Universidad de las Ciencias Informáticas

Facultad 1

Título:
"Soporte Extensible para modos de autenticación del Sistema de Administración de Identidades Gyes."

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

Autores:

Yenisey Calzada Rodríguez. Alexander Pérez Ortiz.

Tutor: Ing. Ernesto Miguel Muñoz.

Ciudad de La Habana, Cuba

Junio 2011

"Un descubrimiento resuelve un gran problema, pero en la solución de cualquier problema hay una pizca de descubrimiento. Tu problema puede ser modesto, pero si es un reto para tu curiosidad y lo resuelves con tus propios medios, experimentarás la tensión y gozarás del triunfo del descubrimiento."

George Polya.

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos la presente a los __ días del mes de _____del 2011.

Firma del Autor: _____
Yenisey Calzada Rodríguez.

Firma del Autor: _____
Alexander Pérez Ortiz.

Firma del Tutor: _____
Ernesto Miguel Muñoz

Ing. Ernesto Miguel Muñoz: Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas, con 3 años de experiencia laboral, pertenece al Departamento de Programación de la Facultad, donde se desempeña como profesor de las Asignaturas Introducción a la Programación y Programación I, está vinculado al Departamento de Seguridad Digital del Centro de Identificación y Seguridad Digital. Dirección de correo electrónico: **emmunoz@uci.cu.**

Se lo dedico a:

Mis viejitos lindos, a mi gran querer, a mi familia en general, a mis amigos y a los que ya no están…

Yeni.

Se lo dedico a:

A mis padres que lo han sido todo para mí en la vida, y a toda mi familia…

Ale.

Son muchas las personas especiales a los que quiero agradecer por su apoyado, su comprensión y su amor en todos los momentos en que lo necesite:

- ✓ *A mis viejitos lindos que son lo más grande que tengo, por pensar que todavía soy una niña y complacer todos mis caprichos, por el sacrificio tan grande que han tenido que hacer para que yo salga adelante.*
- ✓ *A mi novio Ale por ayudarme tanto en el desarrollo de este trabajo, sin el no hubiera podido vencer esta batalla, por estar ahí conmigo hasta altas horas de la noche, por darme todo el amor del mundo y muchas fuerzas para seguir y por soportar mis peleas, mis malos genios, mis defensas bajas, en general por estar junto a mi cuando más lo he necesitado en estos 5 años, agradecer también a su familia, a Adrian, a Guille y a Mayci por acogerme como si fuera una hija porque desde que me conocieron fueron muy cariñosos conmigo.*
- ✓ *A mis hermanos Ori y Guito por apoyarme siempre, a mi sobrina querida Yade y a mi hermanito, postizo, Viti por ocupar mi lugar cuando no estuve.*
- ✓ *A mi familia en general, principalmente a Isaida y a Eida por ayudarme con mis papas cuando lo necesite, a mi tío Niño, a mis tías Beni, Lile y Felicia y a mi abuelito Mario que es el único que me queda, por quererme tanto.*
- ✓ *A mis amigas que siempre están para lo que necesite, Ana, Danay, Ediagnis, Diana Rosa, Yari, Dianiselis y Yumi (…), en fin, menciono algunas porque son muchas, a todas muchas gracias por ganarse mi amistad.*
- ✓ *A mis chicos, Antonio, Yunieski, Yerandy, Ángel, Frank, Addiel y Quevedo que espero no se olviden nunca de mí por la cantidad de veces que los molesté.*
- ✓ *A todas mis amistades muchas gracias y los/las quiero mucho.*

Yenisey Calzada Rodríguez.

