, ya que un error puede influir en el trabajo que se esté ejecutando en esa fiscalía.

Se debe tener en cuenta que los servidores de las fiscalías utilizan como sistema operativo Ubuntu, lo cual significa que se debe tener el conocimiento necesario ya que algunos pasos requieren administración del sistema. Todas las operaciones y configuraciones mencionadas son ejecutados por líneas de comandos, pudiéndose cometer algún error y haciendo que se demore el proceso aún más.

Adicionalmente, el personal designado a esta tarea es variable, ya que existen afectaciones en la planificación realizada, lo cual trae como consecuencia que no se resuelvan en todas las ocasiones los problemas detectados adecuadamente, ni se

registren correctamente las soluciones empleadas en el caso de ocurrir dicha situación.

Todos los problemas mencionados anteriormente pudieran eliminarse con una herramienta que informatice el proceso de instalación y actualización de SIGEF.

Situación por la que surge el siguiente **problema a resolver:** ¿Cómo contribuir a facilitar el proceso de instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía?

El **objeto de estudio** está determinado por el proceso de desarrollo de software.

Se plantea como **objetivo general** desarrollar una aplicación que facilite el proceso de instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía.

Se deriva como **campo de acción** el proceso de desarrollo de software de instaladores para sistemas web.

Por todo lo antes expuesto se plantea la **hipótesis**: si se desarrolla una aplicación que informatice la instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía, se logrará minimizar el tiempo y los errores al realizar dicho proceso sin importar quien lo efectúe.

Se establecen los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Diseñar la solución para el proceso de instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía.
- Implementar la solución para el proceso de instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía.
- Validar los resultados obtenidos.

El proyecto de investigación será desarrollado a través de las siguientes **tareas**:

- Estudio de los principales conceptos relacionados con la instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía.
- Estudio de las principales aplicaciones de instalación y actualización de los sistemas de gestión.
- Selección de los patrones de diseño.

- Elaboración del diseño de la herramienta para informatizar el proceso de instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía.
- Implementación de la herramienta para informatizar el proceso de instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía.
- Validación de la herramienta siguiendo la metodología seleccionada.

Para el correcto cumplimiento de las tareas se utilizan los **métodos científicos de la investigación** teórico y empírico.

Los métodos teóricos permitirán estudiar las características del objeto de investigación que no se observan directamente, de ellos se emplearán:

- **Analítico - Sintético**: se utiliza para el análisis de la información existente y obtener los elementos más importantes.
- **Histórico - Lógico**: se utiliza para el estudio de la evolución de los instaladores.
- **Hipotético - Deductivo**: se utiliza para el apoyo de la hipótesis planteada.
- **Modelación:** se utiliza para la realización de modelos.

Los métodos empíricos permitirán extraer de los fenómenos analizados las informaciones que se necesitan sobre ellos a través de observaciones. De ellos se emplearán:

- **Entrevista**: se utiliza para obtener información sobre cómo se realiza el proceso de instalación y actualización en las fiscalías.
- **Observación**: permitirá detectar la situación real a la hora de realizar el despliegue en las fiscalías.
- **Medición:** se utiliza para la validación de la propuesta final.

Con la elaboración del presente trabajo de diploma se obtendrán los siguientes impactos:

- **Científico**: profundización y sistematización del conocimiento existente sobre el proceso de desarrollo de software de instaladores para sistemas web.
- **Tecnológico**: disposición de una nueva herramienta útil.
- **Político**: incremento de los resultados de la Universidad de las Ciencias Informáticas hacia otros sectores.

- **Social**: incremento de la credibilidad de la Universidad de las Ciencias Informáticas al mejorar los resultados de los convenios realizados.
  Beneficio indirecto del trabajo de los fiscales.
- **Económico**: disminución de tiempo y costo al realizar el proceso de instalación y actualización del Sistema de Informatización de la Gestión de la Fiscalía.

1. Fundamentación teórica.

1.1. Introducción.

En el presente capítulo se realiza un estudio de los principales conceptos relacionados con la instalación y actualización de los sistemas de gestión, necesarios para la comprensión del desarrollo de la investigación.

Teniendo en cuenta que las fiscalías utilizan Ubuntu como sistema operativo en los servidores, durante este capítulo se analizan los tipos de archivos y lenguajes que pueden ser utilizados para cumplir el objetivo general, además, se exponen soluciones relacionadas con este tema para conocer las tendencias actuales existentes, así como sus características y funcionamiento. Se estudian diferentes metodologías y herramientas existentes para determinar cuál es la que más se ajusta a las necesidades del sistema.

1.2. Definiciones principales.

1.2.1. Programa de instalación.

También denominado instalador, es un software que prepara una aplicación para ejecutarse en el equipo. A menos que la aplicación sea un programa de una sola función, se compone de muchos archivos individuales que a menudo se almacenan en varios niveles de carpetas en el ordenador del usuario. En algunos casos existen cientos e incluso miles de archivos que forman parte de una aplicación. (1)

1.2.2. Actualización de software.

Una actualización de software es un paquete de continuación de un software actualizado o en su versión final de desarrollo, un paquete de servicios (Service Pack), una actualización crítica o una actualización de seguridad; utilizado para mejorar o para corregir un producto de software en caso de existir un error.

Estos programas son muy utilizados para mejorar el software sin tener que desinstalarlo una vez que se haya creado una versión superior del mismo, puede ser desde la web o simplemente ejecutarlo desde una ubicación en el ordenador. (2)

1.2.3. Versión de un software.

La versión de un software es un número o nombre que se asigna a un programa informático para mencionar su nivel de desarrollo y su actualización. Lo habitual es que el versionado esté dado por dos números, separados por un punto. El primer número es el mayor, mientras que el segundo es el menor. El mayor se modifica cuando se producen grandes cambios o saltos cualitativos en el desarrollo y el menor varía con las alteraciones o correcciones más pequeñas.

Por ejemplo, la versión 1.5 de un software refiere a la primera gran versión de un programa informático que ya lleva cinco modificaciones menores. Es posible que el siguiente desarrollo sea la versión 1.6 o, si se trata de una modificación importante, pase a denominarse 2.1. (3)

1.2.4. Aplicación web.

Son aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. Es decir, es un software desarrollado en un lenguaje soportado por los navegadores web.

Debido a la facilidad para actualizar y mantenerlas sin tener que distribuir e instalar software a miles de usuarios, son muy utilizadas. Una página web puede contener elementos que permiten la comunicación activa entre el usuario y la información, accediendo a los datos de modo interactivo, como rellenar o enviar formularios, participar en juegos, etc.

Se han desarrollado diferentes tecnologías para la confección de aplicaciones web posibilitando que sean más rápidas y compatibles con los diferentes navegadores. (4)

1.2.5. Sistema de Informatización de la Gestión de la Fiscalía (SIGEF).

Es un proyecto de investigación y desarrollo que consiste en una aplicación web a través de la cual, el fiscal, desde la esfera de trabajo a la que pertenece, puede realizar todas sus actividades o procesos pertinentes, los cuales se encuentran digitalizados en una secuencia lógica de proceder según lo establecido y asistidos por las leyes que en cada caso correspondan. Este sistema constituye una ayuda a la toma de decisiones de los fiscales ya que genera un grupo elevado de reportes, viabiliza y humaniza el trabajo de los mismos, permitiendo reducir al máximo el

margen de errores. El mismo proporciona en gran medida a través de funcionalidades la automatización de los procesos fundamentales en los que interviene la Fiscalía General de la República.

1.2.6. Componentes del sistema.

El Sistema de Informatización de la Gestión de la Fiscalía está integrado por varios componentes que le permiten el correcto funcionamiento como son el servidor de base de datos PostgreSQL 8.4, el servidor web Apache 2 (o superior), el lenguaje de programación PHP 5 (o superior) y la base de datos con la información para iniciar y ejecutarse.

1.2.6.1. Servidor web Apache.

Apache es el servidor web hecho por excelencia, posibilita ser configurado, presenta una gran robustez y estabilidad contribuyendo a que millones de servidores reiteren su confianza en este programa.

Es multiplataforma, lo que le permite correr en varios sistemas operativos, es una tecnología gratuita de código fuente abierto. Es un servidor altamente configurable de diseño modular y permite personalizar la respuesta ante los posibles errores que se puedan surgir en el servidor, conjuntamente proporciona la opción de configurar la creación y gestión de registros. (5)

1.2.6.2. Sistema de gestión de base de datos PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, su código fuente está libremente disponible. Utiliza un modelo cliente-servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Desde sus comienzos hasta la actualidad ha alcanzado estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Está disponible para Linux, UNIX en todas sus variantes y Windows 32/64bit. (6)

1.2.6.3. PHP.

PHP es un lenguaje de secuencia de comandos de servidor, diseñado específicamente para la web. El código es interpretado en el servidor web y genera código HTML.

Fue concebido en 1994 y es fruto del trabajo de Rasmus Lerdorf. Ha sido adoptado por otras personas de talento y ha experimentado tres trasformaciones importantes hasta convertirse en el producto actual. Es un software de código abierto y las siglas PHP significaban inicialmente a Personal Home Page (Página de Inicio Personal), pero se modificaron de acuerdo con la convención de designación de GNU y ahora equivale a Hipertext Preprocessor (Preprocesador de Hipertexto PHP).

Es un lenguaje muy fácil de aprender, se caracteriza por ser un lenguaje muy rápido, soporta en cierta medida la orientación a objeto, clases y herencia. Es un lenguaje multiplataforma con capacidad de conexión con la mayoría de los manejadores de base de datos, además de expandir su potencial utilizando módulos. Posee documentación en su página oficial, la cual incluye descripción y ejemplos de cada una de sus funciones. Es de código libre, por lo que se presenta como una alternativa de fácil acceso para todos y no requiere definición de tipos de variables ni manejo detallado del bajo nivel.

A pesar de todos los beneficios anteriormente dichos es necesaria la instalación un servidor web para su ejecución, la legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP, la programación orientada a objetos es aún muy deficiente para aplicaciones grandes. (7)

1.2.6.4. Base de datos de SIGEF.

Almacena todos los datos correspondientes al trabajo que se realiza en las fiscalías.

1.3. Herramientas para desarrollar instaladores de sistemas de software en plataformas Linux.

En la actualidad existen diversas herramientas capaces de desarrollar instaladores de sistemas de software en Linux. Se seleccionaron las siguientes aplicaciones para el estudio de las mismas, obtener una mejor comprensión del tema y conocer las tendencias actuales.

1.3.1. BitRock InstallBuilder.

BitRock InstallBuilder crea fácilmente instaladores multiplataforma para los sistemas operativos Linux, Windows, Mac OS X, etc. La aplicación genera instaladores nativos con el sistema operativo en el que se instala, es decir, que se integra en su entorno gráfico y funcional sin necesidad de dependencias. Adicionalmente, BitRock InstallBuilder contiene hasta 16 idiomas. Además incorpora instaladores autónomos e incluye una herramienta de instalación que genera paquetes independientes.

El programa genera ficheros de instalación ejecutables optimizados en tamaño y velocidad, independientemente de factores externos, que pueden ejecutarse con interfaz de usuario, texto o modo silencioso. Tiene una licencia libre totalmente funcional para los proyectos de código abierto. De forma predeterminada, InstallBuilder realiza una copia de seguridad de todos los archivos sobrescritos durante la instalación, para que puedan ser recuperados, en caso de un error. Está optimizada en tamaño y velocidad. Esto reduce la descarga, el inicio y tiempo de instalación. Permite a los usuarios elegir los componentes a instalar, lo que garantiza la reutilización de componentes a través de instaladores para acelerar el desarrollo. (8)

1.3.2. InstallJammer.

InstallJammer es una aplicación OpenSource y multiplataforma, destinada a crear interfaces gráficas de instalación para las aplicaciones que se desarrollen con cualquier lenguaje de programación o entorno de desarrollo como Java, C# y VB.Net o Eclipse y Visual Studio.NET. Es multiplataforma y funciona bajo Windows y la mayor parte de las versiones de UNIX, además de soportar escasamente MacOS X.

Entre sus múltiples características se puede destacar que permite una personalización de las instalaciones, así como la edición de los cuadros de diálogo que conformarán el asistente del instalador. Permite el control de temas. Proporciona capacidades para la desinstalación. Es una herramienta bastante adecuada que permite la conclusión de los proyectos de cualquier desarrollador o programador tanto independiente como empresarial. (9)

1.4. Tipos de ficheros que permiten la ejecución de un programa en Linux.

En Linux existen varios tipos de archivos que permiten ejecutar una aplicación, a continuación se muestran algunos de ellos, permitiendo ampliar los conocimientos de los mismos para obtener una mejor comprensión de este tema.

