Universidad de las Ciencias Informáticas

Facultad 1

Plataforma de Instalación para el Sistema de Administración de Identidades.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Greysi Daynielis Villavicencio Del Sol

José Miguel Ruiz Elizondo.

Tutores: Ing. María Lourdes Morilla Faurés.

Ing. Yendry Machado García.

Ciudad de La Habana, Junio de 2013.

"Año 55 de la Revolución".

"El mérito verdadero es el que el hombre adquiere con su voluntad, con su esfuerzo, con su constancia".

Fidel Castro Ruz

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas y a la Facultad 1 los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los____ días del mes de _____ del año 2013.

Greysi Daynielis Villavicencio Del Sol

_____

(Autora)

José Miguel Ruiz Elizondo

_____

(Autor)

María Lourdes Morilla Faurés

_____

(Tutora)

Yendry Machado García

_____

(Tutor)

Greysi

- ✓ *Dedico este trabajo a toda mi familia por todo su apoyo y confianza en mí, por ayudarme siempre en todo lo que he necesitado.*
- ✓ *A mi mamita del alma que ha sido mi guía y apoyo toda la vida, la que con sus consejos ha iluminado mi camino, siempre por el buen sendero, por ser la mejor madre del mundo y enseñarme a no rendirme nunca, por darme todo lo que soy.*
- ✓ *A mi abuela Nélida que la adoro con toda mi vida, por aconsejarme, darme su amor, cariño, comprensión, la que me complace y con su ejemplo a guiado mi forma de ser, mi carácter, y a ser fuerte ante las adversidades.*
- ✓ *A la memoria de mi abuelito que hubiese estado muy orgulloso de mí.*
- ✓ *A mi tío Joseito que no tengo palabras para agradecer lo maravilloso que es conmigo, por enseñarme el valor de la familia, la nobleza de ayudar a los demás y el espíritu emprendedor de seguir siempre adelante, es para mí un ejemplo a seguir por todas sus cualidades que son innumerables.*
- ✓ *A mi papá que ha sido la persona que más cerca he tenido en estos cinco años y siempre ha estado presente cuando lo he necesitado, dándome su apoyo incondicional, su cariño y confianza en todos los momentos, lo admiro mucho y es un extraordinario papá.*
- ✓ *A mi tiatiita, a Yaya, a mi tía Osmara y Mileidis, a mi abuela Bertha que siempre se preocupan por mí, aconsejándome y me dan su apoyo en mis decisiones. A mis bisabuelos Fela y Antonio que los adoro mucho.*
- ✓ *A mis hermanitas que las quiero mucho y me tienen como un ejemplo a seguir.*
- ✓ *A todas mis primas y primos que las adoro y tienen un lugar especial en mi corazón.*
- ✓ *A mis amigas de la Universidad que hemos pasado tantos momentos buenos y malos y siempre han estado conmigo cuando las he necesitado, a Jessica que ha sido para mí una amiga excepcional, siempre apoyándome en todo y dándome buenos consejos, a Yanet, Lianet, Daylis, Yanelis, Islén que todas han sido incondicionales conmigo.*
- ✓ *A mis tutores María Lourdes y Yendry que nos han ayudado mucho en la realización de este trabajo, y especialmente a mi compañero de tesis Jose por ayudarme en esta ardua tarea que hemos llevado a cabo y compartir juntos buenos momentos.*

José Miguel:

- ✓ *Quiero agradecer a mis padres y a todos mis abuelos por darme la mejor educación del mundo: el amor de una familia.*
- ✓ *A mis tíos Lore y Mary por ser como mis segundos padres durante estos cinco años y lo serán por siempre.*
- ✓ *A mi familia por apoyarme siempre y ser la mejor.*
- ✓ *A mis amigos con los cuales he compartido todo momento desde que llegue a esta universidad y a los que se han unido en el camino.*
- ✓ *Agradecer a mi compañera de tesis Greysi, por confiar en mí y por todo el trabajo realizado.*
- ✓ *A todos aquellos que me brindaron una idea o me dieron un consejo durante la realización de este trabajo.*

Greysi

A mi mamita del alma, que la quiero con todo mi corazón.

A mi abuelita por velar cada uno de mis pasos y cuidar de mí.

A la memoria de mi abuelito que hubiese estado muy orgulloso de mí.

A mi tío Joseito por estar siempre pendiente de mí.

A mi papá por apoyarme y ayudarme siempre en todo.

A toda mi familia por ser tan maravillosos conmigo.

Jose Miguel

Quiero dedicarles este trabajo que es el fruto de cinco

años de estudio a esas personas que son el todo en mi vida,

Mis padres y mis abuelos y en especial

A mi abuela Mirta por preocuparse tanto por el niño.

En la Universidad de las Ciencias Informáticas, en el centro CISED, se está desarrollando un Sistema de Administración de Identidades (SAI), sistema integrado por cuatro módulos diferentes los cuales son Auditoría, Autorización, Autenticación y Aprovisionamiento. Este sistema ha sido creado con el objetivo de mejorar la gestión del ciclo de vida de las cuentas de usuarios, mediante la gestión de identidades y los servicios relacionados. En la actualidad la instalación de dicha aplicación se realiza de forma manual y esto resulta engorroso para los responsables de esta actividad en el centro. Por esto surge la necesidad de crear una herramienta de instalación que facilite el despliegue de estos módulos en una plataforma de instalación. El presente trabajo tiene como objetivo el desarrollo de una Plataforma de Instalación para el SAI.

Partiendo del proceso de despliegue como fase fundamental dentro del ciclo de vida del software, se realiza un análisis a varios asistentes de instalación con el objetivo de definir un flujo de instalación completo para dicha plataforma de instalación. El estudio realizado a las principales herramientas que se pudiesen utilizar para el desarrollo de dicha plataforma permitió seleccionar al Visual Studio ya que es la que más condiciones reúne para dar solución a la problemática de la investigación. Dentro de las principales tecnologías utilizadas se encuentra el *Framework .Net 4.0* que permite la compilación y ejecución de la plataforma de instalación, las principales herramientas empleadas fueron el Visual Studio 2010 y como herramienta CASE para el modelado, el *Visual Paradigm*.

Palabras clave:

Despliegue, instalación, instalador, Sistema de Administración de Identidades.

Índice de Figuras

Índice de Tablas

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) hacen referencia a una amplia gama de servicios, aplicaciones, tecnologías, equipos y programas informáticos. En los momentos actuales, estas han evolucionado, por lo que su aplicación se hace necesaria en cada rama de la ciencia, en las cuales ha llegado a tener grandes repercusiones, por esto no deja de ser una realidad el empleo de la informática en todas las esferas de la vida y principalmente en la esfera económica, jugando un papel fundamental en la empresa moderna.

Debido al creciente desarrollo de software durante los últimos años, gran parte del mismo ha sido realizado de forma desorganizada y poco documentada, y considerando el aumento que tendrá en los próximos años, surge la necesidad de lograr métodos de organización para su desarrollo, mediante los procedimientos y herramientas, que provee la ingeniería de software para construir programas de calidad. (1)

Las empresas dedicadas a la producción de software hacen uso de metodologías de desarrollo de software durante el proceso de creación de estos. Una de las fases del ciclo de vida es la fase de despliegue donde entran a jugar un papel fundamental los asistentes de instalación y configuración o también llamados instaladores. La instalación del software es el proceso por el cual los programas desarrollados son transferidos debidamente a la computadora, inicializados y configurados. Actualmente la mayoría de los programas que se desarrollan y distribuyen cuentan con un instalador que permite la correcta instalación y configuración de todos los componentes necesarios para el uso final del software.

La instalación, dependiendo del sistema desarrollado, consiste en una simple copia al disco rígido destino, otra forma más compleja sería en la que los distintos archivos y componentes del software (ejecutables, bibliotecas, datos propios) son descomprimidos y copiados a lugares específicos del disco. En productos de venta masiva las instalaciones completas, son relativamente simples, suelen ser realizadas por los propios usuarios finales, con herramientas propias y de fácil manejo; incluso la configuración suele ser automática. En productos de diseño específico el despliegue queda restringido, normalmente, a personas especialistas involucradas en el desarrollo del software en cuestión. Una vez instalado el software, el mismo pasa a la fase de producción, durante la cual cumple las funciones para las que fue desarrollado, es decir, es finalmente utilizado por los usuarios finales, produciendo los resultados esperados.

La Universidad de las Ciencias Informáticas (UCI), como una de las principales entidades de desarrollo de software en el país, tiene como responsabilidad el desarrollo de aplicaciones para su informatización. En el Centro de Identificación y Seguridad Digital (CISED) se está desarrollando el Sistema de Administración de Identidades, este permitirá el incremento en la seguridad de los procesos de autenticación, autorización, aprovisionamiento y auditoría, además el aumento de la productividad y satisfacción por los usuarios y administradores de las aplicaciones de la organización, reducción de costos por concepto de personal dedicado a la administración, soporte de sistemas y reducción del tiempo de desarrollo de nuevas aplicaciones con elementos de seguridad. El sistema tiene cuatro subsistemas diferentes los cuales son: Autenticación, Autorización, Aprovisionamiento uno por cada proceso clave de la administración de identidades y un cuarto dedicado a las operaciones de Auditoría. Cada uno de estos subsistemas integra diferentes módulos que pueden ser aplicaciones web, servicios web y servicios Windows. El software actualmente no cuenta con una herramienta que instale y configure el sistema, es por esto que se debe realizar una aplicación *desktop* que integre en una única interfaz cada uno de los subsistemas anteriormente mencionados, logrando así una aplicación de software que instale y permita la configuración inicial de cada uno de los subsistemas del Sistema de Administración de Identidades. Partiendo de lo antes expuesto se plantea el siguiente **Problema de investigación:** ¿Cómo facilitar la instalación y despliegue del Sistema de Administración de Identidades? Para enmarcar los límites de la investigación el **Objeto de estudio** se centra en Plataformas de Instalación de software. Para darle solución al problema se define el siguiente **Objetivo general:** Desarrollar la Plataforma de Instalación para el Sistema de Administración de Identidades que facilite el despliegue de dicho sistema. Para darle cumplimiento al mismo se trazan los siguientes **Objetivos específicos:**

- Analizar los mecanismos de instalación existentes.
- Identificar los pre-requisitos y requisitos de instalación para cada componente instalable del Sistema de Administración de Identidades.
- Definir la metodología, las herramientas, los lenguajes y las tecnologías que serán utilizadas en la implementación de la Plataforma de instalación del Sistema de Administración de Identidades.
- Desarrollar la Plataforma de instalación del Sistema de Administración de Identidades.

- Realizar las pruebas a la Plataforma de instalación para el Sistema de Administración de Identidades.

En la presente investigación se tiene como **Idea a Defender** el desarrollo de un Instalador para el Sistema de Administración de Identidades, permitirá facilitar el despliegue y configuración de los componentes de dicho Sistema.

En correspondencia con los objetivos propuestos se trazan las siguientes **Tareas de Investigación:**

- Elaboración del marco teórico para la justificación del trabajo.
- Identificación de las necesidades de la plataforma a desarrollar a partir del estudio de los diferentes subsistemas.
- Definición de la arquitectura de la plataforma de instalación.
- Definición de los pre-requisitos y requisitos de instalación de los diferentes subsistemas teniendo en cuenta la plataforma de despliegue.
- Implementación de la plataforma base de instalación.
- Implementación de las funcionalidades del instalador.
- Realizar las pruebas pertinentes para verificar la calidad de la Plataforma de Instalación.

Los métodos científicos utilizados para darle cumplimiento a las tareas antes expuestas son:

Métodos Teóricos

Analítico-Sintético: El uso de este método permitió analizar los diferentes instaladores de mayor uso en la actualidad y determinar las características que debe tener el instalador propuesto.

Histórico-Lógico: Mediante este método científico, se realizó un estudio del estado del arte, sobre los instaladores que se utilizan en la actualidad. Permitió analizar la evolución de estos sistemas y cuál es su tendencia actual en el mundo.

Métodos Empíricos

Observación: La utilización de este método científico permitió estudiar otros trabajos relacionados con el desarrollo de instaladores con elementos comunes en la investigación. También observar la situación real que se está investigando, así como la identificación de los pasos que conforman el proceso de instalación.

El presente trabajo se divide para su mejor comprensión y análisis, en tres capítulos los cuales quedan estructurados de la siguiente manera:

Capítulo 1: “Fundamentación Teórica”. El objetivo de este capítulo es analizar las diferentes plataformas o marcos de trabajo existentes en la actualidad para el desarrollo de instaladores y los diferentes componentes de despliegue del Sistema de Administración de Identidades. Además se describirán las herramientas, tecnologías, metodologías y lenguajes a utilizar en la solución del problema en cuestión.

Capítulo 2: “Visión y Planificación”. El objetivo que tiene este capítulo es describir el desarrollo de la solución propuesta, se muestra como quedaría el flujo de instalación de la plataforma de instalación, así como los principales artefactos generados para el desarrollo de la plataforma.

Capítulo 3: “Desarrollo y Estabilización”. En este capítulo se especifican la arquitectura y los patrones a utilizar, también los diagramas realizados. Además se describen las pruebas realizadas que permiten validar la solución y se exponen ejemplos de pruebas ejecutadas al sistema que demuestran la integridad del mismo.

Capítulo 1. “Fundamentación teórica”

Introducción

En este capítulo se realiza un estudio detallado de las diferentes herramientas existentes para la realización de instaladores, analizando así los conceptos asociados a estas, se realiza un análisis de las más utilizadas a nivel mundial para de esta manera seleccionar la mejor para el trabajo. En este capítulo también se exponen algunas de las principales características de estos instaladores, cómo funcionan y cómo se realiza la configuración de estos programas. Además se selecciona la metodología que guiará el desarrollo de la aplicación así como los lenguajes de programación y las tecnologías a utilizar. Todo lo anterior tributa a una mejor comprensión del trabajo actual sobre el desarrollo de un instalador para el Sistema de Administración de Identidades.

1.1 Proceso de despliegue de software

Los procesos de despliegue de soluciones de software son, dentro del proceso completo de desarrollo, los más costosos y se ubican entre los más complicados, debido a la carga de personal necesario para esta operación, al numeroso grupo de actividades que conlleva, al soporte que se mantendrá al software, y a que este es el proceso que se realiza con la participación directa de los usuarios finales del software. El despliegue es una etapa de desarrollo de software que se valora muy poco por los desarrolladores, ya que piensan que la etapa fundamental es la Implementación y aunque tiene algo de verídico, no es menos cierto que el despliegue y todo lo que este conlleva, dentro del ciclo de vida del software juega un papel fundamental. Ahora bien, si no se interactúa con el cliente adecuadamente, no se entrega el producto como este exige, del mismo modo si no se utilizan prácticas y técnicas que dejen satisfechos al mismo, este puede ser un fracaso total, aunque el resultado esperado por todos sea el que se diseñó, en principio. Casi siempre para esta etapa final se disponen pocos recursos, conocimientos o no se estudia lo suficiente, ni se prepara el personal para enfrentar el despliegue con calidad.

En el proceso de despliegue de software se deben tener en cuenta varios principios para lograr el éxito de dicho despliegue algunos de estos serían: Se deben administrar las expectativas que el cliente tiene del software. Además ensamblar y probar el paquete de entrega completo, establecer un régimen de soporte antes de entregar el software. Proporcionar material instructivo apropiado a los usuarios finales y arreglar todos los errores encontrados antes de entregar el software. (2)

1.1.1 Proceso de despliegue de software según RUP

RUP divide el proceso en 4 fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. RUP se divide en cuatro fases:

- Inicio (define el alcance del proyecto)
- Elaboración (definición, análisis, diseño)
- Construcción (implementación)
- Transición (fin del proyecto y puesta en producción)

A continuación se muestra una imagen donde aparece la relación entre las cuatro fases del ciclo de vida de RUP y los flujos de trabajos correspondientes en cada una de las fases.

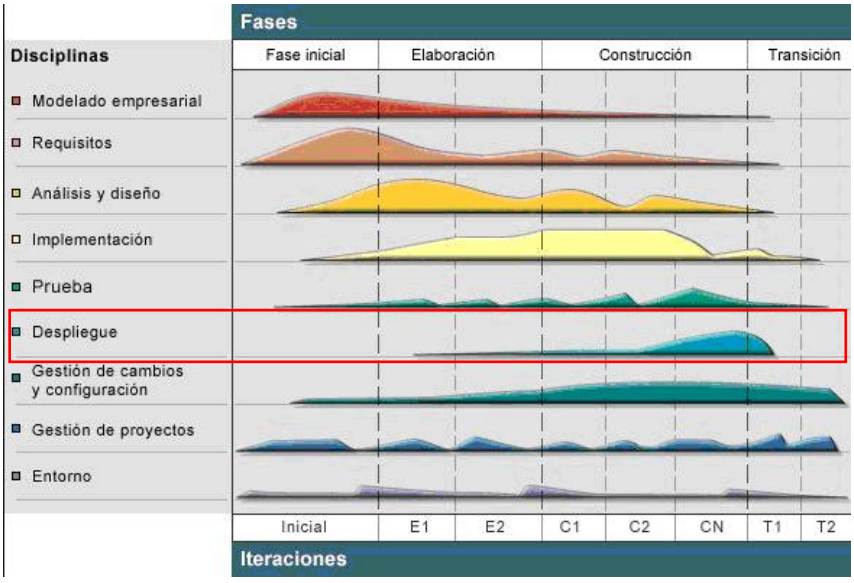


Figura 1: Despliegue como flujo de RUP.

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y realizar tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto. Esta fase se enfoca en asegurar que el software esté disponible para sus usuarios. Esta fase se puede subdividir en varias iteraciones, además incluye pruebas del producto para poder hacer el entregable del mismo. En este punto, la retroalimentación de los usuarios se centra en refinar el producto, configuraciones, instalación y aspectos sobre utilización. Durante esta fase de transición se busca garantizar que se tiene un

producto preparado para su entrega al usuario. El objetivo es traspasar el software desarrollado a la comunidad de usuarios.

Incluye:

- Pruebas Beta para validar el producto con las expectativas del cliente.
- Ejecución paralela con sistemas antiguos.
- Conversión de datos.
- Entrenamiento de usuario.
- Distribuir el producto.

Objetivos:

- Obtener autosuficiencia de parte de los usuarios.
- Concordancia en los logros del producto de parte de las personas involucradas.
- Liberar el producto al mercado.

Principales características:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software. (3)

1.1.2 Proceso de despliegue de software según MSF

La metodología *Microsoft Solution Framework for Agile Software Development* por sus siglas en inglés (MSF) es un modelo de proceso de desarrollo de software basado en fases, dirigido por hitos, y un modelo iterativo que puede ser aplicado para desarrollar y desplegar aplicaciones tradicionales, soluciones empresariales, para comercio electrónico y aplicaciones Web distribuidas. Combina los mejores principios de los modelos en espiral y cascada, combinando la predictibilidad y la creatividad que dan las iteraciones y realimentación continúa. (4)

MSF consiste en cinco fases:

1. Visión
2. Planificación

3. Desarrollo
4. Estabilización
5. Despliegue

A continuación se muestra una imagen con el ciclo de vida completo de la metodología MSF:

Figura 2: Fase de despliegue en la metodología MSF.

Durante la fase de implementación, el equipo implementa la tecnología base y los componentes del sitio, estabiliza el despliegue, las transiciones del proyecto a las operaciones, y obtiene la aprobación final del cliente del proyecto. Después de la implementación, el equipo lleva a cabo una revisión del proyecto y una encuesta de satisfacción del cliente.

El objetivo del proceso de implementación y estabilización es el de identificar y resolver la mayor cantidad de problemas como sea posible para minimizar el total de problemas recurrentes. (4)

Se llevarán a cabo en esta fase los planes de despliegue y el de formación. Los principales trabajos e hitos a conseguir son, en este caso, además de los obvios (implantación de la plataforma, puesta en servicio de todas las funciones, formación a los usuarios y administradores), los siguientes:

- Continuación con las labores de recepción de incidencias, clasificación, tratamiento, resolución e intervención *on-site*.

- Registro de mejoras y sugerencias, funcionalidades no cubiertas y novedades a incorporar en sucesivas versiones de la plataforma, incluyendo mejoras aportadas por los fabricantes de software (nuevas versiones o *Services Packs*).
- Revisión de las Guías y manuales de usuario, rectificación de errores y obtención de los documentos de formación definitivos.
- Entrega de los documentos definitivos en la fase de Visión y *Scope.*
- Revisión de la matriz de riesgos, las métricas de calidad y establecimiento de los estándares de calidad.
- Finalmente, entrega del proyecto y cierre del mismo, con o sin apertura de nuevo proyecto en base a la información y experiencia obtenida. (5)

La duración de la fase de despliegue, puesto que debe planificarse, no puede establecerse a priori, depende de numerosos factores externos al propio proyecto (incluyendo factores de oportunidad política o de negocio) que pueden retardar o acelerar la conclusión.

