UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

[TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE]

Ingeniero en Ciencias Informáticas

[TÍTULO]

Módulo de Registro y Configuración de Proveedores de Servicios asociados al Sistema de Administración de Identidades

Autor: Arniel Couso Linares

Tutor(es): Ing. Roberto Quiñones Bondartchuk

Ing. Yayneris Zambrana Hernández

La Habana, Cuba

Año 54 de la Revolución

El único modo de hacer un gran trabajo es amar lo que haces. Si no lo has encontrado todavía, sigue buscando. No te acomodes. Como con todo lo que es propio del corazón, lo sabrás cuando lo encuentres.

Steven Paul Jobs

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

_____

Arniel Couso Linares

_____ _____

Ing. Yayneris Zambrana Hernández Ing. Roberto Quiñones Bondartchuk

Dedicatoria

A mis padres

A mi hermano

A mi novia linda

A toda mi familia

A todos mis amigos

A todos en general…

Agradecimientos

A mis padres sus profundas instrucciones y enseñanzas, sus constantes preocupaciones, su confianza, su inigualable apoyo y cariño.

A mi hermano mayor que siempre me ha guiado por el buen camino, me ha aconsejado y ayudado mucho.

A mi novia por pasar junto a mí momentos buenos y momentos malos, tanto de día como de madrugada.

A mis tutores por sus buenos consejos, ayuda y revisiones.

A todos mis compañeros de cuarto, de grupo, de aula y de laboratorio por compartir conmigo y por ayudarme cuando lo necesite.

A todos los que sin querer olvido, muchas gracias.

Resumen

En el Centro de Identificación y Seguridad Digital radicado en la Universidad de las Ciencias Informáticas se desarrolla una solución de *software* para la Administración de Identidades, que tiene como principal objetivo gestionar los procesos de autenticación, autorización y aprovisionamiento en aplicaciones empresariales o proveedores de servicios. Además, de estos existen otros procesos que le dan soporte al sistema dentro de los cuales se encuentra el proceso de gestión de integración de los proveedores de servicios y el Sistema de Administración de Identidades en cual se realizan un conjunto de actividades muy importantes para integrar los sistemas.

Con el objetivo de mejorar el proceso de integración entre los sistemas surge el trabajo de diploma que lleva por título “Módulo de registro y configuración de proveedores de servicios asociados al Sistema de Administración de Identidades”. Este trabajo tiene como objetivo general implementar un sistema de configuración y registro que permita mejorar el proceso de integración entre un proveedor de servicios y el Sistema de Administración de Identidades.

La investigación del trabajo estuvo enmarcada fundamentalmente en el análisis de herramientas para la configuración de sistemas y búsqueda de elementos a través de Internet. Además, en el estudio de las principales herramientas, tecnologías y metodología para el desarrollo del sistema, así como en la definición de las características del mismo, planificación, desarrollo y pruebas necesarias para garantizar la calidad del producto final.

Palabras clave: *Configuración, Integración, Proveedor, Registro, Servicio.*

Índice

Índice de tablas

Índice de figuras

Introducción

En la actualidad los delitos informáticos en el ámbito de las tecnologías de la información (TI) se han visto incrementados gracias a la infinidad de posibilidades que existen para acceder a los recursos de la red de manera remota y al gran incremento de las conexiones a Internet. Es por esto que se hace necesario dentro de una organización controlar el acceso de los usuarios mediante niveles de seguridad y restricciones para cada uno mediante la utilización de un Sistema de Administración de Identidades (SAI).

Los sistemas de administración de identidades gestionan de manera centralizada parte de la seguridad de las aplicaciones empresariales o proveedores de servicios de la organización, específicamente los procesos de autenticación, autorización y aprovisionamiento de usuarios. Estos sistemas ayudan a reducir los riesgos potenciales de robo de credenciales y de información, y reducen los costos de implementación de sistemas de seguridad. Además, mediante su aplicación es posible confeccionar directorios que permiten integrar los diversos sistemas informáticos que tenga una empresa a través de una identidad única para el usuario.

Las empresas cubanas no están ajenas a la utilización de este tipo de sistemas, por lo que el Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) desarrolla un proyecto productivo que tiene como cliente al Ministerio del Interior de la República de Cuba (MININT) con el fin de obtener un sistema de este tipo.

Además, de los procesos claves con los que cuenta el sistema autenticación, autorización y aprovisionamiento, para su correcto funcionamiento necesita estar sustentado por otros procesos de soporte a la misma. En los que se encuentra la gestión de integración de los proveedores de servicio con el SAI. Para lograr esta gestión es necesario llevar a cabo una serie de actividades que terminan con la integración de los proveedores de servicios al sistema de administración de identidades. Dichas actividades componen el proceso de Registro y Configuración de Proveedores de Servicios y se detallan a continuación:

Actividades:

- *Adición de parámetros en el archivo de configuración del proveedor de servicios*: Este archivo (*web.config*) almacena la configuración del proveedor de servicio en secciones con diferentes propósitos, que van desde configuraciones de seguridad, hasta configuraciones relacionadas con el hospedaje de la propia aplicación en el servidor web.
- *Búsqueda y registro de recursos del proveedor de servicios*: Con vista a efectuar la posterior autorización de usuarios es necesario identificar los recursos (URLs) que requieren protección y luego registrarlas mediante las interfaces que provee el Sistema de Administración de Identidades.
- *Adición de librerías de clases en el proveedor de servicios*: Los proveedores de servicios de la plataforma "ASP.Net" ofrecen mecanismos de extensión a través de sus archivos de configuración, de esta manera, se puede hacer referencias a implementaciones que redefinen los mecanismos o procesos deseados, entre ellos los de autenticación y autorización. Estas implementaciones se proveen mediante librerías de clases (DLL) que deben ser accedidas por el proveedor de servicio.

Actualmente cada una de estas actividades se realiza de forma manual, lo que requiere cierto grado de especialización en la ejecución de las mismas, dado que el personal que las realiza debe conocer tanto la estructura del archivo de configuración, como la estructura de ficheros del proveedor de servicios. Esta actividad tiene un carácter crítico ya que los errores que puedan ser introducidos en la configuración de un proveedor de servicios pudieran impactar negativamente en su funcionamiento. Por otra parte, los recursos se identifican de uno en uno a través de algún navegador web o gestor de fichero según sea el caso, y a mayor cantidad de recursos, mayor tiempo se empleará en la búsqueda y el registro de los mismos, introduciendo así demoras en el proceso.

A partir de la situación problemática planteada, surge el siguiente **problema científico**: ¿Cómo mejorar el proceso de registro y configuración de Proveedores de Servicios asociados al SAI?

Por lo que se define como **objeto de estudio:** los procesos de gestión de integración de los SAI con los proveedores de servicios.

Para responder al problema de la investigación se tiene como **objetivo general:** Implementar un sistema de configuración y registro, que permita mejorar el proceso de registro y configuración de Proveedores de Servicios asociados al SAI que se desarrolla en el CISED.

Quedando enmarcado como **campo de acción**: Proceso de configuración y registro de proveedores de servicios asociados a los SAI.

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

1. Analizar los referentes teóricos del tema de investigación.
2. Analizar y diseñar la propuesta de solución que se quiere desarrollar.
3. Implementar el módulo de registro y configuración de proveedores de servicios.
4. Validar la propuesta de solución.

Las **tareas** que se trazaron para el desarrollo de la investigación se muestran a continuación:

- Elaboración del marco teórico de la investigación para definir el alcance de la investigación.
- Análisis del estado del arte del tema de investigación obtener los conocimientos necesarios para el desarrollo del trabajo.
- Definición de la metodología y tecnologías para el desarrollo de la propuesta de solución.
- Definición de los requisitos funcionales y no funcionales del sistema a desarrollar.
- Diseño e implementación de la propuesta de solución para dar solución al problema existente.
- Validación de la calidad del sistema desarrollado mediante la realización de pruebas al mismo.

Para el desarrollo de la investigación se definen los métodos científicos que se detallan a continuación:

- Métodos teóricos:
    - Histórico-Lógico: Se utiliza en el análisis de la base teórica de los diferentes procesos de configuración y búsqueda en el mundo viendo su evolución histórica, así como en el estudio de las tecnologías y herramientas a utilizar.
    - Analítico-Sintético: Mediante este método se analiza toda la teoría recopilada en diferentes medios bibliográficos, que puedan servir para el diseño de la solución propuesta.

- Métodos empíricos:
    - Entrevista**:** Se utiliza como una conversación planificada con especialistas relacionados al tema de la investigación, personas con experiencias o debidamente relacionados al fenómeno que se analiza, atribuyendo conocimiento del mismo.

Justificación de la investigación: El desarrollo de la investigación proporcionará al SAI la automatización del proceso de configuración y registro de proveedores de servicios, permitiendo realizar estas acciones de una manera rápida y segura evitando el margen de error al llevar a cabo estos procesos de la forma que se realizan actualmente.

El presente documento estará compuesto por tres capítulos que se detallan a continuación:

Capítulo 1: Fundamentación teórica

En este capítulo se realiza un estudio del estado del arte del tema de investigación, analizando los diferentes sistemas existentes que son similares al que se quiere desarrollar. Además, se analizan algunas de las diferentes metodologías ágiles para el desarrollo de *software*, así como las tecnologías, herramientas y lenguajes que posibilitarán el desarrollo de la propuesta de solución.

Capítulo 2: Análisis y diseño del sistema

En el capítulo se describe detalladamente la propuesta de solución, basado en el análisis previo del objeto de estudio y la problemática planteada. Se definen y especifican las funcionalidades que debe tener el sistema, así como los requisitos de calidad del servicio con los que debe cumplir. Además, se especifica la arquitectura del sistema y los patrones de diseño a usar para la implementación del sistema.

Capítulo 3: Implementación y prueba

En este capítulo se modelan los diferentes diagramas diseñados para la implementación de la solución y se especifica sobre las pruebas que se realizan para la validación de la calidad del sistema.

Capítulo 1: Fundamentación teórica

1.1. Introducción

En este capítulo se realiza un estudio del estado del arte del tema de investigación, analizando los diferentes sistemas existentes que son similares al que se quiere desarrollar. Además, se analizan algunas de las diferentes metodologías ágiles para el desarrollo de software, así como las tecnologías, herramientas y lenguajes que posibilitarán el desarrollo de la propuesta de solución.

1.2. Conceptos asociados al dominio del problema

A continuación se muestra una serie de conceptos asociados al dominio del problema planteado, con el objetivo de lograr un mejor entendimiento del lector.

Entidad: según OASIS SAML[1] una entidad es un elemento activo de una computadora o sistema de red. Por ejemplo, un proceso automatizado o grupo de ellos, un subsistema, una persona o grupo de personas que incorporan un conjunto distinto de funcionalidades (2).

En el libro *Digital Identity* (3 2.1) escrito por Phillip J. Windley se define entidad como una persona, programa informático, maquina u otra cosa que hace una petición para acceder a un recurso.

Principal: según OASIS SAML[2] un principal es una entidad cuya identidad puede ser autenticada (2).

Proveedor de servicios: según OASIS SAML[2] los proveedores de servicios son las entidades que proveen servicios a principales u otras entidades (2).

Proveedor de Identidades: según OASIS SAML[2] un proveedor de identidades es una entidad que crea, mantiene y administra información de identidades para principales y provee autenticación a otros proveedores de servicios (2).

Administración de identidades: es la combinación de procesos de negocio y la tecnología utilizada para administrar los datos en los sistemas informáticos y aplicaciones de los usuarios. Los datos gestionados

[1] Por sus siglas en inglés Security Assertion Markup Language

incluyen los objetos de usuario, los atributos de identidad, los derechos de seguridad y los factores de autenticación (4).

Configuración: en informática la configuración es un conjunto de datos que determina el valor de algunas variables de un programa o de un Sistema Operativo (SO), estas opciones generalmente son cargadas en su inicio y en algunos casos se deberá reiniciar para poder ver los cambios, ya que el programa no podrá cargarlos mientras se esté ejecutando, si la configuración aún no ha sido definida por el usuario (personalizada), el programa o sistema cargará la configuración predeterminada (5)**.**

1.3. Sistema de Administración de Identidades (SAI)

Ante la necesidad de Ministerio del Interior de contar con un sistema que estandarice y controle los procesos de autenticación, autorización y aprovisionamiento de los usuarios en los sistemas en explotación en este organismo, surge en el CISED, específicamente en el Departamento de Seguridad Digital el proyecto Sistema de Administración de Identidades que se identifica como GYES. Este proyecto se divide en cuatro subsistemas, en correspondencia con los procesos de la administración de identidades antes mencionados. Estos subsistemas se dividen a su vez en módulos con objetivos específicos. A continuación se muestra la distribución de los mismos por cada subsistema:

Subsistema de Autenticación:

- Módulo Servidor de Autenticación
- Módulo Cliente de Autenticación
- Módulo proveedores de sesiones
- Módulo Servicios de sesiones
- Módulo Web de Administración

Subsistema de Autorización:

- Módulo de Servicios de Autorización
- Módulo Cliente de Autorización
- Módulo Aplicación de Administración

Subsistema de Aprovisionamiento:

- Motor de Tareas
- Web de Administración
- Servicio de Aprovisionamiento

Subsistema de Reportes y Auditoria:

- Componente proveedor de trazas
- Módulo Web de Auditoria

En el siguiente diagrama se muestra de manera general los diferentes elementos que componen el Sistema de Administración de Identidades y su relación con los recursos y aplicaciones de la organización.

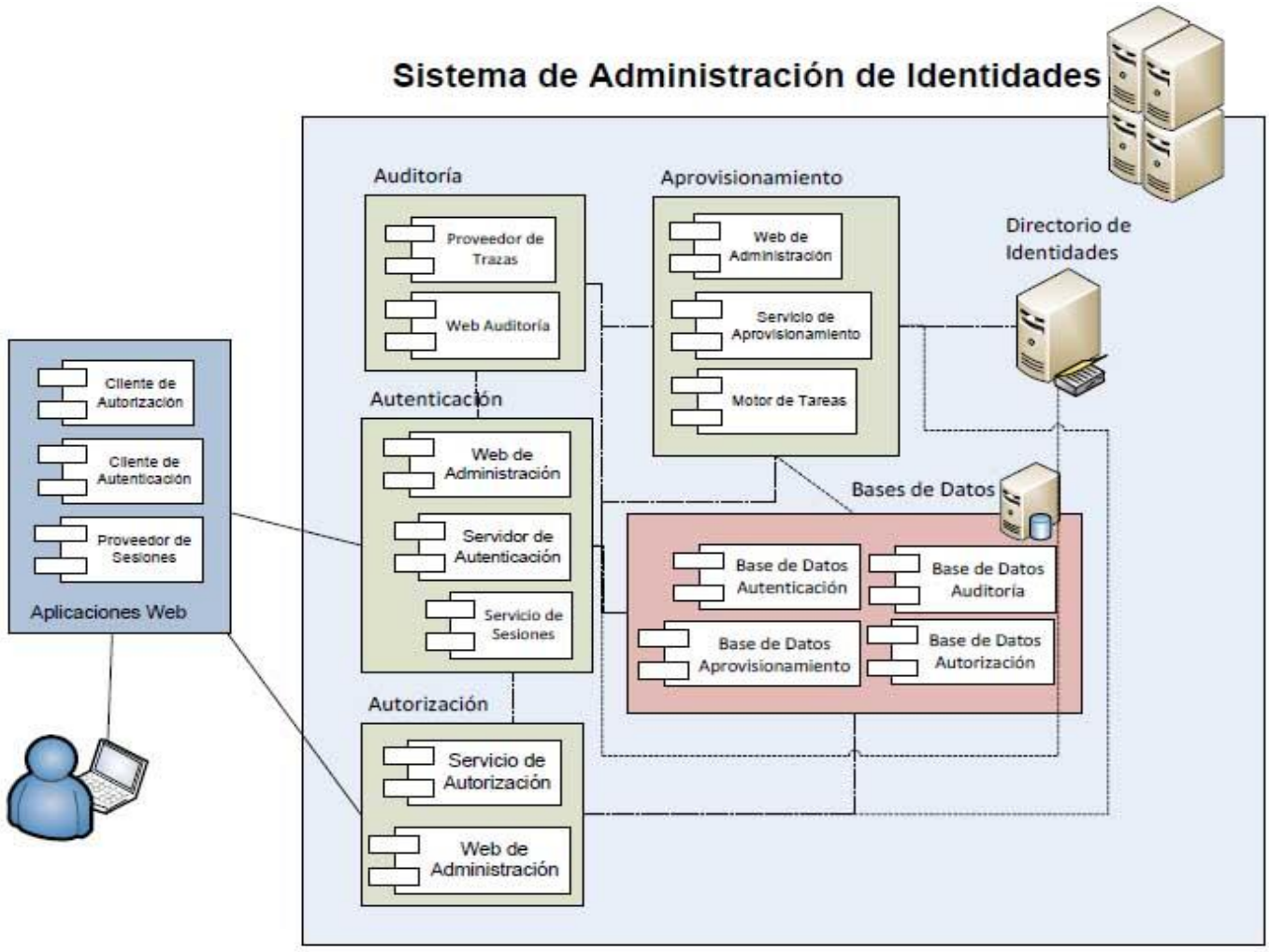

Figura 1: Vista general del Sistema de Administración de Identidades (5).

Hay que destacar que los componentes “Cliente de Autenticación” y “Cliente de Autorización”, constituyen los puntos de contacto entre los proveedores de servicios y el Sistema de Administración de Identidades; en ambos casos, la integración de dichos componentes será a través de la incorporación de librerías de clases y la modificación de los parámetros de configuración de los proveedores de servicios, permitiéndoles intervenir en los procesos de autenticación y autorización respectivamente (5).

A continuación se muestra una tabla con algunos de los parámetros de configuración que se modifican durante este proceso junto con una pequeña descripción sobre estos. Los restantes parámetros se pueden apreciar en el Anexo I.

Nombre	Saml20Federation
Descripción	Contiene las opciones de configuración que se especifican por el protocolo SAML 2.0.
Nombre	ServiceProvider
Descripción	Configura los puntos de entradas Http usados por el proveedor de servicios para comunicarse con los socios de la federación.
Atributos	
Id	El identificador del proveedor de servicios.
server	La URL del servidor que aloja al proveedor de servicios.

Tabla 1: Parámetros de configuración I.

En la Figura 2 se muestra la estructura y algunas de las secciones del archivo de configuración necesarias para la integración entre los sistemas.