1.4.1. Archivos ".deb".

Los archivos con la extensión ".deb" pueden contener archivos ejecutables, bibliotecas, documentación asociada con un programa o conjunto de programas y las instrucciones para su instalación. El sistema de paquetes de Debian permite instalar, actualizar y remover los programas disponibles. Pueden contener programas u otros archivos para distribuir como documentación, temas de iconos, etc. También contienen las instrucciones sobre en qué parte del sistema de archivos instalar el nuevo programa, de que bibliotecas o programas depende, la instalación en curso y guiones (scripts) de configuración. Presentan la desventaja de que exista la posibilidad de que no puedan ser ejecutados en alguna distribución de Linux. (10)

1.4.2. Archivos ".bin".

Los archivos que tienen la extensión ".bin" suelen ser instaladores de programas binarios. La ventaja de instalar un programa con estos formatos es que por lo general van a funcionar bien en todas las distribuciones, mientras que otros formados pre compilados para instalar programas como los "*.deb"* están más limitados. (11)

1.4.3. Scripts.

Un script es un programa simple, que se almacena en un archivo de texto plano y se utiliza para realizar diversas tareas como la combinación de componentes, la interacción con el usuario o con el sistema operativo en cuestión**.**

Facilita la automatización de tareas a través de la creación de pequeñas utilidades. Se emplea mayormente a instancias de la administración de sistemas UNIX y son ejecutados por un intérprete de línea de comandos. (12)

1.5. Lenguajes de programación.

Un lenguaje de programación es una construcción mental del ser humano para expresar programas. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos y una regla principal que resume las demás. (13)

1.5.1. Perl.

Perl es un lenguaje diseñado por Larry Wall en 1987, que originalmente fue desarrollado para ser un lenguaje de manipulación de texto, sin embargo con el pasar de los años se ha formado una verdadera comunidad de personas que lo utilizan para el desarrollo de interfaces gráficas de usuario (GUI), desarrollo de páginas web, administración de sistema, programación en red, entre otras aplicaciones.

Fue creado con el fin de que fuera un lenguaje fácil de usar, completo y eficiente, en lugar de que fuera compacto y elegante. Originalmente este lenguaje fue llamado “Pearl”, que viene de “la parábola de la perla”, sin embargo ya existía un lenguaje con este nombre, por lo que Wall decidió cambiar el nombre a “Perl”. Es potente gracias a su extensibilidad mediante módulos (lo que en otros lenguajes de programación como Java es llamado “bibliotecas”). (14)

Tiene como principales características la facilidad de uso, el soporte para diferentes tipos de programación como lo son la orientada a objetos, la estructural y la programación funcional. Perl toma características de otros lenguajes de programación, su estructura está basada en bloques al estilo de C, lo que lo convierte en un lenguaje imperativo, con variables, expresiones, asignaciones, delimitación de bloques de código mediante llaves, estructuras de control y subrutinas. Es un lenguaje práctico, lo que quiere decir que no determina estrictamente una estructura para programar, sin embargo, hace que muchas veces sea muy difícil la detección de errores.

Perl está disponible para una gran cantidad de sistemas operativos, esto lo hace un lenguaje accesible a cualquier usuario, sirviendo para extender la comunidad del lenguaje y así convertirlo en un lenguaje muy usado y muy confiable.

La principal desventaja de Perl se encuentra en el tiempo de ejecución de un programa ya que se divide en dos fases:

Tiempo de compilación**:** en esta fase se crea el árbol sintáctico del texto del programa, luego el árbol es optimizado antes de iniciar la ejecución del programa**.**

Tiempo de ejecución**:** en esta fase se ejecuta el programa siguiendo el árbol creado en la fase anterior.

Lo anterior pone en evidencia una de las desventajas de Perl y su intérprete, la cual es que cada vez que se corre un programa debe ser compilado, lo que lo hace más lento en tiempo de ejecución que otros lenguajes. (15)

1.5.2. Java.

El lenguaje Java fue diseñado e implementado por un grupo de personas de la empresa Sun Microsystems, ese grupo se encontraba encabezado por James Gosling, considerado en todo el mundo como el padre de Java. En 1990 comenzó el diseño de este nuevo lenguaje de programación, siendo el resultado un lenguaje poderoso, sencillo y confiable, capaz de ejecutarse en todo tipo de procesadores. (16)

Los programas desarrollados en Java presentan diversas ventajas frente a los desarrollados en otros lenguajes: ejecución como aplicación independiente, ejecución como applet, ejecución como servlet, etc. Un applet es una aplicación especial que se ejecuta dentro de un navegador o al cargar una página HTML desde un servidor web. El applet se descarga desde el servidor y no requiere instalación en el ordenador donde se encuentra el navegador. Un servlet es una aplicación sin interface gráfica que se ejecuta en un servidor, la ejecución como aplicación independiente es análoga a los programas desarrollados con otros lenguajes. (17)

Otras características son que es un lenguaje:

- *Simple*. Elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes orientados a objetos.
- *Robusto*. El sistema de Java maneja la memoria de la computadora automáticamente.
- *Seguro*. El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.

- *Portable*. Como el código compilado de Java es interpretado, un programa compilado puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.
- *Independiente a la arquitectura*. Al compilar un programa en Java, el código resultante es un tipo de código binario conocido como byte code. Este es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma.
- *Interpretado*. Java corre en máquina virtual, por lo tanto es interpretado.
- *Dinámico*. No requiere que compiles todas las clases de un programa para que este funcione.

Pero de igual manera hay que mencionar que los programas desarrollados en Java no tienden a ser muy rápidos, también que hay diferentes tipos de soporte técnico para la misma herramienta, por lo que el análisis de la mejor opción se dificulta, algunas herramientas tienen un costo adicional y para manejo a bajo nivel deben usarse métodos nativos, lo que limita la portabilidad. (17)

1.5.3. Python.

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90, cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es similar a Perl, pero con una sintaxis simple, clara y sencilla lo que favorece un código legible. (18)

Se trata de un lenguaje interpretado o de script, que es interactivo y ofrece una gran cantidad de estructuras de datos de alto nivel por medio de un tipado dinámico, por lo tanto, no es necesario declarar el tipo de dato que va a contener una variable determinada, dicho tipo de dato será determinado en tiempo de ejecución según el valor asignado a la variable, además, el tipo de la variable puede cambiar si se le asigna un valor de otro tipo. Es aquel lenguaje que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora. (19)

Es multiplataforma, por lo que puede utilizarse en sistemas operativos como UNIX, Solaris, GNU/Linux, DOS, Microsoft Windows, etc. El principal objetivo que persigue este lenguaje es la facilidad, tanto de lectura, como de diseño. Es de libre distribución y se usa en grandes plataformas como: YouTube y Google. Python es un lenguaje de programación dinámico y orientado a objetos, es multiparadigma, permite varios

estilos: programación orientada a objetos, programación estructural y funcional. Se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation, tiene gran soporte e integración con otros lenguajes y herramientas, también tiene integradas varias bibliotecas estándar y soporta varias bases de datos.

Entre otras de las características de Python se puede mencionar que la gestión de memoria es dependiente de la implementación, es legible por lo que es imposible escribir código ofuscado, también es minimalista ya que todo aquello innecesario no hay que escribirlo (;, {, }, '\n'), soporta objetos y estructuras de datos de alto nivel: strings, listas, diccionarios, etc. Tiene múltiples niveles de organizar código: funciones, clases, módulos, y paquetes, además de ser muy denso: poco código hace mucho.

Presenta el problema de los programas interpretados, que son más lentos que los compilados, sin embargo los programas interpretados suelen ser cortos, en los que la diferencia es inapreciable. (20) (21)

1.5.4. Selección del lenguaje de programación.

Los tres lenguajes analizados tienen varias similitudes en cuanto a características y funcionalidades, pero al analizar cuál es el más adecuado para desarrollar la aplicación que se necesita, teniendo en cuenta el sistema operativo utilizado en los servidores de las fiscalías (Ubuntu), el tiempo para culminar la aplicación, el entorno en que debe funcionar (consola), las operaciones que se deben realizar con archivos y la interacción constante que debe tener con el sistema operativo, se decide utilizar Python, debido a que ahorra un tiempo considerable en el desarrollo de un programa, pues contiene una gran cantidad de librerías que disponen de funciones para el tratamiento de cadenas de caracteres, números, archivos, llamadas al sistema, interfaces gráficas de usuarios (GUI), etc.

A diferencia de Java, Python viene instalado por defecto en el sistema operativo Ubuntu lo cual constituye un paso muy importante de avance, ya que disminuye operaciones para iniciar la aplicación. Tiene además una dependencia mínima, pues no requiere otro programa para funcionar. Ofrece una vía fácil y simple de emplear la programación orientada a objetos, especialmente, cuando se compara con Java.

Es muy fácil de aprender debido a que tiene una sintaxis simple, clara y sencilla favoreciendo un código legible, lo cual no sucede en los otros lenguajes analizados. La velocidad de ejecución del programa en Python sería inapreciable comparada con la de un lenguaje compilado, ya que no es un proyecto muy grande.

1.6. Herramientas y tecnología utilizadas para el desarrollo del software.

1.6.1. Sistema de control de versiones Subversion.

Subversion es un sistema de control libre y de código abierto que maneja ficheros y directorios, y los cambios introducidos en ellos con el tiempo. Esto le permite recuperar versiones antiguas de sus datos o examinar la historia de cómo cambiaron sus datos.

Puede operar a través de las redes, lo que le permite ser utilizado por personas en diferentes equipos lo cual fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto a través del cual todas las modificaciones que se producen. (22)

Se utiliza para almacenar los cambios producidos en el código del sistema web por el equipo de desarrollo de SIGEF, y en la descarga del mismo cuando se va a desplegar si se ha realizado algún cambio desde la última instalación.

1.6.2. Entorno de desarrollo integrado (IDE) Eclipse para Python.

Existen diversos IDE para el lenguaje seleccionado ya que no tiene uno definido, entre los más destacados se encuentran: Python Scripter, Wing IDE 3.0, Komodo y Eclipse siendo el seleccionado para el desarrollo de la aplicación. A continuación se muestran las características que lo resaltan sobre los otros:

Es un IDE escrito en Java, proporciona una plataforma robusta, escalable y de calidad para el desarrollo de software. En un principio fue pensado para el desarrollo de programas Java, pero mediante sus plugins puede extender su ámbito a otros lenguajes.

Precisamente PyDev es un plugin que permite programar en Python usando este IDE. Eclipse es una herramienta Open-Source, y PyDev se distribuye bajo la Eclipse Public License. Es un IDE completo que integra editor de código, depurador e intérprete. Al estar escrito en Java es multiplataforma, por lo que puede ser usado en cualquier sistema operativo que se instale, incorpora un depurador gráfico de Python, resaltado de sintaxis, autocompletado, definiendo las rutas del intérprete instalado (soporta varias versiones) e integrándolo.

Otras ventajas de la integración de Eclipse con PyDev son que brinda posibles soluciones de los errores, define y utiliza plantillas de código que se escriben automáticamente y ordena el código. Eclipse estructura los proyectos en directorios, paquetes, módulos, recursos, lo que ayuda a organizarlos y establecer una jerarquía. (23)

1.6.3. Interfaz gráfica Dialog para Python.

El módulo Dialog le provee una interfaz a Python sencilla y amigable en consola. Contiene una clase del mismo nombre brindando varios cuadros de diálogos mediante los métodos que tiene implementada, para darle el uso deseado.

1.7. Metodologías de desarrollo de software.

Las metodologías deben ser seleccionadas y adaptadas de acuerdo al contexto del proyecto, o sea, del tamaño, la experiencia del personal, el tiempo estimado, entre otros factores. Se han desarrollado varias por la importancia que tienen como guías en el desarrollo del software, por ello se realiza un estudio de las mismas para determinar la que más se ajusta a la problemática existente.

1.7.1. Proceso Unificado de Racional (Rational Unified Process, RUP).

El proceso unificado de desarrollo (RUP) es una metodología para el desarrollo de software que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de la vida del software. El resultado es un proceso basado en los componentes: **dirigido por los casos de uso**, **centrado en la arquitectura**, **iterativo e incremental**.

RUP se divide en cuatro fases:

1. Conceptualización (Concepción o Inicio).
2. Elaboración.
3. Construcción.
4. Transición.

Estas fases se dividen en nueve flujos de trabajo principales, los seis primeros son conocidos como flujos de ingeniería y los otros tres como de apoyo o soporte:

- Modelado del negocio.
- Requisitos.