La experiencia demuestra que no hay una relación directa entre número de máquinas y tiempo necesario para el despliegue. Los factores más relevantes en el cálculo suelen ser la dispersión o concentración geográfica, la complejidad del proceso de migración, el grado de automatización alcanzado, la experiencia y nivel de los técnicos que realizan la operación y condicionantes de calendario, a menudo con restricciones no técnicas, sino de otros tipos. (5)

1.2 Conceptos principales

Instalador: Muchas veces un programa está formado por un conjunto de archivos. El programa en sí mismo solamente es uno, pero necesita de esos otros para funcionar. En muchas ocasiones, esos archivos tienen que estar colocados en diferentes partes del sistema, a veces relacionados con otros archivos. Además de esto, los programas tienen que registrarse en el registro de Sistema Operativo para que el sistema funcione correctamente.
Todas estas tareas se pueden hacer manualmente, pero en muchas ocasiones resultan complejas y engorrosas. Es por eso que generalmente se utiliza un instalador: un programa especial que realiza todas esas tareas de manera automática.

Usar un instalador es sencillo, sólo se va leyendo las indicaciones que se muestran. Generalmente, sólo se trata de hacer clic en Siguiente. Algunos presentan la opción de instalar solamente. Otros, permiten cambiar el directorio donde se colocará el archivo. Al terminar de instalarse el programa, la mayoría coloca accesos directos en el menú, todos los programas y en el escritorio. Es importante conservar siempre los instaladores, pues el programa puede ser que se dañe en algún momento y que se necesite reinstalarlo. Un instalador es un archivo que contiene las instrucciones específicas para que un programa cumpla una determinada función en una Computadora Personal. Dicho archivo contiene librerías de enlace dinámico (DLL por sus siglas en inglés), como también archivos que se copian en el registro y muchos otros, que permiten la funcionalidad de un programa. (6)

1.3 Análisis de Sistemas diseñadores de instaladores

1.3.1 *InstallJammer*

InstallJammer es una aplicación de código abierto 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. *InstallJammer* es un generador de instaladores muy poderoso, el cual admite temas múltiples y de alto nivel de configuración, permite conocer si se encuentran instaladas o no las dependencias que necesita el software para un correcto funcionamiento, información de suma importancia para tratar las dependencias necesarias para el correcto funcionamiento del sistema durante el proceso de instalación y después del mismo, se pueden manipular y modificar mensajes de alertas e informaciones de errores durante todo el proceso de instalación. Los mismos son construidos como un simple fichero ejecutable para una fácil distribución sobre la web, instalando todo lo que se necesita para la aplicación en cualquier plataforma que se ejecute.

Características

- Desarrollo rápido de instaladores.
- Alto nivel de Configuración disponible en todas las etapas de construcción del instalador.
- Soporte multiplataforma.
- Creación de un instalador disponible en varios lenguajes.
- Instalaciones rápidas.
- Instalador ligero.
- Edición de diálogos.

- Control por línea de comandos.
- Posibilita la creación de un desinstalador.
- Ejecución de programas externos(ejemplo: *scripts bash*)
- Detección de paquetes instalados o no en el sistema operativo usado. (7)

1.3.2 *Install4j*

Install4j es un potente generador de instalador de Java, multiplataforma, diseñado con el fin de proporcionar un medio fácil de usar para generar instaladores y lanzadores de aplicaciones nativos personalizados para aplicaciones de Java, se destaca por su facilidad de uso, su compatibilidad con plataformas amplias y su potente sistema de pantalla y acción.

Características:

- Vista enriquecida y sistema de acción.
- Instalador estético.
- Excepcional facilidad de uso.
- Excelente soporte multiplataforma.
- Auto-actualizadores y aplicaciones personalizadas.
- Fácil creación de pantallas personalizadas. (8)

1.3.3 *InstallShield*

InstallShield es una solución de instalación de desarrollo potente y fácil de usar para la creación de instalaciones de *Windows*. Con herramientas automatizadas para producir, empaquetar e instalar sus productos. (9)

Características:

- Soporte para Windows 8, Windows Server 2012 y *Visual Studio* 2012.
- Permite crear funciones y condiciones de los componentes de estos sistemas operativos.
- Mejoras en la suite de instalación.
- Una suite avanzada de interfaz de usuario.
- Comprobación automática de actualizaciones y parches *installer*.
- Secuencias de commandos *PowerShell Support*.

1.3.4 *InstallBuilder*

InstallBuilder es una herramienta de desarrollo para crear instaladores multiplataforma para el software de escritorio y servidor. Se pueden crear rápidamente instaladores profesionales dinámicos, para Linux, Windows, Mac OS X, Solaris y otras plataformas desde un único archivo de proyecto. La funcionalidad de actualización automática hace que sea fácil para entregar actualizaciones directamente a los usuarios una vez que tienen el software instalado. (10)

Características:

- Multiplataforma.
- Actualización automática: Entregar actualizaciones a los usuarios una vez que tienen el software instalado para asegurarse de que están usando siempre la mejor versión disponible.
- Independiente del lenguaje: Puede instalar aplicaciones en cualquier lenguaje, incluyendo: Java, PHP, Perl, Python, Ruby, C / C + +.
- Integración de escritorio: Proporciona aspecto nativo y la integración de escritorio para Windows, KDE y Gnome.
- Instaladores optimizados: Los instaladores están optimizados en tamaño y velocidad y no requieren un paso de extracción automática. Esto reduce el arranque de descarga y el tiempo de instalación.
- Construir soporte multiplataforma: La herramienta de creación de instalador puede ejecutarse en Windows, Mac OS X, Solaris, HP-UX, AIX, FreeBSD, OpenBSD, IRIX y Linux (x86/x64 Intel, Itanium) y generar instaladores para todas las plataformas de destino de un archivo de proyecto único.
- Ahorro de tiempo: Para los usuarios avanzados, un formato amigable proyecto XML soporta la integración de control de origen, el desarrollo de proyectos de colaboración y personalización tanto a mano y usando *scripts* externos. Una interfaz de línea de comandos le permite automatizar e integrar el proceso de construcción.
- Construido en acciones: Acciones incorporadas hacen fácil implementar la funcionalidad de instalación normalmente requerida de una manera multiplataforma. Añadir acciones a su instalador para autodetectar un *Java Runtime*, cambiar los permisos de archivos y la propiedad, sustituyendo el texto en un archivo, añadiendo variables de entorno, añadiendo directorios a la

ruta de acceso, la creación de vínculos simbólicos, cambiando el registro de Windows, el lanzamiento de *scripts* externos y así sucesivamente en sólo unos pocos clics. (10)

1.3.5 *Inno Setup*

Inno Setup es un instalador gratuito para muchos programas de *Windows*. Presentado por primera vez en 1997, *Inno Setup* supera a muchos instaladores en la actualidad, en características y estabilidad. (11)

Características

- Apoyo para cada versión de *Windows* desde el 2000, incluyendo: Windows 8, Windows Server 2012, Windows 7, Windows Vista, Windows Server 2008, Windows XP, Windows Server 2003 y Windows 2000.
- Amplio soporte para la instalación de aplicaciones de 64 bits en las ediciones de 64 bits de *Windows*.
- Estándar de *Windows 2000/XP-style* asistente de interfaz.
- Tipos de configuración personalizables, por ejemplo: completa, mínima, personalizada.
- Completar las capacidades de desinstalación.
- Creación de accesos directos en cualquier lugar, incluso en el menú Inicio y en el escritorio.
- Ejecución de los programas antes, durante o después de la instalación.
- Apoyo para la instalación en varios idiomas.
- Apoyo para las instalaciones y desinstalaciones con firma digital.
- Instalar y desinstalar.
- El código fuente completo está disponible (*Borland Delphi* 2.0-5.0 y 2009).

1.3.6 *InstallAnywhere*

InstallAnywhere es la solución líder de desarrollo multiplataforma para los productores de instalación de aplicaciones ya que provee lo que se necesita para ofrecer una buena experiencia de instalación profesional para entornos físicos y virtuales. Desde un único archivo de proyecto este crea instalaciones confiables para las instalaciones en plataformas Windows, Linux, Mac OS X, Solaris, AIX,

HP-UX, IBM iSeries y le permite llevar los productos de software existentes a nuevas infraestructuras virtuales. (12)

Características

Reducir el tiempo de desarrollo de software.

- Construir un proyecto de instalación en cuestión de minutos con el Asistente de Proyecto y Diseñador Avanzado.
- Instalar el software en múltiples plataformas y entornos tanto físicos como virtuales utilizando un archivo de proyecto.
- Crear instaladores de las aplicaciones más complejas con personalización avanzada y opciones de configuración y soporte de accesibilidad.

Impresiona a los usuarios finales con instalaciones personalizadas.

- Crear instaladores y marca del producto con gráficos personalizados, animaciones e imágenes transparentes.
- Ofrece instalaciones multiplataforma utilizando un amplio conjunto de acciones estándar y API de código personalizado.

1.3.7 *Visual Studio* 2010

Visual Studio proporciona las herramientas que se necesitan para diseñar, desarrollar, depurar y desplegar aplicaciones Web, servicios Web y aplicaciones clientes tradicionales. *Visual Studio* es un conjunto completo de herramientas de desarrollo para crear aplicaciones Web ASP.NET, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C + +, Visual C # y Visual J # todos utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Además, estos lenguajes aprovechan las funciones del *Framework .NET*, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y los servicios Web. *Microsoft Visual Studio 2010 Ultimate* incluye potentes herramientas que simplifican todo el proceso de desarrollo de aplicaciones, de principio a fin. Los equipos pueden observar una mayor productividad y ahorro de costes al utilizar características de colaboración avanzadas, así como herramientas de pruebas y depuración integradas que ayudarán a crear siempre un código de gran calidad. (13)

Administración del ciclo de vida de las aplicaciones (ALM): La creación de aplicaciones de éxito requiere un proceso de ejecución uniforme que beneficie a todos los componentes del equipo. Las herramientas de ALM integradas en *Visual Studio 2010 Ultimate* contribuyen a que las organizaciones colaboren y se comuniquen de forma efectiva en todos los niveles, y a que se hagan una idea precisa del estado real del proyecto, lo que garantiza que se ofrezcan soluciones de gran calidad al tiempo que se reducen los costos.

Herramientas de prueba: Incorpora herramientas avanzadas de pruebas para garantizar la calidad del código en todo momento.

Arquitectura y modelado: El Explorador de arquitectura ayuda a entender los activos de código existentes y otras interdependencias. Los diagramas por capas ayudan a garantizar el cumplimiento de la arquitectura y permiten validar artefactos de código con respecto al diagrama. Además, admite los cinco diagramas de UML más comunes que conviven junto con su código.

Desarrollo de bases de datos: El desarrollo de bases de datos requiere el mismo cuidado y atención que el desarrollo de aplicaciones. Proporciona potentes herramientas de implementación y administración de cambios que garantizan que la base de datos y la aplicación estén siempre sincronizadas.

Compatibilidad con la plataforma de desarrollo: Tanto si se crea soluciones nuevas como si se quiere mejorar las aplicaciones ya existentes, permite hacer realidad una gran variedad de plataformas, entre las que se incluyen *Windows, Windows Server, Web, Office y SharePoint*, entre otras, todo en un único entorno de desarrollo integrado. (13)

1.4 Selección de la herramienta a utilizar

Después del estudio realizado de algunas de las herramientas para crear software de instalación se ha llegado a la conclusión de cuál de estas herramientas anteriormente analizadas, se va a utilizar para solucionar el problema en cuestión, para la selección de dicha herramienta se tuvo en cuenta varios parámetros a medir como son:

- Rápido desarrollo y usabilidad.

- Alto nivel de configuración.
- Uso de dependencias.
- Detección de paquetes instalados o no en el sistema operativo usado.
- Creación de un desinstalador.
- Soporte para varios lenguajes.

Según el análisis realizado a las diferentes herramientas se descarta la utilización de las aplicaciones: *Install4j, InstallAnywhere, InstallBuilder, InstallShield* ya que aunque son herramientas muy utilizadas para la realización de instaladores, son reducidas a los pasos básicos de un proceso de instalación, y se necesita una herramienta que brinde otras opciones para cumplir con las especificaciones que se requieren. En el caso de la herramienta *InnoSetup* no cuenta con los elementos necesarios para darle solución a la problemática planteada debido al bajo nivel de opciones que brindan en cuanto a elementos de configuración, además sus funciones son muy básicas a la hora de desarrollar un instalador. Por todo lo anteriormente expuesto se decide utilizar *Visual Studio*, es la herramienta que más condiciones reúne para dar solución a la problemática de la presente investigación, dado que permite la realización de una aplicación con una agradable interfaz visual, amigable, fácil de usar, entendible, ligera para el sistema, posibilitando una instalación rápida, conociendo dependencias instaladas o no en el sistema, entre otras características, que suplen las necesidades del SAI. Además la aplicación a la que se le va a hacer el instalador necesita un buen manejo de los registros de *Windows*, configuración de IIS 7(*Internet Information Services*) y conexiones a bases de datos.

1.5 Proceso de instalación

Una instalación exitosa es una condición necesaria para el funcionamiento de cualquier software, mientras más complejo sea el software, más archivos contenga, y mayor sea la interdependencia con otros software, mayor es el riesgo de alguna falla durante la instalación. Si la instalación falla aunque sea solo parcialmente, el fin que persigue la instalación posiblemente no podrá ser alcanzado. Por esa razón, sobre todo en casos de software complejo, el desarrollo de un proceso de instalación confiable y seguro es una parte fundamental en el desarrollo del software. A continuación se muestran una serie de pasos de forma genérica los cuales se destacan en un proceso de instalación:

1. Verificación de la compatibilidad: Se debe comprobar si se cumplen los requisitos para la instalación en cuanto a hardware y software. A veces es necesario desinstalar versiones antiguas del mismo software.

2. Verificación de la integridad: Se verifica que el paquete de software es el original, esto se hace para evitar la instalación de programas maliciosos.
3. Creación de los directorios requeridos: Para mantener el orden en el directorio cada sistema operativo puede tener un estándar para la instalación de ciertos archivos en ciertos directorios.
4. Creación de los usuarios requeridos: Para deslindar responsabilidades y tareas se pueden o deben usar diferentes usuarios para diferentes paquetes de software.
5. Concesión de los derechos requeridos: Para ordenar el sistema y limitar daños en caso necesario, se le conceden a los usuarios solo el mínimo necesario de derechos.
6. Copia, desempaque y descompresión de los archivos desde el paquete de software: Para ahorrar ancho de banda y tiempo en la transmisión por internet o espacio de disco duro, los paquetes vienen empacados y comprimidos.

    - Archivos principales, sean de fuente o binarios.
    - Archivos de datos, por ejemplo datos, imágenes, modelos y documentos XML.
    - Documentación.
    - Archivos de configuración.
    - Bibliotecas.
    - Enlaces duros o enlaces simbólicos a otros archivos.

7. Compilación y enlace con las bibliotecas requeridas: En algunos casos no se puede evitar el complicado paso de la compilación y enlace que a su vez tiene severos requerimientos de software al sistema. El enlace con bibliotecas requeridas puede ser un problema si en su instalación no se acataron los estándares establecidos.
8. Configuración: Por medio de archivos de configuración se le da a conocer al software con qué parámetros debe trabajar. Por ejemplo, los nombres de las personas que pueden usar el software, cómo verificar su clave de ingreso, la ruta donde se encuentran los archivos con datos o la dirección del proveedor de correo electrónico.
9. Definir las variables de entorno requeridas: Algunos comportamientos del software solo pueden ser determinados por medio de estas variables. Esto es parte de la configuración, aunque es más dinámica.
10. Registro ante el dueño de la marca: Para el software comercial a veces el desarrollador de este, exige el registro de la instalación si se desea su servicio. (14)

1.6 Metodologías y Herramientas

1.6.1 *Microsoft Solution Framework*

El desarrollo de software no es una tarea fácil, como resultado a este problema ha surgido una alternativa desde hace mucho tiempo, la metodología. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar, inspirado por otras disciplinas de la ingeniería. Como una reacción a estas metodologías, un nuevo grupo de metodologías ha surgido en los últimos años. Durante algún tiempo se conocían como metodologías ligeras, pero el término aceptado ahora es metodologías ágiles. Los métodos ágiles cambian significativamente algunos de los énfasis de los métodos ingenieriles. La diferencia inmediata es que son menos orientados al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. De muchas maneras son más bien orientados al código, siguiendo un camino que dice que la parte importante de la documentación es el código fuente.

Para dar solución al presente trabajo se escoge la *metodología Microsoft Solution Framework for Agile software Development* por sus siglas en inglés (MSF), esta es del tipo de metodologías ágiles, está enfocada a dirigir proyectos o soluciones de innovación, en ella no se detalla ni se hace énfasis en la organización ni el tamaño del equipo de desarrollo, está más bien centrada en la gestión y administración del proyecto para lograr el impacto deseado. Involucra indudablemente la calidad ya que provee liberar una solución si esta aún tiene fallos o desperfectos para ello propone seleccionar un grupo de prueba piloto el cual es una Versión Beta y cumplido un tiempo de prueba ya es liberada la versión formal o Versión Alfa en la cual está garantizada la calidad. (15)

MSF es una flexible e interrelacionada serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Originalmente creado en 1994 para conseguir resolver los problemas a los que se enfrentaban las empresas en sus respectivos proyectos, se ha convertido posteriormente en un modelo práctico que facilita el éxito de los proyectos tecnológicos.

Características

- **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cuál su uso es limitado a un específico lugar.

- **Escalable:** puede organizar equipos tan pequeños entre 3 ó 4 personas, así como también, proyectos que requieren 50 personas o más.
- **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología. (16)

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación. MSF ayuda a las organizaciones a obtener los beneficios de las nuevas tecnologías mediante la aplicación de estos modelos. Estos modelos se aplican a diferentes categorías de problemas que están enmarcadas en la planeación, construcción o implantación, y administración de los sistemas. (17)

Consta de cinco fases:

Fase 1: Visión. En esta fase el equipo y el cliente definen los requerimientos del negocio y los objetivos generales del proyecto. La fase culmina con el hito Visión y Alcance aprobados.
Fase 2: Planeación. Durante la fase de planeación el equipo crea un borrador del plan maestro del proyecto, además de un cronograma del proyecto y de la especificación funcional del proyecto. Esta fase culmina con el hito plan del proyecto aprobado.
Fase 3: Desarrollo. Esta fase involucra una serie de *releases* internos del producto, desarrollados por partes para medir su progreso y para asegurarse que todos sus módulos o partes están sincronizados y pueden integrarse. La fase culmina con el hito Alcance completo.
Fase 4: Estabilización. Esta fase se centra en probar el producto. El proceso de prueba hace énfasis en el uso y el funcionamiento del producto en las condiciones del ambiente real. La fase culmina con el hito *Release Readiness* aprobado.
Fase 5: Despliegue o Implantación. En esta fase el equipo implanta la tecnología y los componentes utilizados por la solución, estabiliza la implantación, apoya el funcionamiento y la transición del proyecto, y obtiene la aprobación final del cliente. La fase termina con el hito Implantación completa. (18)

1.6.2 Visual Paradigm

CASE *Computer Aided software Engineering* es el conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de software. (19)

Las Herramientas CASE son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. La sigla genérica para una serie de programas y una filosofía de desarrollo de software que ayuda a automatizar el ciclo de vida de desarrollo de los sistemas. Una innovación en la organización, un concepto avanzado en la evolución de tecnología con un potencial profundo en la organización. Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.