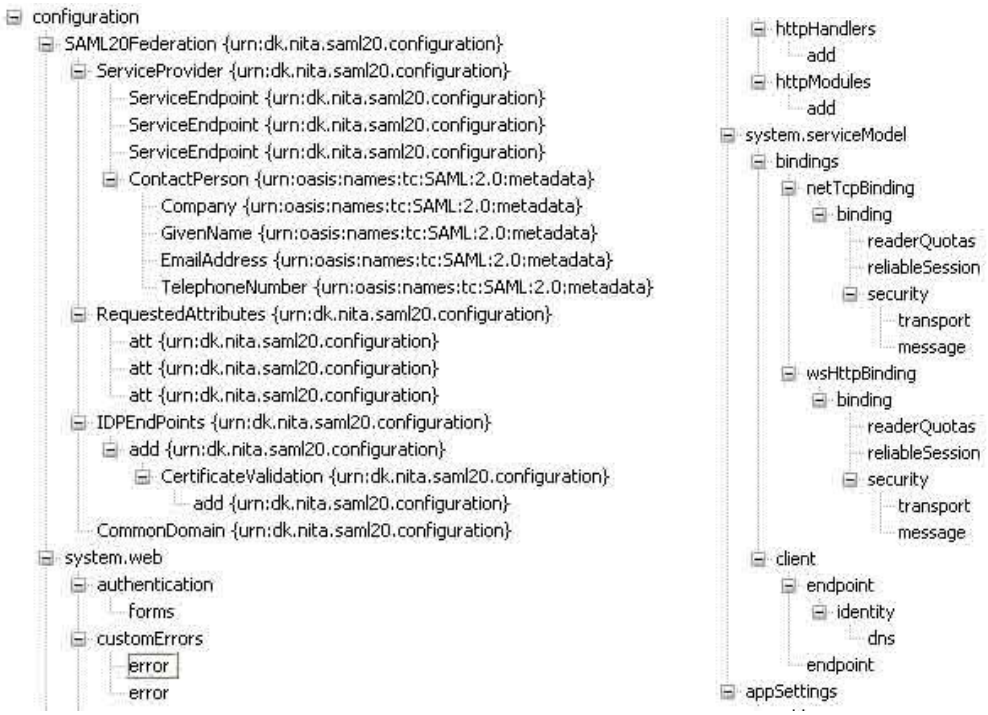

Figura 2: Estructura XML del archivo "web.config".

En la Figura 3 se muestra una sección del archivo de configuración del proveedor de servicios.

```
<?xml version="1.0"?>
<configuration>
  <SAML20Federation xmlns="urn:dk.nita.saml20.configuration">
    <ServiceProvider id="http://localhost.com:61534/" server="http://localhost.com:61534">
  <configSections>
    <!--<section name="gyesAuthz" type="Gyes.WebAm.Bridge.Configuration.GyesAuthorizationSe
    <section name="Federation" type="dk.nita.saml20.config.ConfigurationReader, dk.nita.sam
    <section name="SAML20Federation" type="dk.nita.saml20.config.ConfigurationReader, dk.ni
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,Log4net
  </configSections>
  <log4net>
    <root>
      <level value="All" />
      <appender-ref ref="LogFileAppender" />
    </root>
    <appender name="LogFileAppender" type="log4net.Appender.RollingFileAppender">
      <param name="File" value="C:\Temp\log.txt" />
      <param name="AppendToFile" value="true" />
      <rollingStyle value="Size" />
      <maxSizeRollBackups value="10" />
      <maximumFileSize value="10MB" />
      <staticLogFileName value="true" />
      <layout type="log4net.Layout.PatternLayout">
        <param name="ConversionPattern" value="%-5p%d{yyyy-MM-dd hh:mm:ss}-%m%n" />
```

Figura 3: Sección del archivo de configuración.

Todos los elementos “Web de Administración” de los subsistemas forman un solo componente en el que se realizan todas las tareas de administración del sistema. Una de estas tareas es la “Gestión de Recursos” que permite adicionar, eliminar y modificar los recursos pertenecientes a un proveedor de servicios determinado.

1.4. Análisis de soluciones similares

En la actualidad se pueden encontrar muchas soluciones informáticas que permiten la configuración de otros sistemas y la búsqueda de elementos en el menor tiempo y de la mejor manera posible. Muestra de ello son los ejemplos de sistemas consultados que se describen a continuación:

1.4.1. Sistemas de configuración

La mayoría de las aplicaciones de hoy en día cuentan con un conjunto de variables que determinan el aspecto y el funcionamiento de las mismas. Este conjunto se establece a partir de una serie datos cargados, ya sea desde un archivo o bases de datos al ejecutarse la aplicación. Muchas de estas aplicaciones permiten la configuración de estos datos mediante una interfaz gráfica propia, pero muchas otras hacen uso de otros sistemas para realizar esta tarea. Este tipo de aplicación puede ser de escritorio, de consola o web, además, puede ser sencilla, compleja o de tipo asistente. El conjunto de sistemas que se analizan a continuación son sistemas de este tipo.

SYSEdit

La aplicación de escritorio SYSEdit es un editor simple y especializado que permite la modificación de los cuatro archivos de configuración del sistema operativo Windows de una manera muy sencilla. Estos archivos de configuración son: el autoexec.bat, config.sys, win.ini y el system.ini.

Esta se instala en “*C:\WINDOWS\system32*” junto con el sistema operativo. La misma se puede ejecutar directamente desde la dirección dada anteriormente o desde la línea de comandos mediante el código “systedit”. Una de las características más importantes de esta herramienta es que guarda automáticamente una copia de cada uno de los archivos de configuración cuando se abren por primera vez. Esto propicia que cuando se cometa un error este se pueda corregir recuperado la copia anteriormente hecha.

Una de las desventajas de esta herramienta es que no realiza validaciones de datos, lo que puede traer consigo que se escriban datos con formatos incorrectos haciendo que el Sistema Operativo sea inestable.

MSConfig

MSConfig es una aplicación de escritorio propietaria que se incluye con todas las versiones de sistemas operativos Microsoft Windows desde Windows 98 hasta Windows 7, exceptuando Windows 2000. Esta herramienta tiene como función principal activar o desactivar temporalmente el *software*, los controladores de dispositivos y los servicios de Windows que se ejecutan durante el inicio del sistema operativo. Además, la herramienta permite modificar los cuatros archivos de configuración del sistema operativo a través de un proceso de eliminación con casillas de verificación.

Una de las ventajas de esta herramienta, es que el uso de las casillas de verificación posibilita la reducción en gran medida de los riesgos que se tienen con respecto a la introducción de los datos incorrectos. No obstante, tiene la desventaja que está desarrollada para configurar un conjunto de parámetros de Windows, lo que imposibilita usarla en otros sistemas operativos.

FedUtil

FedUtil es una herramienta que se proporciona con el framework WIF[2], puede ser invocada desde la línea de comando o desde el Visual Studio. Este sistema es un asistente informático o *wizzard* en idioma inglés, que permite establecer una relación de confianza entre un proveedor de servicios y un proveedor de identidad configurando el archivo de configuración del proveedor de servicios.

Esta herramienta muestra de forma continua un conjunto de interfaces gráficas que cada una incluye un conjunto de instrucciones muy intuitivas y fáciles de realizar, que hacen posible la actualización del archivo de configuración después de varios pasos sencillos.

Las funcionalidades que proporciona esta herramienta son:

- Registrar un proveedor de identidad existente como un emisor de confianza del proveedor de servicios.

[2] Por sus siglas en inglés Windows Identity Foundation.

- Ayudar a desarrollar una aplicación para aceptar notificaciones proporcionados por un proveedor de identidad local.
- Adaptar una aplicación existente para aceptar notificaciones.
- Actualizar los metadatos de federación para un proveedor de servicios (6).

1.4.2. Motores de búsqueda

La búsqueda de elementos sin contar con una herramienta automatizada es una tarea que puede no ser tan difícil pero si se invierte bastante tiempo en ella, por tal motivo en el mundo existen múltiples herramientas que permiten realizar esta acción de una manera menos engorrosa y más rápida haciendo uso de diferentes algoritmos y mecanismos con este fin. A continuación se describen algunas de las principales herramientas que existen.

Apache Nutch

Apache Nutch es un robot de exploración web de código abierto basado en Lucene[3], que provee todas las herramientas necesarias para realizar un motor de búsqueda propio (7). Esta herramienta es muy flexible y está desarrollada en el lenguaje de programación Java. La misma pertenece a la organización no lucrativa Apache Software Fundation y permite encontrar todos los hipervínculos de una página web de manera automatizada, lo que trae consigo que se reduzca el trabajo de mantenimiento.

Este sistema está dividido en dos partes fundamentales: la araña web y el buscador. La araña web o *crawler* en inglés obtiene las páginas webs creando un índice de dichas páginas, que el buscador utiliza para responder a las consultas del usuario.

El *crawler* de Nutch funciona repitiendo el siguiente ciclo:

- Generar las URLs para ir a buscarlas al ***crawldb***[4].
- Obtener las URLs.
- Analizar el contenido.

[3] Lucene es una API de código abierto para recuperación de información.
[4] Base de datos de rastreo o crawldb contiene información sobre cada URL que sabe Nutch, incluyendo si se tomó.

- Actualizar el ***crawldb*** con los resultados de la búsqueda y las nuevas URLs que encontró cuando lo analizó.

Htdig

Htdig es un sistema de indexación sobre la Web y motor de búsqueda para un dominio o una intranet. Este sistema no pretende sustituir los grandes sistemas de búsqueda en Internet tales como Google, Yahoo entre otros, no obstante se destina a cubrir las necesidades de búsqueda en una empresa, escuela o sección dentro un sitio web (8). Su funcionamiento se basa en el protocolo HTTP. El proyecto se descontinuó en junio del año 2004. Fue desarrollado en C/C++.

Entre sus principales características se destacan:

- Soporte para HTML y texto plano.
- Búsqueda de expresiones booleanas.
- Resultados de las búsquedas configurables.
- Soporte para la exclusión de robot.txt.
- Búsquedas en una sub-sección de una BD.
- Lanzado bajo la licencia GNU.

Los sistemas analizados anteriormente no constituyen una solución directa al problema de investigación, pues no presentan de manera integrada los requisitos necesarios para adaptarse a las características específicas del Sistema de Administración de Identidades; no obstante se tuvieron en cuenta algunas de sus características y funcionalidades para el desarrollo de la propuesta de solución como son:

De las herramientas de configuración:

- La forma de agrupar los valores de configuración en las interfaces.
- El tipo de aplicación basado en asistentes informáticos.
- El diseño de las interfaces gráficas.
- Las distintas vías de obtención de datos.

De los motores de búsqueda:

- Ciclos y métodos de búsqueda.

Por lo que se hace necesario desarrollar una herramienta que automatice los procesos de configuración y registros del SAI desarrollado en el departamento.

1.5. Metodología de desarrollo de *software*

El desarrollo de *software* es un proceso de muchos riesgos y por ello difícil de controlar, es por esto que es de vital importancia contar con una metodología que indique paso a paso las actividades que deben desarrollarse, las personas involucradas en la realización de las mismas, así como el papel que desempeñan estas en aras de lograr un proceso de desarrollo de *software* exitoso. Para elegir la metodología a usar es necesario determinar antes el alcance del sistema, y de este modo poder seleccionar la más adecuada para guiar el proceso de desarrollo de software.

Microsoft Solution Framework for Agile Software Development

Microsoft Solution Framework Agile se caracteriza por ser de planificación adaptable a los cambios y orientada a las personas. Su proceso introduce ideas importantes del *software* ágil, junto con los principios y prácticas de "*MSF for CMMI*" como por ejemplo: admite una estrategia que utiliza múltiples iteraciones y un enfoque para la construcción de aplicaciones que se basa en escenarios. Además, esta metodología incorpora prácticas para el manejo de la calidad del servicio (el rendimiento y la seguridad) y facilita la automatización y la orientación que se necesita para apoyar el equipo de trabajo, incluyendo la gestión de configuración y de proyectos (9).

En MSF for ASD se establece que el equipo de trabajo debe ser balanceado, con tareas y objetivos bien definidos de manera que se pueda controlar el avance del proyecto. La definición, desarrollo y prueba del producto se realizan en pequeñas iteraciones provenientes del proceso incremental del proyecto, reduciéndose así el margen de error en las estimaciones y proporcionándose información rápida acerca de la exactitud de los planes del proyecto (9).

Esta metodología tiene cinco fases de desarrollo, las cuales soportan 17 flujos de trabajo que son básicos y que contienen a su vez una serie de actividades. Estas fases son:

- Previsión
- Planeamiento
- Desarrollo

- Estabilización
- Implementación

Por cada fase se generan una serie de elementos de trabajo que son esenciales para el entendimiento y ordenamiento del sistema a desarrollar, estos elementos son los siguientes:

- Escenario: Descripción de la necesidad o solicitud del usuario.
- Error: Defecto o desviación entre el comportamiento esperado y el comportamiento observado en el producto.
- Requisito de calidad de servicio: Material resultante esperado del producto final. El mismo puede ser un resultado, un problema resuelto o una característica, entre otros.
- Tarea: Acción independiente que debe realizar una persona o un grupo de personas.
- Riesgo: Evento o condición probable que puede dar resultados potencialmente negativos en el proyecto en un futuro.

Entre las principales ventajas que presenta la metodología se tiene que:

- Mantiene siempre el enfoque al usuario. Es decir, ayudándolo a asegurar que la solución implantada realmente es lo que el usuario necesita, porque mejorará su desempeño.
- Proporciona elementos valiosos a las inevitables preguntas: ¿Cuáles elementos van en el cliente?, ¿Cuáles en el servidor?
- Permite desarrollar siguiendo una filosofía de reutilización de múltiples componentes, lo cual reduce el tiempo necesario para desarrollar nuevas aplicaciones, garantiza la uniformidad e interoperabilidad entre las mismas, y las hace mucho más flexibles para incorporar los cambios que sean necesarios en el futuro.

En el departamento de seguridad digital se tiende a utilizar esta metodología como guía de desarrollo de software, por lo que se selecciona la misma teniendo en cuenta las ventajas que proporciona su aplicación para el proceso de desarrollo del sistema.

1.6. Framework .Net 4.0

Un Framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. Mediante la utilización del mismo se estructura el código fuente, forzando al

desarrollador a crear un código más legible y fácil de mantener. Además, se representa una arquitectura de software que modela las relaciones generales de las entidades del dominio.

Es usado también para el desarrollo de aplicaciones web ASP.NET entre los que se encuentran sitios web dinámicos y servicios web, permitiendo a los desarrolladores escribir código ASP.NET usando cualquier lenguaje admitido en la plataforma .NET. La biblioteca de clases .NET Framework, incluye ADO.NET, ASP.NET, formularios Windows Forms y Windows Presentation Foundation (WPF) (10).

Está formado por una serie de herramientas y librerías con las que se pueden crear todo tipo de aplicaciones, ya sean servicios, aplicaciones de escritorio o web entre otros. Proporciona un entorno de ejecución administrado, un desarrollo e implementación simplificada y la integración con una gran variedad de lenguajes de programación. Los servicios de datos de WCF son componentes de .NET Framework que permiten crear servicios y aplicaciones (11).

Microsoft .NET Framework 4.0 proporciona nuevas mejoras y características que la hacen diferente a sus versiones anteriores, aunque posee la capacidad de funcionar en paralelo con ellas. Algunas de estas mejoras son:

- En el control del *View State*. Obtiene un diccionario con información de estado que le permite guardar y restaurar el estado de vista de un control de servidor en las distintas solicitudes de la misma página.
- En la generación de ID's de los controles ASP.NET.
- Mejoras en la creación de los controles *FormView* y *ListView* para el comportamiento de datos dinámicos.
- ASP.NET Chart Control que permite crear páginas ASP.NET que incluyan gráficos de análisis estadístico o financiero que sean complejos. El control *Chart* admite las características siguientes: Series de datos, áreas de gráfico, ejes, leyendas, etiquetas y títulos.

1.7. Lenguaje de modelado

El lenguaje de modelado seleccionado para guiar el diseño de la aplicación fue el Lenguaje Unificado de Modelado (UML por sus siglas en inglés). UML es la sucesión de una serie de métodos de análisis y diseño orientado a objetos. Los métodos comprenden tanto el lenguaje de modelado como el proceso que

se sigue para obtenerlo. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios (12).

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño. La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de modelado y no el proceso que se siguió para obtenerlo (12).

UML es el lenguaje de modelado de sistemas de *software* más conocido en el mundo. Cuenta con un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación y componentes reutilizables.

Los principales beneficios de UML son que:

- Modela sistemas utilizando conceptos orientados a objetos.
- Establece conceptos y artefactos ejecutables.
- Encamina el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crea un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Tiene un mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones en:

- Visualizar: permite expresar gráficamente un sistema de forma que otra persona lo pueda entender.
- Especificar: permite especificar 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 revisión.

1.8. Herramienta de modelado

La herramienta de modelado Altova UModel es el punto de entrada para el desarrollo de *software* exitoso. Permite crear e interpretar diseños *software* mediante la potencia del estándar UML 2.1. Dibujar su diseño de aplicación y generar código Java o C# partir de sus planos, o hacer ingeniería inversa de programas existentes a diagramas UML claros y precisos para abarcar rápidamente su arquitectura de *software*, el código generado o sus modelos y completar la ronda produciendo automáticamente nuevos diagramas o regenerando el código (13).

Características principales:

- Soporta 14 tipos de diagramas UML.
- Modela esquemas XML en diagramas UML.
- Diagrama el proceso de negocio.
- Genera un código fuente en lenguajes Java, C#, y VB.NET.
- Ingeniería inversa de código fuente y ficheros binarios Java, C# y VB.NET.
- Sincroniza el modelo y el código a través de ingeniería de ida y vuelta.
- Crea diagramas de secuencia desde el código fuente de la ingeniería inversa.
- Genera documentación personalizable del proyecto.
- Se integra con sistemas de control de versiones.
- Estrecha integración con Visual Studio y Eclipse (14).

Altova es una herramienta rápida y eficaz que permite corregir el código generado o sus modelos produciendo automáticamente nuevos diagramas o regenerando el código. UModel es práctico para todos los programadores y gestores de proyecto. Su interfaz visual y usabilidad superior ayudan a incrementar el aprendizaje en términos de UML, permitiendo que desarrolladores, incluso los nuevos en el modelado de *software*, manejen rápidamente UML, potenciando su productividad y maximizando resultados. De manera sencilla UModel permite mantener el proyecto sincronizado y al día.

1.9. Lenguajes de programación

CSharp (C#)

C# es el lenguaje de propósito general diseñado por *Microsoft.* Programar en este es mucho más sencillo e intuitivo que hacerlo con alguno de los otros lenguajes existentes. Entre sus principales características se destacan:(15)

- **Sencillez:** C# elimina elementos que otros lenguajes incluyen y que son innecesarios en .NET, por ejemplo: El código escrito en C# es auto contenido, lo que significa que no necesita de ficheros adicionales tales como ficheros de cabecera. El tamaño de los tipos de datos básicos es fijo e independiente del compilador (15).

- **Orientación a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir plácidamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros) (15).

- **Eficiente:** En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador *unsafe*) y podrán usarse en ellas punteros de forma similar a como se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad de procesamiento muy grande (15).