- ✓ *A mis padres por guiarme en la vida, apoyarme, y ser ejemplo de consagración y sacrificio. A ellos por ser la razón de ser de mi vida y por los cuales siempre he luchado y lucharé siempre.*
- ✓ *A mi novia, que siempre me ha brindado su apoyo incondicional en todos los momentos, por estar ahí siempre.*
- ✓ *A mi familia, por lo especial que es. A todos los amigos que en el transcurso de los 5 años me facilitaron llevar a vías de hechos esta difícil carrera.*
- ✓ *A mi suegros por acogerme en el seno de su familia como si fuera un hijo más, incluyendo en esta lista a mi Tía Querida por darme tanto cariño al tío Manolo a mi cuñado y concuña y también porque no a Dany ese pequeño niño que con solo 4 años es capaz de despertar en cualquier persona sentimientos tan puros como los que solo un niño puede tener.*
- ✓ *A mis amigos especiales: Yarismay ¨ mi hermanita ¨, a Jessi y a Arle amigos que un día entramos a esta gran escuela y que nos unió como si fuéramos todos hermanos, imborrable son mi recuerdos y que nunca perderé aunque ahora el destino nos distancie.*

Alexander Pérez Ortiz.

A nuestro tutor por hacer el mayor esfuerzo para ayudarnos. A todos los profesores que han contribuido en nuestra formación como profesionales .Al colectivo del proyecto, Ruth, Maikel, Roberto, Yayneris, Alejandro y Yaimí, ya que, todos pusieron su granito de arena en el desarrollo de este trabajo. A todos, Gracias.

Autores.

En el Departamento de Seguridad Digital del Centro de Identificación y Seguridad Digital (CISED) y la Universidad de las Ciencias Informáticas (UCI) se está desarrollando el Sistema de Administración de Identidades Gyes con el objetivo de centralizar la seguridad digital común de los proyectos creados en dichas entidades y en colaboración con el Ministerio del Interior (MININT).

Por la necesidad de facilitar y mejorar el proceso de autenticación de usuarios en el Sistema de Administración de Identidades Gyes mediante la creación, modificación y eliminación de contratos de servicios existentes entre el Servidor de Autenticación o IdP y cualquier parte que desee establecer una relación de confianza con él, surge el presente trabajo de diploma titulado *"Soporte Extensible para modos de autenticación del Sistema de Administración de Identidades Gyes"*. En el trabajo se abordan las tendencias actuales relacionadas con los sistemas de administración de identidades y las ventajas que otorga vincular diferentes parámetros de autenticación para la seguridad de los sistemas informáticos. La investigación se enmarca fundamentalmente en el proceso de autenticación de usuarios en el Sistema de Administración de Identidades Gyes, indagando en los sistemas similares existentes tanto a nivel internacional como nacional, y las principales herramientas, tecnologías y metodología a utilizar, se describe el proceso de análisis y diseño de la propuesta definiéndose la arquitectura y los patrones de diseño, se realiza la validación del producto final para garantizar la calidad requerida.

PALABRAS CLAVE

Autenticación, contrato de servicios, Sistema de Administración de Identidades Gyes, Servidor de Autenticación o Idp.

Índice de Contenido

Índice de Tablas

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

Actualmente el éxito de las empresas está relacionado con el uso oportuno de la información y para lograrlo éstas se apoyan en las tecnologías de la información y las telecomunicaciones al automatizar procesos claves de la organización. Por tanto es común que exista una compleja infraestructura donde converjan un gran número de sistemas y servicios, que son usados diariamente por un gran número de personas para cumplir los objetivos de la empresa.

Uno de los requerimientos primordiales de los sistemas informáticos son los mecanismos de seguridad adecuados a la información que se intenta proteger; el conjunto de tales mecanismos ha de incluir al menos un sistema que permita identificar a las entidades que intentan interactuar con los objetos, para que solo puedan acceder a aquellos recursos o servicios a los que estén autorizados. (1)

En esta gran infraestructura tecnológica, la información de las identidades asociadas a los usuarios y sus relaciones con los sistemas de la empresa, frecuentemente es gestionada a través de sistemas de administración de identidades, que no son más que soluciones de software que centralizan y administran los procesos de: autenticación, proceso mediante el cual una entidad puede verificar que otra es, o no, quien dice ser; autorización, proceso mediante el cual una entidad puede verificar que otra cumple, o no, con las políticas de acceso para realizar cierta acción y aprovisionamiento, abarca todo el ciclo de vida de las cuentas de usuarios en la organización.