Visual Paradigm for UML es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas, además es considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos UML. Teniendo en cuenta sus características y los beneficios que brinda para la construcción de software, especialmente referente al modelado, se decidió utilizar *Visual Paradigm for UML* para el modelado de la aplicación. (20)

Permite insertar código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros. (21)

Características

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

Ventajas

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Genera modelos VP-UML instantáneamente a partir de código binario .Net.
- Generación de documentación en formatos HTML y PDF.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X y Solaris.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo *Visio y Rational Rose*.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas. (21)

1.7 Tecnologías y Lenguajes

1.7.1 *Framework* 4.0

El *Framework* .NET es un componente integral de Windows que admite la compilación y la ejecución de la siguiente generación de aplicaciones y servicios Web. Los componentes clave de .NET *Framework* son *Common Language Runtime*, entorno en tiempo de ejecución de lenguaje común, por sus siglas en inglés CLR y la biblioteca de clases .NET *Framework*, que incluye ADO.NET, ASP.NET, formularios *Windows Forms* y *Windows Presentation Foundation* (WPF), .NET proporciona un entorno de ejecución administrado, un desarrollo e implementaciones simplificadas y la integración con una gran variedad de lenguajes de programación. (22)

El *Framework .NET* proporciona los siguientes servicios para los desarrolladores de aplicaciones:

- Gestión de la memoria. En muchos lenguajes de programación, los programadores tienen la responsabilidad de asignar y liberar memoria para el manejo de duración de los objetos. En aplicaciones .NET *Framework*, el CLR proporciona estos servicios a nombre de la aplicación.
- Un sistema de tipo común. En lenguajes de programación tradicionales, tipos básicos son definidos por el compilador, lo que complica la interoperabilidad entre lenguajes.
- Una extensa biblioteca de clases. En lugar de tener que escribir grandes cantidades de código para manejar comunes operaciones de bajo nivel de programación, los programadores pueden utilizar una biblioteca de fácil acceso de .NET *Framework*.
- Marcos de desarrollo y tecnologías. El *Framework* .NET incluye bibliotecas para áreas específicas de desarrollo de aplicaciones, tales como ASP.NET para aplicaciones Web, ADO.NET para el acceso a los datos y *Windows Communication Foundation* para aplicaciones orientadas a servicios.
- Interoperabilidad con otros idiomas. Los compiladores de lenguajes destinados a .NET *Framework* emiten un código intermedio llamado *Common Intermediate Language* (CIL), que, a su vez, se compila en tiempo de ejecución por el *Common Language Runtime*. Con esta función, las rutinas escritas en un idioma son accesibles a otros idiomas, y los programadores pueden centrarse en la creación de aplicaciones en su idioma preferido.
- Compatibilidad de versiones. Con raras excepciones, las aplicaciones que se desarrollan mediante el uso de una versión particular del Marco. NET pueden ejecutarse sin modificación en una versión posterior. (23)

1.7.2 Lenguaje Unificado de Modelado

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

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo pueda entender.

- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura visión. (24)

Utilidad de UML

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

- Como lenguaje de propósito general, se enfoca en el corazón de un conjunto de conceptos para la adquisición, compartición y utilización de conocimientos emparejados con mecanismos de extensión.
- Como un lenguaje para modelamiento ampliamente aplicable, puede ser aplicado a diferentes tipos de sistemas, dominios (negocios, software) y métodos o procesos. Como un lenguaje para modelamiento soportable por herramientas, estas están disponibles para soportar la aplicación del lenguaje para especificar, visualizar, construir y documentar sistemas.
- Como un lenguaje para modelamiento industrialmente estandarizado, no es un lenguaje cerrado, sino más bien, un lenguaje abierto y totalmente extensible reconocido por la industria.(24)

1.7.3 Lenguaje de Programación C#

Es un lenguaje de programación que está diseñado para la construcción de una gran variedad de aplicaciones que se ejecutan en el marco .NET, C# es simple, con seguridad, y orientado a objetos. Las numerosas innovaciones en C# permiten el desarrollo rápido de aplicaciones, manteniendo la expresividad y elegancia de los lenguajes C-estilo.

Visual C# es una implementación del lenguaje C# de *Microsoft*. *Visual Studio* admite *Visual C#* con un editor de código con todas las funciones, compilador, plantillas de proyecto, diseñadores, asistentes de código, un depurador de gran alcance y fácil de usar. El .NET *Framework* proporciona acceso a muchos servicios del sistema operativo y otras clases útiles, bien diseñadas que aceleran el ciclo de desarrollo de manera significativa.

La sintaxis de C# es simple y fácil de aprender, esta será inmediatamente reconocible para cualquiera que esté familiarizado con C, C++ o Java. Los desarrolladores que conocen alguno de estos lenguajes son típicamente capaces de comenzar a trabajar de manera productiva en C# en un tiempo muy corto. La sintaxis de C# simplifica muchas de las complejidades de C++ y proporciona características avanzadas tales como tipos de valor que aceptan valores *NULL*, enumeraciones, expresiones y el acceso directo a la memoria, que no se encuentran en Java. Admite métodos genéricos, que proporcionan una mayor seguridad y rendimiento, y los iteradores, que permiten a los implementadores de clases definir comportamientos de iteración personalizados que son fáciles de utilizar por el usuario.

Como un lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos, incluido el método *Main*, punto de entrada de la aplicación, se encapsulan dentro de definiciones de clase. Una clase puede heredar directamente de una clase padre, pero puede implementar cualquier número de interfaces. En C#, una estructura es como una clase de peso ligero que pueden implementar interfaces, pero no admite la herencia. Es un lenguaje orientado a objetos simple, elegante y con seguridad en el tratamiento de tipos, que permite a los programadores de aplicaciones crear una gran variedad de aplicaciones. También proporciona la capacidad de generar componentes de sistema duraderos en virtud de las siguientes características:

- Gran robustez, gracias a la recolección de elementos no utilizados (liberación de memoria) y a la seguridad en el tratamiento de tipos.
- Seguridad implementada por medio de mecanismos de confianza intrínsecos del código.
- Plena compatibilidad con conceptos de metadatos extensibles.

Además, es posible interaccionar con otros lenguajes, entre plataformas distintas, y con datos heredados, en virtud de las siguientes características:

- Plena interoperabilidad por medio de los servicios .NET *Framework* con un acceso limitado basado en bibliotecas.
- Compatibilidad con XML para interacción con componentes basados en tecnología Web.
- Capacidad de control de versiones para facilitar la administración y la implementación. (25)

1.7.4 Lenguaje XML

XML es el acrónimo de *e**X**tensibe **M**arkup **L**anguaje*. XML es un formato estándar del World Wide Web *Consortium* (W3C) para representar datos estructurados de forma jerárquica (en un árbol). Los documentos XML incluyen una serie de etiquetas que permiten crear documentos auto contenido, en los que los datos van siempre acompañados de sus metadatos correspondientes. XML no es un lenguaje de marcado, es un metalenguaje que permite definir lenguajes de marcado adecuados a usos específicos.

XML se inició como un subconjunto de SGML (*structured generalized markup language*), un estándar ISO por sus siglas en inglés *International Organization for Standardization* para documentos estructurados que es sumamente complejo para poder servir documentos en la web. XML es algo así como SGML simplificado, de forma que una aplicación no necesita comprender SGML completo para interpretar un documento, sino sólo el subconjunto que se defina. Los editores SGML, sin embargo, pueden comprender XML. (26)

Sus características más relevantes son:

- XML es un estándar para escribir datos estructurados en un fichero de texto.
- XML consta de una familia de tecnologías.
- XML no requiere licencia.

Estructura: Un documento XML tiene dos estructuras, una lógica y otra física. Físicamente, el documento está compuesto por unidades llamadas entidades. Una entidad puede hacer referencia a otra entidad, causando que esta se incluya en el documento. Cada documento comienza con una entidad documento, también llamada raíz. Lógicamente el documento está compuesto de declaraciones, elementos, comentarios, referencias a caracteres e instrucciones de procesamiento, todos los cuales están indicados por una marca explícita. (27)

1.8 Conclusiones parciales

Con el estudio de algunas de las herramientas más utilizadas en el mundo para el desarrollo de aplicaciones se llegó a la conclusión de cuál de estas usar para el desarrollo del instalador para el SAI, demostrando que el IDE *Visual Studio* 2010 es el que satisface las expectativas del trabajo que se pretende realizar debido a las características específicas que tiene la Plataforma de Instalación. El análisis de las diferentes metodologías, tecnologías y lenguajes según las necesidades de la solución determinó que la metodología a usar sería MSF, ya que esta metodología según sus características,

no hace énfasis en el tamaño del equipo sino en la calidad del producto. Haciendo uso de la plataforma .NET y usando el IDE de desarrollo *Visual Studio* 2010 se llevará a cabo la implementación de la Plataforma debido a las ventajas que este IDE provee para la puesta en marcha del Instalador, utilizando el lenguaje de programación *C #* y el lenguaje XML, y para el modelado y diseño de diagramas el *Visual Paradigm,* con el fin de obtener un mayor entendimiento de las clases y la estructura de la Plataforma.

Capítulo 2. “Visión y Planificación”

Introducción

La metodología MSF para el desarrollo de software ágil, ha sido una guía para el proceso de desarrollo del presente sistema informático, se han desarrollado dos de las fases que la componen en el presente capítulo. Dicha metodología plantea que se debe adquirir una visión clara de lo que se desea desarrollar y propone la realización de una planificación que guíe al equipo de trabajo hacia la exitosa construcción del sistema. Además se mostrará la descripción de los escenarios de la Plataforma de Instalación, la priorización de estos y la identificación de los requisitos de calidad de servicio.

2.1 Fase Visión

En el presente epígrafe se documentará la fase Visión, explicando profundamente los objetivos que se persiguen con el desarrollo del sistema y proporcionando una idea de cómo sería la lógica del negocio en dicho sistema. Primeramente se incluye la definición de la visión del proyecto, que establece lo que los usuarios podrán realizar cuando el proyecto lance su producto. También se incluye la configuración del equipo, de la infraestructura y de los demás recursos, así como la determinación del proceso de desarrollo.

2.1.1 Definición de las personas

Tabla 1: Descripción de las personas.

Persona	Acciones
Usuario	• Agregar módulos al paquete de instalación. • Configurar cada módulo a instalar. • Instalar el Sistema de Administración de Identidades

2.1.2 Propuesta de solución

A continuación se muestra cómo quedaría la propuesta de solución del instalador para el Sistema de Administración de Identidades.

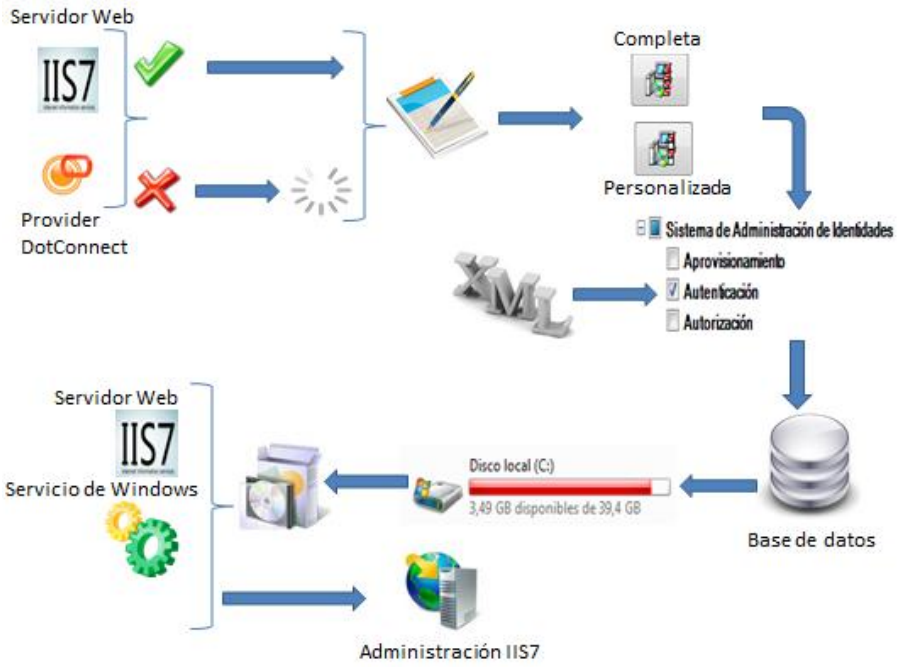


Figura 3: Propuesta de solución para la Plataforma.

Inicialmente la plataforma de instalación debe chequear los pre-requisitos de instalación del software, y proceder a instalarlos en caso que no se encuentren en el equipo. Una vez comprobado que todo esté instalado se procede a aceptar el acuerdo de licencia del software, el usuario debe aceptar para seguir la instalación en caso de no aceptarlo no permitirá continuar con el proceso de instalación. Se debe seleccionar que tipo de instalación se realizará, completa o personalizada, si selecciona la completa se instalarán todos los módulos que se encuentren disponibles para la instalación, si es la personalizada se escoge cuál de los módulos disponibles se desea instalar, estos módulos a instalar estarán disponibles desde un archivo de configuración, (**Ver Figura 19**) además se podrá seleccionar la ruta de destino donde se instalaría la plataforma de instalación.

Una vez seleccionados los subsistemas a instalar y la dirección donde se instalará el Sistema de Administración de Identidades, se pedirán credenciales para verificar la conexión a la base de datos y para crear la cadena de conexión que será guardada en los archivos de configuración de cada módulo. Se verificará que exista espacio para instalar los subsistemas seleccionados, se copiarán los archivos

al directorio de instalación y se publicarán en el servidor *Internet Information Services* (IIS) los sitios y servicios web, además si algún subsistema requiere de un servicio *Windows* se instalará también.
Seguidamente se listarán los servicios y/o sitios web publicados permitiendo cambiarle la configuración básica (nombre, puerto) y por último al finalizar el proceso de instalación se guardarán registros de la aplicación en el Registro del Sistema y se mostrará la opción de abrir la ventana de administración de IIS.
La Plataforma de instalación facilitará al usuario añadir otros subsistemas que no estén instalados, se podrán añadir estos subsistemas una vez configurado el archivo de configuración y copiados los archivos físicos en la carpeta IDM dentro del instalador siguiendo la jerarquía de directorios predeterminada del instalador. El usuario al ejecutar el programa, este verificará que ya está instalado, comprobando que en el registro de *Windows* existan los registros creados en la anterior instalación y mostrará al usuario las opciones de realizar una desinstalación total o permitir añadir nuevos componentes y desinstalar los que ya existan de forma independiente.

2.1.3 Flujo de Actividades de la Plataforma de instalación

A continuación en la **Figura # 4** se muestra cómo quedaría conformado el flujo de acciones que debe chequear el instalador para el correcto funcionamiento del SAI. Los pre-requisitos que tiene el SAI son los siguientes:

- Verificar que se encuentre instalado el *Internet Information Services* (IIS) en su versión 7 en la computadora. En caso que no esté, se procede a la instalación del mismo.
- Verificar que se encuentre instalado el *Dot Connect for Oracle 6.30*, en caso que este no esté instalado, también se efectuaría su instalación.

De esta manera quedarían instalados los pre-requisitos para el despliegue del Sistema de Administración de Identidades, que se realizaría una vez comprobado estos pre-requisitos. Una vez que estén en funcionamiento cada uno de estos, comenzaría la instalación.

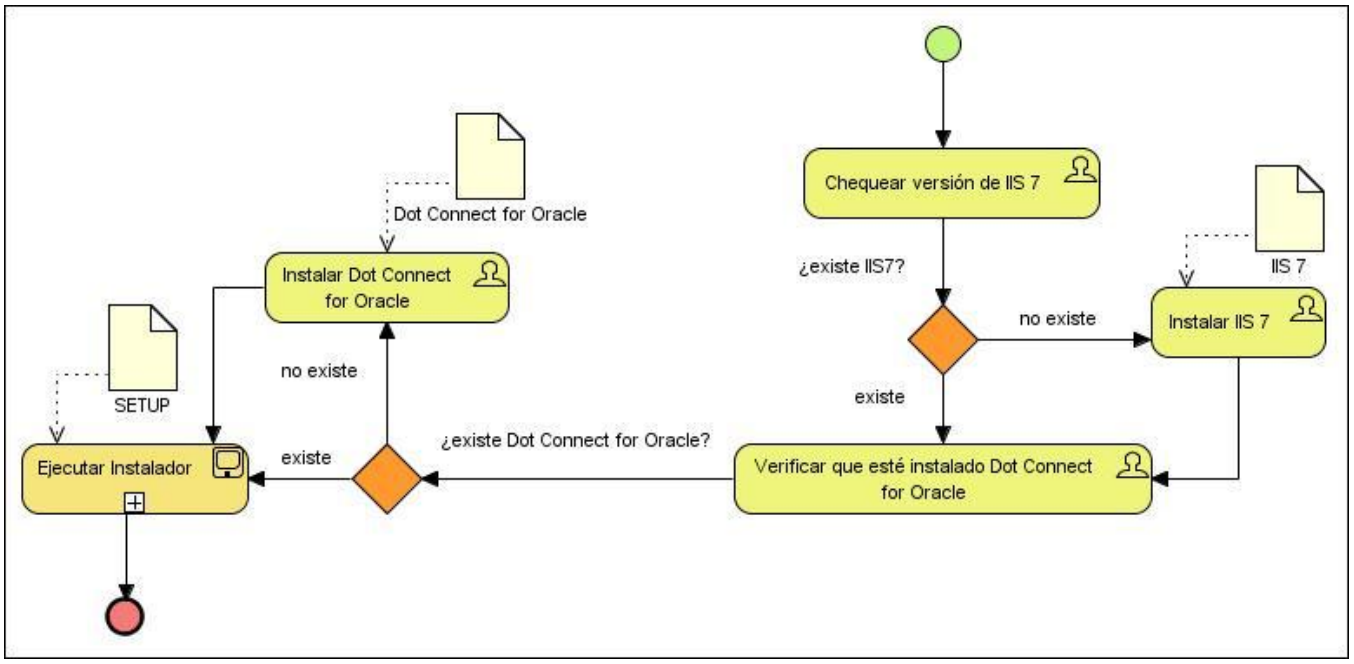


Figura 4: Flujo de instalación de los pre-requisitos.

En la siguiente figura se muestra cómo quedaría conformado el flujo de acciones que comprende la plataforma de instalación una vez ejecutado el instalador. Al ejecutar dicho instalador lo primero que se hace es chequear que los pre-requisitos mencionados anteriormente se encuentren instalados, si al verificar cada uno de estos, la plataforma detecta que hay alguno que no se encuentra en la máquina instalado pues pasaría a la instalación del mismo. Una vez verificada dicha actividad, se procede a la aceptación del acuerdo de licencia del software, en este momento el usuario debe de aceptar o no el contrato para proseguir con la instalación, si no aceptase, inmediatamente se cancelaría la instalación de la plataforma. Luego aparece una vista donde se debe seleccionar el tipo de instalación que se desee, ya sea completa o personalizada, en caso de seleccionar completa se listarían todos los módulos disponibles para la instalación, si no, se debe seleccionar qué módulos se desean instalar sin necesidad de instalarlos todos, también se selecciona la ruta destino donde se instalaría la aplicación.

Una vez seleccionado los módulos a instalar se procede a realizar el trabajo con la base de datos para crear la cadena conexión que será guardada en los archivos de configuración de cada subsistema, se verificará que exista el servidor y que las credenciales sean las correctas, en caso que esta no exista mostraría error y permitiría crear la cadena conexión con esas credenciales entradas por el usuario, además de cargar y ejecutar un *script* en la base de datos en el caso que se haya comprobado que la

conexión exista. Luego se muestra el espacio requerido por lo subsistemas a instalar y el espacio con que cuenta la máquina en el directorio donde se instalaría la aplicación, después se copian los archivos al directorio de instalación y se publican los sitios y servicios web en el servidor, se configuran los archivos de configuración de cada módulo y por último se guardarían registros de la instalación.