JavaScript

JavaScript es un lenguaje de programación que realiza acciones dentro del ámbito de una página web (16). Este es uno de los lenguajes de programación del lado del cliente más utilizado, debido a que es compatible con la mayoría de los navegadores modernos, además, es un lenguaje con muchas posibilidades, pues permite la programación de pequeños *scripts y* programas más grandes, orientados a objetos, con funciones y estructuras de datos complejas. Es bastante sencillo y pensado para hacer las cosas con rapidez, por lo que es fácil de aprender y utilizar.

Con este lenguaje se puede definir interactividades con el usuario y se puede crear efectos especiales en las páginas, donde se tienen dos variantes:

- Permite la ejecución de instrucciones como las respuestas a las acciones del usuario, con lo que se puede crear páginas interactivas
- Los efectos especiales sobre las páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo.

Java Script pone a disposición del programador todos aquellos componentes que forman la página web, para que el mismo pueda acceder a ellos y modificarlos dinámicamente. El mayor recurso o tal vez el único, con que cuenta este lenguaje es el propio navegador, puesto que, el navegador del cliente es el que se encarga de interpretar y ejecutar las instrucciones Java Script (17).

Cascading Style Sheets (CSS)

Las hojas de estilo se desplegaron para remediar los defectos de HTML[5] relacionados con la presentación y el diseño de las páginas. El principio fundamental de las hojas de estilo radica en el uso de un solo documento para almacenar las características de presentación de las páginas asociadas a grupos de elementos (18).

Al utilizar las hojas de estilo, cuando se necesita cambiar el aspecto de un sitio que tiene muchas páginas web lo único que hay que hacer es editar las funciones de la hoja de estilo en un único lugar para cambiar la apariencia del sitio completo. Debido a que se pueden definir múltiples hojas y los estilos pueden aplicarse a todas las páginas es que se denominan "hojas de estilo en cascada". Las hojas de estilo pueden utilizarse para:

- Lograr una apariencia uniforme de todo el sitio al activar una sola definición de estilo en cada página.
- Cambiar un aspecto en todo el sitio web con tan solo editar unas pocas líneas.
- Facilitar la lectura de los códigos HTML, ya que los estilos se definen por separado.
- Permitir que las páginas se carguen más rápido, ya que, hay menos cantidad de código HTML en cada página.
- Posicionar los elementos de la página de una manera más uniforme (18).

[5] Por sus siglas en inglés *Hypertext Markup Lenguaje.*

1.10. Entorno integrado de desarrollo (IDE)

Un entorno integrado de desarrollo (IDE por sus siglas en inglés Integrated Development Environment) es, como su nombre dice, un entorno de desarrollo que incluye un editor para escribir código de programación, que incluye un conjunto de herramientas integradas para la administración de proyectos de *software*. Algunas de las herramientas incluidas en el IDE ayudan al programador a probar el código en busca de errores.

Microsoft Visual Studio 2010

Microsoft Visual Studio es un IDE para sistemas operativos Windows. Dentro de los lenguajes de programación que soporta se encuentra el Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, entre otros. Este permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .Net (19).

Este IDE cuenta con una serie de ventajas que mejoran el proceso de desarrollo, entre las que se encuentran:

- **Productivo:** Visual Studio ofrece un conjunto de herramientas que posibilitan al desarrollador obtener mejores maneras de conseguir más resultados con menos recursos y esfuerzo. Desde editores de código eficaces, IntelliSense, asistentes y varios lenguajes de codificación en un mismo IDE hasta herramientas para el control del ciclo de vida de las aplicaciones. Estas herramientas hacen que los desarrolladores se centren en la solución del problema y no pierdan el tiempo en tareas sin importancia.
- **Integrado:** En Visual Studio se integran varias herramientas, servidores y servicios que benefician al desarrollador de *software*. Estas funcionan bien conjuntamente, no sólo entre ellos, sino también con otro *software* que pertenezca a Microsoft, como son: los productos de servidor de Microsoft y el sistema Microsoft Office.
- **Completo:** Visual Studio es un entorno de desarrollo completo, que cuenta con una inmensa variedad de herramientas, para cualquier fase del desarrollo de software. Es una herramienta diseñada para cualquier persona, tanto para principiantes como para profesionales con mucha

experiencia en el tema. Con Visual Studio, el programador puede desarrollar disímiles sistemas en toda clase de dispositivos, como son los equipos, los servidores y los dispositivos móviles.

1.11. Sistema gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la BD y el usuario o las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos (20). Aunque existen gran cantidad de gestores de bases de datos se decidió utilizar Oracle 11g porque este ya se encuentra definido en la línea de arquitectura del departamento.

Oracle 11g

Oracle 11g es un sistema de gestor de base de datos (SGBD), desarrollado por Oracle Corporation. Se considera como uno de los sistemas de bases de datos más completos destacando soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma (21). Garantiza la seguridad e integridad de los datos, la ejecución de transacciones de forma correcta, una fácil administración y almacenamiento de grandes volúmenes de datos.

Es una herramienta cliente/servidor para la gestión de bases de datos. Posee un lenguaje de diseño de bases de datos PL/SQL, un lenguaje de 5ta Generación, muy potente para tratar y gestionar la BD, y muy versátil, que permite implementar diseños "activos", con triggers y procedimientos almacenados, con una integridad referencial declarativa muy fuerte (22). Permite el uso de particiones para la mejora de la eficiencia, de recopilación e incluso ciertas versiones admiten la administración de bases de datos distribuidas. El *software* del servidor puede ejecutarse en multitud de sistemas operativos.

Oracle es ventajoso para el desarrollo del sistema ya que ha sido diseñado para que las organizaciones puedan controlar y gestionar grandes volúmenes de contenidos no estructurados en un único repositorio con el objetivo de reducir los costos y los riesgos asociados a la pérdida de información.

1.12. Conclusiones

El análisis de los referentes teóricos del tema de investigación permitió obtener los conocimientos necesarios para darle solución a la problemática planteada. El estudio de los sistemas similares hizo posible profundizar en las características y funcionalidades de los mismos de manera que puedan tenerse en cuenta para el desarrollo del sistema. Por último, la selección de la metodología y las tecnologías hará posible un correcto diseño e implementación del sistema.

Capítulo 2: Análisis y diseño del sistema

2.1. Introducción

En el capítulo se describe detalladamente la propuesta de solución, basado en el análisis previo del objeto de estudio y la problemática planteada. Se definen y especifican las funcionalidades que debe tener el sistema, así como los requisitos de calidad del servicio con los que debe cumplir. Además, se especifica la arquitectura del sistema y los patrones de diseño a usar para la implementación del sistema.

2.2. Fase Visión y Alcance

Al inicio del desarrollo de un software es de gran importancia tener definido una visión y un alcance del mismo. Por lo que a continuación se describe la propuesta de solución que se quiere desarrollar para así definir un punto de partida para el equipo de desarrollo.

2.2.1. Propuesta de Solución

Se propone desarrollar un sistema que permita configurar y registrar los proveedores de servicios asociados al SAI. La solución contará con tres módulos (Ver Tabla 2): el módulo de configuración, el módulo de registro y el módulo de administración.

Módulo del sistema	Descripción
Módulo de configuración	Es el encargado de realizar todo el proceso de configuración y registro del proveedor de servicios y de búsqueda de recursos.
Módulo de registro	Es el encargado de realizar todo el proceso de registro de datos y recursos pertenecientes al proveedor de servicios en la BD.
Módulo de administración	Este módulo permite la gestión de los proveedores de servicios y sus recursos registrados.

Tabla 2: Módulos del sistema propuesto.

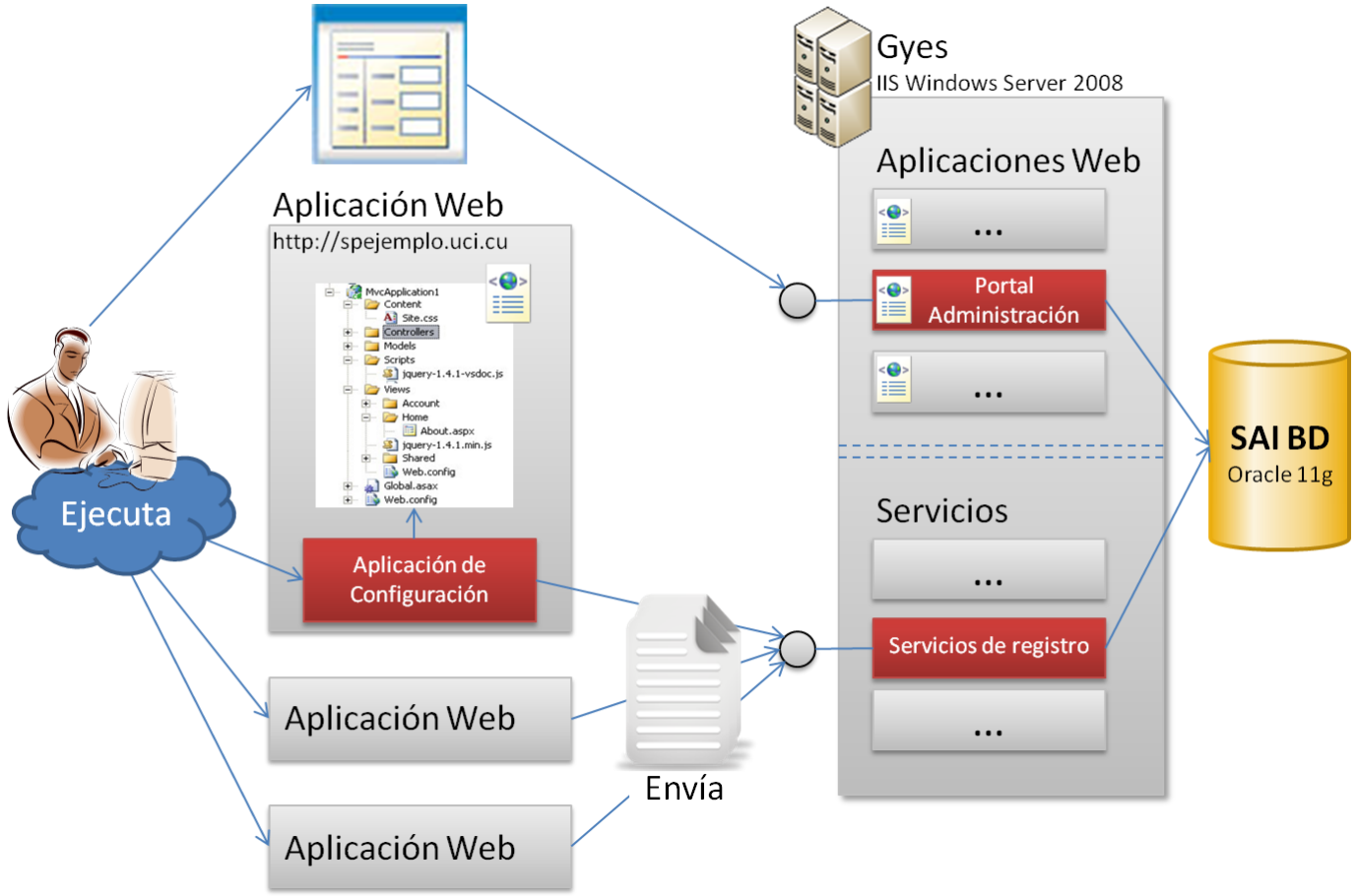

Figura 4: Propuesta de solución.

Como se puede observar en la figura anterior el módulo de configuración está representado por una aplicación de escritorio, que se encargará de cargar, mostrar, modificar y guardar los datos del archivo de configuración del proveedor de servicio a integrar; además, buscará los recursos correspondientes al proveedor de servicios mediante la utilización de una araña web y copiará librerías de clases, así como otros archivos a directorios locales. Para realizar la configuración este debe ejecutarse dentro del Sistema Operativo que aloja al proveedor de servicios ya que este carga directamente desde el disco el *web.config* del mismo.

El módulo de registro está representado por un servicio web que se encargará de registrar los datos pertenecientes al proveedor de servicios en la BD provenientes del módulo de configuración.

El módulo de administración está representado por un portal web que le permitirá a la persona encargada administrar los proveedores de servicios registrados en la BD, así como los recursos que pertenecen a cada uno de estos.

2.2.2. Definición de personas

En la metodología MSF se define a las personas como todos los individuos o sistemas externos que interactúan con la aplicación. Una persona describe las habilidades, necesidades, deseos, hábitos de trabajo y tareas de un particular grupo de usuarios en un carácter ficticio. El objetivo fundamental de esta es describir y crear conocimiento sobre un grupo de usuarios del sistema.

Se define que la persona Administrador será el encargado de realizar la configuración y registro de proveedores de servicios y los recursos de este.

2.3. Fase Planificación

En la fase de Planificación el equipo hace las especificaciones funcionales, un plan maestro para el proyecto y un calendario maestro. Las especificaciones funcionales describen lo que se va a desarrollar, incluyendo contenido como metas de diseño, requerimientos, características y dependencias. Comprende el diseño conceptual, lógico y físico. El plan maestro describe cómo se va a desarrollar la solución y con qué recursos; y el calendario maestro describe cuándo y en qué orden se va a desarrollar la solución. Asimismo se documentan los posibles riesgos que pudieran modificar el plan de desarrollo.

2.3.1. Lista de escenarios del sistema

Un escenario es un tipo de artículo de trabajo, que registra los pasos de interacción del usuario a través del sistema. Cuando una persona intenta alcanzar una meta, el escenario registra los pasos específicos que se realizan para alcanzar el objetivo. Algunos escenarios registrarán los caminos exitosos; mientras que otros registran los no exitosos. A la hora de escribir los escenarios se debe especificar que existen muchos caminos posibles.

Un escenario puede encontrarse en cuatro estados fundamentales que indican en que etapa de desarrollo se encuentra el mismo. En la Figura 5 se muestra los diferentes estados en los que puede encontrase un escenario junto con la relación que existe entre ellos.

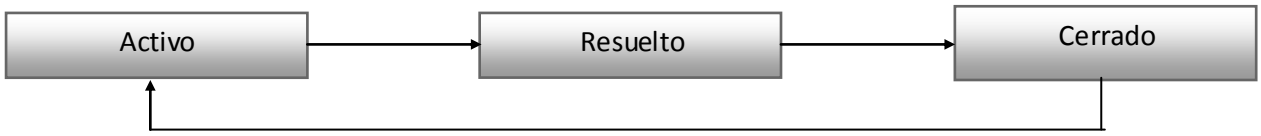

Figura 5: Representación de los estados y sus relaciones.

Para el desarrollo del sistema propuesto se identificaron los siguientes escenarios:

SC 1. Configurar proveedor de servicios.
SC 2. Registrar proveedor de servicios.
SC 3. Gestionar recursos.
  SC 3.1. Adicionar recurso.
  SC 3.2. Eliminar recurso.
  SC 3.3. Buscar recursos.
SC 4. Registrar recursos.
SC 5. Administrar proveedor de servicios.
  SC 5.1. Mostrar lista de proveedores de servicios.
  SC 5.2. Ver recursos de proveedor de servicios.
  SC 5.3. Eliminar proveedor de servicios.
SC 6. Autenticar usuario.

2.3.2. Priorización de la lista de escenarios

La priorización de los escenarios permite identificar los más importantes en el momento de implementar, para que los escenarios que tengan prioridad más alta se realicen en las primeras iteraciones. Calificar al escenario con 5 es equivalente a que tiene una prioridad “Alta”, con 4 “Media” y con 3 “Baja”. (Ver Tabla 3).

Escenarios	Prioridad
Configurar proveedor de servicio	5
Registrar proveedor de servicio	4
Gestionar Recursos	5
Registrar recursos	4
Administrar proveedor de servicios	4
Autenticar usuario	5

Tabla 3: Prioridad por escenario

2.3.3. Requisitos de calidad de servicio

Los requisitos de calidad de servicio representan los requisitos no funcionales en las categorías de plataforma, rendimiento, seguridad y apariencia. Estos documentan características del sistema describiéndose cómo el sistema debe operar y bajo qué condiciones. A continuación se describen requisitos de calidad del servicio definidos para el desarrollo del sistema.

Software:

- El sistema operativo que aloja al proveedor de servicios necesita contar con el framework .Net v4.0.
- El servicio de registro debe alojarse en un servidor que contenga Internet Information Server instalado.
- Para acceder al portal de administración es necesario disponer de un navegador web, este puede ser IE 7, Opera 9, Google Chrome 1 y Firefox 2 o versiones superiores a estos.

Hardware:

- Los servidores deben tener como mínimo: un microprocesador Intel Pentium de 1 GHz, 512 MB de RAM y 2 GB de espacio en disco duro para la instalación.

- Las estaciones de trabajo deben tener como mínimo 256 Mb de memoria RAM y un microprocesador de 2.0 Hz.

Fiabilidad:

- El servicio web deberá estar disponible las 24 horas del día para que las aplicaciones de configuración puedan interactuar con él cuando se requiera de sus funciones.
- Restricciones en el diseño y la implementación:
- El sistema debe implementarse usando el lenguaje C#, sobre la plataforma ASP.NET.
- El SGBD, será Oracle 11g.
- El sistema debe desarrollarse usando el IDE Visual Studio 2010.

Accesibilidad:

- La aplicación sólo va a estar disponible para los usuarios que tengan conocimientos sobre los aspectos necesarios para realizar la configuración.

Usabilidad:

- El sistema podrá ser usado por cualquier persona que posea conocimientos sobre la configuración que debe realizarse para hacer uso del SAI.
- El sistema estará diseñado para una resolución de 1024x768 píxel.
- El sistema será distribuido en idioma español.
- El sistema poseerá una estructura y un diseño homogéneo en todas sus pantallas, que facilite la navegación.

Rendimiento:

- Tener un alto nivel de respuesta, tanto para el acceso a la BD, como para los diferentes procesos que realiza.

2.3.4. Plan de Iteraciones

Para realizar de la manera más adecuada la planificación del desarrollo del sistema, es necesario hacer una estimación del tiempo que le tomará al programador ejecutar la codificación de cada uno de los escenarios. En relación con la prioridad que tenga cada escenario se decide cuáles de ellos se

desarrollarán en las primeras iteraciones, pues las funcionalidades críticas del sistema deben ser codificadas en las iteraciones más tempranas de su ciclo de vida. (Ver Tabla 4)

A partir de la definición de los escenarios con que debe contar el sistema, su priorización y el tiempo con que se cuenta para el desarrollo del mismo, se define dos iteraciones de desarrollo para obtener la solución. A continuación se muestra la distribución de los escenarios por iteraciones:

- **Iteración #1:** Se propone codificar los escenarios que proveen las funcionalidades con mayor prioridad.
- **Iteración #2:** Se codificarán los escenarios que proveen las funcionalidades con una prioridad media.

No	Escenario	Prioridad	Riesgo	Esfuerzo (días)	Iteración
1	Configurar proveedor de servicios.	Alta	Alto	25	1
2	Registrar proveedor de servicios.	Media	Alto	3	2
3	Gestionar recursos.	Alta	Alto	60	1
4	Registrar recursos.	Media	Alto	8	2
5	Administrar proveedor de servicios.	Media	Alto	5	2
6	Autenticar usuario.	Alta	Alto	15	1

Tabla 4: Planificación de escenarios.