Los servicios de gestión de identidades y accesos permiten analizar, planificar, diseñar, implementar, desplegar y mantener un sistema integrado de gestión de identidades. Estandariza la gestión del acceso a todas las plataformas de usuarios, dispositivos, aplicaciones y procesos empresariales, así como a los puntos de seguridad físicos como la tecnología biométrica, los dispositivos de tarjeta inteligente y los identificadores de lectores.

A partir de la creación de varios proyectos del Centro de Identificación y Seguridad Digital (CISED) y la Universidad de las Ciencias Informáticas (UCI) en colaboración con el Ministerio del Interior (MININT), se decidió la implementación del Sistema de Administración de Identidades Gyes como solución de seguridad común a dichos proyectos, el cual está compuesto por tres subsistemas, uno por cada proceso clave de la administración de identidades, el Subsistema de Autenticación, el Subsistema de Autorización y el Subsistema de Aprovisionamiento.

El Subsistema de Autenticación por su parte, es el encargado de los procesos de autenticación de usuarios, el mismo está basado en la autenticación por un solo factor (usuario y contraseña) pero no permite la autenticación mediante otros factores como son: número de activación de la computadora, certificado digital, tarjetas inteligentes o una característica física o acto involuntario del usuario que lo identifica (voz, escritura, huellas dactilares, patrones oculares) y no brinda la posibilidad de modificar la forma en la cual un usuario se autentica en una aplicación determinada sin que se tengan que realizar cambios internos en el software; una de las tendencias actuales en la rama, está encaminada a implementar sistemas que mejoren la seguridad de la información que protegen, combinando diferentes factores, en relación a ello el sistema no permite que un mismo usuario posea diferentes modos de autenticación y no es capaz de manejar una identidad distribuida en recursos o servicios diferentes, lo cual limita todo el potencial del proceso.

Por la necesidad de poder autenticarse de manera flexible incorporando nuevos modos de autenticación dependiendo de condiciones preestablecidas, surge como **problema científico**: ¿Cómo gestionar los diferentes modos de autenticación que pueden usarse en el Sistema de Administración de Identidades Gyes? y como **objeto de estudio**: Los modos de autenticación de los sistemas de administración de identidades.

Quedando enmarcado como **campo de acción**: La gestión de los parámetros que definen a los diferentes modos de autenticación en el Sistema de Administración de Identidades Gyes.

Objetivo general: Desarrollar un módulo extensible mediante esquemas, que soporte diferentes modos de autenticación en el Sistema de Administración de Identidades Gyes.

Idea a defender: Con la implantación del soporte extensible mediante esquemas, se contribuye a aumentar la flexibilidad del proceso de autenticación en las aplicaciones empresariales que utilizan el Sistema de Administración de Identidades Gyes.

Objetivos específicos:

- ✓ Gestionar los contratos de servicios para definir los esquemas de autenticación a utilizar.
- ✓ Rediseñar la estructura para que permita diferenciar los esquemas de autenticación y a quienes pertenecen.
- ✓ Reimplementar el Subsistema de Autenticación, incluyendo la gestión de los mecanismos de autenticación.
- ✓ Validar la solución propuesta.