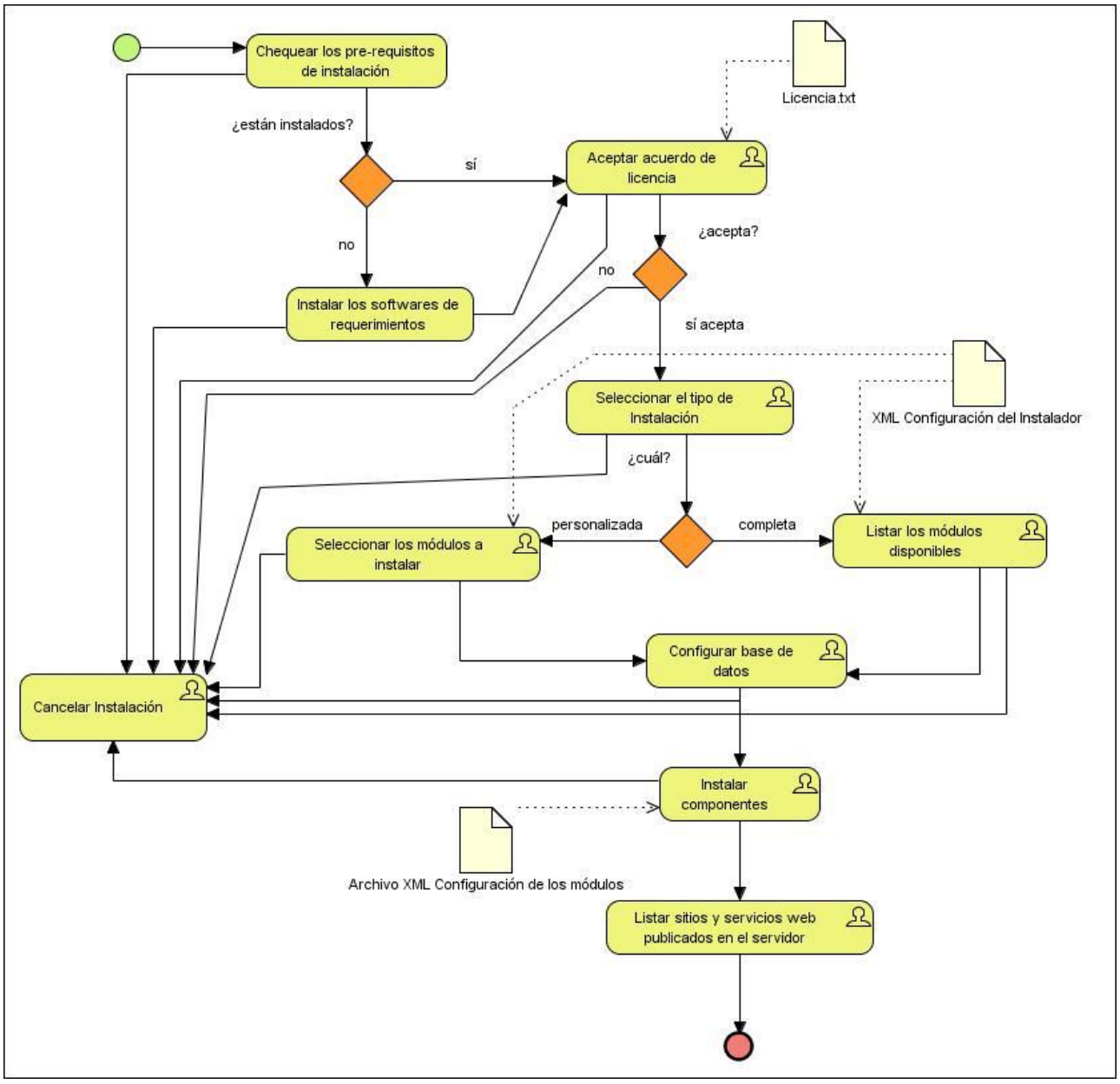


Figura 5: Flujo de Instalación de la Plataforma.

2.2 Fase de Planificación

Una vez abordada la fase de Visión del proyecto de instalación se pasa a la fase de Planificación donde se planifica y se estructura el proyecto, esta se centra en los escenarios y los requerimientos de calidad de servicio, asignando a cada escenario su prioridad, riesgo e iteración en la que se va a desarrollar, además de la descripción de cada uno de estos escenarios para lograr un mayor entendimiento del desarrollo de la aplicación.

2.2.1 Escenarios del sistema

El escenario es un medio por el cual se define una interacción de un usuario del sistema con el sistema para lograr cumplir una meta específica. Se debe por tanto exponer todas las metas del sistema, para validar todas las funcionalidades del mismo y la manera en la cual se relacionan estas funcionalidades con los usuarios. (28)

A continuación se presentan todos los escenarios que se identificaron:

Lista de Escenarios:

- ES 1: Chequear e instalar software de pre-requisitos para la instalación.
- ES 2: Mostrar acuerdo de Licencia de software.
- ES 3: Seleccionar tipo de instalación.
- ES 4: Listar y seleccionar componentes para instalar.
- ES 5: Configurar la base de datos.
- ES 6: Comprobar espacio en disco.
- ES 7: Instalar los componentes.
- ES 8: Listar los servicios y sitios web publicados en el servidor.
    - Configurar sitios y servicios web publicados.
- ES 9: Finalizar proceso de instalación.
- ES 10: Seleccionar tipo de desinstalación.
- ES 11: Listar y seleccionar componentes para desinstalar.
- ES 12: Eliminar configuración de la base de datos.
- ES 13: Desinstalar los componentes.

2.2.2 Priorización de los escenarios

Los escenarios se priorizan según la importancia que tengan cada uno de ellos en el desarrollo de la aplicación, este proceso de priorización tiene como objetivo, identificar cuáles de estos son los más importantes en el momento de implementar el sistema para que esta implementación sea en las primeras iteraciones. Clasificar al escenario con cinco puntos equivale a tener una prioridad “Alta”, con cuatro puntos “Media” y con tres puntos “Baja”.

Tabla 2: Prioridad de escenarios.

#	Escenarios	Prioridad
1	Chequear e instalar software de pre-requisitos para la instalación.	4
2	Mostrar acuerdo de Licencia de software.	3
3	Seleccionar tipo de instalación.	3
4	Listar y seleccionar componentes para instalar.	3
5	Configurar el acceso a la Base de Datos	5
6	Comprobar espacio en disco.	3
7	Instalar los componentes.	5
8	Listar los servicios y sitios web publicados en el servidor.	5
9	Finalizar proceso de instalación.	3
10	Seleccionar tipo de desinstalación.	3
11	Listar y seleccionar componentes para desinstalar.	3
12	Eliminar configuración de la base de datos.	4
13	Desinstalar los componentes.	5

2.2.3 Plan de Iteraciones

Una iteración es un conjunto de tareas que se programan para llevarse a acabo en un período de tiempo, MSF define las iteraciones como un período fijo de tiempo para programar tareas, una iteración es generalmente un período de entre dos y seis semanas. Según el plan de trabajo y fecha de entrega del proyecto de instalación se definen tres iteraciones las cuales tomarían un tiempo de quince semanas para su total desarrollo.

- Iteración #1: Se implementarán los escenarios de mayor prioridad en el Instalador, siendo necesario un total de seis semanas aproximadamente.
- Iteración #2: Se implementarán los escenarios de media prioridad en el Instalador, siendo necesario un total de cinco semanas aproximadamente.

- Iteración #3: Se implementarán los escenarios de baja prioridad en el Instalador, siendo necesario un total de cuatro semanas aproximadamente.

Tabla 3: Planificación de los escenarios.

No.	Escenarios	Prioridad	Riesgo	Esfuerzo (días)	Iteración
1	Chequear e instalar software de pre-requisitos para la instalación.	4	Medio	8	2
2	Mostrar acuerdo de Licencia de software.	3	Bajo	6	3
3	Seleccionar tipo de instalación.	3	Bajo	6	3
4	Listar y seleccionar componentes para instalar.	3	Bajo	5	3
5	Configurar el acceso a la Base de Datos.	5	Alto	10	1
6	Comprobar espacio en disco.	3	Bajo	6	3
7	Instalar los componentes.	5	Alto	10	1
8	Listar los servicios y sitios web publicados en el servidor.	5	Alto	10	1
9	Finalizar proceso de instalación.	3	Bajo	5	3
10	Seleccionar tipo de desinstalación.	3	Bajo	5	3
11	Listar y seleccionar componentes para desinstalar.	3	Bajo	6	3
12	Eliminar configuración de la base de datos.	4	Medio	8	2
13	Desinstalar los componentes.	5	Alto	10	1

2.2.4 Cronometrar escenarios

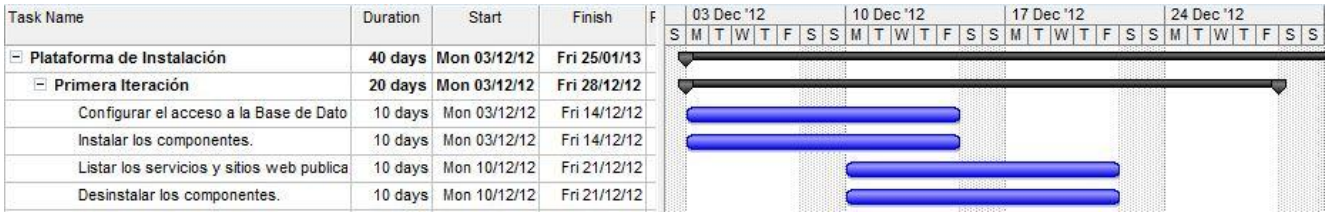
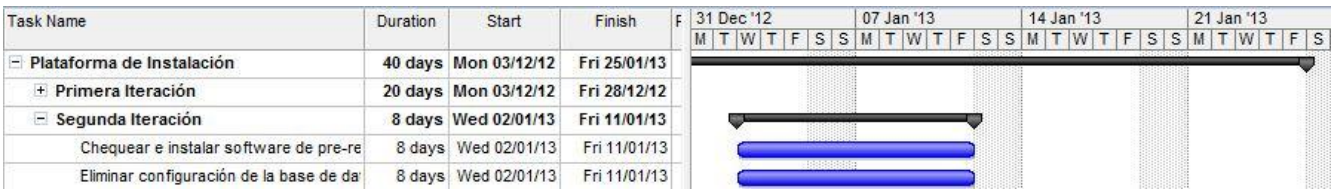
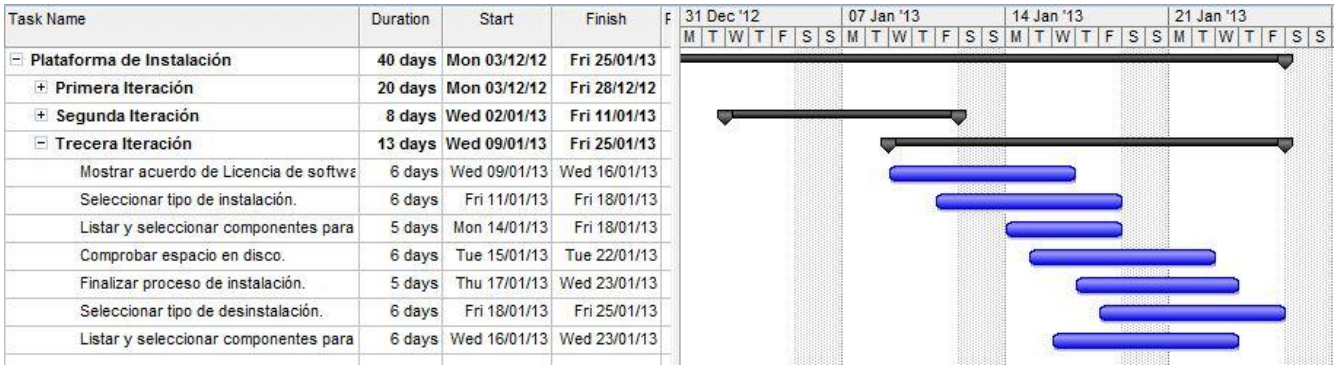


Figura 6: Primera Iteración.



Task Name	Duration	Start	Finish
Plataforma de Instalación	40 days	Mon 03/12/12	Fri 25/01/13
Primera Iteración	20 days	Mon 03/12/12	Fri 28/12/12
Segunda Iteración	8 days	Wed 02/01/13	Fri 11/01/13
Chequear e instalar software de pre-re	8 days	Wed 02/01/13	Fri 11/01/13
Eliminar configuración de la base de da	8 days	Wed 02/01/13	Fri 11/01/13

Figura 7: Segunda Iteración.



Task Name	Duration	Start	Finish
Plataforma de Instalación	40 days	Mon 03/12/12	Fri 25/01/13
Primera Iteración	20 days	Mon 03/12/12	Fri 28/12/12
Segunda Iteración	8 days	Wed 02/01/13	Fri 11/01/13
Trecera Iteración	13 days	Wed 09/01/13	Fri 25/01/13
Mostrar acuerdo de Licencia de softwa	6 days	Wed 09/01/13	Wed 16/01/13
Seleccionar tipo de instalación.	6 days	Fri 11/01/13	Fri 18/01/13
Listar y seleccionar componentes para	5 days	Mon 14/01/13	Fri 18/01/13
Comprobar espacio en disco.	6 days	Tue 15/01/13	Tue 22/01/13
Finalizar proceso de instalación.	5 days	Thu 17/01/13	Wed 23/01/13
Seleccionar tipo de desinstalación.	6 days	Fri 18/01/13	Fri 25/01/13
Listar y seleccionar componentes para	6 days	Wed 16/01/13	Wed 23/01/13

Figura 8: Tercera Iteración.

2.2.5 Requisitos de calidad de servicio

Los requisitos de calidad de servicio según la metodología utilizada definen las características tales como el rendimiento, la accesibilidad, la escalabilidad, la facilidad de mantenimiento, la usabilidad, entre otros. (28)

A continuación se listan los requisitos de calidad que se identificaron para el desarrollo del instalador para el SAI.

Restricciones de diseño:

- La plataforma debe implementarse usando el lenguaje C #.

Usabilidad:

- La plataforma de instalación será distribuida en idioma español.
- La aplicación poseerá una estructura y diseño homogéneos en todas sus ventanas, que facilite su utilización.

Plataforma que soporta:

- *Windows Server* 2008 o superior.

- *Framework* .Net 4.0.
- *Internet Information Services* (IIS) 7.
- El Servidor de Base de Datos debe ser Oracle 11G.
- *DotConnect for Oracle* 6.30.

Apariencia o interfaz externa:

- La plataforma cuenta con notificaciones que informen al usuario.

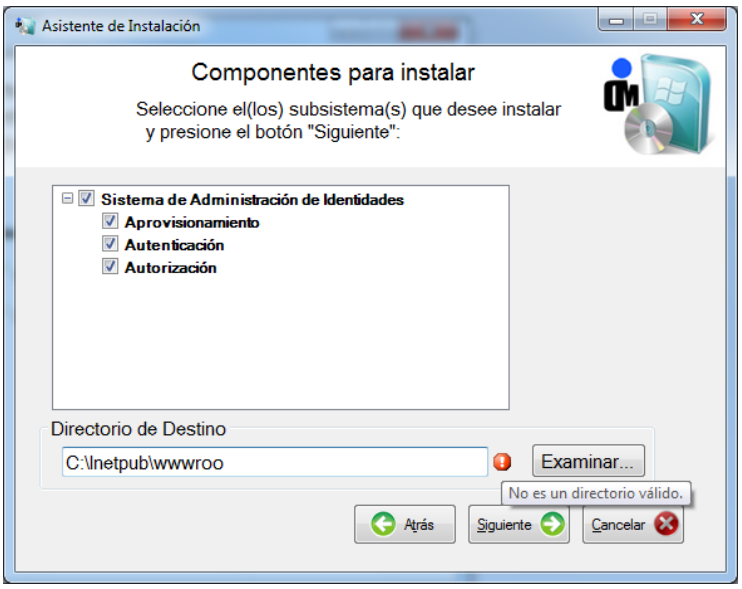
2.2.6 Descripción de los escenarios

A continuación se muestra la descripción del escenario "Listar y seleccionar componentes para instalar", los restantes escenarios se encuentran en el **Anexo 1.**

Tabla 4: Descripción del escenario: "Listar y seleccionar componentes para instalar".

Nombre del escenario: Listar y seleccionar componentes para instalar.		**Identificador:** ES4
Objetivo del escenario: Visualizar una lista de componentes que se va a instalar.		
Persona: Usuario		
Iteración: 3	**Prioridad:**3	**Complejidad:** Baja
Descripción: Se listarán los componentes disponibles para realizar la instalación de estos, en caso de que se haya seleccionado el tipo de instalación completa no se permitirá seleccionar o quitar componentes, solo se permitirá seleccionar o quitar en caso de que haya sido la instalación de tipo personalizada. Además de seleccionar el directorio de instalación del sistema.		

Prototipo de interfaz de usuario:



2.3 Conclusiones parciales

En este capítulo se realizó la descripción de la Plataforma de Instalación para el SAI, así como la planificación del mismo en las diferentes etapas que corresponde desarrollar según la metodología utilizada, para de esta manera obtener una visión más precisa de lo que se quiere lograr con dicho trabajo. También se presenta una propuesta de solución con el objetivo de definir el flujo de instalación de la plataforma de instalación, mencionándose además todas las funcionalidades que esta brindará, los requisitos de calidad de servicio y la especificación de cada uno de estos escenarios para lograr una perspectiva final del sistema. Se muestran los pre-requisitos de instalación que son aquellos que se deben cumplir antes de instalar el sistema con el objetivo de chequear inicialmente la PC donde se desee instalar la plataforma y de esta manera verificar que se cumplan estos antes de la instalación. A través de la realización de este capítulo se ha profundizado en lo que se pretende desarrollar, para tener claro todas las características que debe tener el instalador y llevar a cabo su implementación.

Capítulo 3. “Desarrollo y Estabilización”

Introducción

En el presente capítulo se muestra la arquitectura de la plataforma de instalación, los patrones de diseño que se van a utilizar en el desarrollo de la aplicación y los diferentes diagramas que propone la metodología utilizada en la fase de desarrollo y estabilización con el fin de preparar las bases para la implementación de la plataforma de instalación para el SAI. Además se efectuarán las pruebas a la aplicación con el objetivo de verificar el correcto funcionamiento del sistema y de esta forma corregir los errores encontrados.

3.1 Especificación de la arquitectura

La Arquitectura de software es la organización fundamental de un sistema encargada de sus componentes, las relaciones entre ellos, el ambiente, los principios que orientan su diseño y evolución. Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada Arquitectura de software o Arquitectura lógica. Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado. La AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes y las formas en que estos interactúan y se coordinan para alcanzar la misión del sistema.

La Arquitectura de Software es la organización fundamental de un sistema encargada de sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. (29)

El estilo arquitectural en capas son agrupaciones horizontales lógicas de componentes de software que forman la aplicación o el servicio. Ayuda a diferenciar entre los diferentes tipos de tareas a ser realizadas por los componentes, ofreciendo un diseño que maximiza la reutilización y, especialmente, la mantenibilidad. El dividir una aplicación en capas separadas que desempeñan diferentes roles y funcionalidades, ayuda a mejorar el mantenimiento del código; permite también diferentes tipos de despliegue y, sobre todo, proporciona una clara delimitación y situación de dónde debe estar cada tipo de componente funcional e incluso cada tipo de tecnología. La clave de una aplicación en N-Capas está en la

gestión de dependencias. En una arquitectura N-Capas tradicional, los componentes de una capa pueden interactuar solo con componentes de la misma capa o bien con otros componentes de capas inferiores. Esto ayuda a reducir las dependencias entre componentes de diferentes niveles.

Beneficios de uso de Capas

- El mantenimiento de mejoras en una solución será mucho más fácil porque las funciones están localizadas. Además las capas deben estar débilmente acopladas entre ellas y con alta cohesión internamente, lo cual posibilita variar de una forma sencilla diferentes implementaciones/combinaciones de capas.
- Otras soluciones deberían poder reutilizar funcionalidades expuestas por las diferentes capas, especialmente si se han diseñado para ello.
- Los desarrollos distribuidos son mucho más sencillos de implementar si el trabajo se ha distribuido previamente en diferentes capas lógicas.
- La distribución de capas en diferentes niveles físicos puede, en algunos casos, mejorar la escalabilidad. Aunque este punto hay que evaluarlo con cuidado, pues puede impactar negativamente en el rendimiento. (30)

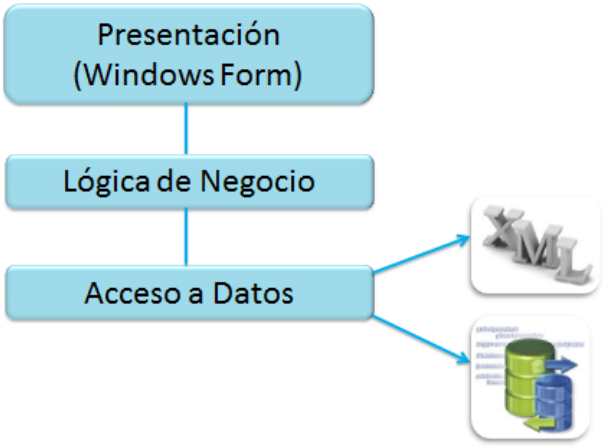


Figura 9: Arquitectura utilizada en la Plataforma de Instalación.

Capa de presentación: es la capa que da cara al usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" para el usuario. Esta capa se comunica únicamente con la capa lógica del negocio.

Capa lógica del negocio: es la capa que contiene los procesos a realizar con la información recibida desde la capa de presentación, las peticiones que el usuario ha realizado, y responsabilizándose de que se le envíen las respuestas adecuadas a la Capa de presentación.

Capa Acceso a Datos: es la capa que contiene el acceso a la base de datos y a los archivos XML, y al servidor *Internet Information Services* (IIS) y mediante esta capa es posible que si ocurriera cualquier cambio en las conexiones no se ve afectada la capa de Lógica del Negocio, ya que están separadas ambas capas.