2.3.5. Descripción de los escenarios

Para el desarrollo del sistema y un mejor entendimiento de las funcionalidades del mismo, se muestra a continuación la especificación de algunos de los principales escenarios que se definieron, los restantes se pueden consultar en el Anexo II.

Nombre del escenario: Configurar proveedor de servicios		**Identificador:** SC1
Objetivo del escenario: Realizar la configuración del proveedor de servicio para lograr la correcta integración con el SAI.		
Persona: Administrador		
Iteración: 1ra	**Prioridad:** Alta	**Complejidad:** 1
Precondiciones:		
Este escenario comienza cuando se presiona el botón “Configurar proveedor de servicio”. Como respuesta el sistema muestra la ventana “Seleccionar archivo de configuración” (Ver Figura 6) que brinda la opción de “Examinar”. Cuando el administrador presiona el botón “Examinar” el sistema muestra una ventana donde debe buscar el archivo de configuración del proveedor de servicios. Inmediatamente que se selecciona el archivo de configuración el sistema carga y muestra en una ventana (Ver Figura 7) los datos de configuración contenidos en el archivo. El administrador después de realizar cambios en los datos presiona el botón “Siguiente”. El sistema muestra una ventana (Ver Figura 8) donde le permite al administrador seleccionar donde desea guardar las librerías de clases. El administrador selecciona copiar las librerías a una carpeta específica, luego presiona el botón “Buscar” y el sistema le muestra una ventana donde el administrador debe buscar la carpeta donde desea copiar las librerías de clases.		

Después que el administrador selecciona la carpeta destino y presiona el botón “Siguiente” el sistema muestra una ventana donde muestra las tareas que está realizando en forma de mensajes, concluyendo de esta forma el escenario.

Validaciones:

- El archivo de configuración debe existir y debe tener una estructura correcta.
- No deben existir campos vacíos.
- Los datos deben estar en un formato correcto.

Prototipo de interfaz de usuario

Figura 6: Prototipo de interfaz gráfica para el escenario SC 1.

Figura 7: Prototipo de interfaz gráfica para el escenario SC 1.

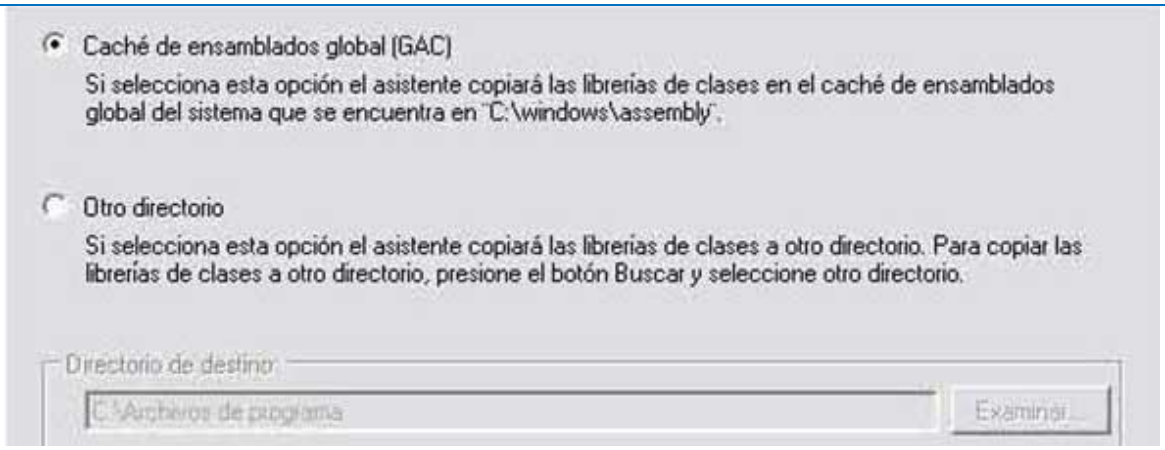

Figura 8: Prototipo de interfaz gráfica para el escenario SC 1.

Requisitos de calidad del servicio	

Tabla 5: Descripción del escenario "Configurar proveedor de servicios".

A continuación, se muestra la descripción de un escenario con prioridad Media. Los restantes se pueden apreciar en el Anexo I.

Nombre del escenario: Registrar proveedor de servicios		**Identificador:** SC 2
Objetivo del escenario: Registrar los datos del proveedor de servicios.		
Persona: Administrador.		
Iteración: 2da	**Prioridad:** Media	**Complejidad:** 1
Precondiciones:		
Este escenario comienza cuando el administrador presiona el botón “Registrar proveedor de servicios”. Como respuesta el sistema muestra un formulario donde el administrador debe introducir los datos del proveedor de servicios a registrar. Después de introducir los datos correspondientes al proveedor de servicio el administrador presiona el		

botón “Siguiente”. El sistema registra el proveedor de servicios y muestra una ventana donde informa al administrador que el proveedor de servicios fue registrado correctamente.
Validaciones: • Verificar que no existan campos vacíos. • Verificar que los datos introducidos tengan un formato correcto.
Prototipo de interfaz de usuario 

Figura 9: Prototipo de interfaz gráfica “Registrar proveedor de servicios”.

Requisitos de calidad del servicio	

Tabla 6: Descripción del escenario "Registrar proveedor de recursos".

2.3.6. Especificación de tareas por escenarios

El escenario “Gestionar recursos” se dividió en tres tareas fundamentales, que proveerán las funcionalidades de “Adicionar”, “Buscar” y “Eliminar” recursos. A continuación se muestra la descripción de una de estas tareas las demás pueden apreciarse en el Anexo III.

Nombre del escenario: Adicionar recurso	**Identificador:** SC 3.1
Objetivo del escenario: Adicionar un recurso al árbol de recursos.	
Persona: Administrador.	

Iteración: 2da	**Prioridad:** Media	**Complejidad:** 1
Precondiciones:		
Esta tarea comienza cuando el administrador presiona el botón “Adicionar”. Como respuesta el sistema muestra un formulario (Ver Figura 10). Luego el administrador introduce la dirección del recurso y presiona el botón “Aceptar”. El sistema como respuesta agrega al árbol de recursos el nuevo recurso creado.		
Validaciones: • Verificar que no existan campos vacíos. • Verificar que se haya seleccionado un recurso del árbol de recursos. En este escenario no es necesario realizar validaciones.		
Prototipo de interfaz de usuario		
Requisitos de calidad del servicio		

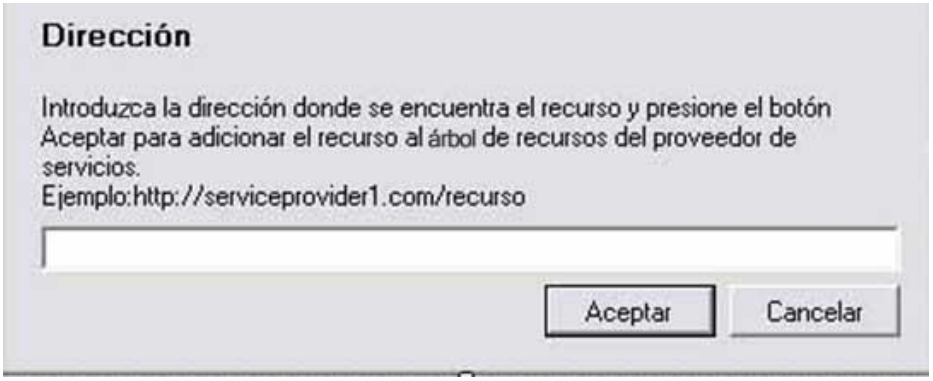

Figura 10: Prototipo de interfaz gráfica "Adicionar recurso".

Tabla 7: Descripción del escenario "Adicionar recurso”.

El escenario “Administrar proveedores de servicios” se dividió en dos tareas fundamentales, que proveerán las funcionalidades de “Ver recursos” y “Eliminar” proveedores de servicios. A continuación se muestra la descripción de una de estas tareas las demás pueden apreciarse en el Anexo III.

<table>
<tr><td colspan="2">Nombre del escenario: Ver recursos</td><td>Identificador: SC 5.2</td></tr>
<tr><td colspan="3">Objetivo del escenario: Mostrar los recursos de un proveedor de servicios.</td></tr>
<tr><td colspan="3">Persona: Administrador.</td></tr>
<tr><td>Iteración: 2da</td><td>Prioridad: Media</td><td>Complejidad: 1</td></tr>
<tr><td colspan="3">Precondiciones:</td></tr>
<tr><td colspan="3">El administrador accede a la aplicación. El sistema muestra un listado de todos proveedores de servicios existentes y brinda la posibilidad de seleccionar el proveedor de servicios del cual desea ver los recursos. El administrador realiza la selección y oprime el botón “Ver recursos”.

Finalmente, el sistema muestra todos los recursos pertenecientes al proveedor de servicios seleccionado.</td></tr>
<tr><td colspan="3">Validaciones:
En este escenario no es necesario realizar validaciones.</td></tr>
<tr><td colspan="3">Prototipo de interfaz de usuario</td></tr>
<tr><td>Requisitos de calidad del servicio</td><td colspan="2"></td></tr>
</table>

Tabla 8: Descripción del escenario “Ver detalles”.

2.4. Estilo arquitectónico a utilizar

Desde los inicios de la arquitectura de *software*, se observó que en la práctica del diseño y la implementación ciertas regularidades de configuración aparecían una y otra vez como respuesta a similares demandas. El número de esas formas no parecía ser muy grande. Muy pronto se les llamó estilos, por analogía con el uso del término en arquitectura de edificios. Un estilo describe entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Esas piezas se encuentran repetidamente en la práctica, trasuntando la existencia de decisiones estructurales coherentes. Una vez que se han identificado los estilos, es lógico y natural pensar en re-utilizarlos en situaciones semejantes que se presenten en el futuro (23). Igual que los patrones de arquitectura y diseño, todos los estilos tienen un nombre: cliente-servidor, modelo-vista-controlador, tubería-filtros, arquitectura en capas, entre otros (24).

Estilo arquitectónico en capas

Los sistemas o arquitecturas en capas constituyen uno de los estilos que aparecen con mayor frecuencia mencionados como categorías mayores del catálogo o, por el contrario, como una de las posibles encarnaciones de algún estilo más envolvente. En algunos ejemplares, las capas internas están ocultas a todas las demás, menos para las capas externas adyacentes, y excepto para funciones puntuales de exportación; en estos sistemas, los componentes implementan máquinas virtuales en alguna de las capas de la jerarquía. En otros sistemas, las capas pueden ser sólo parcialmente opacas. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas (24).

En un estilo en capas, los conectores se definen mediante los protocolos que determinan las formas de la interacción. Los diagramas de sistemas clásicos en capas dibujaban las capas en adyacencia, sin conectores, flechas ni interfaces; en algunos casos se suele representar la naturaleza jerárquica del sistema en forma de círculos concéntricos (24).

Las ventajas del estilo en capas son obvias. Ante todo, el estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. En segundo lugar, el estilo admite muy naturalmente optimizaciones y refinamientos. En tercer lugar, proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la

medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas (24).

A continuación se muestra la vista lógica de la arquitectura del sistema de manera general. En esta figura se muestran la capaz que compondrán el sistema y la relación que existe entre ellas.

Figura 11: Vista lógica de la arquitectura del sistema.

2.4.1. Descripción de las capas de la Arquitectura

Capa de presentación: en esta capa se encuentran todas las interfaces gráficas que presenta el sistema al usuario, con el propósito de mostrar o capturar información. Esta capa tiene una interacción directa con la capa de negocio.

Capa de servicios: esta capa está formada por clases *Service* dónde cada método de estas clases representa una regla de negocios. La responsabilidad de esta capa es exponer a la capa de presentación todas las operaciones que se correspondan con la lógica de negocios.

Capa de negocio: esta capa contiene todas las clases necesarias para realizar de manera correcta todas las funcionalidades que dan solución a los requerimientos del negocio. Contiene las clases Controladoras, que son las que manejan todas las operaciones sobre las entidades del negocio definidas y las clases Entidades que representan lo que se gestiona en la aplicación. Esta capa se comunica con la capa de

presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de BD almacenar o recuperar datos de él.

Capa de acceso a datos: Esta capa sirve como puente entre la capa lógica de negocio y el proveedor de datos. Pretende encapsular las especificidades del proveedor de datos tales como (Oracle y archivos XML), a la siguiente capa, para cambiar en una sola capa el proveedor de datos si se necesita.

Capa de datos: esta capa está constituida por todo el conjunto de documentos y procedimientos que permiten el almacenamiento de la información recolectada.

2.5. Patrones de Diseño

Los patrones solucionan un problema en un contexto particular**.** Además, impone una regla sobre la arquitectura y es usado junto a un estilo arquitectónico para determinar la forma de la estructura general de un sistema. Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, e identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.

Patrones GRASP

GRASP es un acrónimo de General Responsibility Assignment *Software* Patterns (patrones generales de *software* para asignar responsabilidades). El nombre se eligió para sugerir la importancia de aprender estos principios para diseñar con éxito el *software* orientado a objetos. Los patrones GRASP son utilizados para describir los principios fundamentales del diseño y la asignación de responsabilidades (26). Se considera que más que patrones, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de *software*. Estos patrones se dividen en varias categorías, y las que se usan para desarrollar la aplicación son las siguientes:

Alta cohesión: La cohesión es una medida de fuerza de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con baja cohesión hace muchas cosas no relacionadas, o hace demasiado trabajo. Tales clases no son beneficiosas ya que son difíciles de mantener, de reutilizar y de entender. Una clase con alta cohesión mejora la claridad y la facilidad de su uso, su mantenimiento se simplifica y es fácil de reutilizar. Este patrón se evidencia en la aplicación ya que la clase ¨ConfigurationControler¨ y ¨RegistrationControler¨, son las encargadas de realizar todas las operaciones

de configuración y registro de proveedores de servicios, y de afectarse estas clases no afectarían las demás funcionalidades, por lo que se considera que tiene alta cohesión.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que un elemento está conectado a otro, un elemento con bajo acoplamiento no depende demasiado de otros elementos. Una clase con alto acoplamiento confía en muchas otras clases, tales clases podrían no ser deseables ya que son muy complicadas de entender, son difíciles de reutilizar y los cambios realizados en las clases relacionadas fuerzan cambios locales. Este patrón se evidencia dentro de la aplicación en las clases de acceso a datos ya que cada uno tiene bastante independencia de las clases de abstracción de datos, lo que permite una mayor reutilización.

Experto en información: En este patrón cada objeto es responsable por mantener su propia información. Si tiene relaciones de agregación con otros objetos, también será responsable de: conocer la información de ellos, de crearlos y delegarles las operaciones. Este patrón se evidencia en la clase “ConfigManager”, porque esta es la que maneja toda la información referente a la configuración.

Creador: La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad y reutilización. Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se favorece el bajo acoplamiento. En la aplicación se pone de manifiesto este patrón, cuando la clase “RegistrationControler” crea una instancia de la clase “Crawler” para invocar al método de búsqueda de recursos. (Ver Figura 12)

```
_crawler = new Crawler(depth);
_crawler.NewFolderFound += new Crawler.NewFolderCallback(NewFolderFound);
_crawler.ErrorSearching += new Crawler.ErrorSearchingCallback(CrawlerErrorSearching);
_crawler.CrawlFinished += new Core.Crawler.CrawlCompletedCallback(CrawlerCrawlFinished);
_crawler.StartScan(path);
```

Figura 12: Creación de una instancia de la clase “Crawler”.

2.6. Conclusiones

Con los conocimientos adquiridos en el estudio del estado del arte, y la aplicación de la metodología de desarrollo fue posible determinar el alcance de la propuesta de solución que se desea desarrollar, por lo que se pudo realizar una planificación acorde al tiempo con que se cuenta, definiendo así las iteraciones de desarrollo. Además, con la definición y especificación de los requisitos se tiene todo el diseño del sistema para su correcto funcionamiento.

La aplicación de los patrones de arquitectura seleccionados permitió realizar una programación ordenada y legible durante la implementación del software. La selección del estilo de arquitectura en capa para la implementación de la solución permitió al grupo de desarrollo dividir el problema en una secuencia de pasos incrementales, haciendo posible efectuar modificaciones en secciones determinadas del sistema, sin tener que cambiar el mismo completamente.

Capítulo 3: Implementación y prueba

3.1. Introducción

En el presente capítulo se abordan las fases de Desarrollo y Estabilización que propone la Metodología. Respondiendo a las actividades de estas fases, se modelan los diferentes diagramas necesarios para el desarrollo del sistema y se realizan las pruebas al sistema para comprobar así su correcto funcionamiento, y la calidad del mismo.

3.2. Fase de Desarrollo

En esta etapa el equipo se enfoca en desarrollar el sistema basándose en la arquitectura y las funcionalidades definidas para el mismo. Esta etapa involucra una serie de entregas internas de componentes del desarrollo, realizadas en paralelo y en segmentos, para medir el avance del desarrollo y asegurar que los componentes interactúen entre sí.

3.2.1. Diagrama de aplicación

El diagrama de aplicación es primordial en el momento de definir la arquitectura del sistema, según la metodología MSF para el desarrollo de *software* ágil. El mismo es una solución de ámbito, se crea con el objetivo de representar todos los componentes que se relacionan con la aplicación, como los servicios, las aplicaciones web, las aplicaciones de escritorio, las aplicaciones de BD y de servicios externos. Además, muestra las conexiones entre estas aplicaciones reflejando la actual configuración de la solución. (Ver Figura 13)

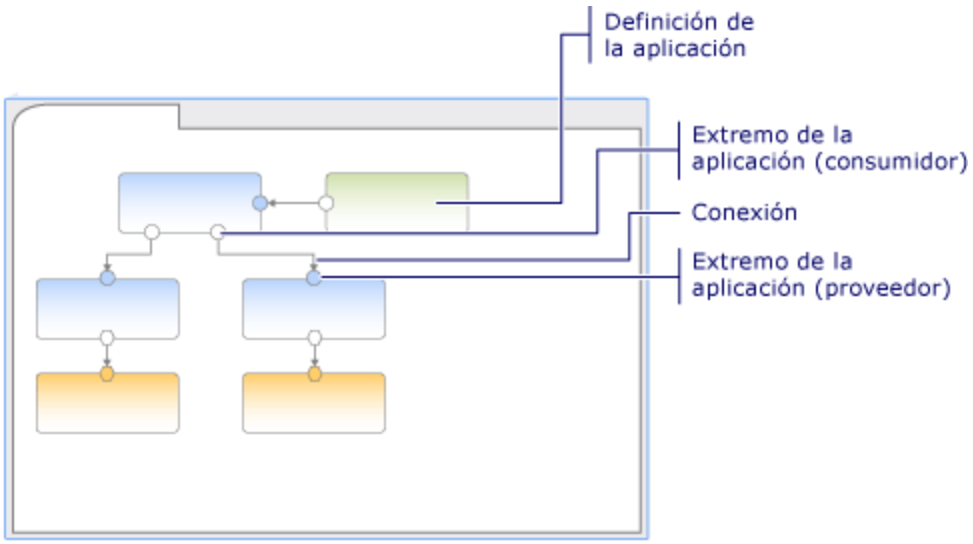

Figura 13: Representación del Diagrama de Aplicación (27).