Para dar cumplimiento a los objetivos planteados se trazaron las siguientes **tareas**:

- ✓ Realización del marco teórico – metodológico de la investigación.
- ✓ Realización de un estudio de las soluciones existentes sobre los modos de autenticación y mecanismos de almacenamiento para la solución.
- ✓ Investigación de posibles tecnologías y herramientas para la solución del problema.
- ✓ Identificación de las tecnologías y herramientas que se utilizarán para la realización del problema planteado.
- ✓ Estudio de la arquitectura y las funcionalidades existentes en el Subsistema de Autenticación.
- ✓ Documentación del análisis y diseño realizado de la estructura para diferenciar los esquemas de autenticación.
- ✓ Documentación del análisis y diseño sobre la refactorización en el Subsistema de Autenticación.
- ✓ Implementación del Soporte Extensible para modos de autenticación del Sistema de Administración de Identidades Gyes.
- ✓ Implementación de la aplicación para la gestión de los contratos de servicios.
- ✓ Implementación de la refactorización en el Subsistema de Autenticación.
- ✓ Validación del correcto funcionamiento de la solución.

Los ***Métodos Científicos*** utilizados fueron:

Teóricos:

- ✓ Analítico-Sintético: Se analiza y resume la documentación recopilada, a través de los diferentes medios bibliográficos, para de esa forma desarrollar un mejor diseño e implementación del sistema que se propone.
- ✓ *Modelación:* Se modela el problema planteado, definiendo una arquitectura, que cumple con el objetivo de dar una mejor solución a la situación presente.
- ✓ *Análisis sistémico:* se plantea el problema y su solución como un todo, realizando un estudio de cada uno de los componentes de la evaluación de proyectos de software, estableciendo dependencias y conexiones entre cada una de las fases para poder lograr un resultado integral e instaurar así un modelo sostenible.
- ✓ *Análisis histórico – lógico*: para analizar la trayectoria y evolución de la metodología de desarrollo de software y demás herramientas que se utilizan durante el trabajo.

Empíricos:

- ✓ *Observación:* Es usado en la recolección de la información, es factible y práctico, permite a partir de examinar la situación presente llegar a nuevos conocimientos para lograr una solución práctica del problema planteado, llevando a cabo el registro visual del fenómeno en cuestión.

Aportes prácticos esperados del trabajo

Con este trabajo se espera obtener una aplicación web que permita:

- ✓ Mejorar el proceso de autenticación en el Sistema de Administración de Identidades Gyes.
- ✓ Actualizar, controlar y procesar información referente a los contratos de servicios.
- ✓ Disminuir los costos en el soporte y autenticación de usuario.
- ✓ Aumentar la productividad de los usuarios al habilitar una experiencia de inicio de sesión sencilla.

El contenido del presente trabajo está dividido en 3 capítulos, distribuidos de la manera siguiente:

- ✓ **Capítulo 1:** *Fundamentación teórica:* se realizará un estudio del estado del arte abordando las características principales que se van a utilizar como base para desarrollar el sistema propuesto, así como un análisis de las tendencias, metodologías, lenguajes y programas más actuales que serán utilizados en el proceso de desarrollo de la aplicación.
- ✓ **Capítulo 2:** *Características del sistema*: en este capítulo se describirá y caracterizará la solución propuesta partiendo de un análisis previo del objeto de estudio y la problemática. Además se realizará una especificación de las funcionalidades del sistema a través del levantamiento y descripción de los escenarios y los requerimientos de calidad de servicio que debe cumplir la solución.
- ✓ **Capítulo 3:** *Desarrollo y estabilización de la solución propuesta*: en este capítulo se establecerá la arquitectura y los patrones de diseño a utilizar, necesarios para la implementación de la solución; además se modelará el diagrama de clases, el del centro de datos lógicos y el de aplicación, se realizarán un conjunto de pruebas a la aplicación para comprobar que cumple con los requisitos propuestos y se dará una evaluación a dichas pruebas en forma de resumen.

1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

El siguiente capítulo es el resultado de una investigación acerca de los conceptos esenciales para comprender los procesos de negocio asociados al dominio del problema, los software existentes relacionados con el campo de acción, las nuevas tecnologías en este campo, las tendencias actuales, lenguajes de implementación y una breve reseña del proceso de desarrollo de software utilizado para darle solución al problema planteado.

1.2. La administración de identidades.

Identidad está definida como un grupo de características representativas de una persona.