- Análisis y Diseño.
- Implementación.
- Instalación.
- Administración de configuración y cambios.
- Administración del proyecto.
- Ambiente.

Esta metodología permite realizar la temprana mitigación de posibles altos riesgos, el conocimiento adquirido en una iteración puede aplicarse en todas, los usuarios están involucrados continuamente y funciona bien en proyectos de innovación. Se realiza un seguimiento detallado en cada una de las fases del estado y el progreso. (24)

1.7.2. Programación Extrema (Extreme Programming, XP).

La programación extrema (XP) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en una serie de valores, prácticas y principios para el éxito de un proyecto los cuales son:

Valores:

- **Retroalimentación:** debe existir una retroalimentación continua entre el cliente y los miembros del equipo de desarrollo.
- **Comunicación**: debe existir una fluida comunicación entre todos los participantes, XP ayuda mediante sus prácticas a fomentarla.
- **Simplicidad**: La simplicidad y la comunicación se complementan, cuanto más simple es el sistema menos debes comunicarte con él.
- **Coraje:** El coraje es un valor muy importante dentro de la programación extrema. Un miembro de un equipo de desarrollo extremo debe de tener el coraje de exponer sus dudas, miedos, experiencias sin embellecer éstas de ninguna manera. Detrás de este valor se encuentra el lema "si funciona, mejóralo", que choca con la práctica habitual de no tocar algo que funciona, por si acaso. .

Principios:

- Realimentación veloz.

- Modificaciones incrementales.
- Trabajo de calidad.
- Asunción de simplicidad.
- Aceptando el cambio.

Los principios fundamentales se apoyan en los valores, suponen un puente entre los valores y las prácticas, que están más ligadas a las técnicas que se han de seguir. (25)

Prácticas:

- **El juego de la planificación (The planning game):** Hay una comunicación frecuente el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
- **Entregas pequeñas (Small releases):** Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses.
- **Metáfora (Metaphor):** El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).
- **Diseño simple (Simple design):** Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
- **Pruebas (Testing):** La producción de código está dirigida por las pruebas unitarias, las cuales son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.
- **Refactorización (Refactoring)**. Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.

- **Programación en parejas (Pair programming):** Toda la producción de código debe realizarse con trabajo en parejas de programadores. Las principales ventajas de introducir este estilo de programación son: muchos errores son detectados conforme son introducidos en el código (inspecciones de código continuas), por consiguiente la tasa de errores del producto final es más baja, los diseños son mejores y el tamaño del código menor (continua discusión de ideas de los programadores), los problemas de programación se resuelven más rápido, se posibilita la transferencia de conocimientos de programación entre los miembros del equipo, varias personas entienden las diferentes partes del sistema, los programadores conversan mejorando así el flujo de información y la dinámica del equipo, y finalmente, los programadores disfrutan más su trabajo. Dichos beneficios se consiguen después de varios meses de practicar la programación en parejas.
- **Propiedad colectiva del código (Collective ownership):** Cualquier programador puede cambiar cualquier parte del código en cualquier momento. Motiva a todos a contribuir con nuevas ideas en todos los segmentos del sistema, evitando a la vez que algún programador sea imprescindible para realizar cambios en alguna porción de código.
- **Integración continúa (Continuous integration):** Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
- **40 horas por semana (40-hour week):** Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.
- **Cliente en el equipo (On-site customer):** El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Este es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.
- **Estándares de programación (Coding standards):** XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras como se ilustra en la Figura 1 (ver anexos), donde una línea entre dos prácticas significa que las dos prácticas se refuerzan entre sí. La mayoría de las prácticas propuestas por XP no son novedosas sino que en alguna forma ya habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica. El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo. (25)

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico, y es desarrollado en cuatro fases: planificación, diseño, desarrollo y pruebas.

Los objetivos de XP son muy simples:

- *La satisfacción del cliente*: Esta metodología trata de dar al cliente el software que necesita y cuando lo necesita, por lo que se debe responder rápido a las necesidades del cliente incluso cuando los cambios sean al final de ciclo de la programación.
- *Potenciar al máximo el trabajo en grupo*: Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.

Tiene de positivo que se puede obtener una programación organizada, de ahí que se obtenga menor taza de errores y la satisfacción del programador, es apropiado para entornos volátiles, se puede hacer una planificación transparente para los clientes, ya que conocen las fechas de entrega y funcionalidades. Permite definir en cada iteración cuales son los objetivos de la siguiente, así como tener una retroalimentación de los usuarios lo cual es muy útil. La presión está a lo largo de todo el proyecto y no en una entrega final. Como negativo tiene que se emplea solo en proyectos a corto plazo y tiene altas comisiones en caso de fallar. (25)

1.7.3. Scrum.

Scrum es una metodología de desarrollo ágil desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, indicada para proyectos con un rápido cambio de requisitos. Las principales características que la definen es que el desarrollo de software se realiza mediante iteraciones, denominados Sprints, con una duración de 30 días. El

resultado de cada Sprint es un incremento del producto que se muestra al cliente. Otra cuestión peculiar de Scrum son las reuniones a lo largo del proyecto, de ellas la más importante es la que se efectúa diariamente durante 15 minutos por parte del equipo de desarrollo para coordinar e integrar el trabajo.

Las ventajas más importantes son que se obtiene la entrega de un producto funcional al finalizar cada Sprint, da la posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente, normalmente adquiere equipos integrados y comprometidos con el proyecto, toda vez que ellos definieron el alcance y se auto-administran. Tiene un alcance acotado y viable, se obtiene software lo más rápido posible y este cumple con los requerimientos más importantes, se trabaja en iteraciones cortas, de alto enfoque y total transparencia, se acepta que el cambio es una constante universal y se adapta el desarrollo para integrar los cambios que son importantes, se incentiva la creatividad de los desarrolladores haciendo que el equipo sea auto administrado, se mantiene la efectividad del equipo habilitando y protegiendo un entorno libre de interrupciones e interferencias, y permite producir software de una forma consistente, sostenida y competitiva.

Al mismo tiempo hay que tener en cuenta que no genera toda la evidencia o documentación de otras metodologías, requiere delegar responsabilidades al equipo, incluso permite fallar si es necesario, y es una metodología que difiere del resto, esto causa cierta resistencia en su aplicación para algunas personas. (26)

1.7.4. Selección de la metodología de desarrollo de software.

Teniendo en cuenta el estudio de las metodologías, es seleccionada la metodología XP, pues se puede alegar que RUP representa un proceso muy detallado para desarrollarlo en proyectos a corto plazo, como es el caso, debido a la extensa documentación que genera cada actividad de cada fase y esto representa una gran inversión de tiempo.

Scrum por su parte es una metodología ágil y está diseñada para mostrar resultados lo antes posible cumpliendo con los requisitos más importantes, no genera toda la evidencia o documentación de otras metodologías y permite fallar si es necesario. Admite producir software de una forma competitiva por la duración de cada Sprint, no siendo el caso ya que es necesario el software en el tiempo establecido, el cual es corto, pero con la calidad requerida.

En cambio XP actualmente es la metodología ágil más extendida y documentada, adaptándose a las necesidades de cualquier equipo de desarrollo. Exige que se establezca una comunicación más fluida con el cliente y que éste tenga mayor participación en el desarrollo del software, logrando que se involucre más en el proceso. Se realizan pruebas constantemente del sistema y se le ofrece al usuario versiones continuas del mismo, esto provoca que se consigan productos usables con mayor rapidez que a su vez satisfacen las necesidades del usuario con mayor exactitud.

El proceso de integración es continuo, por lo que el esfuerzo final para la integración es nulo. Se consigue que los desarrolladores apliquen las buenas prácticas que se les ofrecen con esta metodología. XP se basa en unos principios y prácticas los cuales no se han hecho a priori o porque sí, sino que tienen un por qué a partir de una forma global de desarrollar software.

1.8. Conclusiones del capítulo.

Mediante la investigación que se ha realizado en el capítulo se concluye que:

- El estudio de las principales definiciones, las herramientas para desarrollar instaladores de sistemas de software en plataformas Linux y los tipos de ficheros que permiten la ejecución de un programa, permitió una mejor comprensión del objetivo general del trabajo.
- Se realizó un estudio de los lenguajes de programación fundamentando la elección de Python, para el cual se analizan las herramientas, tecnologías y metodologías necesarias para el desarrollo del software, concluyendo que el Sistema de control de versiones Subversion será utilizado para la descarga del sistema web, el IDE Eclipse para el desarrollo del software utilizando el plugin PyDev para la integración con Python, Dialog como interfaz gráfica de dicho lenguaje, y XP para guiar el desarrollo del software.

2. Planificación.

2.1. Introducción.

En el presente capítulo se describen las principales características del sistema, se hace referencia a la fase de planificación, propia de la metodología XP utilizada para el desarrollo de la solución propuesta. Se detallan las historias de usuario (HU) para luego establecer el orden en que serán implementadas atendiendo a su prioridad, esto constituye uno de los primeros pasos de la metodología seleccionada. También se muestran los artefactos generados que se requirieron complementariamente.

2.2. Fase Planificación.

En esta etapa el cliente crea las HU que son de interés para la entrega del producto. Durante la fase se realiza una estimación del esfuerzo que costará implementar cada HU, para la misma se utiliza como medida el punto, el cual se considera como una semana. Las HU necesitan de una a tres semanas de desarrollo, si la estimación es superior a tres semanas, se divide en dos o más historias, y si es menor de una semana, se combina con otra historia. (25)

2.2.1. Historias de usuario.

Tienen el mismo propósito que los casos de uso, se describen desde la perspectiva del cliente aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. El contenido de estas debe ser concreto y sencillo (ver tablas de historias de usuario en anexos).

Historia de Usuario	
Número: 1	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Instalar programas para el funcionamiento del sistema web.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Puntos estimados: 1	**Iteración Asignada:** 1
Descripción: Inicia cuando el usuario instala el sistema web o la base de datos.	
Observaciones:	

Tabla 1: Representación de la Historia de Usuario No.1

2.2.2. Flujo actual de los procesos.

El proceso de despliegue de SIGEF se realiza hoy en día entregándole al personal encargado de llevar a cabo dicho proceso, un dispositivo de almacenamiento con los archivos necesarios y los pasos a realizar. De manera manual se realiza la edición de varios ficheros en el caso de la configuración, y la ejecución de comandos necesarios para iniciar, detener, o reiniciar los servicios, asignar permisos, instalar programas, entre otras operaciones de administración. Estos procesos se realizan de manera física en el servidor.

2.2.3. Objeto de informatización.

Para efectuar la instalación o actualización de SIGEF existen numerosos pasos, que de forma manual resulta tediosa y en muchas ocasiones algo complicada, como instalar el sistema web, la base de datos, configurar los archivos necesarios y las salvas de la base de datos; todos ellos serán objeto de informatización pidiéndole solamente al usuario los datos necesarios.

2.2.4. Propuesta del sistema a desarrollar.

Para darle solución al problema planteado, el presente trabajo de diploma propone desarrollar una herramienta que brinde un conjunto de funcionalidades acordes al trabajo que se realiza actualmente de manera manual, con una interfaz gráfica sencilla, con la menor cantidad de pasos que lo permita, ejecutándose rápidamente y registrando los errores en caso de que ocurran.

2.2.5. Personal relacionado con el sistema.

Personal relacionado con el sistema	Descripción
Miembros del despliegue.	Es el personal que posee los permisos necesarios del sistema y de la fiscalía en cuestión para la instalación o actualización de SIGEF.

Tabla 2: Personal relacionado con el sistema.

2.2.6. Requisitos del sistema.

Los requisitos no son más que una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo, además tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. (27)

2.2.6.1. Requisitos funcionales.

No. RF	Descripción	Prioridad
RF1	Instalar programas para el funcionamiento del sistema web.	Alta
RF2	Copiar ficheros de configuración de Apache y PHP.	Baja
RF3	Descomprimir fichero del sistema web.	Media
RF4	Configurar parámetros de Base de Datos en el sistema web.	Media
RF5	Configurar parámetros del sistema web.	Baja
RF6	Instalar sistema web en el servidor.	Baja
RF7	Establecer permisos de propietario y grupo al sistema web.	Alta
RF8	Cambiar contraseñas de administrador y usuario de Postgres.	Alta
RF9	Crear roles de usuarios en la base de datos.	Media
RF10	Crear usuarios en la base de datos.	Media
RF11	Crear base de datos.	Alta
RF12	Cargar Backup en la base de datos creada.	Alta
RF13	Ejecutar funciones “CambiarSecuenciasBD” y “CambiarSecuenciasEspecialBD” en la base de datos.	Alta
RF14	Crear sistema de salvas de la base de datos.	Alta
RF15	Descargar actualización del sistema web.	Alta
RF16	Actualizar el sistema web.	Alta

Tabla 3: Requisitos funcionales.