3.2 Patrones de diseño

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

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño, para que una solución sea considerada un patrón debe poseer ciertas características, una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores, otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (31)

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas de software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente. (32)

Categorías de patrones

Según la escala o nivel de abstracción:

- Patrones de arquitectura: Aquellos que expresan un esquema organizativo estructural fundamental para sistemas de software.
- Patrones de diseño: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.
- Dialectos: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

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

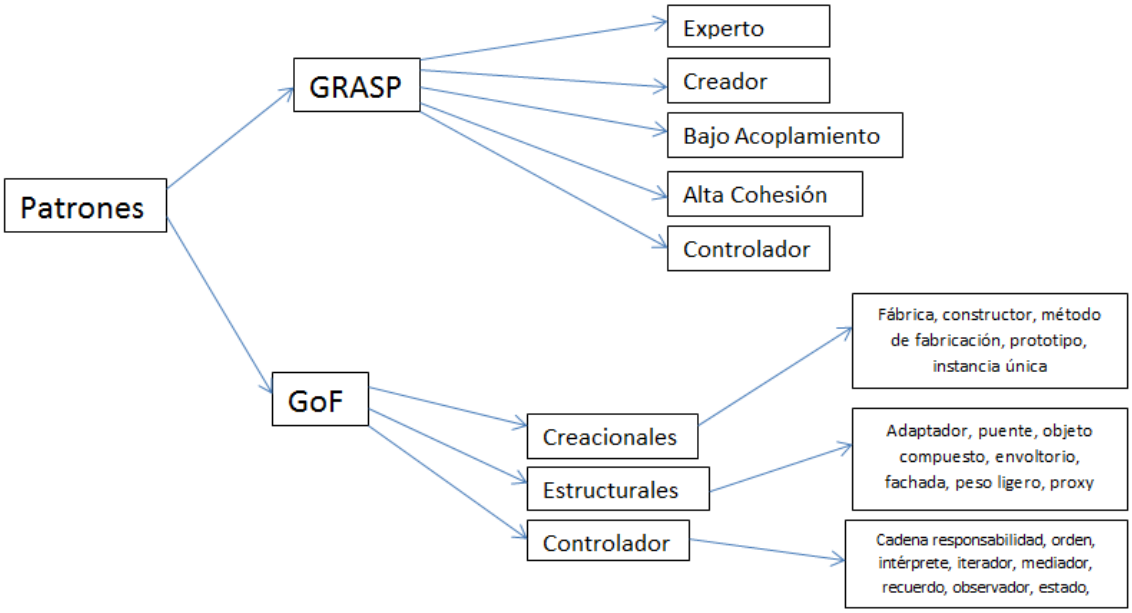


Figura 10: Relación de familias de patrones de diseño.

3.2.1 Patrones GRASP utilizados

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

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

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. GRASP es un acrónimo que significa *General Responsability Assignment software Patterns*. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos. (33)

Experto

Asignar una responsabilidad al experto en información; la clase que tiene la información necesaria para llevar a cabo la responsabilidad.

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con este no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. (33)

Este patrón se evidencia en la clase ***Connection.***

```
public class Connection
{
    private string _connectionString;
    private  OracleConnection _connectionOracle;


    public Connection(string userDb, string password, string serverDb,string sid,OracleConnectMode mode)
    {
        _connectionOracle = new OracleConnection { UserId = userDb, Password = password,
                                        Server = serverDb, Sid = sid, Direct = true, ConnectMode = mode };
        _connectionString =ConnectionOracle.ConnectionString;
        GlobalIdentity.Instance.Connection = ConnectionOracle;
    }
```

Figura 11: Ejemplo Patrón Experto.

Creador

Crear una nueva instancia por la clase que:

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

- Contiene o agrega la clase.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. En un diagrama de clases se registran las relaciones muy frecuentes entre las clases. El patrón Creador indica que la clase incluyente del contenedor o registro es idónea para asumir la responsabilidad de crear la directriz contenida o registrada. (33)

Este patrón se evidencia en la clase ***LoadConfigurationFile.***

```
var xsubsystems = from xsubsystem in _xDocument.Element("Configuration").
                        Element("System").Elements("Subsystem") select xsubsystem;

SubsystemsList = xsubsystems.Select(xsubsystem => new Subsystem(xsubsystem)).ToList();
```

Figura 12: Ejemplo de Patrón Creador.

Controlador

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

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

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

determinada (por ejemplo, un objeto similar a una ventaja) que rara vez puede utilizarse en otras aplicaciones. En cambio, el hecho de delegar a un controlador la responsabilidad de la operación de un sistema entre las clases del dominio soporta la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras. (33)

Se evidencia en la clase ***Installer***.

```
public class Installer
{
    private  List<Subsystem> _subsystemslist;
    private Requeriments _checkrequeriments;

    public Installer()...

    public List<Subsystem> SubsystemsList()...

    public Requeriments Checkrequeriments...

    public List<Subsystem> SubsystemToInstall()...

    public List<Subsystem> SubsystemToUninstall()...
```

Figura 13: Ejemplo de Patrón Controlador.

3.3 Diagrama de Aplicación

Un elemento fundamental que contiene la metodología utilizada es el diagrama de aplicación, se utiliza para modelar el hardware utilizado en las implementaciones del sistema y las relaciones entre sus componentes, muestra los elementos de despliegue así como los servicios web, aplicaciones Windows, aplicaciones web y las aplicaciones de referencia, estas pueden ser las bases de datos externas y servicios web externos. Los elementos usados por este tipo de diagrama son nodos, componentes y asociaciones. (34)

A continuación se mostrará dicho artefacto que en este caso el instalador se ejecutará en la PC cliente que será donde estará el servidor web y este se comunicará mediante el protocolo TNS[1] por sus siglas en inglés (*Transparent Network Substrate)*, con el servidor de base de datos que puede estar en la misma PC cliente o en otra diferente.

[1] TNS: *Transparent Network Substrate*, traducido al español Sustrato de red transparente, es una capa de comunicación que utilizan las bases de datos Oracle.

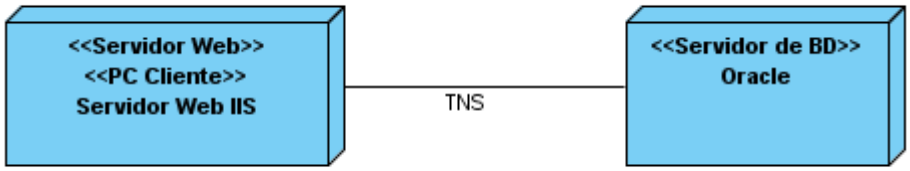


Figura 14: Diagrama de Aplicación.

3.4 Diagrama de Clases

Un diagrama de clases describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre cada uno de ellos. (35)

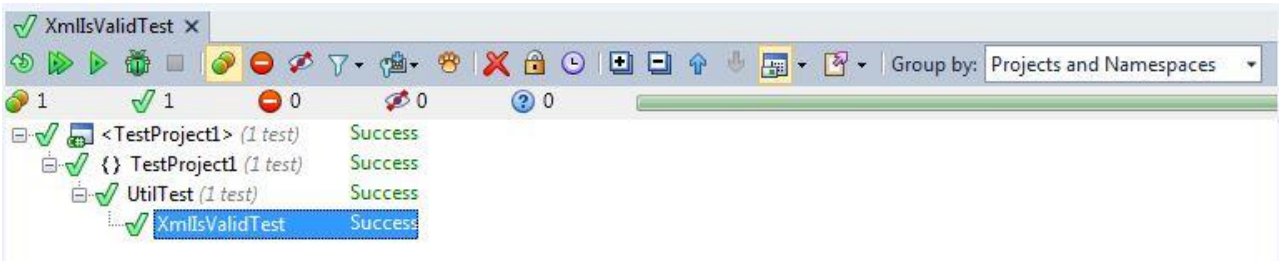
En el **Anexo 2** se mostrará dicho artefacto que este se encuentra dividido en 3 capas la de presentación, lógica del negocio y acceso a datos.

3.5 Pruebas Unitarias

En programación, una prueba unitaria es una forma de probar el correcto funcionamiento de cada unidad del código. Esto sirve para asegurar que cada una de las partes que integran la aplicación funcione correctamente por separado. Las ventajas de las pruebas unitarias frente otras formas de *test* son que, pueden aplicarse de una forma automatizada con antelación a la puesta en producción de un código. (36) Pruebas de caja blanca es también conocida como pruebas estructurales, y las pruebas de caja de vidrio, las connotaciones de "caja transparente" y "cuadro de vidrio" indican que tiene una visibilidad completa del funcionamiento interno del producto de software, en concreto, la lógica y la estructura del código. (37)

EL IDE *Visual Studio* 2010 permite realizar pruebas unitarias a los métodos, las pruebas aplicadas a algunas de las funcionalidades de la Plataforma de Instalación se muestran en el **Anexo 3**, a continuación se muestra un ejemplo de la prueba realizada al método “XmlIsValidTest”.

Tabla 5: Prueba realizada al método: "XmlIsValidTest".

Prueba de Unidad		
Nombre Prueba: XmlIsValidTest		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ejecución:** 15-5-2013
Ejecutado por: José Miguel Ruiz	**Verificado por:**	
Descripción: Para la ejecución de esta prueba se debe cargar la configuración del archivo xml e introducirlo como parámetro al método y este devolverá "true" en caso que esté correcto sino lanzará una excepción.		
Entrada: string xml		
Criterio de aceptación:		
Resultado: 		

3.6 Pruebas de Caja Negra

Las pruebas de caja negra son, pruebas funcionales dedicadas a "mirar" en el exterior de lo que se prueba. Estas pruebas se denominan de varias formas, pruebas de caja "opaca", pruebas de entrada/salida, pruebas inducidas por datos, los sinónimos son variados. Las pruebas de caja negra se centran principalmente en lo que "se quiere" de un módulo, o sección específica de un software, es decir, es una manera de encontrar casos específicos en ese módulo que atiendan a su especificación. Las pruebas de caja negra se limitan a que se pruebe con "datos" de entrada y se estudie cómo salen, sin preocuparse de lo que ocurre en el interior. (38)

Los resultados arrojados a partir de le realización de dicha prueba a la Plataforma de Instalación se encuentran en el **Anexo 4,** a continuación se muestra el caso de prueba realizado al escenario Seleccionar tipo de instalación.

Tabla 6: Descripción del caso de prueba del escenario: "Seleccionar tipo de instalación".

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar opción "Completa"	*Permitir seleccionar opción " Completa"*	*Completa*	*Muestra otra interfaz donde se listan los subsistemas disponibles para instalar, inicialmente inhabilitados.*	*Se selecciona la Opción Completa*
EC 1.3 Seleccionar opción "Personalizada"	*Permitir seleccionar opción "Personalizada"*	*Personalizada*	*Muestra otra interfaz donde se listan los subsistemas disponibles para instalar, dando la opción de seleccionar los que desee el usuario.*	*Se selecciona la Opción Personalizada*
EC 1.4 Seleccionar la opción "Atrás"	*Permitir seleccionar "Atrás"*	*Atrás 2*	*Regresa a la interfaz anterior que es la de "Acuerdo de Licencia de software"*	*Se selecciona la Opción Atrás*
EC1.5 Seleccionar la opción "Cancelar"	*Permitir seleccionar "Cancelar"*	*Cancelar 4*	*Muestra un mensaje preguntando ¿Desea cancelar la instalación?, si el usuario oprime "Si", sale de la instalación sino sigue la misma.*	*Se selecciona la Opción Cancelar*

3.6.1 Resultados de las pruebas de Caja Negra

A partir de los resultados obtenidos luego de realizadas las pruebas de caja negra a la plataforma de instalación, se le realizó tres iteraciones a cada uno de los escenarios del sistema, a continuación se muestra una tabla con los resultados obtenidos:

Tabla 7: Resultados de las pruebas de caja negra.

No Conformidades	1ra Iteración	2da Iteración	3ra Iteración
Detectadas	11	5	2
Resueltas	8	4	2
Pendientes	3	1	0

A continuación se muestra una gráfica de barras con los resultados obtenidos anteriormente:

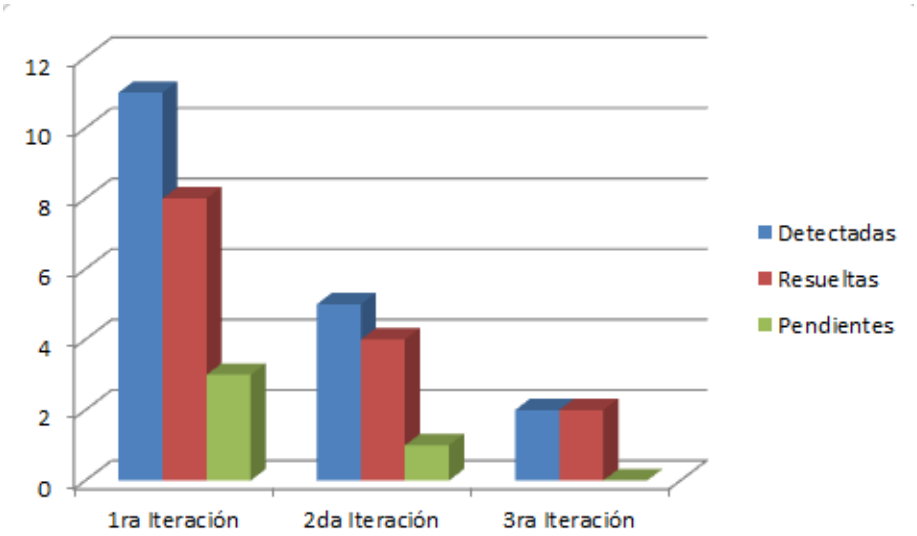


Figura 15: Gráfica con los resultados de las pruebas de caja negra.

Dichos resultados son satisfactorios ya que demuestran el avance de la implementación de la Plataforma ya que a medida que se detectaban los problemas se iban solucionando para de esa manera resolver cada error encontrado y la aplicación cumpla con los requisitos que se plantean para la misma.

3.7 Conclusiones parciales

Con la realización de este capítulo se define la estructura básica que tendrá la Plataforma de Instalación para el SAI que estará representada por una arquitectura en 3 capas ya que es la más adecuada para el trabajo que se propone realizar, además se identificaron los patrones de diseño que se utilizan en la implementación del mismo. También se muestran todos los diagramas para realizar la implementación de la aplicación, como son el diagrama de clases del sistema, el diagrama de aplicación para un mayor

entendimiento del sistema y se realizaron las pruebas pertinentes para validar el correcto funcionamiento de la plataforma de instalación. Se describieron y realizaron un conjunto de casos de pruebas a la interfaz mediante el método de Caja Negra y además se realizaron las pruebas unitarias con el fin de comprobar cada uno de los métodos del sistema, ya que estas se encargan de chequear el código fuente de la aplicación y asegurar que cada una de las partes que la integran funcionen correctamente por separado.

Conclusiones generales

El estudio realizado a diferentes plataformas de instalación existentes, permitió diseñar el flujo de instalación para la Plataforma de Instalación del SAI, ofreciendo al usuario una interfaz amigable que cumple con las especificaciones requeridas.

El análisis de cada uno de los subsistemas que conforman el SAI permitió definir los pre-requisitos de instalación de la plataforma y los requisitos para su despliegue.

El estudio realizado a las diferentes herramientas existentes en el mundo para desarrollar instaladores, permitió seleccionar cuál de estas utilizar para la implementación de la plataforma.

La metodología seleccionada permitió guiar el proceso de desarrollo de la solución mediante el uso de las herramientas y tecnologías seleccionadas.

Se desarrolló una plataforma de instalación que cumple con las especificaciones requeridas para realizar la instalación del SAI, según las características específicas del instalador, haciendo un correcto uso de los patrones de diseño y la arquitectura del sistema.

Las pruebas unitarias y de caja negra que fueron aplicadas a la plataforma de instalación permitieron verificar la calidad de la aplicación, y corregir errores existentes tanto en el código de la aplicación como en las interfaces gráficas.

Recomendaciones

- ✓ Implementar nuevas funcionalidades para la integración con otros gestores de base de datos.
- ✓ Implementar una funcionalidad para manejar la seguridad del instalador mediante una clave.

Bibliografía

1. **Cataldi, Lage, Pessacq, y García Martínez, R.** INGENIERIA DE SOFTWARE EDUCATIVO. [En línea] 2006. http://www.iidia.com.ar/rgm/comunicaciones/c-icie99-ingenieriasoftwareeducativo.pdf.

2. *Procedimiento para el despliegue de soluciones de software.* **Miranda Pardo, Daily y Tamayo Hernandez, Juniel.** La Habana : s.n., 2005.

3. [En línea] [Citado el: 1 de 3 de 2013.] http://es.scribd.com/doc/62560417/RUP-FASE-DE-TRANSICION.

4. **María A. Mendoza Sanchez.** http://www.informatizate.net - Metodologías De Desarrollo De Software. [En línea] [Citado el:] http://www.informatizate.net/articulos/metodologías_de_desarrollo_de_software_07062004.html.

5. **Paredes, Roberto José Silva.** Metodología msf. [En línea] http://www.slideshare.net/bebeyom/metodología-msf-4861508.

6. **Iván Lasso.** proyectoautodidacta. [En línea] 9 de 5 de 2008. [Citado el:] http://www.proyectoautodidacta.com/comics/%C2%BFque-es-un-instalador/.

7. *InstallJammer.* [En línea] [Citado el: 24 de 1 de 2013.] http://installjammer.com/docs/.

8. EJ Technologies. [En línea] [Citado el: 29 de 1 de 2013.] http://www.ej-technologies.com/products/install4j/overview.html

9. FlexeraSorftware. [En línea] 1 de 2 de 2013. http://www.installshield.com/.

10. BitRock Install Builder. [En línea] [Citado el: 2013 de 1 de 29.] http://installbuilder.bitrock.com/.

11. **Jordan Russell.** Inno Setup. [En línea] http://www.jrsoftware.org/isinfo.php.

12. FlexeraSoftware. [En línea] [Citado el: 30 de 1 de 2013.] http://www.flexerasoftware.com/products/installanywhere.htm

13. Visual Studio. [En línea] [Citado el: 30 de 1 de 2013.] http://www.microsoft.com/visualstudio/esn/team-foundation-service

14. Manual De Estándares Para La Instalación De Software. *BuenasTareas.com*. [Citado el: 27 de 01 de 2013] http://www.buenastareas.com/ensayos/Manual-De-Est%C3%A1ndares-Para-La Instalaci%C3%B3n/4102002.html

15. **José H. Canós, Patricio Letelier yMª Carmen Penadés.** *Métodologías Ágiles en el Desarrollo de Software.* DSIC -Universidad Politécnica de Valencia : 46022 Valencia.

16. ITSystem. [En línea] [Citado el: 21 de 1 de 2013.] http://www.itsystems.com.uy/Metodologia.aspx.

17. **María A. Mendoza Sanchez.** informatizate. Metodologías De Desarrollo De Software. [En línea] TeamSoft Perú S.A.C., 7 de 6 de 2004. [Citado el: 2 de 1 de 2013.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

18. Inet Consulting Services. [En línea] 2011. [Citado el: 12 de 1 de 2013.] http://www.inet.com.sv/metodologia.asp.

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

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

21. Visual Paradigm for UML . [En línea] [Citado el: 1 de 1 de 2013.] http://www.visual-paradigm.com/product/vpuml/.

22. Microsoft, Información general y conceptual sobre .NET Framework. [En línea] [Citado el: 1 de 2 de 2013.] http://msdn.microsoft.com/es-es/library/zw4w595w%28v=vs.80%29.aspx.

23. Microsoft, Guía de desarrollo para .NET Framework. [En línea] [Citado el: 2 de 12 de 2012.] http://msdn.microsoft.com/es-es/library/hh156542.aspx.

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

25. **Ruiz, Diego.** *C# La Guia Total Del Programador.* s.l. : MP-Ediciones SA, 2005.

26. **Ray, Erik T.** *Learning XML.* s.l. : O'Reilly Media, 2003.

27. **Brochard, Johnny.** *XML: conceptos e implementación.* Barcelona : ENI-Ediciones, 2003.

28. **RAMIRO, VILLARROEL GONZALEZ LUIS.** *APLICACIÓN DE LA METODOLOGÍA MSF V4.0 A LA.* Sangolquí : 2008.

29. **Reynoso, Carlos Billy.** *Introducción a la Arquitectura de Software.* BUENOS AIRES : s.n., 2004.

30. **Cesar de la Torre Llorente, Unai Zorrilla Castro, Javier Calvarro Nelson, Miguel Anglel Ramos Barroso.** *Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0.* España : KRASIS PRESS, 2010. Microsoft Ibérica S.R.L..