La noción de identidad está relacionada con la caracterización de una persona ya sea física o moral. Por ejemplo, identidad puede ser categorizada de acuerdo con diferentes aspectos tal como identidad personal, identidad biológica (ADN), identidad social (membresía), o identidad legal y la integración de todas estas en diferentes situaciones (tal como actividades de ocio, negocios, trabajo o interacción social). La identidad de la persona intervendrá en el contexto laboral, en escenarios donde la competencia representa un importante factor de éxito en el cumplimiento de una meta. (2)

La identidad digital está estrechamente relacionada con la noción de identidad personal, la misma está formada por todos aquellos datos y servicios utilizados por un único individuo, que se encuentran relacionados o pertenecen a una red, definen al mismo dentro de la red y pueden ser verificados mediante documentos que los confirman. No debe confundirse “Identidad Digital” con “Identidad Electrónica”. Esta última tiene como característica utilizar una sola tarjeta o identificación para acceder a ciertos servicios que existen en el mundo físico. (3)

Perder la identidad digital, no se refiere solamente a que roben los datos de la cuenta de correo electrónico de un usuario. En ocasiones puede llegar a convertirse en una verdadera pesadilla con alcances insospechados. Por lo que las empresas consientes de la importancia de proteger las identidades de los usuarios que utilizan sus recursos y servicio, crean estrategias y mecanismos encaminados a preservar la integridad de estos datos. Uno de estos mecanismos son los Sistemas de Administración de Identidades.

1.2.1. Sistema de Administración de Identidades.

Se denomina sistema de administración de identidades a un sistema integrado de políticas y procesos organizacionales que pretende facilitar y controlar el acceso a los sistemas de información y a las instalaciones. Representa una categoría de soluciones interrelacionadas que se utilizan para administrar autenticación de usuarios, derechos y restricciones de acceso, perfiles de cuentas, contraseñas y otros atributos necesarios para la administración de perfiles de usuario en una aplicación. (4)

Los sistemas de administración de identidades, son una necesidad de aquellas organizaciones en las que debido a su crecimiento, el número de repositorios de identidad, usuarios y/o servicios ha alcanzado un tamaño lo suficientemente grande como para que sea complicado continuar cumpliendo con las reglas de negocio de la organización sin dedicar grandes cantidades de recursos a ello. Los mismos no son más que soluciones de software que centralizan y administran, los procesos de: Autenticación, proceso mediante el cual una entidad A puede verificar que otra entidad B es, o no, quien dice ser; Autorización, proceso mediante el cual una entidad A puede verificar que otra entidad B cumple, o no, con las políticas de acceso para realizar cierta acción. La autenticación es un prerrequisito para que ocurra la autorización; Aprovisionamiento, proceso de dar alta, mantenimiento y baja a las cuentas de usuarios asociadas a entidades de la organización de los sistemas de la misma, es decir, las funciones de aprovisionamiento abarcan todo el ciclo de vida de las cuentas de usuarios en la organización.

Entre los principales beneficios de este tipo de soluciones están la reducción de gastos en personal dedicado a la administración de sistemas y soporte, aumento de los niveles de seguridad de la organización a través de la incorporación de elementos como esquemas de autenticación más seguros, e incremento de la productividad a partir de modelos como Single Sign-On, el cual evita que los usuarios tengan que memorizar un sin número de credenciales o tener que presentarlas a cada uno de los sistemas con los cuales trabajan. (5)

1.3. Autenticación.

La autenticación o autentificación es el acto de establecimiento o confirmación de algo (o alguien) como auténtico, es decir que reclama hecho por, o sobre la cosa es verdadero. La autenticación de un objeto puede significar la confirmación de su procedencia, mientras que la autenticación de una persona a menudo consiste en verificar su identidad. (6)

1.3.1. Mecanismo general de autenticación.

La mayor parte de los sistemas informáticos mantienen de uno u otro modo una relación de identidades personales asociadas normalmente con un perfil de seguridad, roles y permisos. La autenticación de usuarios permite a estos sistemas asumir con una seguridad razonable, que quien se está conectando es quien dice ser, para que las acciones que se ejecuten en el sistema puedan ser referidas a esa identidad.