En la Figura 14 se muestra el diagrama de aplicación del sistema a desarrollar. En el mismo se muestra una aplicación de escritorio, un servicio de Windows, una aplicación web y una BD externa. La aplicación de escritorio es la encargada de realizar los procesos de configuración y registro. Esta se conecta con el servicio de Windows brindándole los datos necesarios para realizar el registro de datos.

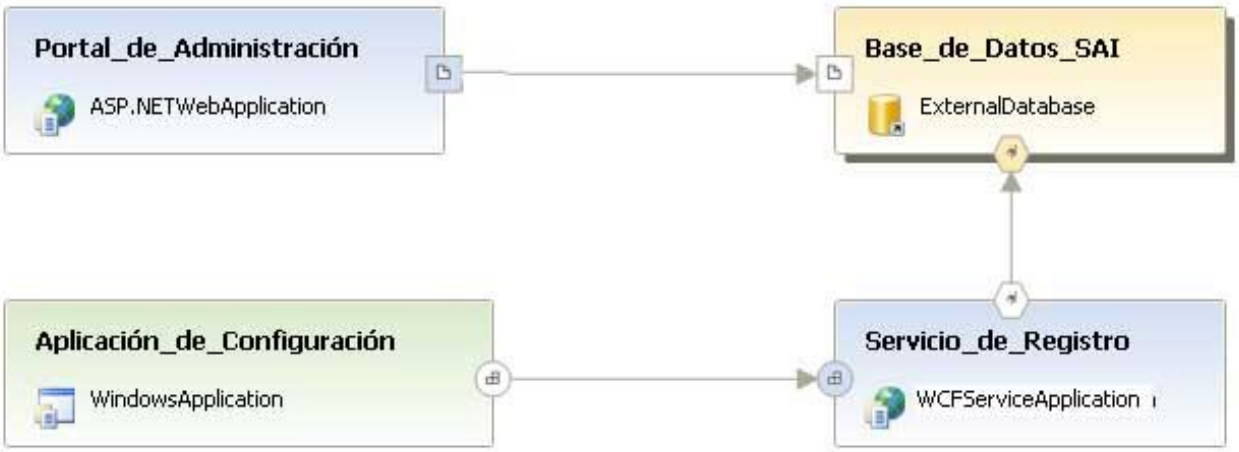

Figura 14: Diagrama de Aplicación.

Se puede observar en la figura anterior el servicio de Windows mediante el cual se establece la comunicación entre la aplicación de escritorio y la BD. La aplicación web se encarga de administrar los proveedores de servicios haciendo peticiones a la BD mediante la conexión entre ambas. La BD es la encargada de almacenar toda la información sobre los proveedores de servicios y los recursos de estos proveedores.

3.2.2. Diagrama lógico del centro de datos

El diagrama lógico de centro de datos representa y resume parte de un centro de datos físico utilizando servidores lógicos, extremos, conexiones y zonas como se muestra en la Figura 15. Este diagrama comunica información importante a los desarrolladores sobre el entorno de destino en el que se implementarán los sistemas de aplicación. La infraestructura física de un centro de datos no suele ser significativa a un desarrollador, que necesita comprender qué aplicaciones que alojan entornos están presentes, y cómo se configuran, restringen y conectan (28).

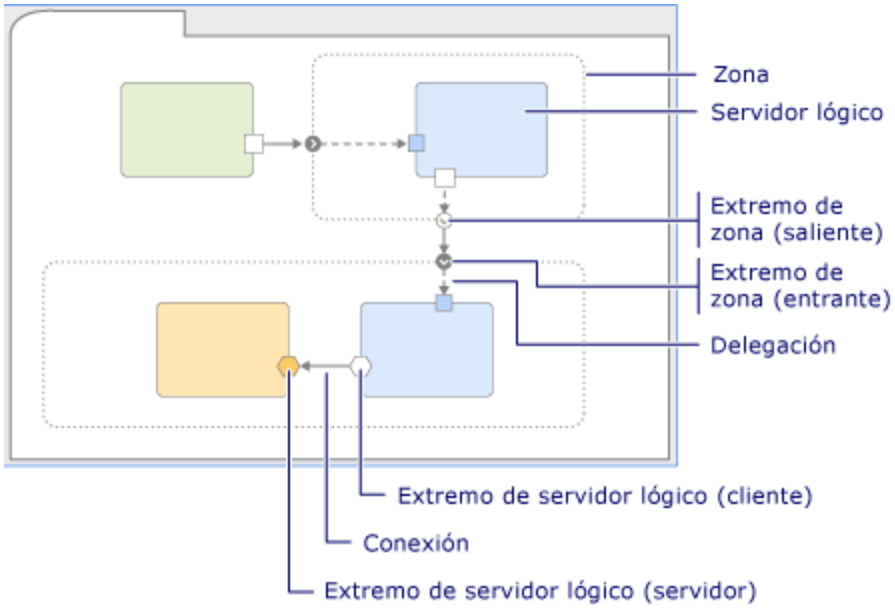

Figura 15: Representación del Diagrama de Centro de Datos Lógicos (28).

El diagrama de centros de datos lógicos no describe los equipos físicos o incluso los tipos de equipo en un centro de datos. Por el contrario, se utiliza para definir o documentar configuraciones concretas del

software del servidor de aplicaciones, como Internet Information Server, SQL Server o BizTalk Server, y mostrar cómo se interconectan estos servidores lógicos (28).

Las instrucciones siguientes describen qué es y qué no es un centro de datos lógico:

- Describe los tipos de servidores de aplicaciones y qué tipos de aplicación pueden alojar, no cuántos servidores físicos están presentes.
- Describe la configuración del servicio dependiente, no las características físicas.
- Describe qué protocolos están disponibles, no si se utiliza la tabla de enrutamiento IP.
- Describe los límites de comunicación, no los *firewalls*, VLANs, modificadores y enrutadores.
- Describe requisitos de autenticación de aplicaciones, no el cifrado en el nivel de conexión.
- Describe las restricciones en la configuración de la aplicación, sin tener que paginar el desarrollador (28).

En la Figura 16 se muestra el diagrama lógico de centro de datos para la solución propuesta. Este diagrama está conformado por tres servidores lógicos:

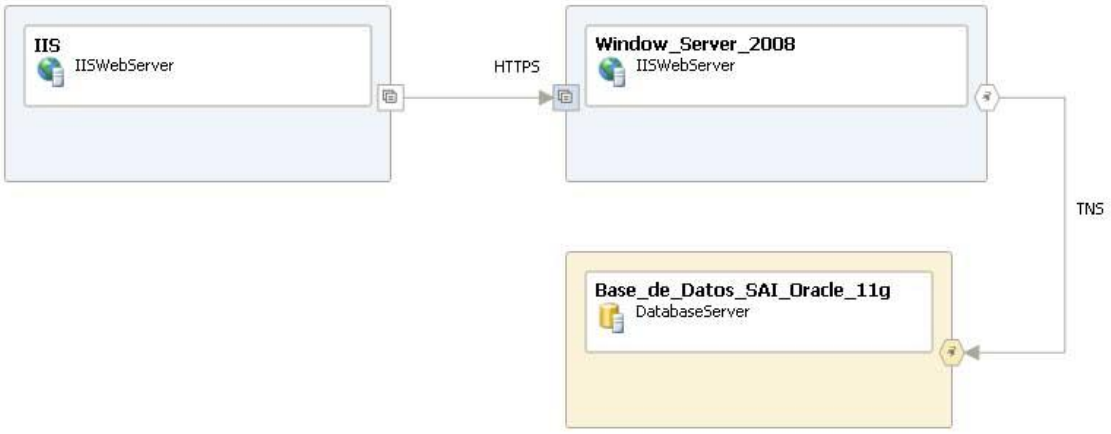

Figura 16: Diagrama Lógico de Centro de Datos.

IIS: Representa el servidor Internet Information Server (IIS) donde se alojará el proveedor de servicios. En este servidor también se ejecutará la aplicación de configuración.

Window_Server_2008: Representa el servidor de los Servicios de Internet Information Server (IIS) que alojará la aplicación Web de administración y el servicio de registro.

Base_de_Datos_SAI_Oracle_11g: Representa y documenta un servidor de BD.

3.2.3. Diagrama de clases

El diagrama de clases es un tipo de diagrama estático, que describe la estructura de un sistema representando las clases que serán utilizadas, sus atributos y las relaciones que existen entre ellas. Nos sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de convencimiento. Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso. A continuación se muestran algunas de las clases que componen el sistema:

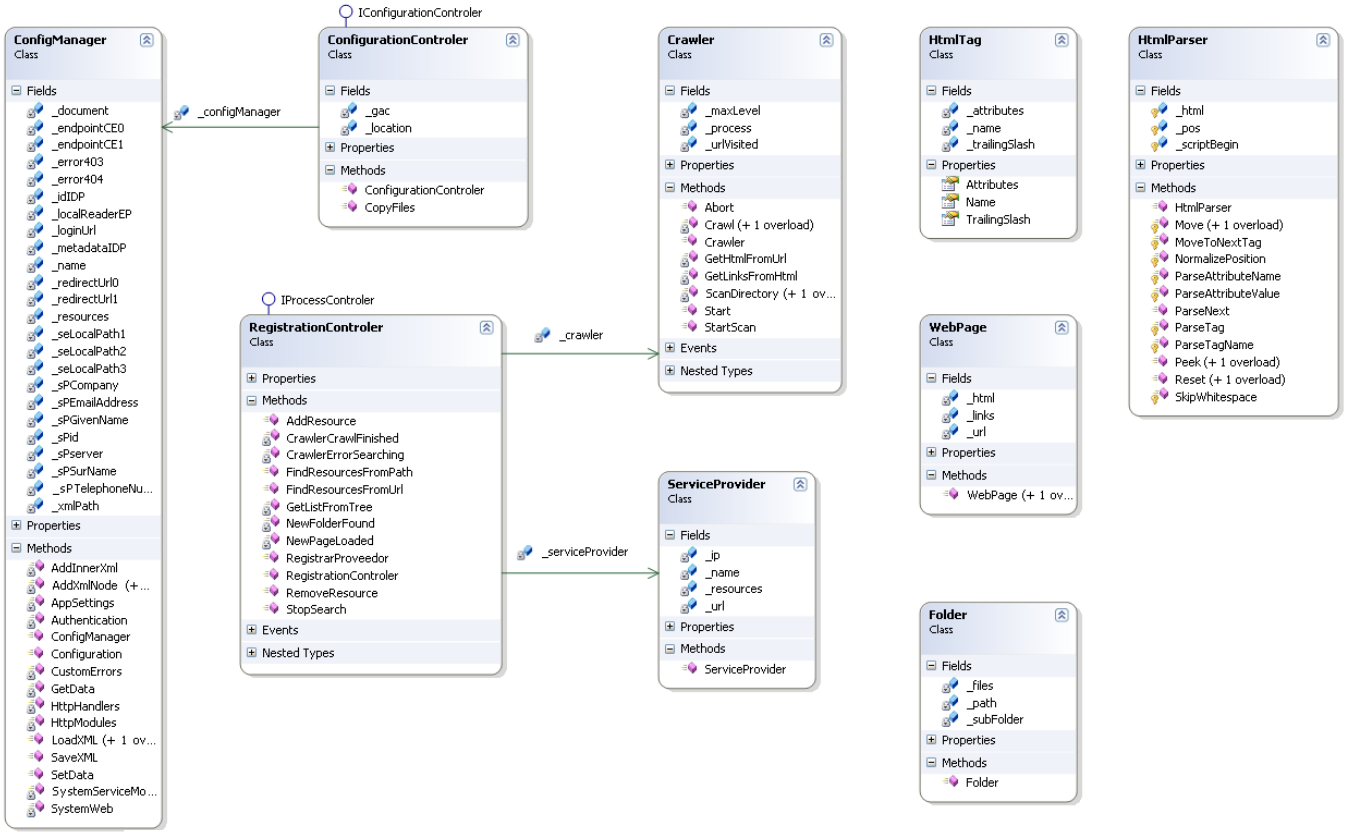

Figura 17: Diagrama clases del Módulo de Configuración.

A continuación se presentan dos descripciones de las clases definidas, las restantes podrán ser consultadas en el Anexo IV.

Nombre	*Crawler*	
Tipo de Clase	Controladora	
Descripción	Permite realizar búsquedas.	
Atributo	**Tipo**	**Descripción**
_maxLevel	int	Máximo nivel de profundidad de búsqueda.
_process	Thread	Hilo para realizar el trabajo en un subproceso.
_urlVisited	Hashtable	Url visitadas.
Método	**Descripción**	
Crawler()	Constructor de la clase.	
Crawl(string url):void	Permite escanear una página web a partir de una Url.	
Crawl(string url, int level):void	Permite escanear una página web a partir de una Url.	

Abort():void	Detiene el proceso de búsqueda.
GetHtmlFromUrl(string url, ref string html):bool	Obtiene el código HTML de una página web a partir de la URL.
GetLinksFromHtml(string html):List<string>	Obtiene los enlaces que se encuentran en una cadena HTML.
ScanDirectory(string path):void	Permite escanear una carpeta a partir de una dirección.
ScanDirectory(string path, int level):void	Permite escanear una carpeta a partir de una dirección.
Star(string url):void	Inicia el proceso de búsqueda.
StarScan(string url, int level):void	Inicia el proceso de búsqueda.

Tabla 9: Descripción de la clase "Crawler".

Nombre	*HtmlParser*	
Tipo de Clase	Controladora	
Descripción	Parsea código HTML.	
Atributo	**Tipo**	**Descripción**
_html	string	Código HTML.

_pos	int	Posición del cursor.
_scriptBegin	bool	Indica si existe código *script*.

Método	Descripción
HtmlParser()	Constructor de la clase.
Move():void	Mueve la posición actual hacia delante un caracter.
Move(int ahead)	Mueve la posición actual hacia delante, el número especificado de caracteres.
MoveToNextTag():bool	Mueve el cursor hacia el inicio de la siguiente etiqueta.
NormalizePosition():void	Normaliza la posición actual. Esto es principalmente, para el manejo de las condiciones en que indexOf () y otros, devuelven valores negativos porque el elemento buscado no fue encontrado.
ParseAttributeName():string	Parsea el atributo name de la etiqueta. La posición actual debe ser el primer caracter del name.
ParseAttributeValue():string	Parsea el valor del atributo name de la etiqueta. La posición actual debe ser la del primer caracter seguido del signo igual diferente al espacio en blanco.
ParseNext(string name, HtmlTag	Parsea la siguiente etiqueta que encaja con la etiqueta

tag):bool	especificada.
ParseTag(string name, HtmlTag tag):bool	Parsea el contenido de la etiqueta HTML.
ParseTagName():string	Parsea la etiqueta name.
Peek():char	Devuelve el caracter en la posición actual, o un caracter nulo, si estamos en el final del código HTML.
Peek(int ahead):char	Devuelve el caracter en la posición especificada a partir de la posición actual, o un caracter nulo, si estamos en el final del código HTML.
Reset():void	Reinicia la posición actual para comenzar desde el principio del código HTML.
Reset(string html):void	Restablece el código HTML y reinicia la posición actual para comenzar desde el inicio.
SkipWhiteSpace()	Salta los espacios en blanco.

Tabla 10: Descripción de la clase "HtmlParser".

3.2.4. Modelo de datos

Un modelo de datos es un conjunto de conceptos, reglas y convenciones que nos permiten describir y en ocasiones manipular los datos, de un cierto mundo real, que se desea almacenar en la BD. Al producto del modelo de datos se le llama esquema y a los datos en concreto, ocurrencia del esquema.

Para garantizar la seguridad e integridad de los datos relevantes de este proyecto se seleccionó como SGBD[6] a Oracle 11g porque propicia la ejecución de transacciones de forma correcta sin causar inconsistencias y permite manejar de manera clara, sencilla y ordenada grandes volúmenes de datos.

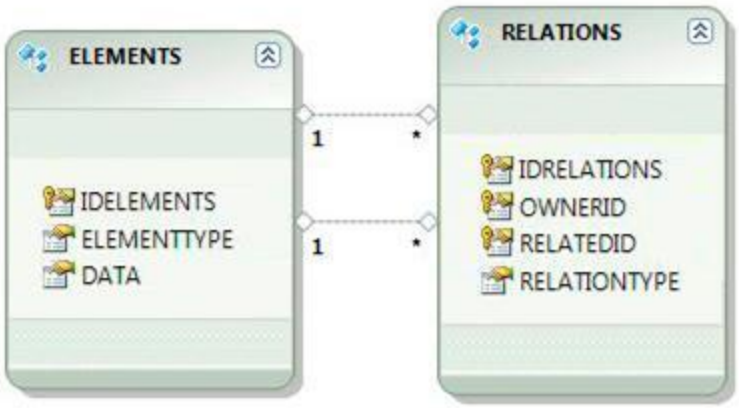

Figura 18: Modelo de datos.

Las clases persistentes en una aplicación implementan las entidades del problema de negocio. No todas las instancias de una clase persistente se consideran que estén en el estado persistente, una instancia puede en cambio ser transitoria o estar separada. A continuación se muestran las descripciones de las clases persistentes:

Nombre	*Elements.*	
Descripción	Representa un elemento	
Atributo	**Tipo**	**Descripción**
IDELEMENTS	string	Representa el identificador de un elemento y es único.

[6] Sistema gestor de bases de datos.

ELEMENTTYPE	string	Representa el tipo de elemento.
DATA	Xml	Archivo XML que contiene los atributos del elemento.

Tabla 11: Descripción de la clase persistente "Elements".

Nombre	*Relations.*	
Descripción	Representa la relación que existe entre dos elementos.	
Atributo	**Tipo**	**Descripción**
IDRELATIONS	string	Representa el único identificador de la relación.
OWNERID	string	Representa el identificador del elemento al que le pertenece la relación.
RELATEDID	string	Representa el identificador del elemento con el que se relaciona el OWNERID.
RELATIONTYPE	string	Representa el tipo de relación que existe entre los dos elementos relacionados.

Tabla 12: Descripción de la clase persistente "Relations".

3.2.5. Interfaz gráfica

Interfaz “Gestionar recursos”: permite gestionar todas las acciones que se realizan sobre los recursos del proveedor de servicios. Cuenta con las opciones adicionar, eliminar y buscar.