2.2.6.2. Requisitos no funcionales.

Apariencia o interfaz

RNF1. Presentar una interfaz amigable que permita la fácil interacción con la herramienta.

RNF2. Posibilitar a los usuarios sin experiencia una rápida adaptación brindando la correcta e inequívoca información de la acción a realizar.

RNF3. Mostrar mensajes de errores en la introducción de datos de una forma sencilla y explicativa, la entrada de datos incorrecta será detectada claramente por la herramienta.

Usabilidad

RNF4. El software debe adaptarse al lenguaje y términos utilizados por los clientes, con vista a una mayor comprensión de la herramienta de trabajo.

Seguridad

RNF5. Las contraseñas no serán almacenadas.

RNF6. Confiabilidad: La información manejada por el sistema debe estar protegida de acceso no autorizado.

RNF7. Integridad: La información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción.

Fiabilidad

RNF8. Garantizar capacidad para capturar excepciones.

RNF9. Registrar errores: El software debe ser capaz de registrar los errores que ocurran durante su ejecución.

Software

RNF10. Sistema Operativo Ubuntu 10.10 o superior.

RNF11. Python 2.6 o superior.

Hardware

RNF12. Mínimo PC Pentium 3, 256 Mb de memoria Ram, 2GB de espacio libre en disco duro para poder ejecutar la herramienta.

2.2.7. Estimación de esfuerzos por historia de usuario.

Permite tener una medida de la velocidad de progreso del proyecto y brindan una guía a la cual ajustarse.

Historia de Usuario	Puntos de Estimación (semanas)
Instalar programas para el funcionamiento del sistema web.	1

Copiar ficheros de configuración de Apache y PHP.	0.5
Descomprimir fichero del sistema web.	0.5
Configurar parámetros de Base de Datos en el sistema web.	1
Configurar parámetros del sistema web.	1
Instalar sistema web en el servidor.	1
Establecer permisos de propietario y grupo al sistema web.	1
Cambiar contraseñas de administrador y usuario de Postgres.	1
Crear roles de usuarios en la base de datos.	0.5
Crear usuarios en la base de datos.	0.5
Crear base de datos.	0.5
Cargar Backup en la base de datos creada.	0.5
Ejecutar funciones “CambiarSecuenciasBD” y “CambiarSecuenciasEspecialBD” en la base de datos.	1
Crear sistema de salvas de la base de datos.	2
Descargar actualización del sistema web.	1
Actualizar el sistema web.	2
Total	**15**

Tabla 4: Estimación de esfuerzos por Historia de Usuario.

2.2.8. Plan de iteraciones:

Este plan define las HU que serán implementadas en cada iteración y las posibles fechas de sus entregas. Se identificaron tres iteraciones de acuerdo a las opciones que brindará la aplicación, las cuales se especifican a continuación:

Iteración 1: En esta iteración se implementan HU con alta, media y baja prioridad en el negocio, creando una de las funcionalidades del sistema. Estas HU son: 1, 2, 3, 4, 5, 6, 7. Además, se obtendrá la primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación.

Iteración 2: El objetivo de esta iteración es la implementación de la instalación de la base de datos, la cual posee funcionalidades con prioridad alta y media. Con la culminación de la misma se tendrán implementadas las HU 8, 9, 10, 11, 12, 13, 14. Al terminar la iteración se contará con otra versión de prueba del producto.

Iteración 3: El objetivo de la iteración está dado por la implementación de las funcionalidades restantes, las HU 15 y 16. Al terminar esta iteración se contará con la versión final del producto. A partir de este momento el sistema será puesto a prueba por un período de tiempo para evaluar el desempeño del mismo.

2.2.9. Plan de duración de las iteraciones:

El objetivo fundamental del plan de duración de las iteraciones es mostrar la duración de cada iteración y el orden en que serán implementadas las HU en cada una de éstas.

Iteraciones	Orden de las historias usuario a implementar	Duración de las iteraciones
Iteración 1	Instalar programas para el funcionamiento del sistema web.	6
	Copiar ficheros de configuración de Apache y PHP.	
	Descomprimir fichero del sistema web.	
	Configurar parámetros de Base de Datos en el sistema web.	
	Configurar parámetros del sistema web.	
	Instalar sistema web en el servidor.	
	Establecer permisos de propietario y grupo al sistema web.	
Iteración 2	Cambiar contraseñas de administrador y usuario de Postgres.	6
	Crear roles de usuarios en la base de datos.	
	Crear usuarios en la base de datos.	
	Crear base de datos.	
	Cargar Backup en la base de datos creada.	
	Ejecutar funciones “CambiarSecuenciasBD” y “CambiarSecuenciasEspecialBD” en la base de datos.	
	Crear sistema de salvas de la base de datos.	
Iteración 3	Descargar actualización del sistema web.	3
	Actualizar el sistema web.	
	Total	15

Tabla 5: Plan de duración de las iteraciones.

2.2.10. Plan de entregas.

El propósito del plan de entregas es establecer las HU que serán agrupadas para conformar una entrega, y el orden de las mismas. En este plan se unen las funcionalidades referentes a un mismo tema en módulos, esto permite un mayor entendimiento en la fase de implementación.

Módulos	Historias de usuario que abarca
Instalar sistema web	Instalar programas para el funcionamiento del sistema web.
	Copiar ficheros de configuración de Apache y PHP.
	Descomprimir fichero del sistema web.
	Configurar parámetros de Base de Datos en el sistema web.
	Configurar parámetros del sistema web.
	Instalar sistema web en el servidor.
	Establecer permisos de propietario y grupo al sistema web.
Instalar base de datos	Cambiar contraseñas de administrador y usuario de Postgres.
	Crear roles de usuarios en la base de datos.
	Crear usuarios en la base de datos.
	Crear base de datos.
	Cargar Backup en la base de datos creada.
	Ejecutar funciones “CambiarSecuenciasBD” y “CambiarSecuenciasEspecialBD” en la base de datos.
	Crear sistema de salvas de la base de datos.
Descargar y actualizar sistema web	Descargar actualización del sistema web.
	Actualizar el sistema web.

Tabla 6: Funcionalidades por módulos.

Módulos	Final 1ra Iteración 2da semana de Marzo	Final 2da Iteración 3ra semana de abril	Final 3ra Iteración 2da semana de mayo
Instalar sistema web	v1.0	v1.1 Final	Finalizado
Instalar base de			v1.4 Final

datos		v1.2	
Descargar y actualizar sistema web		v1.3	V1.5 Final

Tabla 7: Plan de duración de entrega.

2.3. Conclusiones del capítulo.

Durante el transcurso de este capítulo se expusieron las principales características del sistema y se desarrolló la fase Planificación que propone la metodología seleccionada concluyendo que:

- Se identifican y planifican las HU, quedando establecidas sus prioridades y las necesidades del cliente.
- Se realizó la estimación del esfuerzo necesario en cada una de las HU para tener una medida de la velocidad del progreso del proyecto.
- Se realizó el plan de duración de las iteraciones, mostrando la duración de cada iteración y el orden en que serán implementadas las HU.
- Se realizó el plan de entregas, agrupando las HU para conformar una entrega, y estableciendo el orden de las mismas.

3. Diseño, implementación y pruebas.

3.1. Introducción.

En el presente capítulo se describen las fases de diseño, implementación y pruebas de la metodología de desarrollo XP, generando los artefactos correspondientes como las tarjetas Clase, Responsabilidad y Colaboración, así como las tareas de programación desarrolladas en cada una de las iteraciones. También se hace referencia a las pruebas realizadas sobre el sistema para comprobar su funcionamiento.

3.2. Fase Diseño.

3.2.1. Tarjetas Clase, Responsabilidad y Colaboración (CRC).

Las tarjetas Clase, Responsabilidad y Colaboración (CRC) son utilizadas para representar las responsabilidades de las clases y sus interacciones. Permiten trabajar con una metodología basada en objetos, permitiendo que el equipo de desarrollo, contribuya en la tarea del diseño. Se definen las tarjetas CRC con la finalidad de obtener un diseño simple y no incurrir en la implementación de características que no son necesarias (ver tablas de tarjetas CRC en anexos). (28)

3.2.2. Patrones de diseño.

Un patrón de diseño es una solución estándar para un problema común de programación, una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios, un proyecto o estructura de implementación que logra una finalidad determinada. El uso de patrones ayuda a obtener un software de calidad (reutilización y extensibilidad). (29)

La representación puede observarla en la figura 3 en los anexos.

3.2.2.1. Patrones Gang of Four (GoF).

Los patrones de diseño como son descritos en el libro de los conocidos como Gang of Four están divididos en tres partes y cada una describe los patrones según el propósito. Aplicados al ambiente de programación de Python se utilizaron:

De creación: conciernen al proceso de creación de objetos.

- Solitario (Singleton):

Su utilidad es asegurar que una clase tiene una sola instancia y proporcionar un punto de acceso global a ella. Es necesario cuando hay clases que tienen que gestionar de manera centralizada un recurso, una variable global no garantiza que sólo se instancie una vez.

Es utilizado para mantener una única instancia de la clase Fachada la cual contiene los objetos de las clases InstalarWeb, InstalarBaseDatos, DescargarWeb y ActualizarWeb. En el caso de la implementación del patrón, sustituyó los constructores privados con un mecanismo de excepción y un método de clase con una función regular pero el patrón subyacente es claramente el Solitario. La naturaleza flexible y dinámica del lenguaje provee una buena base para una variedad de distintas y elegantes soluciones.

De estructura: tratan la composición de clases y objetos.

- Fachada (Facade):

Simplifica el acceso a un conjunto de clases, proporcionando una única clase que todos utilizan para comunicarse con dicho conjunto de clases. Los clientes no necesitan conocer las clases que hay tras la clase Fachada, se pueden cambiar las clases “ocultadas” sin necesidad de cambiar los clientes, sólo hay que realizar los cambios necesarios en la clase Fachada (29).

Se utiliza para que la clase InterfazPrincipal acceda a los métodos de las clases InstalarWeb, InstalarBaseDatos, DescargarWeb y ActualizarWeb mediante la clase Fachada, centralizando las operaciones para obtener un acceso único a ellos y más fácil, ya que los usuarios no necesitan conocer dichas clases, posibilitando realizar los cambios necesarios en la clase Fachada.

3.2.2.2. Patrones General Responsibility Assignment Software Patterns (GRASP).

En el desarrollo del sistema, se utilizan los patrones GRASP para la asignación de responsabilidades en el diseño de objetos.

- **Bajo Acoplamiento:** es utilizado para diseñar clases más independientes, que reduzcan el impacto de los cambios y sean más reutilizables.
- **Alta Cohesión:** es utilizado para que cada clase realice una labor única dentro del sistema.

- **Experto:** se utilizó para que cada objeto realizara la funcionalidad de acuerdo a la información que domina, por lo que aseguró que se le asigne determinada responsabilidad a la clase que mayor información posee para cumplir dicha tarea.
- **Controlador:** es utilizado en la clase Fachada para la centralización de actividades en el sistema.
- **Creador:** es utilizado para que sea creado el objeto de una clase por la que utilizará algún recurso de ella.

3.2.3. Estándar de codificación.

XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

El cumplimiento de esta tarea permite que las personas del equipo de SIGEF en futuras versiones puedan modificar cualquier parte del código, por lo que es imprescindible que el estilo de codificación sea consistente. Se listan distintas convenciones utilizadas en el código Python comprendido en la librería estándar de la distribución principal para el desarrollo de este proyecto.

3.2.3.1. Formateo del código.

Indentación: Usa 4 espacios por cada nivel de indentación.

Tabuladores o espacios: Se utiliza la forma más popular de indentar en Python sólo espacios. El código indentado con una mezcla de tabuladores y espacios no es aconsejable.

Tamaño máximo de línea Todas las líneas están limitadas a un máximo de 79 caracteres.