31. **Nicolás Tedeschi.** Microsoft, ¿Qué es un Patrón de Diseño? . [En línea] [Citado el: 21 de 12 de 2012.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

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

33. **Larman, Craig.** *UML y Patrones Introducción al análisis y diseño orientado a objetos.*

34. Microsoft, Definir aplicaciones en diagramas de aplicaciones. [En línea] 2013. [Citado el: 2 de 12 de 2012.] http://msdn.microsoft.com/es-es/library/ms181832%28v=vs.80%29.aspx.

35. **Weitzenfeld, Alfredo.** *Ingeniería de software orientada a objetos con UML.* s.l. : Thomson, 2009.

36. Microsoft, Trabajar con pruebas unitarias. [En línea] 2013. [Citado el: 11 de 12 de 2012.] http://msdn.microsoft.com/es-es/library/ms182515%28v=vs.80%29.aspx.

37. **Srinivasan Desikan, Gopalaswamy Ramesh.** *Software Testing: Principles and Practices.* New Delhi, India : Pearson Education, 2008.

38. **Adám, Victor Gomez.** Globe Testing. [En línea] http://www.globetesting.com/2012/08/pruebas-de-caja-negra/.

Bibliografía consultada

1. **Hielscher, Jose Daniel Moñoz Frías y Rafael Palacios.** *Fundamentos de la programación utilizando el lenguaje C.* España : Servicios editoriales S.L., 2006.

2. Instalaciones silenciosas. [En línea] http://foros.softonic.com/software/.

3. Artículo de Oracle Oracle Database 11g en Windows. *Desarrollo e Implementación.* [En línea] Septiembre de 2010.

4. Paquetes MSI. [En línea] http://www.installshield.com.

5. Instalaciones Silenciosas. [En línea] http://www.psicofxp.com/forums/programacion.313/244849-instalacion-silenciosa-msxml3-0-a.htm.

6. Visual C#. [En línea] http://msdn.microsoft.com/es-es/library/kx37x362(v=vs.80).aspx.

7. Crear Delegate. [En línea] http://msdn.microsoft.com/es-es/library/ms173171(v=vs.80).aspx.

8. **Birnios, Mariano.** *Microsoft Visual Basic. NET Guía Del Programador.* s.l. : MP Ediciones S.A, 2007.

9. **R. Powell, Richard L. Weeks.** *C Sharp and the .NET Framework: The C++ Perspective.* 2004.

10. **Ramírez, José Felipe Ramírez.** *Aprenda Visual Basic Practicando.* s.l. : Pearson Educación de Mexico, 2006.

11. **Team, Microsoft IIS.** *Internet Information Services (IIS) 6 Resource Kit.* 2004.

12. **Bai, Ying.** *Practical Database Programming with Visual Basic.NET.* Canadá : IEEE PRESS, 2012.

13. **Zehoo, Edmund.** *Pro ODP.NET for Oracle Database 11g.* New York : APress, 2010.

14. **Groussard, Thierry.** *Recursos Informáticos C# 4 - Los fundamentos del lenguaje - Desarrollar con Visual Studio 2010.* Barcelona : ENI Ediciones, 2011.

15. **Consulting, Krasis.** *Guía de Arquitectura N-Capas Orientada Al Dominio Con .Net 4.0.* 2011.

16. **Kroenke, David.** *Procesamiento de bases de datos: fundamentos, diseño e implementación.* México : Pearson Educación, 2004.

17. **Campderrich, Benet.** *Ingeniería de software.* Eureca Media : UOC, 2003.

18. **María A. Mendoza Sanchez.** http://www.informatizate.net - Metodologías De Desarrollo De Software. [En línea] [Citado el:] http://www.informatizate.net/articulos/metodologías_de_desarrollo_de_software_07062004.html.

19. UML CASE. *Tool for software development.* [En línea] [Citado el: 27 de Enero de 2013.] http://www.visual-paradigm.com/product/vpuml/.

20. Microsoft. [En línea] http://msdn.microsoft.com/es-es/library/aa287554%28v=vs.71%29.aspx.

21. W3e. [En línea] [Citado el: 10 de Enero de 2013.] http://www.w3.org/XML/..

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

23. **Larman, Craig.** *UML y Patrones Introducción al análisis y diseño orientado a objetos.*

24. **RAMIRO, VILLARROEL GONZALEZ LUIS.** *APLICACIÓN DE LA METODOLOGÍA MSF V4.0 A LA.* Sangolquí : 2008.

25. **Reynoso, Carlos Billy.** *Introducción a la Arquitectura de Software.* BUENOS AIRES : s.n., 2004.

26. **Kurt Bittner, IAN AUTOR SPENCE.** *Use Case Modeling.* Boston : Pearson Education,INC, 2003.

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

Glosario de Términos

A

ActiveX: Es una tecnología de Microsoft que habilita diferentes aplicaciones para compartir funcionalidades a través de Internet. Los programas creados usando *ActiveX* (también llamados controles *ActiveX*) pueden ser agregados a las web y realizar diferentes funciones.

Authenticode: Es una tecnología de Microsoft que utiliza criptografía estándar del sector para firmar el código de aplicación con certificados digitales que comprueban la autenticidad del editor de la aplicación. Utilizando *Authenticode* para la implementación de aplicaciones, ayuda a evitar los fenómenos de los "caballos de Troya", en los que terceras personas malintencionadas hacen parecer que un virus u otro programa dañino es un programa legítimo de una fuente solvente y de confianza.

B

Borland Delphi: Es un entorno de desarrollo flexible y potente. Además es intérprete de un lenguaje llamado *Object* Pascal. La programación Delphi no es solo un intérprete, sino que incluye otras herramientas para facilitar la escritura del código y el diseño de la aplicación.

C

CLR: *Common Language Runtime* ("entorno en tiempo de ejecución de lenguaje común") es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET. El CLR es el encargado de compilar una forma de código intermedio llamada *Common Intermediate Language* mediante un compilador en tiempo de ejecución.

CAB: Abreviatura de *Cabinet*, es el formato nativo de archivo comprimido de *Microsoft Windows*. Soporta compresión y firma digital, y se utiliza en una variedad de motores de instalación de Microsoft: Setup API, *Device Installer*, AdvPack (para la instalación de componentes ActiveX de *Internet Explorer*) y *Windows Installer.*

CASE: (*Computer Aided software Engineering*, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto.

H

HP-UX: Es la versión de Unix desarrollada y mantenida por Hewlett-Packard desde 1983, ejecutable típicamente sobre procesadores HP PA RISC y en sus últimas versiones sobre Intel Itanium.

HTML: Siglas de *HyperText Markup Language* (lenguaje de marcado hipertexto), hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes.

I

IIS: Es un servidor web y un conjunto de servicios para el sistema operativo *Microsoft Windows*. Este servicio convierte a una PC en un servidor web para Internet o una intranet, es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente. Se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas.

J

Java: Es un lenguaje de programación y la primera plataforma informática creada por *Sun Microsystems* en 1995. Es la tecnología subyacente que permite el uso de programas punteros, como herramientas, juegos y aplicaciones de negocios.

JRE: *Java Runtime Environment* es una máquina virtual de Java y su función es hacer de intermediario entre una aplicación programada en Java y el sistema operativo que se esté usando. De este modo, cualquier aplicación puede funcionar en cualquier sistema operativo que disponga del JRE.

M

MSI: Los paquetes MSI (*Microsoft Installer*) se definen como instaladores de Microsoft, a saber, aquellos paquetes de software que contienen la información necesaria para automatizar su instalación, minimizando la intervención manual del usuario, ya que toda la información iría contenida en el propio fichero "msi".

Metalenguaje: Es un lenguaje que se usa para hablar acerca de otro lenguaje. Al lenguaje acerca del cual se está hablando se le llama el lenguaje objeto. El metalenguaje puede ser idéntico al lenguaje objeto.

N

.NET: Es un *framework* de Microsoft que hace énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permite un rápido desarrollo de aplicaciones.

P

***PowerShell*:** Es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible. Dicho sistema está definido por un estándar alrededor del cual se determina una arquitectura de hardware y una plataforma de software.

PDF: Sigla del inglés *portable document format*, formato de documento portable, es un formato de almacenamiento de documentos digitales independiente de plataformas de software o hardware.

R

***Releases*:** Nueva versión de una aplicación informática. Liberar, publicar, lanzar.

RUP: *Rational Unified Process* en inglés, es un proceso de desarrollo de *software* desarrollado por la empresa *Rational software*, actualmente propiedad de IBM. Junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

S

***Suite*:** Es un conjunto de programas de software que permite realizar distintas tareas típicas del mundo laboral. Redactar documentos, realizar balances contables e imprimir información forman parte de las funciones habituales de una *suite*. Esta recopilación de programas informáticas se desarrolla para brindar mayor comodidad al usuario, que no debe comprar cada programa por separado.

***Scripts*:** Un *script* es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de comandos y usualmente son archivos de texto.

T

TNS: *Transparent Network Substrate*, es una capa de comunicación que utilizan las bases de datos Oracle. TNS Service Name es el nombre por el que se conocen las instancias de una base de datos Oracle en una red. El nombre de servicio TNS se asigna al configurar la conectividad a la base de datos de Oracle. La replicación utiliza el nombre de servicio TNS para identificar al suscriptor y establecer las conexiones.

U

UML: Lenguaje Unificado de Modelado por sus siglas en inglés, *Unified Modeling Language*, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (*Object Management Group*). Es un lenguaje gráfico para visualizar, especificar, construir y

documentar un sistema. UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

W

***Windows PowerShell*:** Es una interfaz de consola (CLI) con posibilidad de escritura y conjunción de comandos por medio de guiones (*scripts* en inglés). Esta interfaz de consola está diseñada para su uso por parte de administradores de sistemas, con el propósito de automatizar tareas o realizarlas de forma más controlada. Es un intérprete de comandos orientado a objetos. La información de entrada y de salida en cada etapa del proceso es un conjunto de instancias de objeto, a diferencia de lo que ocurre con los intérpretes de comandos tradicionales, que sólo devuelven y reciben texto.

WWW: Es un sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de Internet. Con un navegador web, un usuario visualiza sitios web compuestos de páginas web que pueden contener texto, imágenes, videos u otros contenidos multimedia, y navega a través de esas páginas usando hiperenlaces.

Anexos

Anexo1: Descripción de los escenarios.

Tabla 8: Descripción del escenario: "Chequear e instalar software de pre-requisitos de instalación".

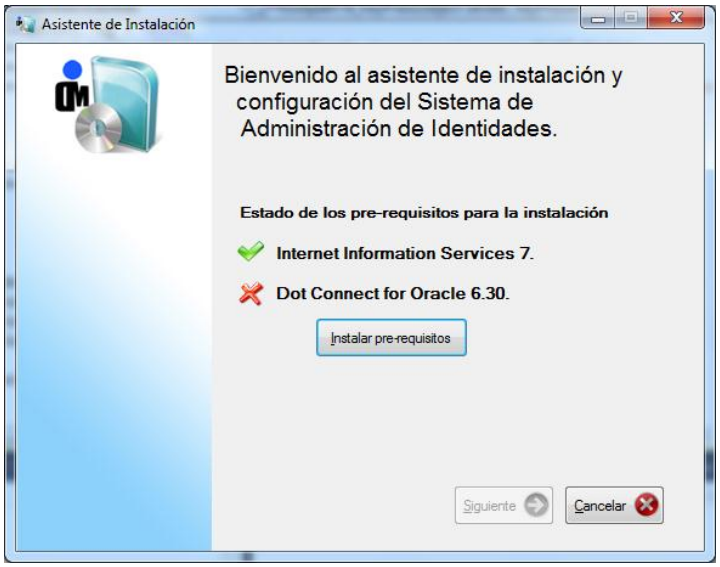
Nombre del escenario: Chequear e instalar software de pre-requisitos para la instalación.		**Identificador:** ES1
Objetivo del escenario: Chequear los pre-requisitos de instalación y en caso que no estén instalados instalarlos en la máquina.		
Persona: Usuario		
Iteración: 2	**Prioridad:**4	**Complejidad:** Media
Descripción: Comprobar que el equipo cumple con todos los pre-requisitos de instalación e instalar el software necesario en cado de que no cumpla con alguno.		
Prototipo de interfaz de usuario: 		

Tabla 9: Descripción del escenario: "Seleccionar tipo de instalación".

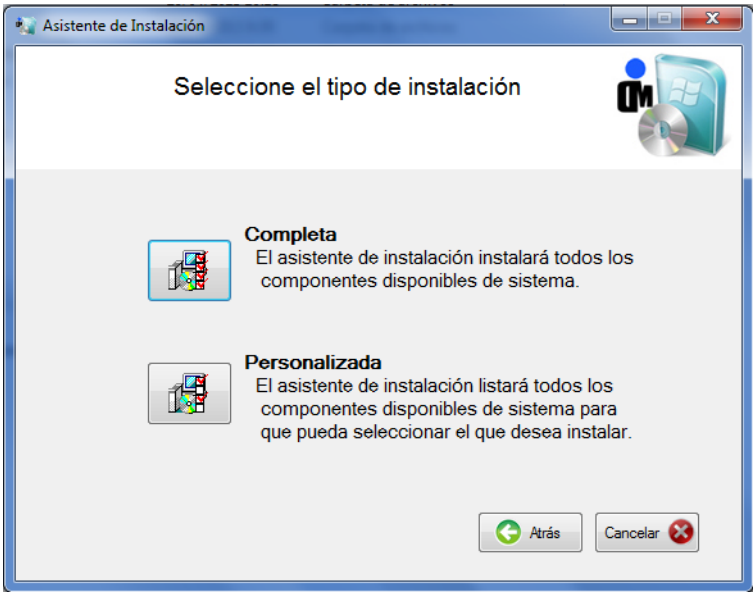
Nombre del escenario: Seleccionar tipo de instalación.		**Identificador:** ES3
Objetivo del escenario: Elegir el tipo de instalación que se desee realizar.		
Persona: Usuario		
Iteración: 3	**Prioridad:**3	**Complejidad:** Baja
Descripción: El administrador que será el usuario que realizará la instalación al llegar a esta paso hará la selección del tipo de instalación que desee realizar mediante un botón para la instalación completa y otro botón para la personalizada.		
Prototipo de interfaz de usuario: 		

Tabla 10: Descripción del escenario: "Mostrar acuerdo de Licencia de *software*".

Nombre del escenario: Mostrar acuerdo de Licencia de software.	**Identificador:** ES 2
Objetivo del escenario: Mostrar al usuario que esté realizando la instalación, el acuerdo de licencia de software.	

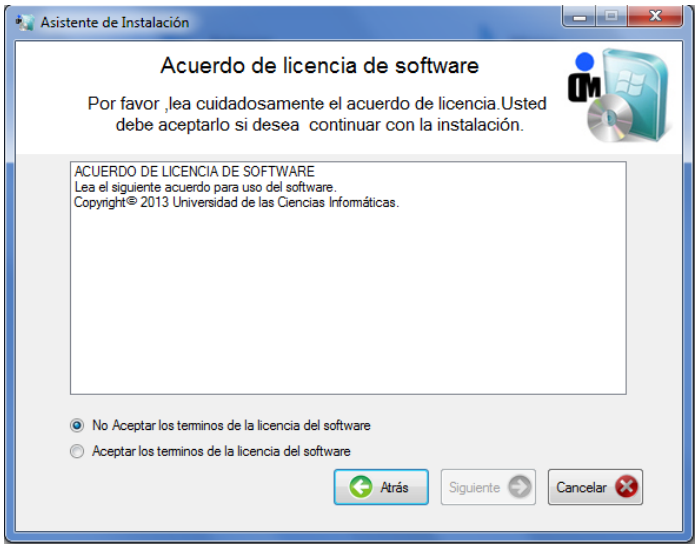
Persona: Administrador		
Iteración: 3	**Prioridad:**3	**Complejidad:** Baja
Descripción: Se mostrará el acuerdo de licencia del software que se instalará, el usuario que está realizando la instalación deberá leer el acuerdo para que se active el control (*radiobutton*) al cual está asociada la acción, de estar de acuerdo con dicha licencia puede continuar con la secuencia de instalación.		
Prototipo de interfaz de usuario: 		

Tabla 11: Descripción del escenario: "Configurar la base de datos".

Nombre del escenario: Configurar la base de datos		**Identificador:** ES5
Objetivo del escenario: Crear la cadena conexión que será guardada en los archivos de configuración de los subsistemas.		
Persona: Usuario		
Iteración: 1	**Prioridad:**5	**Complejidad:** Alta

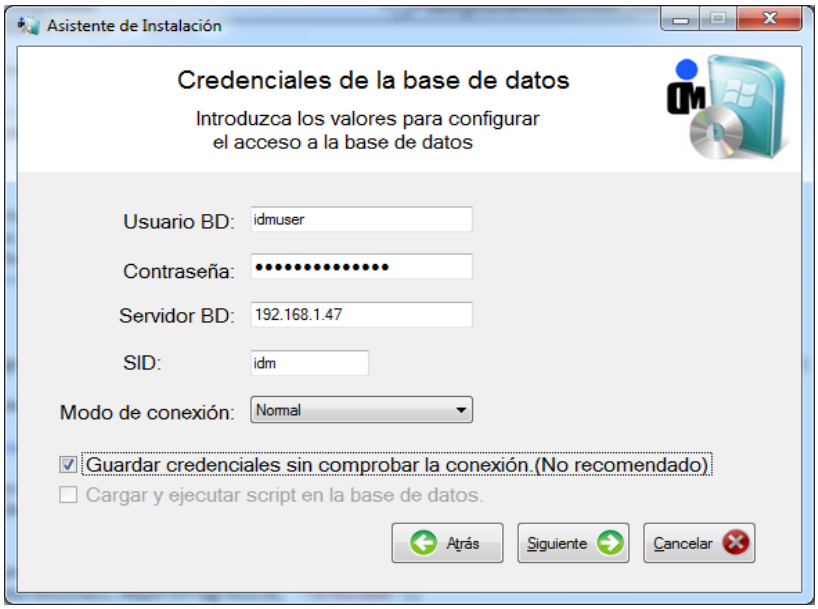
Descripción: Se mostrarán los campos para que el usuario pueda introducir las credenciales de conexión a la base de datos, además permitirá cargar y ejecutar un *script* en la base de datos, en caso de que no se pueda comprobar la conexión el usuario podrá seleccionar un campo para que se guarden las credenciales sin comprobar conexión, en este caso no se podrá ejecutar el *script*.
Validaciones: La conexión a la base de datos debe estar habilitada.
Prototipo de interfaz de usuario: 

Tabla 12: Descripción del escenario: "Comprobar espacio en disco".

Nombre del escenario: Comprobar espacio en disco.		**Identificador:** ES6
Objetivo del escenario: Comprobar que exista espacio para instalar en el directorio de instalación.		
Persona: Usuario		
Iteración: 3	**Prioridad:**3	**Complejidad:** Baja
Descripción: Se mostrará en la pantalla información respecto al espacio que ocupan los subsistemas que se instalarán y el espacio disponible para realizar la instalación.		
Validaciones: La conexión a la base de datos debe estar habilitada.		

Prototipo de interfaz de usuario:

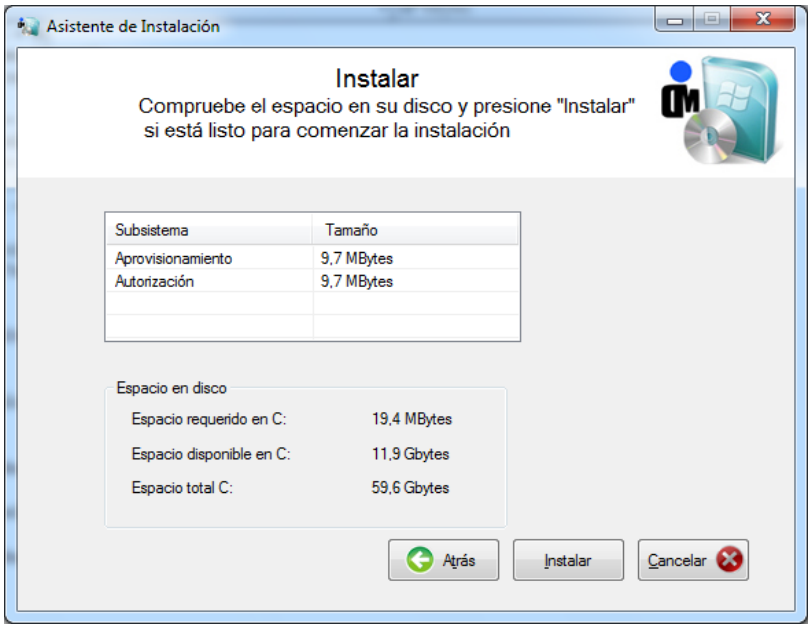


Tabla 13: Descripción del escenario: "Instalar los componentes".