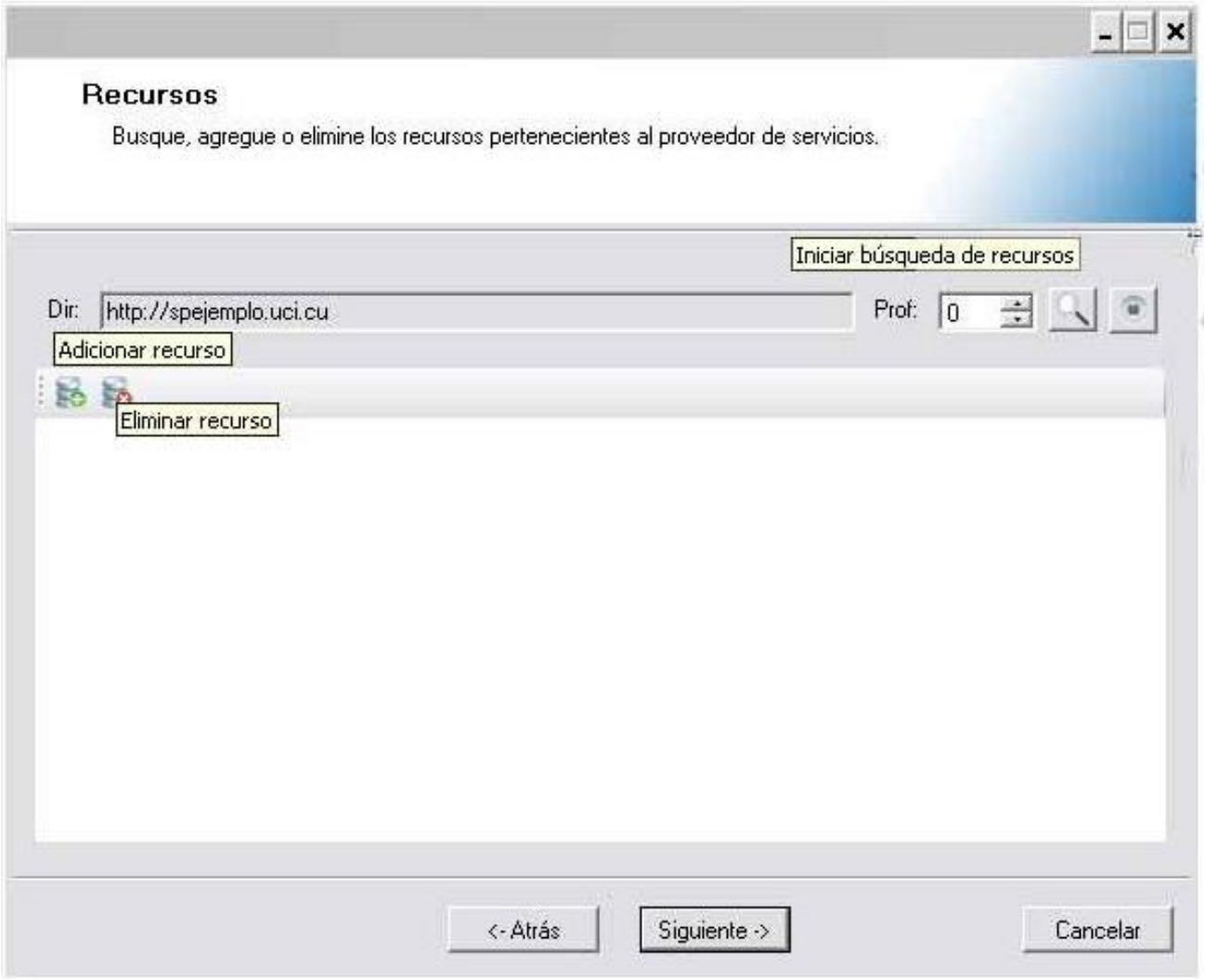

Figura 19: Interfaz gráfica “Gestionar recursos”.

Las restantes interfaces de la solución pueden ser consultadas en el Anexo V.

3.3. Fase de Estabilización

En esta fase se realizan un conjunto de pruebas a la solución para controlar que esta cumpla con los requisitos mínimos de operatividad y para garantizar la calidad de la misma. Las pruebas realizadas durante esta etapa enfatizan en el uso y operación de la solución bajo condiciones específicas y reales. La duración de esta fase es variable y depende de factores ajenos a la complejidad de la propia solución, como es la adecuada selección del entorno de pruebas.

3.3.1. Aplicación de Pruebas de Caja Negra

Las pruebas de caja negra conocidas también por pruebas funcionales, pruebas de caja opaca o pruebas de entrada/salida, se llevan a cabo sobre la interfaz del *software*, obviando por completo el comportamiento interno y la estructura del programa (Ver Figura 20). Esta puede ser llevada a cabo por cualquier persona aunque no tenga con conocimientos de programación.

Figura 20: Representación de la Prueba de Caja Negra.

Los casos de prueba de la caja negra pretenden demostrar que:

- Las funciones que proporciona el *software* son operativas.
- Los parámetros de entrada son aceptados de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Con la aplicación de las pruebas de caja negra al *software* se trata de encontrar un conjunto de errores que influyen en la calidad del mismo como son:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

A continuación se presentan dos descripciones de casos de pruebas, las restantes estarán en el Anexo VI.

Escenario: “Gestionar recursos”.

Sección del Escenario	Entrada	Resultado	Condiciones
Buscar recursos correctamente	El administrador introduce el nivel de búsqueda correctamente.	El sistema deshabilita todos los controles y comienza la búsqueda de recursos y medida que encuentra recursos los agrega al árbol de recursos.	
Adicionar recurso correctamente.	El administrador selecciona un recurso del árbol de recursos y presiona el botón “Adicionar recurso”. El administrador introduce la Url del recurso que quiere adicionar.	El sistema adiciona el recurso creado al nodo seleccionado.	
Adicionar recurso sin seleccionar un recurso.	El administrador presiona el botón “Adicionar recurso”.	El sistema muestra una ventana de notificación en la que informa al usuario que debe seleccionar un recurso.	El administrador no selecciona un recurso del árbol de recursos.

Adicionar recurso dejando campos vacíos.	El administrador selecciona un recurso del árbol de recursos. El administrador presiona el botón “Adicionar recurso”. El administrador deja los campos vacíos y presiona el botón “Adicionar”.	El sistema muestra una ventana de notificación donde le informa al administrador que no pueden existir campos vacíos.	El administrador deja campos vacíos.
Eliminar recurso correctamente.	El administrador selecciona un recurso del árbol de recursos. El administrador presiona el botón “Eliminar recurso”.	El sistema elimina del árbol de recursos el recurso seleccionado.	
Eliminar recurso sin seleccionar un recurso.	El administrador presiona el botón “Eliminar recurso”.	El sistema muestra una ventana de notificación en la que informa al usuario que debe seleccionar un recurso.	El administrador no selecciona un recurso.

Tabla 13: Descripción del caso de pruebas del escenario "Gestionar recursos".

Escenario “Registrar proveedor de servicio”.

Sección del Escenario	Entrada	Resultado	Condiciones

Registrar proveedor de servicios correctamente.	El administrador introduce los datos del proveedor de servicio correctamente.	El sistema registra el proveedor de servicios. El sistema muestra en la ventana de sucesos una notificación donde informa que se ha realizado el registro correctamente.	
Registrar proveedor de servicios dejando campos en blanco.	El administrador deja los campos vacíos.	El sistema muestra una ventana de notificación informando al usuario que no puede dejar campos vacíos.	El administrador deja campos vacíos.
Registrar proveedor de servicios introduciendo datos incorrectos.	El administrador introduce los parámetros incorrectamente.	El sistema muestra una ventana de notificación donde le informa al administrador que el formato de los datos introducidos es incorrecto.	El administrador introduce datos incorrectos.

Tabla 14: Descripción del caso de pruebas del escenario "Registrar proveedor de servicio".

Como parte de la automatización de este tipo de pruebas, se pueden realizar en Visual Studio 2010 las pruebas de interfaz de usuario (IU), conocidas como pruebas de IU codificadas proporcionan pruebas funcionales de la interfaz de usuario y validación de los controles de la interfaz de usuario. Las pruebas de IU automatizadas permiten probar que la interfaz de usuario funciona correctamente después de los cambios del código. Su ejecución es más rápida que las pruebas manuales (29).
En la Figura 21 se muestra la prueba de interfaz de usuario realizada a la aplicación de configuración.

```
public void RecordedMethod1()
{
    Variable Declarations
    // Click 'Unknown Name' list box
    Mouse.Click(ulItemList, new Point(510, 212));
    // Doble-Click 'Gyes.Configuration.SPRegClnt.exe'
    Mouse.DoubleClick(ulGyesConfigurationSPRListItem, new Point(32, 17));
    // Click en el enlace 'Registra proveedor de servicios'
    Mouse.Click(ulRegistraproveedordesHyperlink, new Point(132, 11));
    // Escribir 'ProveedorServicios' en el campo de texto 'textBox1'
    ulTextBox1Edit.Text = this.RecordedMethod1Params.UITextBox1EditText;
    // Escribir 'http://intranet2.uci.cu' en el campo de texto 'textBox2'
    ulTextBox2Edit.Text = this.RecordedMethod1Params.UITextBox2EditText;
    // Click en el botón 'Siguiente ->'
    Mouse.Click(ulSiguienteButton, new Point(38, 13));
    // Click en el botón 'button1' button
    Mouse.Click(ulButton1Button, new Point(14, 16));
    // Click en el botón 'Aceptar' button
    Mouse.Click(ulAceptarButton, new Point(26, 15));
    // Click en el botón 'Siguiente ->' button
    Mouse.Click(ulSiguienteButton, new Point(49, 10));
    // Esperar a que el botón 'Salir' este activo
    ulSalirButton.WaitForControlPropertyEqual("Enabled", true);
    // // Click en el botón 'Salir' button
    Mouse.Click(ulSalirButton, new Point(38, 17));
}
```

Figura 21: Prueba de interfaz de usuario de la aplicación de configuración.

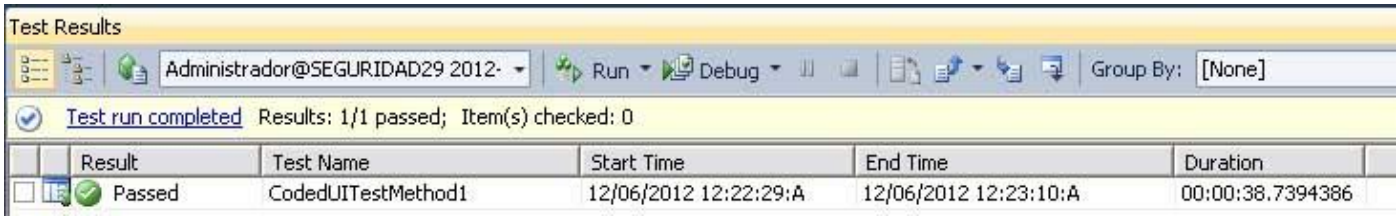

Figura 22: Resultados de prueba

3.3.2. Pruebas Unitarias

Las pruebas unitarias se realizan sobre una sección de código del sistema para probar su correcto funcionamiento de manera individual. Esto permite asegurar que todos los módulos del sistema funcionen correctamente por separado. Dichas pruebas se utilizan para asegurar el correcto funcionamiento de secciones de código, mucho más reducidas que el conjunto, y que tienen funciones concretas con cierto grado de independencia. El propósito principal de estas pruebas es encontrar discrepancias entre la especificación de la interfaz del módulo y su comportamiento real.

A continuación se presenta una descripción de una prueba unitaria realizada al sistema, las restantes estarán en el Anexo VII.

```
[TestMethod()]
[DeploymentItem("Gyes.Configuration.SPRegClnt.Core.dll")]
public void GetHtmlFromUrlTest()
{
    int param0 = 0;
    Crawler_Accessor target = new Crawler_Accessor(param0);
    string url = "http://localhost/caso_estudio/";
    string html = string.Empty;
    string htmlExpected = @"
    <!DOCTYPE html PUBLIC ""-//W3C//DTD XHTML 1.0 Transitional//EN"" ""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"">
    <html xmlns=""http://www.w3.org/1999/xhtml"" xml:lang=""en"" lang=""en"">
        <head>
            <meta http-equiv=""Content-Type"" content=""text/html; charset=utf-8"" />
            <title>Grupo docente</title>
            <link type=""text/css"" rel=""stylesheet"" href=""css/styles.css"" />
            <script type=""text/javascript"" src=""js/functions.js""></script>

        </head>

        <body>
            <ul>
                <li><a href=""adicionar_estudiante.php"">Adicionar estudiante</a></li>
                <li><a href=""listar_estudiantes.php"">Listar estudiantes</a></li>
            </ul>
        </body>
    </html>";
    bool expected = true;
    bool actual;
    actual = target.GetHtmlFromUrl(url, ref html);
    Assert.AreEqual(htmlExpected, html);
    Assert.AreEqual(expected, actual);
}
```

Figura 23: Prueba Unitaria al método "GetHtmlFromUrl".

Prueba de Unidad		
Nombre de la Prueba	GetHtmlFromUrl	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	20/05/2012
Ejecutado por	**Verificado por**	
Arniel Couso Linares	Arniel Couso Linares	

Descripción	Se introduce la dirección de la página web y se devuelve el código HTML.
Entrada	*string* url, ref *string* html
Criterio de aceptación	Retorna verdadero y el código HTML si lo encuentra.

Resultado

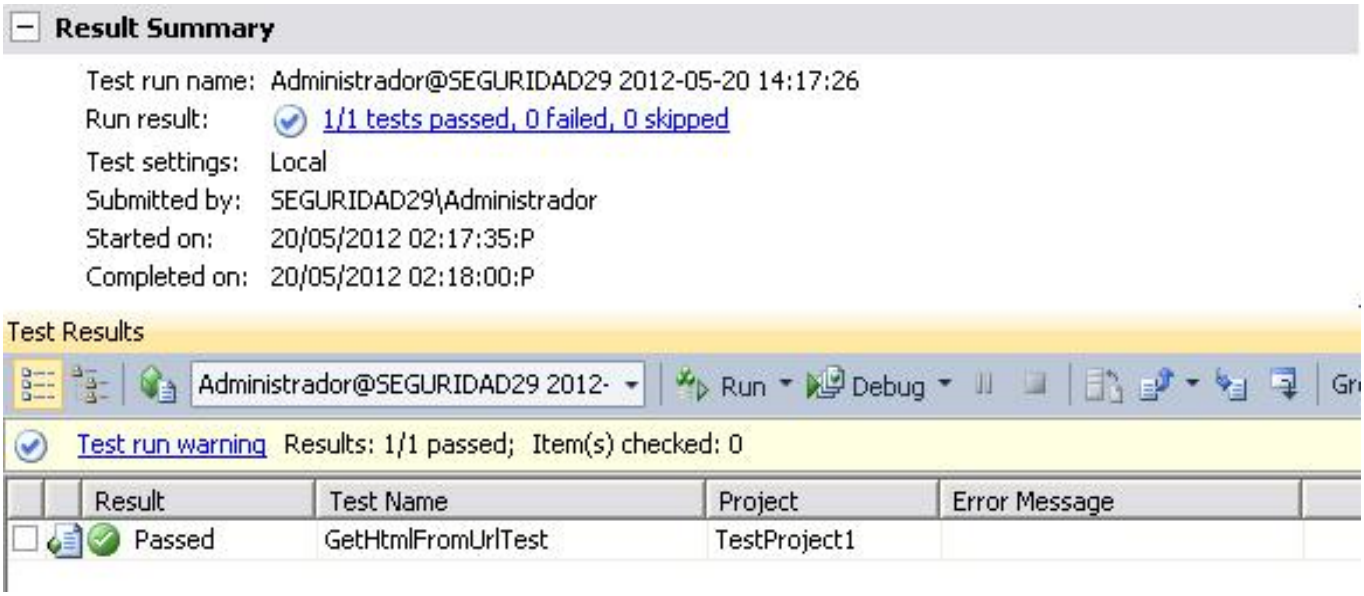

Figura 24: Resultado de prueba unitaria al método "GetHtmlFromUrl".

Tabla 15: Prueba unitaria al método "GetHtmlFromUrl".

Una vez desarrollado el sistema para el registro y configuración de proveedores de servicios se decidió realizar una comparación entre los tiempos empleados realizando el proceso tanto de manera manual como automática. Se realizó una simulación del proceso en condiciones específicas para un total de 60 recursos, de forma tal que se obtuvieran los tiempos empleados según la forma de realización.

El proceso consiste en adicionar los parámetros de configuración al proveedor de servicios, copiar las librerías de clase necesarias, realizar la búsqueda de los recursos y por último el registro de los mismos. El tiempo empleado una vez realizado el proceso manera manual fue de 5 minutos y 17 segundos; y el tiempo empleado al realizarlo de forma automática fue de solo 37 segundos. A partir de estos resultados

se puede expresar, que la utilización del sistema implementado da solución a la problemática existente reduciendo considerablemente el tiempo empleado en realizar el proceso.

3.4. Conclusiones

Con la elaboración de los diferentes diagramas modelados se logró una mejor comprensión del funcionamiento del sistema. Con la aplicación de las pruebas se pudo detectar las vulnerabilidades del código, así como los errores en el funcionamiento del sistema permitiendo validar la calidad de la aplicación para así cumplir con las expectativas del cliente.

Conclusiones

Con el desarrollo del presente trabajo se logró implementar un sistema que permite el registro y la configuración de los proveedores de servicios asociados al SAI. De esta manera es posible concluir que:

- El análisis de las herramientas similares al sistema que se quiere desarrollar, permitió conocer las principales funcionalidades y características para dar solución al problema existente en el SAI.
- La aplicación de la metodología, las tecnologías y herramientas hizo posible que se realizara un correcto diseño y una rápida implementación del sistema, cumpliendo así con las expectativas y necesidades identificadas inicialmente.
- La definición de los escenarios del sistema permitió conocer cuáles son las características que tiene el mismo y sentó las bases para dar inicio a la fase de desarrollo del sistema.
- La realización de las pruebas al sistema permitió detectar y corregir a tiempo las no conformidades generadas, contribuyendo así a elevar la calidad del mismo con el objetivo de cumplir con las expectativas del cliente.
- La utilización del sistema desarrollado permitió al SAI del CISED reducir considerablemente el tiempo empleado para realizar el proceso de registro y configuración.

Por todo lo anteriormente expuesto se concluye que el objetivo propuesto para el presente trabajo ha sido cumplido satisfactoriamente, ya que la aplicación da solución a la situación problemática que le dio origen.

Recomendaciones

Se recomienda desarrollar una nueva versión de la herramienta para hacer uso de la misma en Sistemas Operativos basados en plataformas libres.