Todavía existen muchos dispositivos que están limitados a 80 caracteres por línea; además limitando el ancho de las ventanas a 80 caracteres posibilitas el tener varias ventanas una al lado de otra. El ajuste de línea por defecto en este tipo de dispositivos no da buenos resultados. Por lo tanto, se limitan todas las líneas a un máximo de 79 caracteres. Para cadenas de texto largas (cadenas de documentación o comentarios), se limitan a 72 caracteres. (30)

Ejemplo tomado de la clase InterfazPrincipal:

```
    def interfaz_instalar_bd(self):
        self.__cinterfaz.titulo("Instalar base de datos")
        modo = self.comprobar_repo()
        self.__fachada.instalar_programas_bd(modo)

        self.__cinterfaz.ventana_mensaje("""A continuación creará la \
contreña de postgres para linux:""", 6, 63)
        self.__fachada.cambiar_pass_pg_admin()
```

Líneas en blanco.

Separa las funciones no anidadas y las definiciones de clases con dos líneas en blanco. Las definiciones de métodos dentro de una misma clase se separan con una línea en blanco. Se usan líneas en blanco extras (de forma reservada) para separar grupos de funciones relacionadas. Las líneas en blanco se omiten entre un grupo de funciones con una sola línea (por ejemplo, con un conjunto de funciones sin implementación). Se usan líneas en blanco en las funciones, de forma limitada, para indicar secciones lógicas. (30)

3.2.3.2. Codificación de caracteres.

Todos los módulos descritos has sido usando la codificación UTF-8 lo que es especificado al intérprete en la segunda línea de todos los módulos.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

3.2.3.3. Módulos importados.

Los módulos importados han sido colocados en distintas líneas, por ejemplo:
Sí:

```
import os
import sys
```

No:

```
import sys, os
```

Los imports se han agrupado siguiendo el siguiente orden:

1. imports de la librería estándar
2. imports internos del paquete

3. imports de externos de paquetes relacionados se añade una línea en blanco después de cada grupo de imports.

Ejemplo tomado de la clase InstalarWeb:

```
import os
import subprocess
#internos
import c_programas
import c_operaciones_archivos
import c_operaciones_documentos
```

3.2.3.4. Espacios en blanco en expresiones y sentencias.

Se usan espacios en blanco extra de la siguiente forma:
Inmediatamente después de entrar en un paréntesis o antes de salir de un paréntesis, corchete o llave.

Sí: spam(ham[1], {eggs: 2})
No: spam(ham[1], { eggs: 2 })

Inmediatamente antes de una coma, punto y coma, o dos puntos:
Sí:
if x == 4: print x, y; x, y = y, x
No:
if x == 4 : print x , y ; x , y = y , x

Inmediatamente antes de abrir un paréntesis para una lista de argumentos de una llamada a una función:
Sí: spam(1)
No: spam (1)

Inmediatamente antes de abrir un paréntesis usado como índice o para particionar:
Sí: dict['key'] = list[index]
No: dict ['key'] = list [index]

Más de un espacio alrededor de un operador de asignación (u otro operador) para alinearlo con otro.

Sí:
```
x = 1
y = 2
long_variable = 3
```

No:
```
x             = 1
y             = 2
long_variable = 3
```

Se usa espacios alrededor de los operadores aritméticos:

Sí:
```
i = i + 1
submitted += 1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a - b)
```

No:
```
i=i+1
submitted +=1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
```

No se usan espacios alrededor del signo '=' cuando se use para indicar el nombre de un argumento o el valor de un parámetro por defecto.

Sí:
```
def complex(real, imag=0.0):
return magic(r=real, i=imag)
```

No:
```
def complex(real, imag = 0.0):
return magic(r = real, i = imag)
```

Aunque a veces es adecuado colocar un if/for/while con un cuerpo pequeño en la misma línea, nunca se hace para sentencias multi-cláusula.

Preferiblemente no:
```
if foo == 'blah': do_blah_thing()
for x in lst: total += x
while t < 10: t = delay()
```

Definitivamente no:

```
if foo == 'blah': do_blah_thing()

else: do_non_blah_thing()

try: something()

finally: cleanup()

do_one(); do_two(); do_three(long, argument,

                                    list, like, this)

if foo == 'blah': one(); two(); three()
```

3.2.3.5. Convenciones de nombres.

Estilos de nombrado:

- minusculas_con_guiones_bajos
- PalabrasEnMayusculas (CapWords o CamelCase -- llamado así por el parecido de las mayúsculas con las jorobas de los camellos). Algunas veces también se llaman StudlyCaps.

Nota: Cuando se usan abreviaturas en CapWords, se mantiene en mayúsculas todas las letras de la abreviatura. Por lo tanto HTTPServerError es mejor que HttpServerError.

- capitalizacionMezclada (se diferencia de PalabrasEnMayusculas porque la primera letra está en minúsculas)

Nombres de paquetes y módulos:

Los módulos tienen nombres cortos formados en su totalidad por letras minúsculas. Se utilizan guiones bajos en el nombre del módulo si mejora la legibilidad. Los paquetes Python también tienen nombres cortos formados de letras minúsculas.

Nombres de clases:

Los nombres de clases usan la convención CapWords.

Nombres de funciones:

Los nombres de funciones están en letras minúsculas, con palabras separadas mediante guiones bajos según sea necesario para mejorar la legibilidad.

Argumentos de funciones y métodos:

Se usa siempre 'self' como primer argumento de los métodos de instancia.

Nombres de métodos y variables de instancia:

Se utilizan las reglas de los nombres de funciones: minúsculas con palabras separadas por guiones bajos cuando es necesario para mejorar la legibilidad. Se usan guiones bajos al inicio sólo para métodos no públicos y variables de instancia. Para evitar colisiones de nombres con subclases, se utilizan dos guiones bajos al principio del nombre para invocar las reglas de planchado de nombres de Python. (30)

3.3. Fase Implementación.

3.3.1. Tareas de programación.

Las tareas de programación se realizan para detallar mejor las HU, facilitando el entendimiento en el proceso de implementación. Cada HU puede contener una o más tareas de programación en caso de necesitarla en dependencia de la complejidad de la funcionalidad a desarrollar. Para definirlas se cuenta con una plantilla que permite definir cada una de las actividades a las que están asociadas las HU (ver tablas de tareas de programación en anexos).

Tarea	
Número tarea: 1	**Número historia de usuario**: 1
Nombre tarea: Instalar programas para el funcionamiento del sistema web.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 01/02/12	**Fecha fin**: 04/02/12
Descripción: Comprueba si están instalados los programas, procediendo a instalarlos en caso negativo.	

Tabla 8: Representación de la tarea de programación: “Instalar programas para el funcionamiento del sistema web.”

Campos de la tarjeta tarea de programación:

- ✓ **Número historia de usuario:** Número de la historia de usuario a la que corresponde. Índice de la historia de usuario a la que se corresponde esta tarea de programación.
- ✓ **Número tarea:** Índice de la tarea de programación. Es un número único que se le asigna a cada tarea de programación que pertenece a una historia de usuario determinada con el fin de lograr una mejor organización de éstas.
- ✓ **Nombre tarea:** Nombre de la tarea de programación. Debe ser descriptivo, en la medida de las posibilidades, de lo que se realizará y no muy extenso.
- ✓ **Tipo de tarea:** Informa el tipo de tarea a realizar. Las tareas pueden ser:
  - **Desarrollo:** Tarea que se realizará por primera vez.
  - **Corrección:** Tarea que se realiza a partir de una anterior que no se realizó correctamente, es decir no pasó todos los casos de prueba que le corresponden correctamente.
  - **Mejora:** Tarea que se realiza a partir de una anterior que se realizó correctamente pero se incorporan nuevos requerimientos para la misma, ahora tendrá que ser modificada para pasar satisfactoriamente los nuevos casos de prueba adicionados.
  - **Otra:** Tarea que no corresponde con ninguna de las anteriormente descritas.
- ✓ **Fecha inicio:** Fecha en que se inicia la implementación de la tarea de programación.
- ✓ **Fecha fin:** Fecha en que concluye la implementación de la tarea de programación.
- ✓ **Descripción:** Se describe qué es lo que se desea realizar. La descripción debe ser corta y precisa.
- ✓ **Puntos estimados:** Tiempo estimado que demorará la implementación de la tarea de programación. Tiempo ideal en que se estima se implementará la tarea de programación.

3.3.1.1. Iteración 1.

Iteración	Historias de usuario
1	Instalar programas para el funcionamiento del sistema web.
	Copiar ficheros de configuración de Apache y PHP.
	Descomprimir fichero del sistema web.
	Configurar parámetros de Base de Datos en el sistema web.
	Configurar parámetros del sistema web.
	Instalar sistema web en el servidor.

	Establecer permisos de propietario y grupo al sistema web.

Tabla 9: Representación de las historias de usuario de la iteración 1.

3.3.1.2. Iteración 2.

Iteraciones	Orden de las historias usuario a implementar
2	Cambiar contraseñas de administrador y usuario de Postgres.
	Crear roles de usuarios en la base de datos.
	Crear usuarios en la base de datos.
	Crear base de datos.
	Cargar Backup en la base de datos creada.
	Ejecutar funciones “CambiarSecuenciasBD” y “CambiarSecuenciasEspecialBD” en la base de datos.
	Crear sistema de salvas de la base de datos.

Tabla 10: Representación de las historias de usuario de la iteración 2.

3.3.1.3. Iteración 3.

Iteración	Historias de usuario
3	Descargar actualización del sistema web.
	Actualizar el sistema web.

Tabla 11: Representación de las historias de usuario de la iteración 3.

3.4. Fase Pruebas.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias (pruebas de caja blanca), desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación (pruebas de caja negra), destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente realizando las pruebas sobre la interfaz del software. (25)

Los resultados generales pueden ser observados en la gráfica 1 (ver en anexos).

3.4.1. Pruebas unitarias.

Con las pruebas unitarias del software se comprueban los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado.

Debido a la propia filosofía de Python, es difícil encontrar tanto herramientas de prueba como depuradores. Al ser un lenguaje interpretado que no necesita compilarse ni enlazarse, se supone que el propio código (usando el intérprete) es todo lo necesario para probar y depurar un programa.

Dentro de la gama de este tipo de herramientas se encuentran varias de ellas como: PyUnit, Py.Test, TestOOB, NOSE, TRIAL y QUNITTEST, de las cuales las más factible fue PyUnit.

PyUnit es la herramienta oficial para hacer pruebas unitarias en Python. Se incluye en la librería estándar (en el módulo unittest) y ha sido creado por los desarrolladores de JUnit para facilitar la creación y gestión de tests de prueba en módulos Python. Es soportado perfectamente por el IDE Eclipse con el plugin PyDev, permite separar el código de pruebas del código del propio proyecto para mejorar la comprensión, refactorización y la facilidad para hacer cambios en el programa. Por otra parte, se pueden realizar los tests por línea de comandos.

Es capaz de realizar son pruebas de igualdad, excepciones, fallo, pruebas de cordura (comprobar que no se producen situaciones imposibles) y pruebas de mayúsculas (relacionadas con las cadenas de caracteres). Debido a todo esto, y a que se trata de la herramienta estándar de Python, tiene mucha difusión y existe mucha documentación sobre esta librería en la red.

Para poder hacer un caso de prueba propio en PyUnit es necesario hacer una subclase de "TestCase", que es un objeto que puede correr un único método de prueba. Para realizar una prueba de algo en Python se usa el comando "assert", si este comando falla cuando se ejecuta el caso de prueba PyUnit dará el caso por fallido, de lo contrario el caso el resultado es correcto. Para cada método se le hicieron la cantidad de pruebas necesarias para su correcta ejecución.

Ejemplo del método comprobar_rutavarwwwsgf cuando existe la ruta:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

```
import unittest
import c_instalar_web

class MisPruebas(unittest.TestCase):
    def runTest(self):
        self.__ciweb = c_instalar_web.InstalarWeb()
        self.assertEqual(self.__ciweb.comprobar_rutavarwwwsgf(), True)

testCase = MisPruebas()
runner = unittest.TextTestRunner()
runner.run(testCase)
```

Mostrándose en cada caso el resultado como se muestra en la figura 2 (ver anexos).

3.4.2. Pruebas de aceptación.

Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las HU. Durante las iteraciones las HU seleccionadas serán traducidas a pruebas de aceptación, en ellas se especifican desde la perspectiva del cliente, los escenarios para probar que una HU ha sido implementada correctamente. Una HU puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de éstas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable. Una HU no se considera completa hasta que no ha pasado por sus pruebas de aceptación. (25)

3.5. Conclusiones del capítulo.

Durante la elaboración del presente capítulo se expusieron los artefactos construidos como parte de las actividades de diseño, implementación y pruebas que propone la metodología XP concluyendo que:

- Se confeccionaron las tarjetas CRC como artefacto de la fase de diseño representando las responsabilidades de las clases y sus interacciones.
- Se realizaron las tareas de programación generadas por cada historia de usuario permitiendo obtener la herramienta en la fase de implementación.
- Se le aplicaron las pruebas que propone la metodología de desarrollo XP asegurando la calidad del producto.