El primer elemento necesario y suficiente para la autenticación es la existencia de identidades biunívocamente identificadas con un identificador único. Los identificadores de usuarios pueden tener muchas formas siendo la más común una sucesión de caracteres conocida comúnmente como *login*.

El proceso general de autenticación consta de los siguientes pasos:

- ✓ El usuario solicita acceso a un sistema.
- ✓ El sistema solicita las credenciales al usuario.
- ✓ El usuario aporta las credenciales que le identifican y permiten verificar la autenticidad de la identificación.
- ✓ El sistema verifica, según sus reglas, si las credenciales aportadas son suficientes para dar acceso al usuario o no. (7)

1.3.2. Métodos de autenticación.

Los métodos de autenticación están en función de lo que utilizan para la verificación y estos se dividen en tres categorías:

- ✓ Sistemas basados en **algo se sabe (código secreto)**. Ejemplo, una contraseña.
- ✓ Sistemas basados en **algo que se posee (apoyo físico)**. Ejemplo, una tarjeta de identidad, una tarjeta inteligente, un dispositivo USB tipo *epass token, smartcard o dongle criptográfico*.
- ✓ Sistemas basados en **algo que es (biométrica):** una característica física del usuario o un acto involuntario del mismo que lo identifican. Ejemplo, verificación de voz, de escritura, de huellas, de patrones oculares. (7)

1.3.3. Autenticación por multi-factor.

La autenticación por multi-factor para los seres humanos se clasifica en dependencia de la cantidad de factores que la componen, por ejemplo:

- ✓ Autenticación de un factor:

- identificador + contraseña (elemento **que se sabe**).
- definición sin contacto (elemento **que se posee**).
- Biométrica o identificador + biométrica (elemento **que es**).

✓ Autenticación de dos factores:

- Tarjeta inteligente + código PIN (elementos **que se posee y que se sabe**).
- Tarjeta inteligente + biométrica (elemento **que se posee y que es**).
- Biométrica + contraseñas (elemento **que es y que se sabe**).

✓ Autenticación de tres factores:

- Tarjeta inteligente + cifra PIN + biométrica (elementos **que se posee** y **que se sabe** y **que es**). (8)

1.3.4. Single Sign-On (SSO).

Single Sign-On (SSO) es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación.

Hay cinco tipos principales de SSO:

✓ Enterprise Single Sign-On (E-SSO), también llamado *legacy single sign-on*, funciona para una autenticación primaria, interceptando los requerimientos de *login* presentados por las aplicaciones secundarias para completar los mismos con el usuario y contraseña.

✓ Web Single Sign-On (Web-SSO), también llamado *web access management* (Web-AM), trabaja sólo con aplicaciones y recursos accedidos vía web. Los accesos son interceptados con la ayuda de un servidor proxy o de un componente instalado en el servidor web destino. Los usuarios no autenticados que tratan de acceder son redirigidos a un servidor de autenticación y regresan solo después de haber logrado un acceso exitoso.

✓ Kerberos, es un método popular de externalizar la autenticación de los usuarios. Los usuarios se registran en el servidor Kerberos y reciben un "ticket", luego las aplicaciones cliente lo presentan para obtener acceso.

✓ Identidad Federada, es una nueva manera de concebir este tema, también para aplicaciones Web. Utiliza protocolos basados en estándares para habilitar que las aplicaciones puedan identificar los clientes sin necesidad de autenticación redundante.