Referencias bibliográficas

1. HODGES, Jeff, PHILPOTT, Neustar Rob, MALER, RSA Security Eve y MICROSYSTEMS, Sun (eds.). *Glossary for the OASIS SecurityAssertion Markup Language (SAML)V2.0* [En línea], 15 de Marzo del 2005. S.l.: s.n. Disponible en: http://docs.oasis-open.org/security/saml/v2.0/.
2. WINDLEY, Phillip J. *Digital Identity*. S.l.: O'Reilly, 2005. ISBN 0-596-00878-3.
3. *Defining Identity Management* [En línea]. S.l.: s.n. [Citado el: 13 de Enero del 2012]. Disponible en: http://hitachi-id.com/identity-manager/docs/identity-management-defined.html.
4. Definición de Configuración » Concepto en Definición ABC. [En línea]. [Citado el: 10 de Mayo del 2012]. Disponible en: http://www.definicionabc.com/tecnologia/configuracion.php.
5. Proyecto técnico - Sistema de Administración de Identidades.
6. Establecer confianza de un servicio de usuario de confianza de WCF a un STS mediante FedUtil. [En línea]. [Citado el: 11 de Abril del 2012]. Disponible en: http://msdn.microsoft.com/es-es/library/ee517264.aspx.
7. Introduction to Nutch, Part 1: Crawling | Java.net. [En línea]. [Citado el: 7 de Junio del 2012]. Disponible en: http://today.java.net/pub/a/today/2006/01/10/introduction-to-nutch-1.html.
8. Debian -- Details of package htdig in sid. [En línea]. [Citado el: 17 de Junio del 2012]. Disponible en: http://packages.debian.org/es/sid/htdig.
9. CALZADA RODRÍGUEZ, Yenisey y PÉREZ ORTIZ, Alexander. *Soporte Extensible para modos de autenticación del Sistema de Administración de Identidades Gyes* [En línea]. S.l.: Universidad de las Ciencias Informáticas (UCI), 2011. Disponible en: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_0499/1/TD_0499.pdf.
10. .NET Framework 4. [En línea]. [Citado el: 7 de Marzo del 2012]. Disponible en: http://msdn.microsoft.com/es-es/library/w0x726c2.aspx.
11. Windows Workflow Foundation. [En línea]. [Citado el: 7 de Marzo del 2012]. Disponible en: http://msdn.microsoft.com/es-es/netframework/aa663328.
12. El Lenguaje de Modelado Unificado (UML). [En línea]. [Citado el: 16 de Febrero del 2012]. Disponible en: http://www.docirs.cl/uml.htm.
13. UML Tool. [En línea]. [Citado el: 17 de Junio del 2012]. Disponible en: http://www.altova.com/umodel.html.
14. *UModeldatasheet.pdf* [En línea]. S.l.: s.n. [Citado el: 17 de Junio del 2012]. Disponible en: http://www.altova.com/documents/UModeldatasheet.pdf.
15. c# en castellano. Programación en Castellano. [En línea]. [Citado el: 13 de Enero del 2012]. Disponible en: http://www.programacion.com/csharp.
16. Programación del lado del cliente. [En línea]. [Citado el: 17 de Junio del 2012]. Disponible en:

http://prograweb.com.mx/pweb/0202ladoCliente.html.
17. *¿Qué es Javascript?* | *Maestros del Web* [En línea]. S.l.: s.n. [Citado el: 6 de Abril del d 2012]. Disponible en: http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.
18. CSS: Hojas de estilo. [En línea]. [Citado el: 17 de Junio del 2012]. Disponible en: http://es.kioskea.net/contents/css/cssintro.php3.
19. Barrera, Ana Maria: GLOSARIO DE PROGRAMACION. [En línea]. [Citado el: 17 de Junio del 2012]. Disponible en: http://ambarreraf.blogspot.com/2011/08/glosario-de-programacion.html.
20. CAVSI. [En línea]. Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.
21. Foros de debate, grupos de desarrolladores de Oracle. [En línea]. [Citado el: 17 de Junio del 2012]. Disponible en: http://www.clubdesarrolladores.com/foro/categoria/37-oracle.
22. *Capitulo III.pdf* [En línea]. S.l.: s.n. [Citado el: 17 de Junio del 2012]. Disponible en: http://dspace.ups.edu.ec/bitstream/123456789/1725/5/Capitulo%20III.pdf.
23. KAZMAN, Rick. Software Architecture. *Handbook of Software Engineering y Knowledge Engineering*. 2011,
24. REYNOSO, Carlos and KICILLOF, Nicolás. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft* [En línea]. 5 de Marzo del 2004. S.l.: s.n. [Citado el: 5 de Mayo del 2012]. Disponible en: http://carlosreynoso.com.ar/archivos/arquitectura/Estilos.PDF.
25. GARLAN, David y SHAW, Mary. *An introduction to software architecture*. S.l. CMU Software Engineering Institute Technical Report, 1994.
26. LARMAN, Craig. *UML y Patrones: Introducción al análisis y programación orientada a objetos.* S.l.: s.n., [sin fecha].
27. *MSDN Librery para Visual Studio 2005*. S.l.: Microsoft, [sin fecha].
28. Información general sobre el Diseñador de centros de datos lógicos. [En línea]. [Citado el: 7 de Mayo del 2012]. Disponible en: http://msdn.microsoft.com/es-es/library/ms181931(v=VS.80).aspx#WhatisLDD.
29. Probar la interfaz de usuario con pruebas de IU automatizadas. [En línea]. [Citado el: 17 de Junio del 2012]. Disponible en: http://msdn.microsoft.com/es-es/library/dd286726.aspx.
30. *White Box Testing* [En línea]. S.l.: s.n. [Citado el: 24 de Abril del 2012]. Disponible en: https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/white-box/259-BSI.html.

Glosario de Términos

Administración de identidades: es la combinación de procesos de negocio y la tecnología utilizada para administrar los datos en los sistemas informáticos y aplicaciones de los usuarios. Los datos gestionados incluyen los objetos de usuario, los atributos de identidad, los derechos de seguridad y los factores de autenticación.

Application Programming Interface (API): es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción. Son usados generalmente en las bibliotecas.

ASP.NET: es un *framework* para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML.

Base de Datos (BD): conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente. En una BD, la información se organiza en campos y registros.

Clases: conjunto de objetos que comparten atributos, operaciones, relaciones y semántica, las mismas representan los conceptos fundamentales del sistema.

Configuración: en informática la configuración es un conjunto de datos que determina el valor de algunas variables de un programa o de un Sistema Operativo.

Diagrama de Clases: representación de los conceptos de importancia en el área de la aplicación, así como de las relaciones entre estos.

Entidad: elemento activo de una computadora o sistema de red. Por ejemplo, un proceso automatizado o grupo de ellos, un subsistema, una persona o grupo de personas que incorporan un conjunto distinto de funcionalidades

***Framework*:** un *framework*, en el desarrollo de *software* es una estructura de soporte definida, en la cual otro proyecto de *software* puede ser organizado y desarrollado. Típicamente, puede incluir soporte de

programas, bibliotecas y un lenguaje interpretado entre otros *software* para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

HTML: Lenguaje de Marcado de Hipertexto.

MSF (Microsoft Solution *Framework*): f*ramework* que brinda Microsoft para guiar el desarrollo de sistemas. Es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos.

Principal: es una entidad cuya identidad puede ser autenticada.

Proveedor de servicios: son las entidades que proveen servicios a principales u otras entidades (1).

Proveedor de Identidades: es una entidad que crea, mantiene y administra información de identidades para principales y provee autenticación a otros proveedores de servicios.

Escenario (SC): abreviatura definida para identificar los escenarios del sistema.

UML: lenguaje de modelado de sistemas de *software* más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de un desarrollo de *software*.

Anexos

Anexo I: Descripción de parámetros de configuración

Nombre	ServiceEndpoint
Descripción	Configura los puntos de entradas Http usados por el proveedor de servicios para comunicarse con los compañeros de la federación. Cada punto de entrada debe corresponder con un manejador en el servicio.
Atributos	
Type	Determina la función de este punto de entrada. Debe ser uno de los siguientes: • sign on • logout • metadata
localpath	La dirección donde el punto de entrada esta.
redirectUrl	La URL a donde el usuario será re-direccionado después que el control termine de ejecutar la acción.
Nombre	NameIdFormats
Descripción	Permite especificar cual *NameIdFormats* es soportado por el proveedor de servicios.
Atributos	

all	verdadero/falso Verdadero cuando todos SAML *NameIdFormats* son soportados.
Nombre	CommonDomain
Descripción	Contiene las opciones de configuración para el lector de *cookie*.
Atributos	
Enabled	Verdadero si el lector de *cookie* está activo sino falso
localReaderEndpoint	La URL del lector local de *cookie*.
Nombre	RequestedAttributes
Descripción	Lista de atributos que el proveedor de servicios quiere confirmar.
Atributos	
Name	Identificador del atributo.
isRequired	Atributo opcional que especifica que el atributo debe encontrarse en el metadato del proveedor de servicios.
Nombre	IDPEndPoints
Descripción	Determina como el proveedor de servicios debe conectarse con los socios de la federación.

Atributos	
metadata	La dirección de la carpeta donde el metadato puede encontrarse
idpSelectionUrl	URL de la página web por defecto.

Tabla 16: Parámetros de configuración II.

Anexo II: Descripción de los escenarios

<table>
<tr><td colspan="2">Nombre del escenario: Autenticar usuario</td><td>Identificador: SC 6</td></tr>
<tr><td colspan="3">Objetivo del escenario: Verificar que el Administrador tenga acceso al sistema.</td></tr>
<tr><td colspan="3">Persona: Administrador.</td></tr>
<tr><td>Iteración: 2da</td><td>Prioridad: Media</td><td>Complejidad: 1</td></tr>
<tr><td colspan="3">Precondiciones:</td></tr>
<tr><td colspan="3">El escenario inicia cuando el Administrador introduce su usuario y contraseña, el sistema verifica la autenticidad de los datos, si los datos son correctos el Administrador accede al sistema.</td></tr>
<tr><td colspan="3">Validaciones:
Verificar que no existan campos vacíos.</td></tr>
<tr><td colspan="3">Prototipo de interfaz de usuario
 
</td></tr>
</table>

Figura 25: Prototipo de interfaz gráfica "Autenticar usuario"	
Requisitos de calidad del servicio	

Tabla 17: Descripción del escenario "Autenticar usuario".

Nombre del escenario: Gestionar recursos		**Identificador:** SC 3
Objetivo del escenario: Gestionar los recursos del proveedor de servicios.		
Persona: Administrador		
Iteración: 1ra	**Prioridad:** Alta	**Complejidad:** 1
Precondiciones:		
Este escenario comienza cuando el administrador presiona el botón “Siguiente” en la interfaz gráfica que captura los datos correspondientes al proveedor de servicio. Como respuesta el sistema muestra la ventana “Gestionar recursos” donde le brinda las opciones al usuario de “Buscar”, “Adicionar” y “Eliminar” recurso (Ver Figura 26). Nota: Las opciones de “Buscar”, “Adicionar” y “Eliminar” han sido omitidas debido a que estas se describen como tareas del escenario en cuestión.		
Validaciones: • En este escenario no es necesario realizar validaciones.		
Prototipo de interfaz de usuario		

Figura 26: Prototipo de interfaz gráfica "Gestionar recursos".

Requisitos de calidad del servicio	

Tabla 18: Descripción del escenario "Gestionar recurso".

Nombre del escenario: Administrar proveedor de servicios		**Identificador:** SC 5
Objetivo del escenario: Administrar los proveedores de servicios que se encuentran registrados.		
Persona: Administrador.		
Iteración: 2da	**Prioridad:** Media	**Complejidad:** 1
Precondiciones:		
El usuario accede a la aplicación. El sistema muestra un listado de todos los proveedores de servicios que han sido registrados y brinda la posibilidad de seleccionar las opciones de “Ver recursos” y “Eliminar”. Nota: Las opciones de “Ver recursos” y “Eliminar” han sido omitidas debido a que estas se describen como tareas del escenario en cuestión.		
Validaciones:		

• En este escenario no es necesario realizar validaciones.
Prototipo de interfaz de usuario
Requisitos de calidad del servicio

Tabla 19: Descripción del escenario "Administrar proveedor de servicios".

Nombre del escenario: Registrar recursos		**Identificador:** SC 4
Objetivo del escenario: Registrar los recursos pertenecientes al proveedor de servicios.		
Persona: Administrador.		
Iteración: 2da	**Prioridad:** Media	**Complejidad:** 1
Precondiciones:		
Este escenario comienza cuando el administrador presiona el botón “Siguiente” después de haber gestionado los recursos pertenecientes al proveedor de servicios. Como respuesta el sistema registra los recursos al proveedor de servicios y muestra un mensaje en una ventana informando el éxito del registro.		
Validaciones: • Verificar que se haya realizado la gestión de los recursos.		
Prototipo de interfaz de usuario		
Requisitos de calidad del servicio		

Tabla 20: Descripción del escenario “Registrar recursos”.

Anexo III: Descripción de tareas por escenarios

Nombre del escenario: Buscar recursos		**Identificador:** SC 3.3
Objetivo del escenario: Buscar los recursos pertenecientes al proveedor de servicios.		
Persona: Administrador.		
Iteración: 2da	**Prioridad:** Media	**Complejidad:** 1
Precondiciones:		
Esta tarea comienza cuando el administrador presiona el botón “Buscar recursos”. El sistema comienza la búsqueda y a medida que encuentra los recursos los adiciona de forma automática al árbol de recursos.		
Validaciones: • En este escenario no es necesario realizar validaciones.		
Prototipo de interfaz de usuario (Ver Figura 26)		
Requisitos de calidad del servicio		

Tabla 21: Descripción del escenario “Buscar recursos”.

Nombre del escenario: Eliminar recurso	**Identificador:** SC 3.2
Objetivo del escenario: Eliminar un recurso del árbol de recursos.	

Persona: Administrador.		
Iteración: 2da	**Prioridad:** Media	**Complejidad:** 1
Precondiciones:		
Esta tarea comienza cuando el administrador selecciona un recurso del árbol de recursos y presiona el botón “Eliminar recurso”. El sistema muestra una ventana de confirmación para eliminar el recurso. El administrador presiona el botón “Aceptar” y el sistema elimina el recurso del árbol de recursos.		
Validaciones: • Verificar que el administrador haya seleccionado un recurso del árbol de recursos.		
Prototipo de interfaz de usuario (Ver Figura 26)		
Requisitos de calidad del servicio		

Tabla 22: Descripción del escenario “Eliminar recurso”.

Nombre del escenario: Eliminar		**Identificador:** SC 3
Objetivo del escenario: Elimina un proveedor de servicios recurso seleccionado.		
Persona: Administrador.		
Iteración: 2da	**Prioridad:** Media	**Complejidad:** 1
Precondiciones:		

El administrador accede a la aplicación con la intención de eliminar un proveedor de servicios. El sistema le muestra un listado de todos los proveedores de servicios existentes y brinda la posibilidad de seleccionar uno para ser eliminado. El administrador realiza la selección y oprime la opción Eliminar. El sistema muestra una ventana de confirmación preguntando si desea eliminar el proveedor de servicios seleccionado. El administrador confirma la operación presionando el botón “Aceptar”. Finalmente, el sistema elimina el proveedor de servicios.	
Validaciones: • En este escenario no es necesario realizar validaciones.	
Prototipo de interfaz de usuario	
Requisitos de calidad del servicio	

Tabla 23: Descripción del escenario “Eliminar recursos”.

Anexo IIIV: Descripción de clases

Nombre	*RegistrationControler*.	
Tipo de Clase	Controladora	
Descripción	Representa la clase que controla el proceso de registro de proveedores de servicios.	
Atributo	**Tipo**	**Descripción**
_serviceProvider	ServiceProvider	Proveedor de servicios
_crawler	Crawler	Clase que contiene los métodos para realizar las búsquedas.
Método	**Descripción**	
RegistrationControler()	Constructor de la clase.	
RegistrarProveedor():bool	Permite registrar los datos de un proveedor de servicios.	
FindResourcesFromPath(int depth, string path):void	Busca los recursos físicos.	
FindResourcesFromUrl(int depth,	Busca los recursos lógicos.	

string url):void	
StopSearch():void	Detiene el proceso de búsqueda.
AddResource(TreeNode father, string address):void	Adiciona un recurso al árbol de recursos.
RemoveResource(TreeNode node):void	Elimina un recurso del árbol de recursos.
GetListFromTree(TreeNode node):List<string>	Retorna una lista de cadenas URL correctas obtenidas a partir de un árbol de nodos.
StopSearch():void	Método que detiene la búsqueda de recursos.
NewPageLoaded(WebPage page):void	Método privado que se ejecuta cuando se ha comenzado una búsqueda y se ha encontrado una página web. Se encarga de adicionar los recursos encontrados al árbol de recursos.
NewFolderLoaded(Folder folder):void	Método privado que se ejecuta cuando se ha comenzado una búsqueda y se ha encontrado una carpeta. Se encarga de adicionar los recursos encontrados al árbol de recursos.
CrawlerCrawFinished():void	Método privado que se ejecuta cuando se termina la búsqueda.
CrawlerErrorSearching(string url, string message):void	Método privado que se ejecuta cuando ocurre un error durante la búsqueda. Se encarga de buscar el nodo con el error en el árbol de nodos para adicionarle un mensaje de error.

Tabla 24: Descripción de la clase controladora "RegistrationControler".

Nombre	*ConfigurationControler*.	
Tipo de Clase	Controladora.	
Descripción	Representa la clase que controla el proceso de configuración del proveedor de servicios.	
Atributo	**Tipo**	**Descripción**
_gac	bool	Representa si se desea guardar las librerías al GAC.
_location	string	Representa la dirección a donde se desea guardar las librerías de clases.
_configManager	ConfigManager	Administrador de configuración.
Método	**Descripción**	
ConfigurationControler()	Constructor de la clase.	
CopyFiles():bool	Método que se encarga de copiar las librerías de clases al directorio seleccionado.	

Tabla 25: Descripción de la clase controladora "ConfigurationControler".

Nombre	*ServiceProvider*.	
Tipo de Clase	Entidad	
Descripción	Contiene todo los datos del proveedor de servicio.	
Atributo	**Tipo**	**Descripción**
_ip	string	Dirección IP del proveedor de servicio.
_name	string	Nombre del proveedor de servicio.
_resource	TreeNode	Árbol de recursos del proveedor de servicios.
_url	string	Url del proveedor de servicios.
Método	**Descripción**	
ServiceProvider ()	Constructor de la clase.	

Tabla 26: Descripción de la clase "Service Provider".

Nombre	*Folder*.

Tipo de Clase	Entidad	
Descripción	Contiene todo los datos de una carpeta.	
Atributo	**Tipo**	**Descripción**
_files	string[]	Arreglo de direcciones de los archivos de la carpeta.
_path	string	Dirección de ubicación de la carpeta.
_subFolder	string[]	Arreglo con las direcciones de las sub-carpetas contenidas en la carpeta.
Método	**Descripción**	
Folder()	Constructor de la clase.	

Tabla 27: Descripción de la clase "Folder".

Nombre	*WebPage*
Tipo de Clase	Entidad
Descripción	Contiene todo los datos de una página web.

Atributo	Tipo	Descripción
_html	string	Código HTML de la página web.
_links	string[]	Arreglo con los *links* que contiene esta página web.
_url	string	Dirección URL de la página web.
Método	**Descripción**	
WebPage()	Constructor de la clase.	

Tabla 28: Descripción de la clase "WebPage".

Nombre	*HtmlTag*	
Tipo de Clase	Entidad	
Descripción	Contiene todo los datos de una etiqueta de código HTML.	
Atributo	**Tipo**	**Descripción**
_attributes	Dictionary<string, string>	Colección de nombres y valores de atributo para este etiqueta HTML.