Conclusiones generales.

Con la elaboración del presente trabajo se desarrollaron una serie de tareas para darle cumplimiento al objetivo general planteado que se derivan en:

- Se realizó un estudio de las principales tendencias a nivel internacional, lo cual permitió seleccionar el lenguaje de programación, la metodología, el entorno de desarrollo integrado y la interfaz gráfica más adecuada.
- Se realizó el diseño de la aplicación lo cual permitió el desarrollo de una herramienta para la instalación y actualización de SIGEF satisfaciendo las necesidades del cliente.
- Se logró la implementación de la herramienta comprobándose la hipótesis elaborada, ya que se informatizaron los procesos, mejorando considerablemente el tiempo de instalación y actualización de SIGEF, y los errores al realizar dicho proceso sin importar quien lo efectúe.
- Se validaron los resultados obtenidos a partir de la metodología seleccionada, lo cual asegura la calidad del producto.

Recomendaciones.

- ✓ Realizar mantenimiento de la herramienta propuesta para su utilización en futuras versiones de SIGEF.
- ✓ Incorporar la herramienta propuesta dentro del paquete de soluciones que se entregan a la fiscalía.

Referencias bibliográficas.

1. PC Magazine Encyclopedia. [En línea] [Citado el: 13 de 12 de 2011.] http://www.pcmag.com/encyclopedia_term/0%2C2542%2Ct%3Dinstall+program&i%3D45034%2C00.asp.

2. **Microsoft.** Descripción de la terminología estándar utilizada para describir las actualizaciones de software de Microsoft. [En línea] [Citado el: 13 de 12 de 2011.] http://support.microsoft.com/kb/824684/es.

3. **Definición.de.** Definición.de. [En línea] 2008-2011. [Citado el: 03 de 02 de 2012.] http://definicion.de/version/.

4. **Hooping.** Hooping. [En línea] 1995-2008. [Citado el: 03 de 02 de 2012.] http://www.hooping.net/glossary/aplicaciones-web-146.aspx.

5. **Ciberaula.** Ciberaula. [En línea] 2010. [Citado el: 2012 de 02 de 16.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

6. **Martínez, Rafael.** PostgreSQL. [En línea] 2009-2011. [Citado el: 08 de 02 de 2012.] http://www.postgresql.org.es/sobre_postgresql.

7. **Welling, Luke y Thomson, Laura.** *Desarrollo Web con PHP y MySQL.*

8. **BitRock, Support.** bitrock.com. *bitrock.com.* [En línea] http://support.bitrock.com/article/installbuilder-action-lists..

9. **Ingeman, Anders.** osalt.com. *osalt.com.* [En línea] [Citado el: 25 de 03 de 2012.] http://www.osalt.com/installjammer.

10. **Colunga Aurea, Mario Chirinos.** aurea-dt.com. [En línea] 12 de 10 de 2010. [Citado el: 22 de 02 de 2012.] http://www.aurea-dt.com/documentos/ADT_PaquetesDeb_Manual.pdf.

11. **Proyecto Pinguino.** Proyecto Pinguino. [En línea] [Citado el: 14 de 02 de 2012.] http://proyectopinguino.blogspot.com/2009/02/ejecutar-archivos-en-linux-bin-run-sh-y.html.

12. **Definición ABC.** Definición ABC. [En línea] 2007-2011. [Citado el: 03 de 02 de 2012.] http://www.definicionabc.com/general/script.php.

13. **Enciclopedia.** Enciclopedia libre en Español. *Enciclopedia libre en Español.* [En línea] [Citado el: 14 de 04 de 2012.] http://enciclopedia.us.es/index.php/Lenguaje_de_programaci%C3%B3n.

14. **Charte Ojeda, Francisco.** *Programación GNU/Linux.* s.l. : Anaya Multimedia, 2003. pág. 720. 9788441515444.

15. **Pérez Hidalgo, Andrey Francisco.** *Descripción de los aspectos fundamentales del lenguaje de programación PERL.* Universidad de Costa Rica, Escuela de Ciencias de la Computación e Informática. San José, Costa Rica : s.n., 2007. pág. 9.

16. **López, Angel.** *JAVA la programación del futuro.* Primera edición. Montevideo : s.n., 1997. 987-9131-38-X.

17. **García de Jalón, Javier, y otros.** *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2000. pág. 155.

18. **González Duque, Raúl.** *Python para todos.* s.l. : Creative Commons V2.5. pág. 160.

19. **Mazzarri S., Milton R.** SlideShare. [En línea] 2010. [Citado el: 30 de 01 de 2012.] http://slideshare.net/doknos/qu-es-python.

20. **van Rossum, Guido.** *Guía de aprendizaje de Python.* [ed.] Jr, Fred L. Drake. 2005. pág. 118. Vol. Release 2.4.1a0.

21. **Marzal, Andrés y Gracia, Isabel.** *Introducción a la programación con Python.* 2003. pág. 399.

22. **Collins-Sussman, Ben, W. Fitzpatrick, Brian y Michael Pilato, C.** Control de versiones Open Source de siguiente generación. [En línea] 2002-2011. http://svnbook.red-bean.com/nightly/en/index.html.

23. **Ubuntu.** Guía documentada para Ubuntu. [En línea] 2011. [Citado el: 08 de 02 de 2012.] http://www.guia-ubuntu.org/index.php?title=Eclipse.

24. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Pearson Addison-Wesley, Madrid., 2000. pág. 464. Vol. 1. 9788478290369.

25. **Beck, Kent.** *Extreme Programming Explained.* Primera Edición. 1999. pág. 224. 0201616416.

26. **H. Canós, José , Letelier, Patricio y Penadés, María Carmen.** *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n. 46022.

27. **Pressman, Roger S.** *Ingeniería del Software.* [ed.] Darrel Ince. Quinta. s.l. : Mc Graw Hill. pág. 601.

28. **Mestras, P J.** [En línea] 2009. [Citado el: 26 de 03 de 2012.] http://www.fdi.ucm.es.

29. **MonoNeurona.** MonoNeurona. *MonoNeurona.* [En línea] 2002-2012. [Citado el: 26 de 04 de 2012.] http://www.MonoNeurona.org.

30. **González Duque, Raúl.** Guía de estilo del código Python. [En línea] [Citado el: 14 de 04 de 2012.] http://mundogeek.net/traducciones/guia-estilo-python.htm.

Glosario de términos.

A continuación en orden alfabético, se muestra el significado de algunos términos usados en el documento cuyo uso no es común y puede dificultar la compresión del mismo:

GUI: es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

Hardware: Corresponde a todas las partes físicas y tangibles de una computadora.

HTTP: Protocolo de Transferencia de Hipertexto. Modo de comunicación para solicitar páginas web.

HTML (Lenguaje de Marcado de Hipertexto): Lenguaje de marcado predominante para la construcción de páginas web.

HU: Historias de usuario.

Multihilo: Varios caminos, en el ámbito informático, varias vías de ejecución.

Plugin: “Parche” para un programa que le añade características nuevas.

Software: Se refiere al equipamiento lógico o soporte lógico de una computadora digital.

UCI: Universidad de las Ciencias Informáticas.

Unix: Sistema operativo portable, multitarea y multiusuario; desarrollado en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McElroy.

Web: World Wide Web (también conocida como "la Web"), el sistema de documentos (o páginas web) interconectados por enlaces de hipertexto, disponibles en Internet.

Anexos.

Figuras.

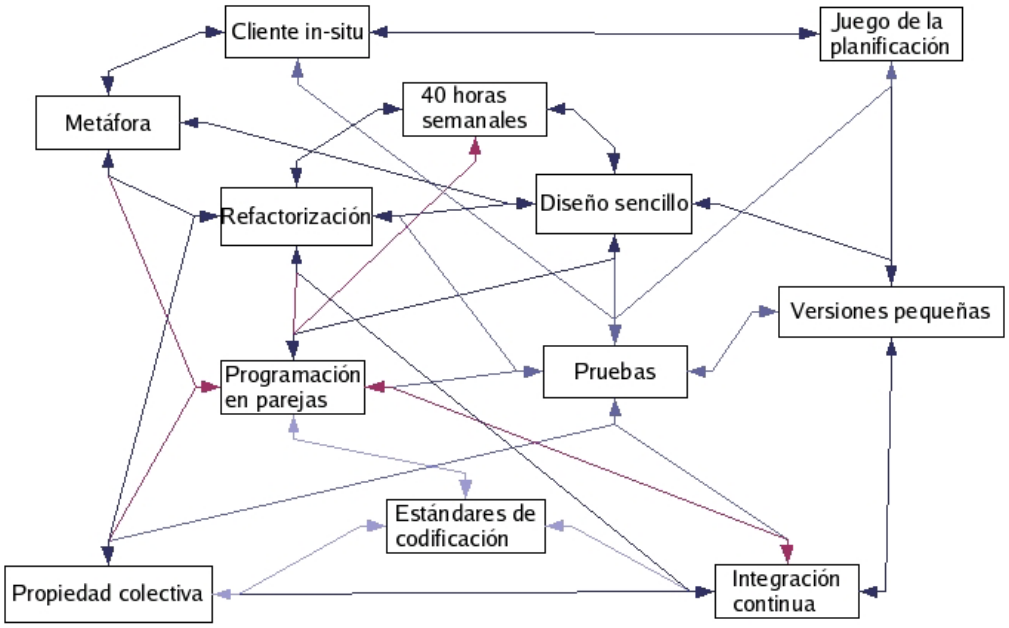

Figura 1. Relación de todas las prácticas.

```
glc@glc-pc:~/workspace/ISIGEF$ sudo src/./pruebaunitaria.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
glc@glc-pc:~/workspace/ISIGEF$
```

Figura 2. Resultado de prueba unitaria.

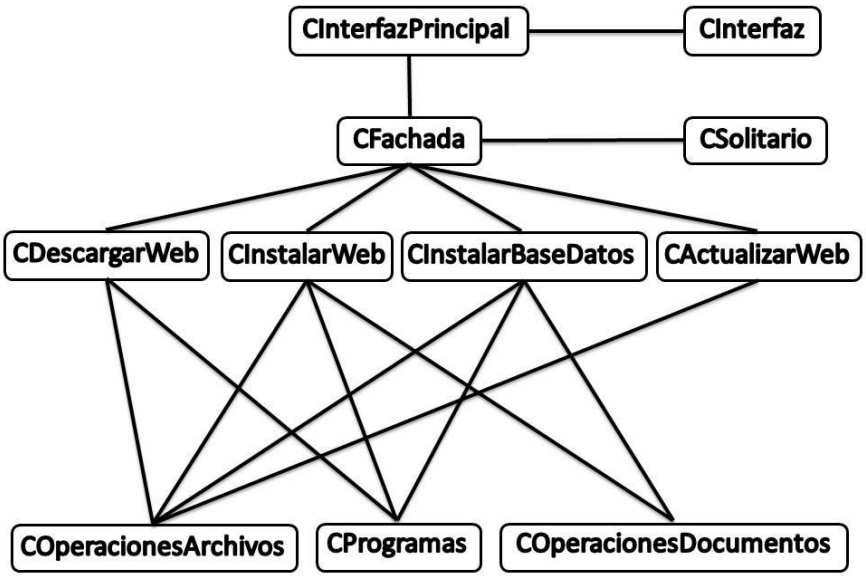

Figura 3. Representación de patrón Fachada aplicado a la herramienta.

Tablas de historias de usuario.

Historia de Usuario	
Número: 1	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Instalar programas para el funcionamiento del sistema web.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Puntos estimados: 1	**Iteración Asignada:** 1
Descripción: Inicia cuando el usuario instala el sistema web o la base de datos.	
Observaciones:	

Tabla 12: Representación de la Historia de Usuario No.1

Historia de Usuario	
Número: 2	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Copiar ficheros de configuración de Apache y PHP.	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Bajo
Puntos estimados: 1	**Iteración Asignada:** 1
Descripción: Inicia cuando el usuario instala el sistema web justo después de la HU1 copiando los archivos de configuración en las direcciones predeterminadas.	
Observaciones: Los ficheros deben existir en la ubicación predeterminada.	

Tabla 13: Representación de la Historia de Usuario No.2

Historia de Usuario	
Número: 3	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Descomprimir fichero del sistema web.	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 1	**Iteración Asignada:** 1
Descripción: Inicia cuando el usuario instala el sistema web justo después de la HU2 descomprimiéndose el fichero que contiene el sistema web.	
Observaciones: El fichero debe existir en la ubicación predeterminada.	