Nombre del escenario: Instalar los componentes.		**Identificador:** ES6
Objetivo del escenario: Instalar los componentes en sus respectivos directorios.		
Persona: Usuario		
Iteración: 1	**Prioridad:**5	**Complejidad:** Alta
Descripción: Se copiarán los archivos, se publicarán los servicios y sitios web en el servidor *Internet Information Services* 7, se guardará la cadena conexión en los archivos de configuración y se guardarán registros de la aplicación el registro de *Windows*.		

Prototipo de interfaz de usuario:

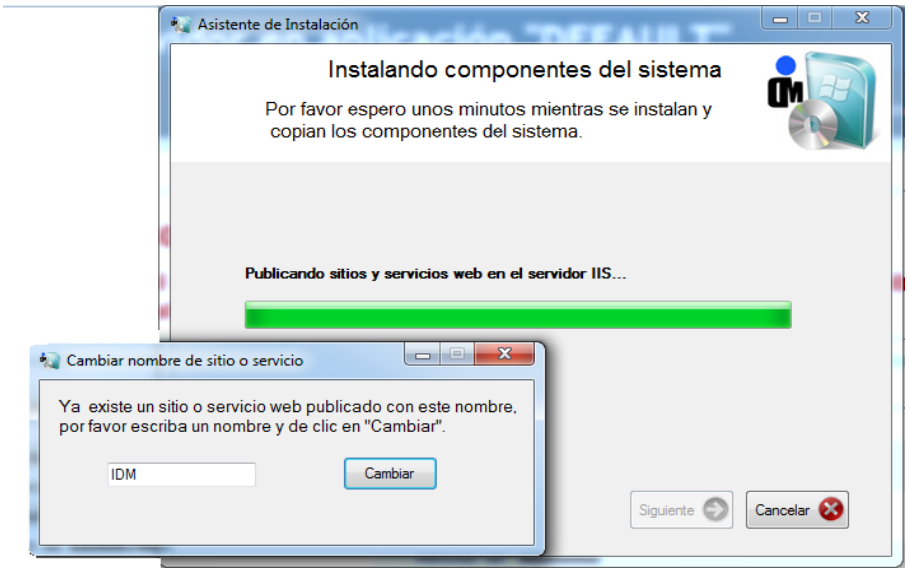


Tabla 14. Descripción del escenario: "Listar los servicios y sitios web publicados en el servidor".

Nombre del escenario: Listar los servicios y sitios web publicados en el servidor.		**Identificador:** ES7
Objetivo del escenario: Permite cambiarle la configuración básica (nombre y puerto) de los sitios web publicados.		
Persona: Usuario		
Iteración: 1	**Prioridad:**5	**Complejidad:** Alta
Descripción: Se mostrará un listado de todos los sitios web publicados en el servidor, permitiendo cambiarle la configuración al hacer clic derecho sobre el nombre de cada sitio web.		
Validaciones:		

Prototipo de interfaz de usuario:

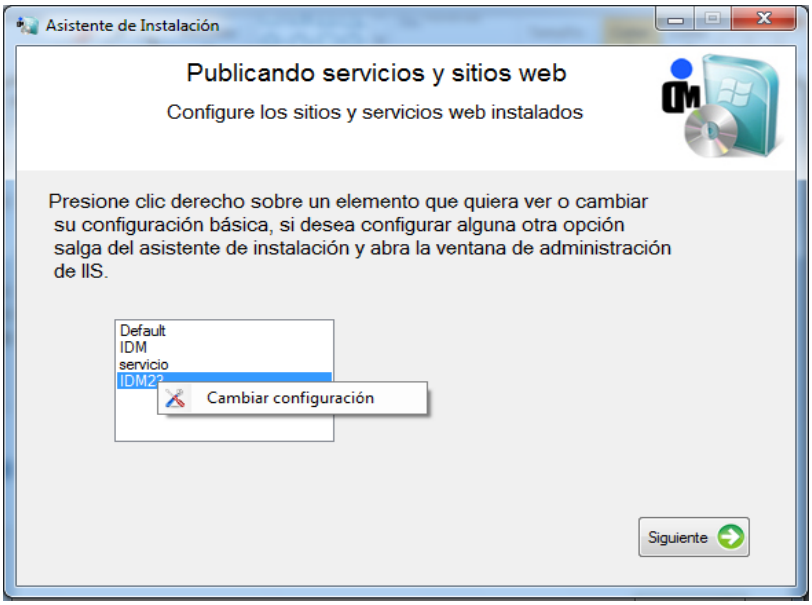


Tabla 15: Descripción del escenario: "Finalizar proceso de instalación".

Nombre del escenario: Finalizar proceso de instalación.		**Identificador:** ES8
Objetivo del escenario: Terminar el proceso de instalación.		
Persona: Usuario		
Iteración: 3	**Prioridad:**3	**Complejidad:** Baja
Precondiciones:		
Descripción: Se mostrará un campo para marcar en caso de que el usuario desea abrir la venta de administración del servidor *IIS* al presionar el botón "Finalizar", además de cerrar el asistente de instalación.		
Validaciones:		

Prototipo de interfaz de usuario:

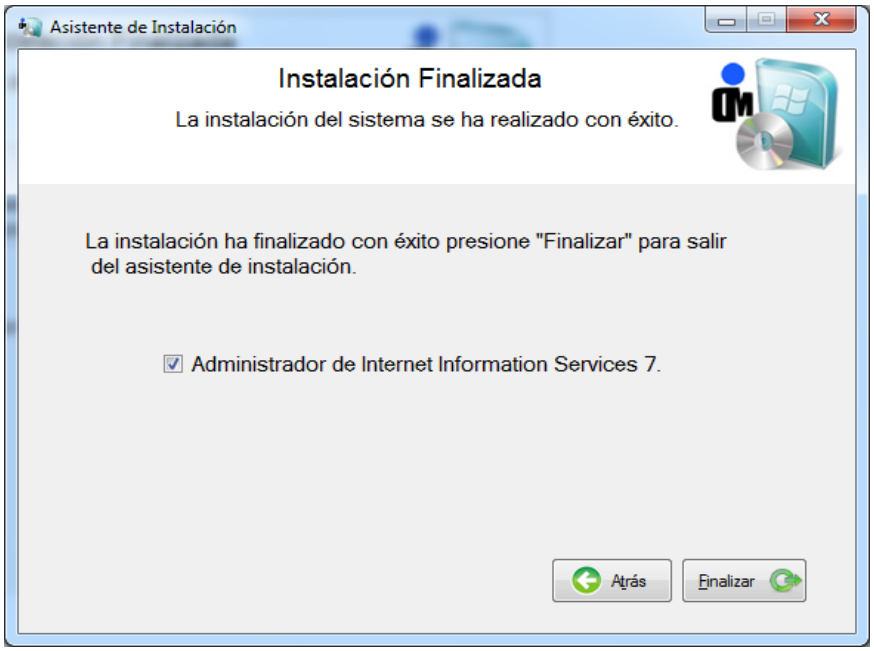


Tabla 16: Descripción del escenario: "Seleccionar tipo de desinstalación".

Nombre del escenario: Seleccionar tipo de desinstalación.		**Identificador:** ES9
Objetivo del escenario: Seleccionar el tipo de desinstalación que se desea realizar.		
Persona: Usuario		
Iteración: 3	**Prioridad:**3	**Complejidad:** Baja
Precondiciones:		
Descripción: Se comprueba que esté instalado el sistema en el equipo y se visualizan 2 botones en el cual se puede escoger la opción de desinstalar completamente o personalizada.		
Validaciones:		

Prototipo de interfaz de usuario:

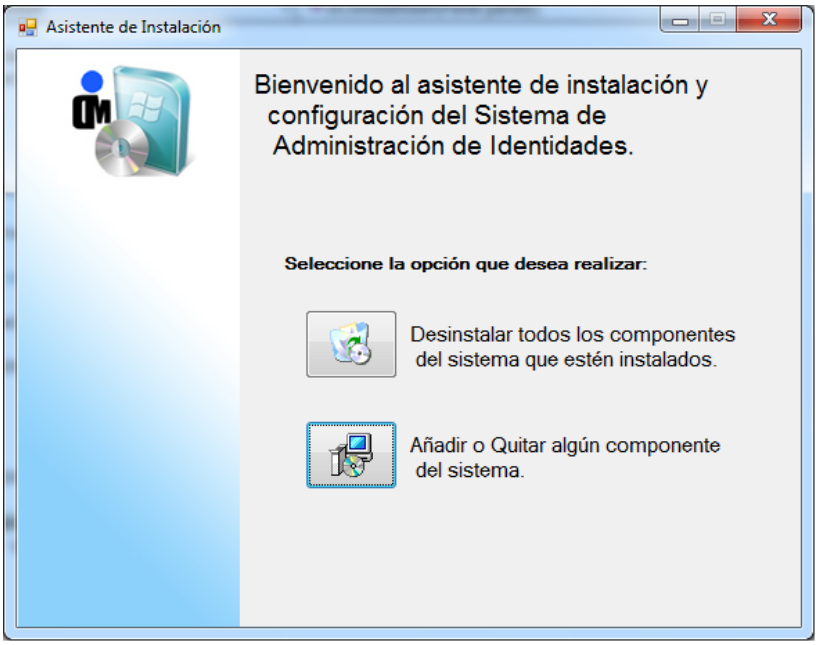


Tabla 17: Descripción del escenario: "Listar y seleccionar componentes para desinstalar".

Nombre del escenario: Listar y seleccionar componentes para desinstalar.		**Identificador:** ES10
Objetivo del escenario:		
Persona: Usuario		
Iteración: 3	**Prioridad:**3	**Complejidad:** Baja
Precondiciones:		
Descripción: Se listarán los componentes que estén instalados en el equipo y los nuevos que se encuentren disponibles para instalar, dará la opción de seleccionar el componente que desee desinstalar en el caso que el tipo de desinstalación sea personalizada, no permitiendo realizar la acción de seleccionar los componentes en el caso de que sea desinstalación completa.		
Validaciones:		

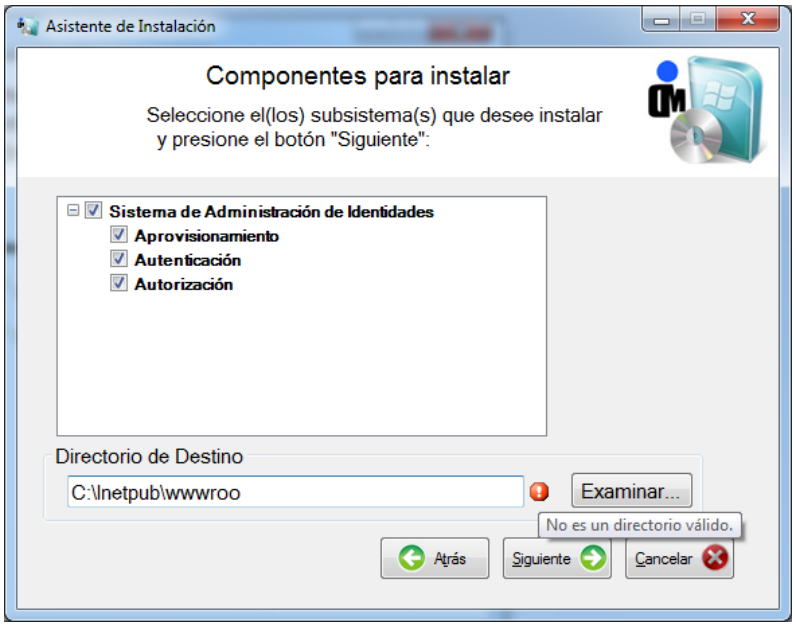
Prototipo de interfaz de usuario:


Tabla 18: Descripción del escenario: "Eliminar configuración de la base de datos.".

Nombre del escenario: Eliminar configuración de la base de datos.		**Identificador:** ES11
Objetivo del escenario: Permitir eliminar o no la base de datos.		
Persona: Usuario		
Iteración: 2	**Prioridad:**4	**Complejidad:** Media
Descripción: Se le mostrará al usuario para que pueda seleccionar si desea eliminar la base de datos o no, en caso de que quiera eliminarla se le pedirán credenciales de conexión a la base de datos.		
Validaciones:		
Prototipo de interfaz de usuario:		

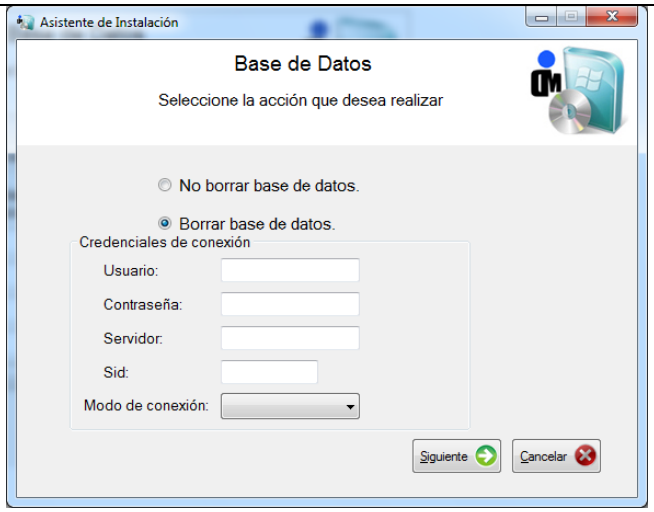
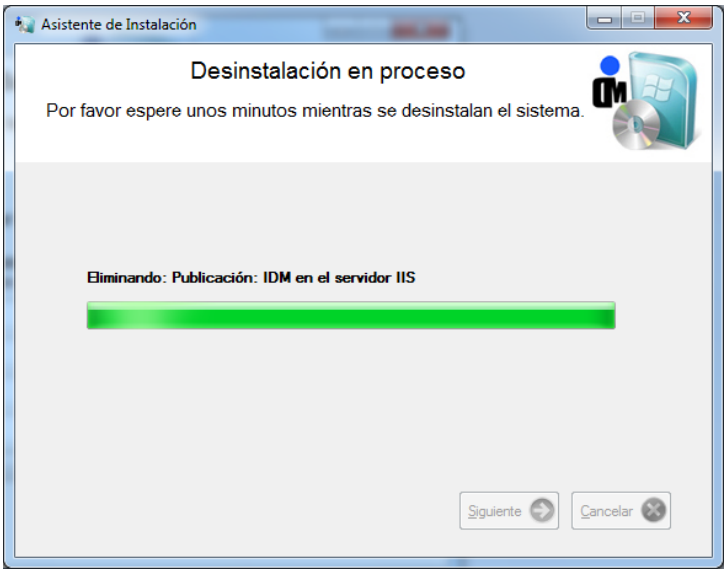


Tabla 19: Descripción del escenario: "Desinstalar los componentes".

Nombre del escenario: Desinstalar los componentes.		**Identificador:** ES12
Objetivo del escenario: Permitir la desinstalación de los componentes e instalación de los nuevos.		
Persona: Usuario		
Iteración: 1	**Prioridad:**5	**Complejidad:** Alta
Precondiciones:		
Descripción: Se borrarán todos los archivos de cada componente que se vaya a desinstalar y se instalarán los que haya seleccionado el usuario que no estaban instalados.		
Validaciones:		

Prototipo de interfaz de usuario:



Anexo 2: Diagrama de Clases

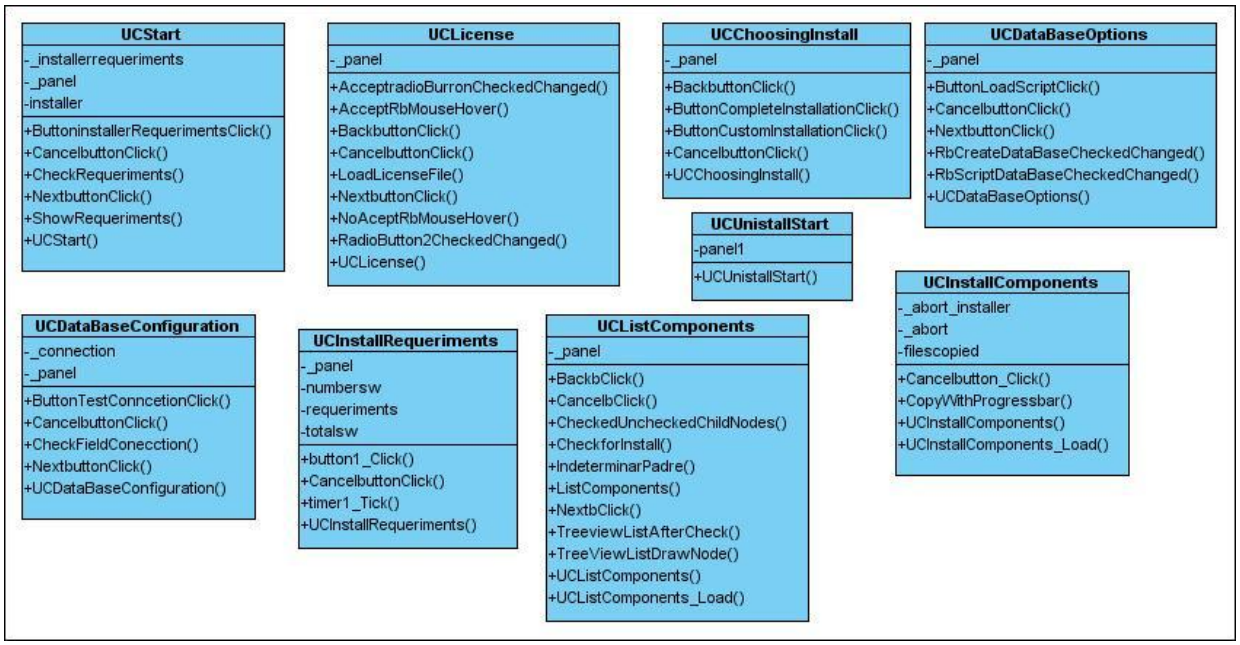


Figura 16: Diagrama de clases de la Capa de presentación.

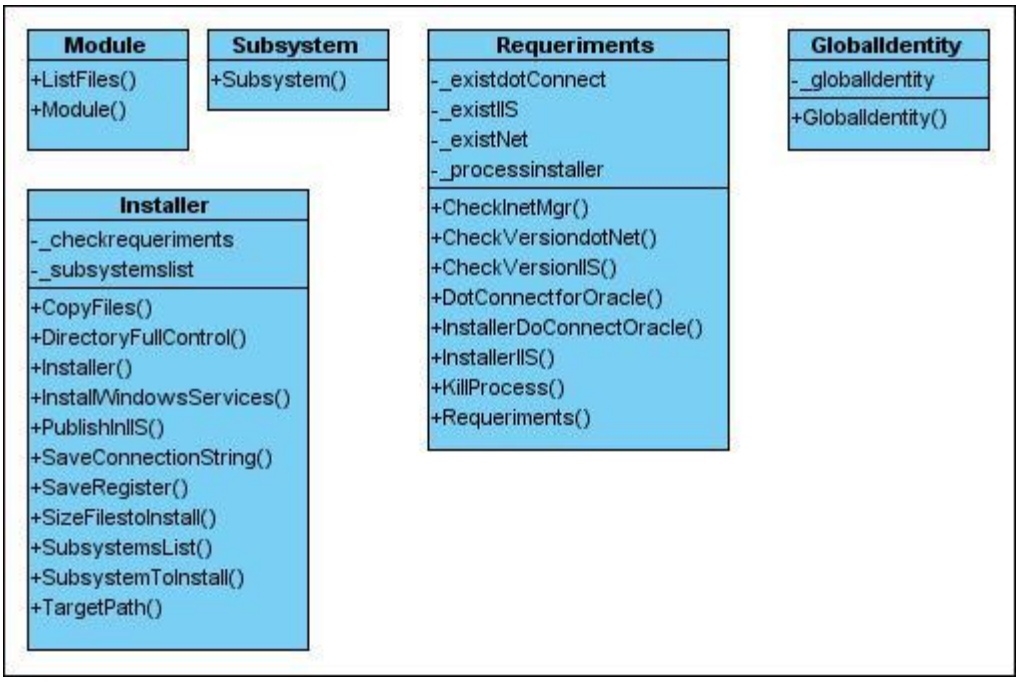


Figura 17: Diagrama de clases de la Capa de lógica del negocio.