_name	string	Nombre de la etiqueta HTML.
_trailingSlash	bool	True si esta etiqueta contiene una barra inclinada al final.
Método	**Descripción**	
HtmlTag()	Constructor de la clase.	

Tabla 29: Descripción de la clase "HtmlTag".

Anexo IV: Descripciones de interfaces

Interfaz “Proveedor de servicios”: permite capturar los datos de un proveedor de servicio para ser registrados.

Figura 27: Interfaz gráfica "Proveedor de servicios".

Interfaz “Seleccionar archivo de configuración” del escenario “Configurar proveedor de servicios”: brinda al administrador una breve descripción y permite seleccionar la ubicación del archivo de configuración del proveedor de servicios.

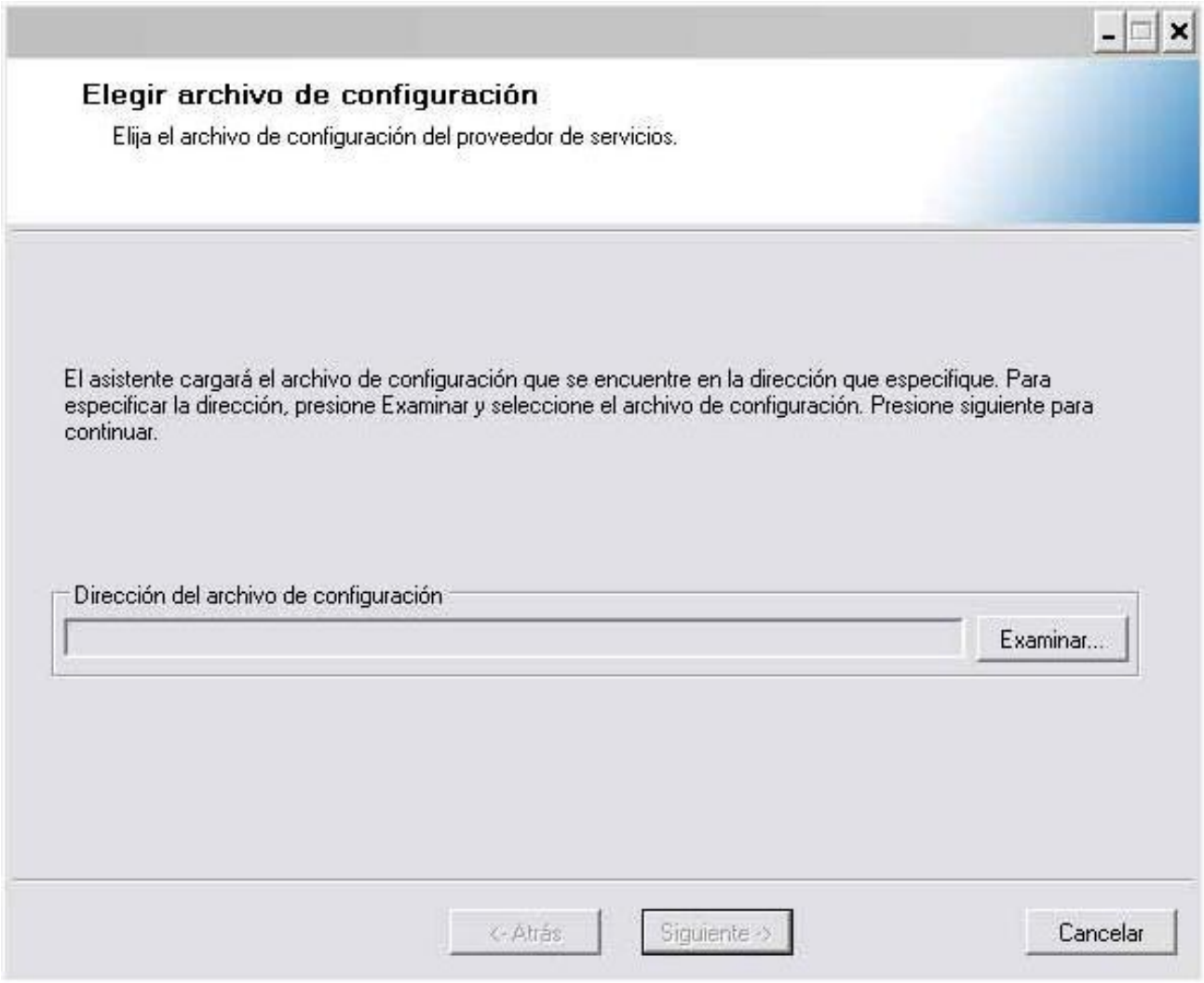

Figura 28: Interfaz gráfica "Elegir archivo de configuración".

Interfaz “Valores de configuración” del escenario “Configurar proveedor de servicios”: brinda al administrador un conjunto de campos necesarios para la correcta configuración del proveedor. Permite al usuario modificar los campos.

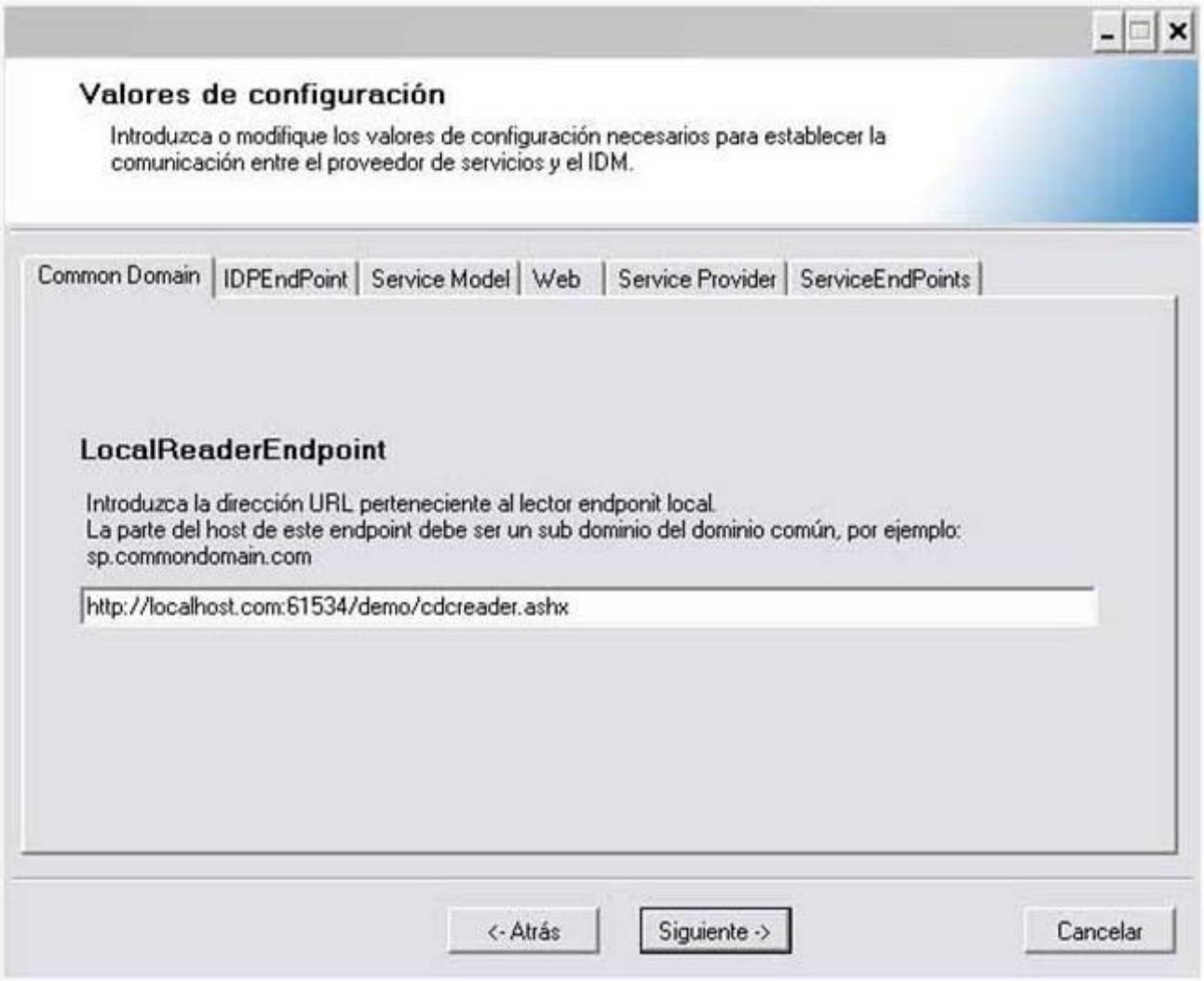

Figura 29: Interfaz gráfica "Valores de configuración".

Interfaz “Elegir directorio para las librerías de clases” del escenario “Configurar proveedor de servicios”: permite al administrador seleccionar la carpeta destino para las librerías de clases.

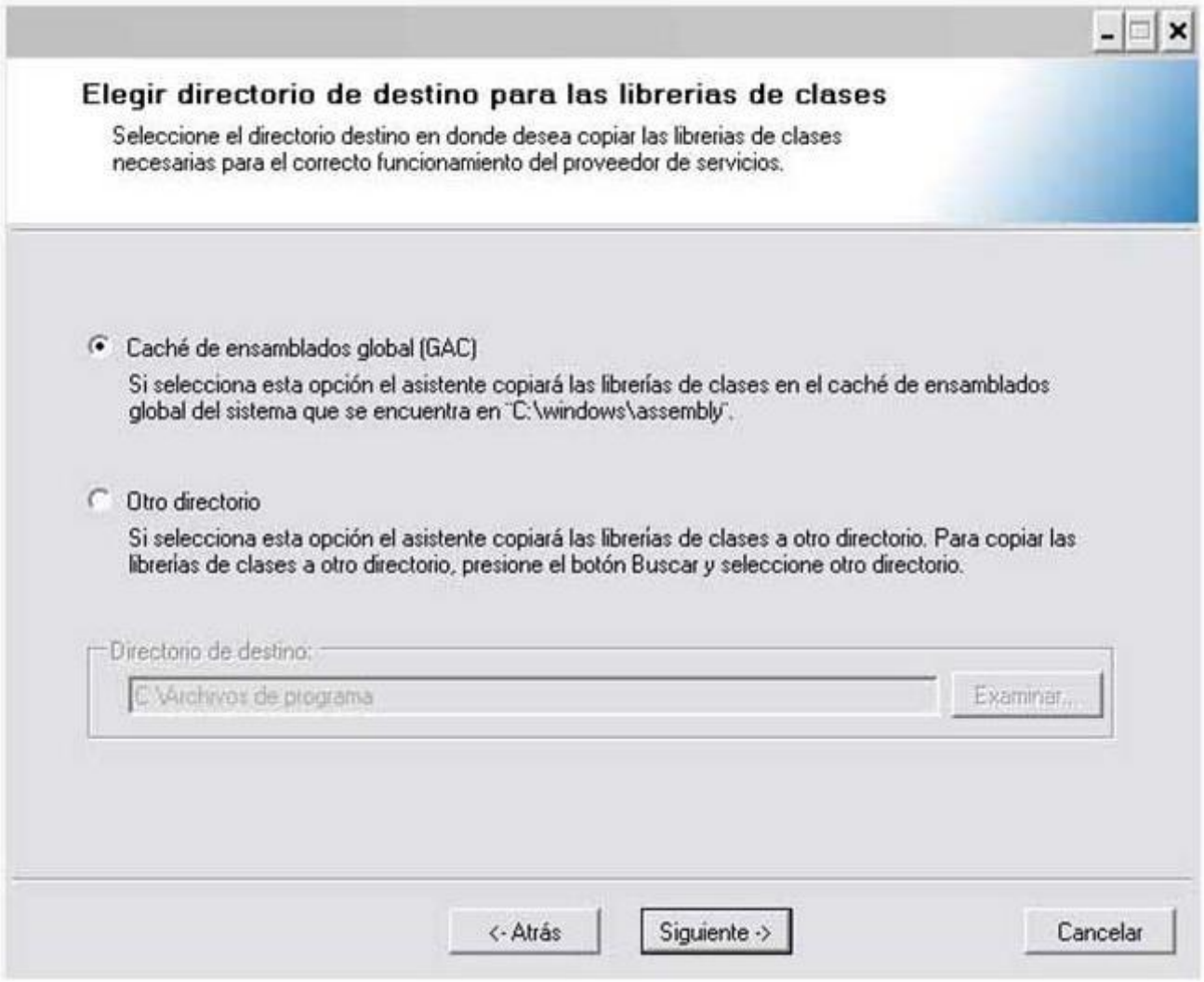

Figura 30: Interfaz gráfica “Seleccionar directorio para librerías de clases”.

Anexo VI: Casos de pruebas

Escenario “Configurar proveedor de servicios”.

Sección del Escenario	Entrada	Resultado	Condiciones
Configurar proveedor de servicios correctamente.	El administrador introduce los parámetros de configuración correctamente.	El sistema muestra una ventana de notificación donde informa que se ha realizado la configuración correctamente.	
Configurar proveedor de servicios seleccionando un archivo de configuración erróneo.	El administrador selecciona un archivo de configuración erróneo.	El sistema muestra una ventana de notificación donde le informa al administrador que ocurrió un error cuando se cargaba el archivo.	El administrador selecciona un archivo de configuración erróneo.

Tabla 30: Descripción del caso de pruebas del escenario “Configurar proveedor de servicios”.

Escenario “Autenticar usuario”.

Sección del Escenario	Entrada	Resultado	Condiciones
Autenticarse de	El administrador introduce	El sistema muestra una	

forma correcta.	los parámetros de para autenticarse en el sistema correctamente.	página con el listado de proveedores de servicios registrados.	
Autenticarse de introduciendo datos incorrectos.	El administrador introduce los datos de autenticación en el sistema incorrectamente.	El sistema muestra una ventana de notificación donde le informa al administrador que la cuenta no existe.	El administrador introduce los datos incorrectos para la autenticación en el sistema.

Tabla 31: Descripción del caso de prueba del escenario "Autenticar usuario".

Anexo VII: Pruebas unitarias

```
[DeploymentItem("Gyes.Configuration.SPRegClnt.Core.dll")]
public void GetLinksFromHtmlTest()
{
    int param0 = 0; // TODO: Initialize to an appropriate value
    Crawler_Accessor target = new Crawler_Accessor(param0); // TODO: Initialize to an appropriate value
    string html = @"
    <!DOCTYPE html PUBLIC ""-//W3C//DTD XHTML 1.0 Transitional//EN"" ""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtc
    <html xmlns=""http://www.w3.org/1999/xhtml"" xml:lang=""en"" lang=""en"">
        <head>
            <meta http-equiv=""Content-Type"" content=""text/html; charset=utf-8"" />
            <title>Grupo docente</title>
            <link type=""text/css"" rel=""stylesheet"" href=""css/styles.css"" />
            <script type=""text/javascript"" src=""js/functions.js""></script>

        </head>

        <body>
            <ul>
                <li><a href=""adicionar_estudiante.php"">Adicionar estudiante</a></li>
                <li><a href=""listar_estudiantes.php"">Listar estudiantes</a></li>
            </ul>
        </body>
    </html>";
    List<string> expected = new List<string>();
    expected.Add("adicionar_estudiante.php");
    expected.Add("listar_estudiantes.php");

    List<string> actual;
    actual = target.GetLinksFromHtml(html);
    Assert.AreEqual(expected[0], actual[0]);
    Assert.AreEqual(expected[1], actual[1]);
}
```

Figura 31: Prueba Unitaria al método "GetLinksFromHtml".

Prueba de Unidad		
Nombre de la Prueba	GetLinksFromHtml	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	20/05/2012
Ejecutado por	**Verificado por**	
Arniel Couso Linares	Arniel Couso Linares	

Descripción	Para poder realizar la prueba se deben pasar una cadena que constituye el código HTML de una página web. Este código tiene que contener etiquetas <href>.
Entrada	*string* html
Criterio de aceptación	Retorna una lista de los *links* contenidos dentro del código HTML pasado.

Resultado

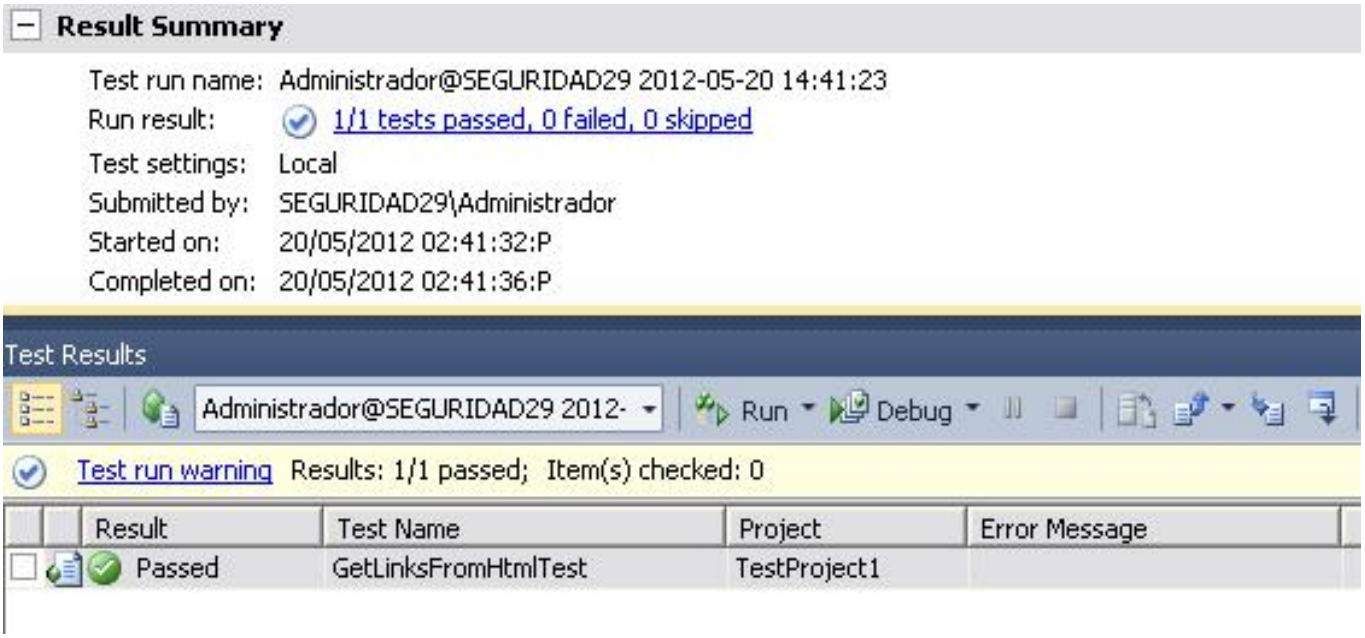

Figura 32: Resultado de prueba unitaria "GetLinksFromHtml"

Tabla 32: Prueba unitaria al método "GetLinksFromHtml".