Tabla 14: Representación de la Historia de Usuario No.3

Historia de Usuario	
Número: 4	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Configurar parámetros de Base de Datos en el sistema web.	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 1	**Iteración Asignada:** 1
Descripción: Inicia cuando el usuario instala el sistema web justo después de la HU3 modificando el fichero que contiene la información de conexión a la base de datos pidiendo los datos necesarios.	
Observaciones: El fichero que contiene el sistema web debe haberse descomprimido correctamente.	

Tabla 15: Representación de la Historia de Usuario No.4

Historia de Usuario	
Número: 5	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Configurar parámetros del sistema web.	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Bajo
Puntos estimados: 1	**Iteración Asignada:** 1
Descripción: Inicia cuando el usuario instala el sistema web justo después de la HU4 modificando el fichero que contiene la información del sistema pidiendo los datos necesarios.	
Observaciones: El fichero que contiene el sistema web debe haberse descomprimido correctamente.	

Tabla 16: Representación de la Historia de Usuario No.5

Historia de Usuario	
Número: 6	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Instalar sistema web en el servidor.	
Prioridad en Negocio: Baja	**Prioridad en Negocio:** Baja
Puntos estimados: 1	**Puntos estimados:** 1
Descripción: Inicia cuando el usuario instala el sistema web justo después de la HU5 moviendo el directorio descomprimido del sistema web hacia el directorio del servidor web predeterminado.	
Observaciones: El fichero que contiene el sistema web debe haberse descomprimido correctamente.	

Tabla 17: Representación de la Historia de Usuario No.6

Historia de Usuario	
Número: 7	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Establecer permisos de propietario y grupo al sistema web.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 1	**Iteración Asignada:** 1
Descripción: Inicia cuando el usuario instala el sistema web justo después de la HU6 asignando los permisos de propietario y grupo, necesarios para el correcto funcionamiento del sistema.	
Observaciones:	

Tabla 18: Representación de la Historia de Usuario No.7

Historia de Usuario	
Número: 8	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Cambiar contraseñas de administrador y usuario de Postgres.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 1	**Iteración Asignada:** 2
Descripción: Inicia cuando el usuario instala la base de datos pidiéndole al usuario que ingrese la nueva contraseña.	
Observaciones:	

Tabla 19: Representación de la Historia de Usuario No.8

Historia de Usuario	
Número: 9	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Crear roles de usuarios en la base de datos.	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 1	**Iteración Asignada:** 2
Descripción: Inicia cuando el usuario instala la base de datos justo después de la HU8.	
Observaciones:	

Tabla 20: Representación de la Historia de Usuario No.9

Historia de Usuario	
Número: 10	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Crear usuarios en la base de datos.	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 1	**Iteración Asignada:** 2
Descripción: Inicia cuando el usuario instala la base de datos justo después de la HU9.	
Observaciones:	

Tabla 21: Representación de la Historia de Usuario No.10

Historia de Usuario	
Número: 11	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Crear base de datos.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 1	**Iteración Asignada:** 2
Descripción: Inicia cuando el usuario instala la base de datos justo después de la HU10.	
Observaciones:	

Tabla 22: Representación de la Historia de Usuario No.11

Historia de Usuario	
Número: 12	**Usuario:** Miembro del despliegue

Nombre Historia de Usuario: Cargar Backup en la base de datos creada.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Puntos estimados: 1	**Iteración Asignada:** 2
Descripción: Inicia cuando el usuario instala la base de datos justo después de la HU11.	
Observaciones: El fichero debe existir en la ubicación predeterminada.	

Tabla 23: Representación de la Historia de Usuario No.12

Historia de Usuario	
Número: 13	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Ejecutar funciones "CambiarSecuenciasBD" y "CambiarSecuenciasEspecialBD" en la base de datos.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 1	**Iteración Asignada:** 2
Descripción: Inicia cuando el usuario instala la base de datos justo después de la HU12 pidiéndole al usuario el número de la fiscalía.	
Observaciones:	

Tabla 24: Representación de la Historia de Usuario No.13

Historia de Usuario	
Número: 14	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Crear sistema de salvas de la base de datos.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 2	**Iteración Asignada:** 2
Descripción: Inicia cuando el usuario instala la base de datos justo después de la HU13 pidiéndole al usuario los datos de configuración.	
Observaciones:	

Tabla 25: Representación de la Historia de Usuario No.14

Historia de Usuario	
Número: 15	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Descargar actualización del sistema web.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 1	**Iteración Asignada:** 3

Descripción: El sistema descarga la actualización del servidor pidiéndole al usuario la dirección.
Observaciones:

Tabla 26: Representación de la Historia de Usuario No.15

Historia de Usuario	
Número: 16	**Usuario:** Miembro del despliegue
Nombre Historia de Usuario: Actualizar el sistema web.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 2	**Iteración Asignada:** 2
Descripción: El sistema actualiza el sistema web guardando la configuración y copiando los ficheros nuevos.	
Observaciones:	

Tabla 27: Representación de la Historia de Usuario No.16

Tablas de tarjetas CRC.

Clase: Interfaz	
Responsabilidad	Colaboración
Brindar ventanas a mostrar	

Tabla 28: Tarjeta CRC Clase interfaz.

Clase: InterfazPrincipal	
Responsabilidad	Colaboración
Mostrar la interfaz principal con las operaciones a realizar	Interfaz Fachada

Tabla 29: Tarjeta CRC Clase InterfazPrincipal.

Clase: Fachada	
Responsabilidad	Colaboración
Controla todas las operaciones que se realizan para realizar las funcionalidades que brinda la herramienta.	InstalarWeb InstalarBaseDatos DescargarWeb

	ActualizarWeb

Tabla 30: Tarjeta CRC Clase Fachada.

Clase: Programas	
Responsabilidad	Colaboración
Realiza las operaciones de instalación y comprobación de los programas.	

Tabla 31: Tarjeta CRC Clase Programas.

Clase: OperacionesDocumentos	
Responsabilidad	Colaboración
Realiza las operaciones relacionadas con la edición de documentos.	

Tabla 32: Tarjeta CRC Clase OperacionesDocumentos.

Clase: OperacionesArchivos	
Responsabilidad	Colaboración
Realiza las operaciones relacionadas con archivos.	

Tabla 33: Tarjeta CRC Clase OperacionesArchivos.

Clase: InstalarWeb	
Responsabilidad	Colaboración
Realiza las operaciones relacionadas con la instalación del sistema web.	OperacionesArchivos OperacionesDocumentos Programas

Tabla 34: Tarjeta CRC Clase InstalarWeb.

Clase: InstalarBaseDatos	
Responsabilidad	Colaboración
Realiza las operaciones relacionadas con la instalación de la base de datos.	OperacionesArchivos OperacionesDocumentos Programas

Tabla 35: Tarjeta CRC Clase InstalarBaseDatos.

Clase: ActualizarWeb	
Responsabilidad	Colaboración
Realiza las operaciones relacionadas con la actualización del sistema web.	OperacionesArchivos

Tabla 36: Tarjeta CRC Clase ActualizarWeb.

Clase: DescargarWeb	
Responsabilidad	Colaboración
Realiza las operaciones relacionadas con la descarga del sistema web del repositorio.	OperacionesArchivos Programas

Tabla 37: Tarjeta CRC Clase DescargarWeb.

Tablas de tareas de programación.

Iteración 1.

Tarea	
Número tarea: 1	**Número historia de usuario**: 1
Nombre tarea: Instalar programas para el funcionamiento del sistema web.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 01/02/12	**Fecha fin**: 04/02/12
Descripción: Comprueba si están instalados los programas, procediendo a instalarlos en caso negativo.	

Tabla 38: Representación de la tarea de programación: “Instalar programas para el funcionamiento del sistema web.”

Tarea	
Número tarea: 2	**Número historia de usuario**: 2
Nombre tarea: Copiar ficheros de configuración de Apache y PHP.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 0.5
Fecha inicio: 06/02/12	**Fecha fin**: 08/02/12

Descripción: Se copian los ficheros a sus respectivos destinos.

Tabla 39: Representación de la tarea de programación: "Copiar ficheros de configuración de Apache y PHP."

Tarea	
Número tarea: 3	**Número historia de usuario**: 3
Nombre tarea: Descomprimir fichero del sistema web.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 0.5
Fecha inicio: 09/02/12	**Fecha fin**: 11/02/12
Descripción: Se descomprime el sistema web en el directorio predeterminado.	

Tabla 40: Representación de la tarea de programación: "Descomprimir fichero del sistema web."

Tarea	
Número tarea: 4	**Número historia de usuario**: 4
Nombre tarea: Configurar parámetros de Base de Datos en el sistema web.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 13/02/12	**Fecha fin**: 18/02/12
Descripción: Se modifica el archivo con los datos pedidos al usuario.	

Tabla 41: Representación de la tarea de programación: "Configurar parámetros de Base de Datos en el sistema web."

Tarea	
Número tarea: 5	**Número historia de usuario**: 5
Nombre tarea: Configurar parámetros del sistema web.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 20/02/12	**Fecha fin**: 25/02/12
Descripción: Se modifica el archivo con los datos pedidos al usuario.	

Tabla 42: Representación de la tarea de programación: "Configurar parámetros del sistema web."

Tarea	
Número tarea: 6	**Número historia de usuario**: 6

Nombre tarea: Instalar sistema web en el servidor.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 27/02/12	**Fecha fin**: 03/03/12
Descripción: Se mueve el sistema web hacia el directorio de publicación del servidor web.	

Tabla 43: Representación de la tarea de programación: "Instalar sistema web en el servidor."

Tarea	
Número tarea: 7	**Número historia de usuario**: 7
Nombre tarea: Establecer permisos de propietario y grupo al sistema web.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 05/03/12	**Fecha fin**: 10/03/12
Descripción: Se establecen los permisos al sistema web.	

Tabla 44: Representación de la tarea de programación: "Establecer permisos de propietario y grupo al sistema web."

Iteración 2.

Tarea	
Número tarea: 1	**Número historia de usuario**: 8
Nombre tarea: Cambiar contraseñas de administrador y usuario de Postgres.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 12/03/12	**Fecha fin**: 17/03/12
Descripción: El sistema le pide al usuario que introduzca la contraseña de Postgres para cada tipo de usuario.	

Tabla 45: Representación de la tarea de programación: "Cambiar contraseñas de administrador y usuario de Postgres."

Tarea	
Número tarea: 2	**Número historia de usuario**: 9
Nombre tarea: Crear roles de usuarios en la base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 0.5
Fecha inicio: 19/03/12	**Fecha fin**: 21/03/12

Descripción: Se crean los roles de los usuarios.

Tabla 46: Representación de la tarea de programación: "Crear roles de usuarios en la base de datos."

Tarea	
Número tarea: 3	**Número historia de usuario**: 10
Nombre tarea: Crear usuarios en la base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 0.5
Fecha inicio: 22/03/12	**Fecha fin**: 24/03/12
Descripción: Se crean los usuarios del sistema.	

Tabla 47: Representación de la tarea de programación: "Crear roles de usuarios en la base de datos."

Tarea	
Número tarea: 4	**Número historia de usuario**: 11
Nombre tarea: Crear base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 0.5
Fecha inicio: 26/03/12	**Fecha fin**: 28/03/12
Descripción: Se crea la base de datos pidiéndole al usuario los datos correspondientes.	

Tabla 48: Representación de la tarea de programación: "Crear base de datos."

Tarea	
Número tarea: 5	**Número historia de usuario**: 12
Nombre tarea: Cargar Backup en la base de datos creada.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 0.5
Fecha inicio: 29/03/12	**Fecha fin**: 31/03/12
Descripción: Se restaura el backup en la base de datos creada.	

Tabla 49: Representación de la tarea de programación: "Cargar Backup en la base de datos creada."

Tarea	
Número tarea: 6	**Número historia de usuario**: 13

Nombre tarea: Ejecutar funciones “CambiarSecuenciasBD” y “CambiarSecuenciasEspecialBD” en la base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 02/04/12	**Fecha fin**: 07/04/12
Descripción: El sistema le muestra las fiscalías al usuario para que realice la selección y ejecutar las funciones.	

Tabla 50: Representación de la tarea de programación: “Ejecutar funciones “CambiarSecuenciasBD” y “CambiarSecuenciasEspecialBD” en la base de datos.”

Tarea	
Número tarea: 7	**Número historia de usuario**: 14
Nombre tarea: Crear sistema de salvas de la base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 09/04/12	**Fecha fin**: 21/04/12
Descripción: El sistema le pide al usuario los datos correspondientes para configurar las salvas de la base de datos.	