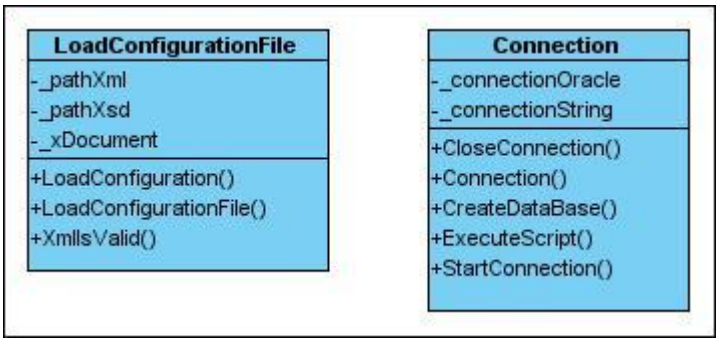
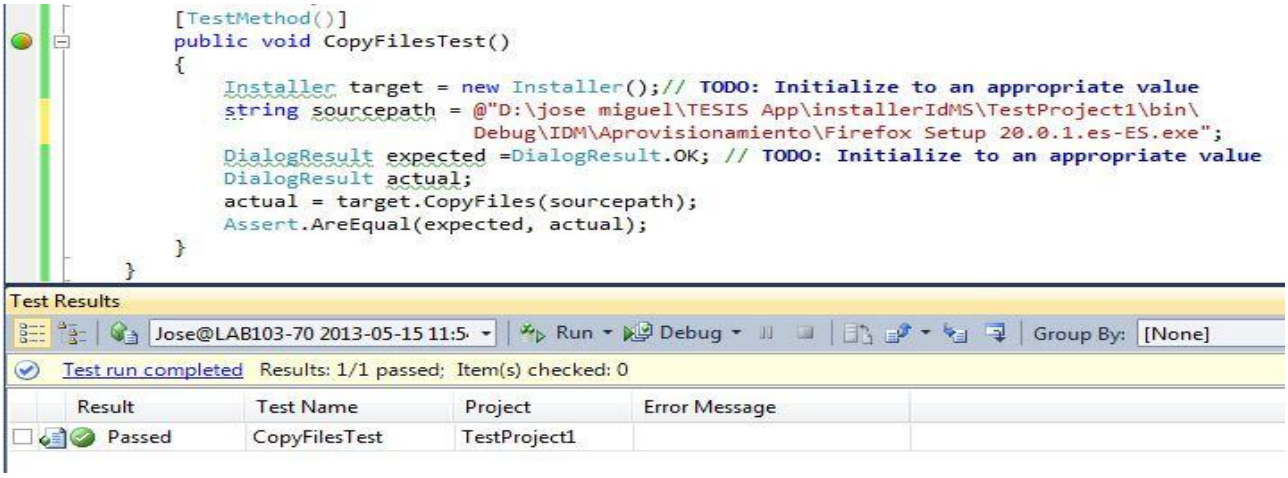


Figura 18: Diagrama de clases de la capa Acceso a datos.

Anexo 3: Pruebas Unitarias

Tabla 20: Prueba realizada al método: "CopyFilesTest".

Prueba de Unidad		
Nombre Prueba: CopyFilesTest		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ejecución:** 15-5-2013
Ejecutado por: José Miguel Ruiz	**Verificado por:** Yendry Machado García	
Descripción: Para la ejecución de esta prueba se necesita pasar por parámetro la dirección del archivo a copiar y este devolverá un valor DialogResult.OK cuando se copie de forma correcta.		
Entrada: string sourcepath		
Criterio de aceptación:		
Resultado: 		

```
[TestMethod()]
public void CopyFilesTest()
{
    Installer target = new Installer();// TODO: Initialize to an appropriate value
    string sourcepath = @"D:\jose miguel\TESIS App\installerIdMS\TestProject1\bin\
                          Debug\IDM\Aprovisionamiento\Firefox Setup 20.0.1.es-ES.exe";
    DialogResult expected =DialogResult.OK; // TODO: Initialize to an appropriate value
    DialogResult actual;
    actual = target.CopyFiles(sourcepath);
    Assert.AreEqual(expected, actual);
}
}
```


Tabla 21: Prueba realizada al método: "DirectoryFullControlTest".

Prueba de Unidad		
Nombre Prueba: DirectoryFullControlTest		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ejecución:** 15-5-2013
Ejecutado por: José Miguel Ruiz	**Verificado por:** Yendry Machado García	

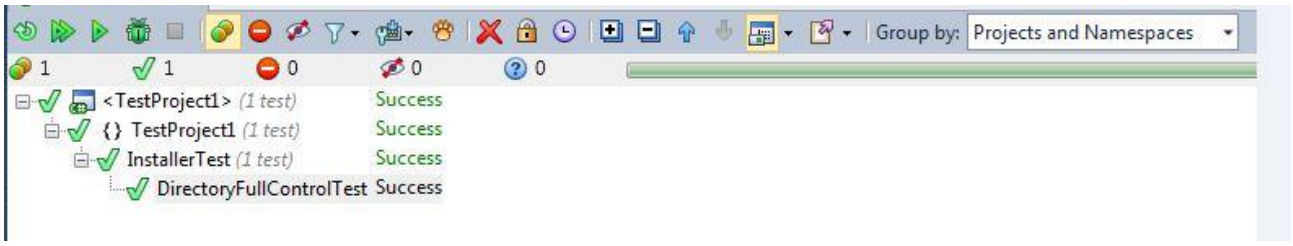
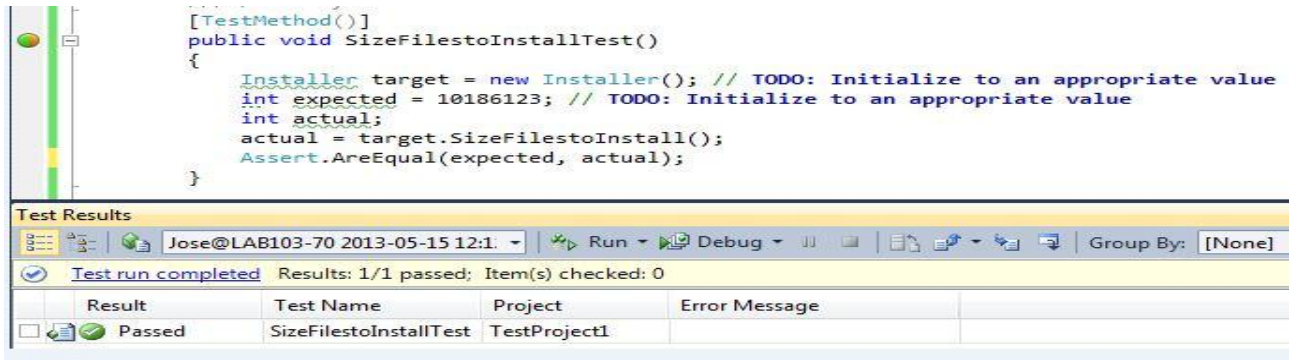
Descripción: Para la ejecución de esta prueba se necesita pasar por parámetro la dirección del directorio al cuál se le va a modificar los permisos de acceso.
Entrada: **DirectoryInfo** directory
Criterio de aceptación: Modifica los permisos de acceso al directorio entrado por parámetro.
Resultado: 

Tabla 22: Prueba realizada al método: "SizeFilestoInstallTest".

Prueba de Unidad		
Nombre Prueba: SizeFilestoInstallTest		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ejecución:** 15-5-2013
Ejecutado por: José Miguel Ruiz	**Verificado por:** Yendry Machado García	
Descripción: Para la ejecución de esta prueba se deben seleccionar los subsistemas que se van instalar y la aplicación calcula la cantidad de bytes que deberá copiar.		
Resultado: 		

Anexo 4: Pruebas de Caja Negra

Tabla 23: Chequear e instalar *software* de pre-requisitos para la instalación.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC1 .1 Chequear estado de pre-requisitos para la instalación.	*Verificar que estén instalados los pre-requisitos para la instalación.*		*Cambio en los iconos que indican si están instalados o no, el software, en caso de ser la flecha verde está instalado, sino muestra un icono de advertencia que significa que no está instalado.*	*Pantalla Principal.*
EC 1.2 Seleccionar opción Instalar pre-requisitos	*Permitir instalar pre-requisitos*	*Instalar*	*Se mostrará una interfaz con una barra de progreso que representa la instalación de los pre-requisitos.*	*Se selecciona la Opción Instalar*
EC 1.3 Seleccionar opción Siguiente en la interfaz de instalar pre-requisitos	*Permite seleccionar la opción "Siguiente" en la interfaz de instalar pre-requisitos*	*Siguiente1*	*Mostrará la interfaz de acuerdo de licencia de software.*	*Se selecciona la Opción Siguiente*
EC 1.4 Seleccionar opción Cancelar en la interfaz de instalar pre-requisitos	*Permite seleccionar la opción "Cancelar" en la interfaz de instalar pre-requisitos*	*Cancelar1*	*Muestra un mensaje preguntando ¿Desea cancelar la instalación?, si el usuario oprime si, sale de la instalación sino sigue la misma.*	*Se selecciona la Opción Cancelar*
EC 1.5 Seleccionar opción Siguiente	*Permite seleccionar la opción "Siguiente"*	*Siguiente 2*	*Mostrará la interfaz de acuerdo de licencia de software.*	*Se selecciona la Opción Siguiente*
EC 1.6 Seleccionar opción Cancelar	*Permite seleccionar la opción "Cancelar"*	*Cancelar 2*	*Muestra un mensaje preguntando ¿Desea cancelar la instalación?, si el usuario oprime "Si", sale de la instalación sino sigue la misma.*	*Se selecciona la Opción Cancelar*

Tabla 24: Mostrar acuerdo de Licencia de *software*.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar la opción "No aceptar licencia del software"	*Permitir seleccionar la opción "No aceptar la licencia del software"*	*No Aceptar*	*Se habilita el botón "Atrás" y el botón "Cancelar"*	*Se selecciona la Opción "No aceptar la licencia de software".*
EC 1.2 Seleccionar "Atrás"	*Permitir seleccionar la opción "Atrás"*	*Atrás 1*	*Regresa a la pantalla principal de la plataforma de instalación*	*Se selecciona la Opción "Atrás".*
EC 1.3 Seleccionar la opción "Aceptar licencia del software"	*Permitir seleccionar la opción "Aceptar la licencia del software"*	*Aceptar*	*Se habilita el botón "Siguiente"*	*Se selecciona la Opción "Aceptar la licencia de software".*
EC 1.4 Seleccionar "Siguiente"	*Permitir seleccionar la opción "Siguiente"*	*Siguiente 3*	*Muestra la interfaz de selección del tipo de instalación "Completa" y "Personalizada".*	*Se selecciona la Opción "Siguiente"*
EC 1.5 Seleccionar la opción "Cancelar"	*Permitir seleccionar la opción "Cancelar"*	*Cancelar 3*	*Muestra un mensaje preguntando ¿Desea cancelar la instalación?, si el usuario oprime "Si", sale de la instalación sino sigue la misma.*	*Se selecciona la Opción Cancelar*

Tabla 25: Listar y seleccionar componentes para instalar.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar subsistemas a instalar	*Permite seleccionar subsistemas*	*Subsistemas*	*Se habilita el botón siguiente*	*Se selecciona la Opción Subsistemas a instalar*
EC 1.2 Introducir Directorio de Destino	*Permite escribir la ubicación donde se desean guardar el paquete de instalación.*	*Directorio*	*Permite introducir un directorio de destino y se habilita el botón siguiente, si se introduce una ruta no válida muestra un mensaje "No es un directorio válido".*	*Se selecciona la Opción Directorio de destino*
EC 1.3 Seleccionar opción Examinar	*Permite escoger la ruta de ubicación de la instalación.*	*Examinar*	*Muestra una ventana para abrir programas y así seleccionar la ruta de ubicación, para la instalación.*	*Se selecciona la Opción Examinar*
EC 1.4 Seleccionar opción Atrás	*Permite seleccionar "Atrás"*	*Atrás 3*	*Regresa a la interfaz anterior de seleccionar el tipo de instalación.*	*Se selecciona la Opción Atrás*
EC1.5 Seleccionar opción Siguiente	*Permite seleccionar "Siguiente"*	*Siguiente 4*	*Muestra la interfaz de configuración de la Base de datos.*	*Se selecciona la Opción Siguiente*
EC1.6 Seleccionar opción Cancelar	*Permite seleccionar "Cancelar"*	*Cancelar 5*	*Muestra un mensaje preguntando ¿Desea cancelar la instalación?, si el usuario oprime "Si", sale de la instalación sino sigue la misma.*	*Se selecciona la Opción Cancelar*

Tabla 26: Configurar la base de datos.

Escenario	Descripción	Variable 1	Variable 2	Variable3	Variable 4	Variable 5	Variable 6	Variable 7	Respuesta del sistema	Flujo entral
EC 1.1 Introducir los valores para configurar la Base de datos.	Permitir configurar la base de datos.	Usuario josetest	V Contraseña josetest1	V Servidor 192.168.1.47	V SID idm	Conexión	Guardar	Cargar		Se selecciona la Opción Introducir valores de configuración de la Base de Datos.
		I Usuario jose***	V Contraseña josetest1	V Servidor 192.168.1.47	V SID idm	Conexión	Guardar	Cargar	Muestra un mensaje de error "Todos los campos son datos requeridos, llénelos por favor"	
		V Usuario josetest	I Contraseña $%7)@	V Servidor 192.168.1.47	V SID idm	Conexión	Guardar	Cargar	Muestra un mensaje de error "Todos los campos son datos requeridos, llénelos por favor"	
		V Usuario josetest	V Contraseña josetest1	I Servidor a.qwe.12.$	V SID idm	Conexión	Guardar	Cargar	Muestra un mensaje de error "Todos los campos son datos requeridos, llénelos por favor"	
		V Usuario josetest	V Contraseña josetest1	V Servidor 192.168.1.47	I SID $&&	Conexión	Guardar	Cargar	Muestra un mensaje de error "Todos los campos son datos requeridos, llénelos por favor"	

		V Usuario josetest	V Contraseña josetest1	V Servidor 192.168.1.47	V SID idm	Conexión	Guardar	Cargar	Muestra un mensaje de error "Todos los campos son datos requeridos, llénelos por favor"	
		I Usuario jose***	I Contraseña $%7)@	I Servidor a.qwe.12.$	I SID $&&	Conexión	Guardar	Cargar	Muestra un mensaje de error "Todos los campos son datos requeridos, llénelos por favor"	
EC 1.4 Seleccionar la opción "Atrás"	Permitir seleccionar el botón "Atrás"	Atrás 4							Regresa a la interfaz de selección de los subsistemas a instalar	Se selecciona la Opción Atrás
EC 1.5 Seleccionar la opción "Siguiente"	Permitir seleccionar el botón "Siguiente"	Siguiente 5							Muestra la interfaz de comprobar el espacio en disco para los subsistemas a instalar en la PC.	Se selecciona la Opción Siguiente
EC Seleccionar opción guardar credenciales	Permite seleccionar la opción guardar credenciales	Guardar credenciales							Desactiva la opción "Cargar y ejecutar scripts en la base de datos".	Se selecciona la Opción Guardar credenciales

EC 1.6 Seleccionar la opción Cargar y ejecutar scripts	*Permite seleccionar la opción Cargar y ejecutar scripts*	*Cargar*							*Muestra un openfiledialog para cargar el fichero con el script.*	*Se selecciona la Opción Cargar y ejecutar scripts*
EC 1.7 Seleccionar el Modo de Conexión	*Permite seleccionar el modo de conexión*	*Modo de conexión*							*Despliega una lista con los posibles modos de conexión a seleccionar.*	*Se selecciona la Opción Modo de Conexión*
EC 1.8 Seleccionar la opción "Cancelar"	*Permitir seleccionar el botón "Cancelar"*	*Cancelar 6*							*Muestra un mensaje preguntando ¿Desea cancelar la instalación?, si el usuario oprime "Si", sale de la instalación sino sigue la misma.*	*Se selecciona la Opción Cancelar*

Tabla 27: Comprobar espacio en disco.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar la opción "Atrás"	*Permitir seleccionar opción "Atrás"*	*Atrás 5*	*Regresa a la interfaz de configuración de la Base de datos.*	*Se selecciona la Opción Atrás*
EC 1.2 Seleccionar la opción "Instalar"	*Permitir seleccionar opción "Instalar"*	*Instalar 2*	*Muestra una interfaz con una barra de progreso donde indica el proceso de instalación de los subsistemas.*	*Se selecciona la Opción Instalar*

EC 1.3 Seleccionar la opción "Cancelar"	*Permitir seleccionar opción "Cancelar"*	*Cancelar 7*	*Muestra un mensaje preguntando ¿Desea cancelar la instalación?, si el usuario oprime "Si", sale de la instalación sino sigue la misma.*	*Se selecciona la Opción Cancelar*

Tabla 28: Instalar los componentes.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar la opción "Siguiente"	*Permitir seleccionar opción "Siguiente"*	*Siguiente 6*	*Muestra la interfaz donde visualiza todos los servicios y sitios web que existen en el servidor IIS 7.*	*Se selecciona la Opción Siguiente*
EC 1.1 Seleccionar la opción "Cancelar"	*Permitir seleccionar opción "Cancelar"*	*Cancelar 8*	*Muestra un mensaje preguntando ¿Desea cancelar la instalación?, si el usuario oprime si, sale de la instalación sino sigue la misma.*	*Se selecciona la Opción Cancelar*

Tabla 29: Seleccionar tipo de desinstalación.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar la opción "Desinstalación total"	*Permite seleccionar la opción para desinstalar todos los componentes que estén instalados.*	*Desinstalación total*	*Muestra una interfaz con todos los componentes que se van a desinstalar.*	*Se selecciona la Opción Desinstalación total*

EC 1.1 Seleccionar la opción "Añadir o eliminar algún componente"	*Permite seleccionar la opción para añadir o eliminar componentes en específicos.*	*Añadir o quitar componentes*	*Muestra una interfaz para seleccionar los componentes a instalar o desmarcar los que desee desinstalar.*	*Se selecciona la Opción Añadir o quitar componentes*

Tabla 30: Finalizar proceso instalación.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar opción "Administrador de IIS7"	*Permite seleccionar ese campo.*	*Administrar*		*Se selecciona la Opción Administrador de IIS*
EC 1.2 Seleccionar la opción "Atrás"	*Permite seleccionar la opción "Atrás".*	*Atrás 6*	*Muestra la interfaz anterior de listar los servicios y sitios web existentes en el servidor.*	*Se selecciona la Opción Atrás*
EC 1.3 Seleccionar la opción "Finalizar"	*Permite seleccionar la opción "Finalizar".*	*Finalizar*	*Cierra el asistente de aplicación.*	*Se selecciona la Opción Finalizar*

Tabla 31: Listar los servicios y sitios web publicados en el servidor.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Seleccionar la opción "Siguiente"	*Permite seleccionar la opción "Siguiente".*	*Siguiente*	*Muestra la última interfaz del asistente de instalación.*	*Se selecciona la Opción Siguiente*

EC 1.2 Seleccionar la opción "Configuración"	*Permite seleccionar la opción "Configuración" de un sitio o servicio web.*	*Configuración*	*Muestra una nueva ventana con los datos del servicio o sitio web seleccionado.*	*Se selecciona la Opción Configuración*

Tabla 32: Configurar sitios y servicios web publicados.

Escenario	Descripción	Variable 1	Variable 2	Respuesta del sistema	Flujo central
EC 1.1 Introducir valores para configurar el sitio o servicio web.	*Permite introducir los valores*	*V Nombre nombre*	*V Puerto 34342*		*Se selecciona la Opción introducir los valores de configurar sitios y servicios web publicados*
		V Nombre mismo	*I Puerto 66598*	*Muestra un mensaje de error "El puerto es un número hasta 65535"*	
		I Nombre 23$@&	*I Puerto 69890*	*Muestra un mensaje de error.*	
EC 1.2 Seleccionar la opción "Publicar"	*Permite seleccionar la opción "Publicar"*	*publicar*		*Se guardan los datos correctamente y se cierra la ventana*	*Se selecciona la Opción Publicar*

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <Configuration>
3       <System>
4       <Subsystem subsystem_name="Aprovisionamiento" >
5       <Modules>
6         <Module module_name="IDM" module_type="WebSite">
7           <Add config_filename ="Web.config"/>
8         </Module>
9         <Module module_name="servicio"  module_type="WebService">
10          <Add config_filename ="Gyes.Provisioning.Service.dll.config"/>
11          <Add config_filename ="Web.config"/>
12        </Module>
13      </Modules>
14      </Subsystem>
15      <Subsystem subsystem_name="Autorización">
16      <Modules>
17        <Module module_name ="IDMAut" module_type="WebSite">
18          <Add config_filename="Web.config"/>
19        </Module>
20      </Modules>
21      </Subsystem>
22    </System>
23    </Configuration>
```

Figura 19: Archivo de configuración XML.