Tabla 51: Representación de la tarea de programación: “Crear sistema de salvas de la base de datos.”

Iteración 3.

Tarea	
Número tarea: 1	**Número historia de usuario**: 15
Nombre tarea: Descargar actualización del sistema web.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 23/04/12	**Fecha fin**: 28/04/12
Descripción: El sistema le pide al usuario que introduzca la dirección del repositorio, descarga la actualización del sistema web y lo comprime.	

Tabla 52: Representación de la tarea de programación: “Descargar actualización del sistema web.”

Tarea	
Número tarea: 2	**Número historia de usuario**: 16
Nombre tarea: Actualizar el sistema web.	

Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 30/04/12	**Fecha fin**: 12/05/12
Descripción: El sistema descomprime el sistema web, guarda la configuración del que se encuentra instalado, lo elimina y copia el nuevo.	

Tabla 53: Representación de la tarea de programación: "Actualizar el sistema web."

Tablas de casos de prueba de aceptación.

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario**: Instalar programas para el funcionamiento del sistema web.
Nombre: Comprobar que estén instalados los programas necesarios.	
Descripción: Prueba para verificar si detecta los programas que necesita el sistema web para su ejecución.	
Condiciones de Ejecución: Los programas no deben estar instalados.	
Entrada/Pasos de ejecución: El usuario selecciona la opción Instalar sistema web o Instalar Base de Datos.	
Resultados Esperados: El sistema muestra un listado con los programas que necesita instalar.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 54: Caso de prueba 1 HU1_P1

Caso de Prueba de Aceptación	
Código: HU1_P2	**Historia de Usuario**: Instalar programas para el funcionamiento del sistema web.
Nombre: Instalar programas para el funcionamiento del sistema web.	
Descripción: Instala los programas para el funcionamiento del sistema web.	
Condiciones de Ejecución: Los programas no deben estar instalados.	
Entrada/Pasos de ejecución: El usuario selecciona la opción Instalar sistema web o Instalar Base de Datos.	
Resultados Esperados: El sistema instala los programas.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 55: Caso de prueba 2 HU1_P2

Caso de Prueba de Aceptación	
Código: HU2_P1	**Historia de Usuario**: Copiar ficheros de configuración de Apache y PHP.
Nombre: Copiar ficheros de configuración de Apache y PHP.	
Descripción: El sistema copia los archivos con las configuraciones necesarias hechas que se encuentran en el directorio de la aplicación hacia el directorio predeterminado donde se encuentran los archivos del mismo nombre.	
Condiciones de Ejecución: Los archivos deben existir y estar en la ubicación predeterminada.	
Entrada/Pasos de ejecución: El usuario selecciona la opción "Instalar Sistema Web" en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 56: Caso de prueba 3 HU2_P1

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario**: Descomprimir fichero del sistema web.
Nombre: Descomprimir fichero del sistema web.	
Descripción: El sistema descomprime el sistema web que se encuentra ubicado en el directorio de la aplicación.	
Condiciones de Ejecución: El fichero debe existir y estar en la ubicación predeterminada.	
Entrada/Pasos de ejecución: El usuario selecciona la opción "Instalar Sistema Web" en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 57: Caso de prueba 4 HU3_P1

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de Usuario**: Configurar parámetros de Base de Datos en el sistema web.
Nombre: Configurar parámetros de Base de Datos en el sistema web.	
Descripción: La aplicación le pide al usuario los datos necesarios para configurar la base de datos, modificándolos en el archivo que contiene dicha información en el sistema web.	
Condiciones de Ejecución: El archivo que contiene la configuración del sistema web debe existir y estar en la ubicación predeterminada al descomprimirse el sistema.	

Entrada/Pasos de ejecución: El usuario selecciona la opción “Instalar Sistema Web” en el menú de la pantalla principal.
Resultados Esperados: Se muestra el progreso de la instalación.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 58: Caso de prueba 5 HU4_P1

Caso de Prueba de Aceptación	
Código: HU5_P1	**Historia de Usuario**: Configurar parámetros del sistema web.
Nombre: Configurar parámetros del sistema web.	
Descripción: La aplicación le pide al usuario los datos necesarios para configurar el archivo que contiene la información del sistema web.	
Condiciones de Ejecución: El archivo que contiene la configuración del sistema web debe existir y estar en la ubicación predeterminada al descomprimirse el sistema.	
Entrada/Pasos de ejecución: El usuario selecciona la opción “Instalar Sistema Web” en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 59: Caso de prueba 6 HU5_P1

Caso de Prueba de Aceptación	
Código: HU6_P1	**Historia de Usuario**: Instalar sistema web en el servidor.
Nombre: Instalar sistema web en el servidor.	
Descripción: El directorio del sistema web se mueve hacia el servidor web.	
Condiciones de Ejecución: El fichero que contiene el sistema web comprimido debe haberse descomprimido exitosamente.	
Entrada/Pasos de ejecución: El usuario selecciona la opción “Instalar Sistema Web” en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 60: Caso de prueba 7 HU6_P1

Caso de Prueba de Aceptación	
Código: HU7_P1	**Historia de Usuario**: Establecer permisos de propietario y grupo al sistema web.

Nombre: Establecer permisos de propietario y grupo al sistema web.
Descripción: Se le asignan los permisos necesarios al directorio del sistema web.
Condiciones de Ejecución: El directorio debe haberse movido al directorio del servidor web.
Entrada/Pasos de ejecución: El usuario selecciona la opción “Instalar Sistema Web” en el menú de la pantalla principal.
Resultados Esperados: Se muestra el progreso de la instalación.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 61: Caso de prueba 8 HU7_P1

Caso de Prueba de Aceptación	
Código: HU8_P1	**Historia de Usuario**: Cambiar contraseñas de administrador y usuario de Postgres.
Nombre: Cambiar contraseñas de administrador y usuario de Postgres.	
Descripción: El sistema le pide al usuario que introduzca la nueva contraseña de UNIX en el caso del administrador, en el caso del usuario se le muestra la consulta que debe ejecutar.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: El usuario selecciona la opción “Instalar Base de Datos” en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 62: Caso de prueba 9 HU8_P1

Caso de Prueba de Aceptación	
Código: HU9_P1	**Historia de Usuario**: Crear roles de usuarios en la base de datos.
Nombre: Crear roles de usuarios en la base de datos.	
Descripción: El sistema crea los roles.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: El usuario selecciona la opción “Instalar Base de Datos” en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 63: Caso de prueba 10 HU9_P1

Caso de Prueba de Aceptación

Código: HU10_P1	**Historia de Usuario**: Crear usuarios en la base de datos.
Nombre: Crear usuarios en la base de datos.	
Descripción: El sistema crea los usuarios en la base de datos pidiendo la contraseña en el caso del que lo necesite.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: El usuario selecciona la opción "Instalar Base de Datos" en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 64: Caso de prueba 11 HU10_P1

Caso de Prueba de Aceptación	
Código: HU11_P1	**Historia de Usuario**: Crear base de datos.
Nombre: Crear base de datos.	
Descripción: El sistema crea la base de datos pidiéndole al usuario el nombre.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: El usuario selecciona la opción "Instalar Base de Datos" en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 65: Caso de prueba 12 HU11_P1

Caso de Prueba de Aceptación	
Código: HU12_P1	**Historia de Usuario**: Cargar Backup en la base de datos creada.
Nombre: Cargar Backup en la base de datos creada.	
Descripción: El sistema carga el backup en la base de datos creada.	
Condiciones de Ejecución: El backup debe existir en la ubicación predeterminada y la base de datos debe haberse creado.	
Entrada/Pasos de ejecución: El usuario selecciona la opción "Instalar Base de Datos" en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 66: Caso de prueba 13 HU12_P1

Caso de Prueba de Aceptación	
Código: HU13_P1	**Historia de Usuario**: Ejecutar funciones “CambiarSecuenciasBD” y “CambiarSecuenciasEspecialBD” en la base de datos.
Nombre: Ejecutar funciones “CambiarSecuenciasBD” y “CambiarSecuenciasEspecialBD” en la base de datos.	
Descripción: El usuario debe seleccionar la fiscalía a la que pertenece y el sistema ejecuta dichas funciones con el número de la fiscalía seleccionada.	
Condiciones de Ejecución: El usuario debe seleccionar la fiscalía a la que pertenece.	
Entrada/Pasos de ejecución: El usuario selecciona la opción “Instalar Base de Datos” en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 67: Caso de prueba 14 HU13_P1

Caso de Prueba de Aceptación	
Código: HU14_P1	**Historia de Usuario**: Crear sistema de salvas de la base de datos.
Nombre: Crear sistema de salvas de la base de datos.	
Descripción: El sistema le pide al usuario la fecha y la hora en que deben ser creadas.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: El usuario selecciona la opción “Instalar Base de Datos” en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 68: Caso de prueba 15 HU14_P1

Caso de Prueba de Aceptación	
Código: HU15_P1	**Historia de Usuario**: Descargar actualización del sistema web.
Nombre: Descargar actualización del sistema web.	
Descripción: El sistema le pide al usuario la dirección del repositorio.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: El usuario selecciona la opción “Descargar actualización del sistema web” en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la instalación.	

Evaluación de la Prueba: Prueba satisfactoria.

Tabla 69: Caso de prueba 16 HU15_P1

Caso de Prueba de Aceptación	
Código: HU16_P1	**Historia de Usuario**: Actualizar el sistema web.
Nombre: Actualizar el sistema web.	
Descripción: El sistema salva la configuración del sistema web instalado, lo elimina y copia el nuevo con la configuración guardada.	
Condiciones de Ejecución: El sistema web instalado debe existir.	
Entrada/Pasos de ejecución: El usuario selecciona la opción “Actualizar el sistema web” en el menú de la pantalla principal.	
Resultados Esperados: Se muestra el progreso de la descarga.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 70: Caso de prueba 17 HU16_P1

Gráficos.

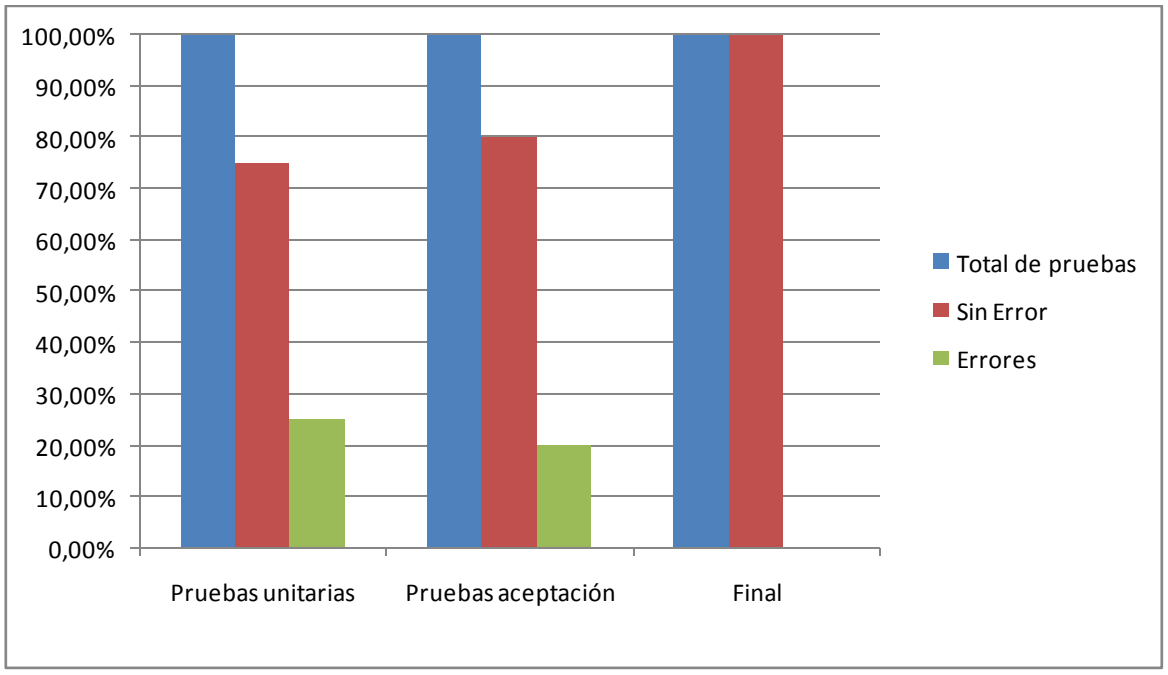

Gráfico 1. Resultados de pruebas.