- ✓ OpenID, es un proceso de SSO distribuido y descentralizado donde la identidad se compila en una *Url* que cualquier aplicación o servidor puede verificar. (9)

1.4. Sistema de Administración de Identidades Gyes.

En el Centro de Identificación y Seguridad Digital se está desarrollando el Sistema de Administración de Identidades Gyes compuesto por tres subsistemas; Autenticación, Autorización y Aprovisionamiento, que funcionan de manera conjunta para centralizar la seguridad digital de las aplicaciones empresariales en una organización. En la siguiente figura se muestra, de manera general, los elementos del sistema y su relación con los recursos y aplicaciones de la organización.

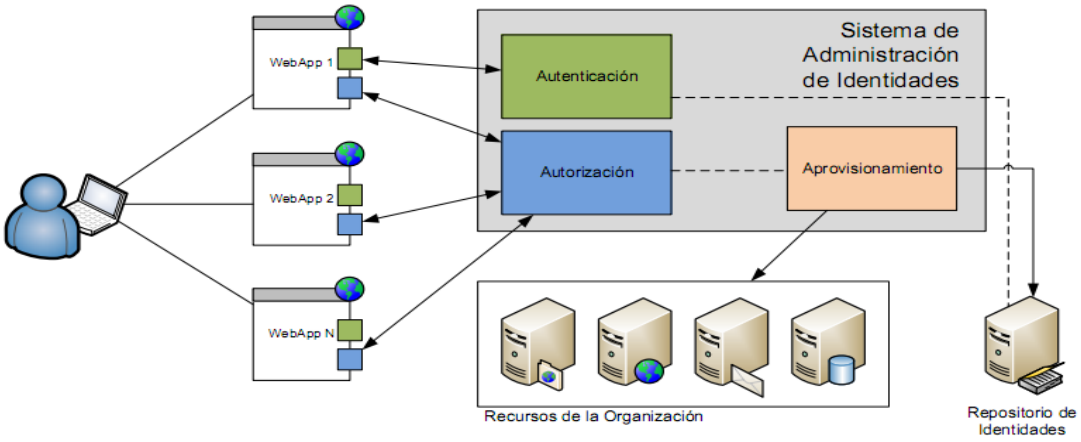

Figura 1: Vista General de Sistema de Administración de Identidades Gyes.

El Subsistema de Aprovisionamiento, de manera general cuenta con un servicio web en función de Proveedor de Servicios de Aprovisionamiento (PSP) que traduce las peticiones SPML en acciones de aprovisionamiento sobre los recursos de la organización y elabora la respuesta, que es enviada a la entidad emisora (RA) que generó la petición, las peticiones pueden ser generadas desde el portal de administración o desde el motor de tareas.

El Subsistema de Autorización centraliza las políticas de acceso a los recursos de la organización para una mejor administración. El proceso de autorización está basado en el modelo RBAC definido para el control de acceso. Este modelo es ideal para organizaciones con una estructura bien definida donde los roles están bien identificados. Cada rol tiene asignado un grupo de permisos, que no son más que operaciones sobre recursos, y los usuarios adquieren los permisos a través de los roles que posean.

El Subsistema de Autenticación está diseñado según el modelo Web Single Sign-On (Web SSO) y hace uso de librerías que implementan el estándar SAML versión 2.0 de la OASIS. En este modelo el usuario introduce sus credenciales sólo una vez y queda autenticado en más de una aplicación o recursos dentro de la organización. Esto es posible a partir de la utilización de tickets de autenticación firmados por un servidor y que son almacenados en el navegador del cliente, así las aplicaciones que sean accedidas después de la primera autenticación sólo verificarían la existencia e integridad de la información almacenada en el ticket. De esta forma el usuario no necesita memorizar múltiples credenciales y se aumenta la seguridad del proceso de autenticación al minimizar los intentos frente a varias aplicaciones. Este subsistema se divide en los siguientes componentes de software:

- ✓ Servidor de Autenticación: tiene asignado el rol de proveedor de identidades (IdP según el modelo Web SSO), es el encargado de interpretar las peticiones de autenticación en formato SAML emitidas por los proveedores de servicios (SP según el modelo Web SSO), ofrecer los mecanismos de autenticación predefinidos y a partir de la interacción directa con el repositorio de identidades de la organización determinar si una identidad es válida o no.
- ✓ Cliente de Autenticación: componente de software que se incluye a las aplicaciones o SPs de la organización para la comunicación con el Servidor de Autenticación o IdP. Intercepta las peticiones de acceso a recursos protegidos en el SP y las transforma a peticiones SAML que son enviadas al IdP.
- ✓ Aplicación de Contratos: la comunicación entre los SPs y el IdP será posible gracias a una relación de confianza establecida a partir de un contrato de servicio y es precisamente esta aplicación la encargada de gestionarlos.

1.5. Principales sistemas existentes.

1.5.1. A nivel internacional.

IBM Tivoli Access Manager for e-Business.

Tivoli Access Manager for e-business es una solución versátil para la gestión de problemas de autenticación y autorización. Centrado principalmente en las aplicaciones web, las implementaciones de *Access Manager* abarcan desde el inicio único de sesión (SSO) hasta despliegues de infraestructura de seguridad más complejos.

Tivoli Access Manager for e-business ayuda a:

- ✓ Definir y gestionar de forma centralizada la autenticación para una amplia gama de iniciativas empresariales, como portales de empleados, clientes y socios, sistemas de CRM, aprovisionamiento electrónico, proyectos de SSO que cubren toda la empresa y proyectos de externalización de servicios.
- ✓ Habilitar un SSO flexible para aplicaciones *web* que pueda abarcar varios sitios o dominios con una serie de opciones de SSO, a fin de contribuir a eliminar la necesidad de llamadas al servicio de atención al usuario y otros problemas de seguridad asociados a la existencia de varias contraseñas. Mediante su integración con otros proveedores de SSO, *Access Manager* supera el inicio de sesión limitado y contribuye a implementar una única autenticación para el usuario en todas sus interacciones en el sistema. Para la autenticación estandarizada entre dominios mediante SAML y *Liberty* ID/FF.

Tivoli Access Manager for e-business incorpora una amplia gama de mecanismos de autenticación de usuarios: identificadores y contraseñas de usuario, certificados de cliente, identidades para sistemas móviles e inalámbricos y soporte para señales de *RSA SecurID*. Es posible combinar varios mecanismos a fin de realizar una autenticación más detallada para los recursos más sensibles. Ofrece a los clientes con necesidades especiales de autenticación que puedan utilizar la nueva interfaz de autenticación externa que se incluye con este *release* o bien seguir utilizando las opciones de ampliación de autenticación que ya existían. Este producto ayuda a ampliar los procesos empresariales a sus socios comerciales y a sus asociados, además soporta plataformas como: *Windows NT, NetWare, Linux y UNIX*.

Interfaz de autenticación externa.

Access Manager siempre ha tenido un mecanismo basado en C que sirve para agregar mecanismos de autenticación personalizados además de los ya soportados en el producto. Ahora, con este *reléase* hay una nueva opción de autenticación que permite la comunicación de definiciones de autenticación sobre *HTTP*. Esto significa que el servicio de autenticación externo es prácticamente ilimitado en cuanto a la forma en que se implemente. Por ejemplo, puede ejecutarse en un servidor de aplicaciones *J2EE o .NET*, interactuar con el usuario, acceder a los datos que sean necesarios para tomar la decisión de autenticación y añadir los atributos necesarios a la credencial del usuario. (10)

Oracle Identity Manager.

Oracle Identity Manager es un sistema de gestión de identidades empresariales altamente flexible y escalable que controla centralmente las cuentas de los usuarios y los privilegios de acceso dentro de los recursos empresariales. Ofrece funcionalidades para la administración de roles e identidades, la administración de aprobaciones y pedidos, la administración de derechos basados en políticas, la integración de tecnología y la automatización para el cumplimiento y las auditorias. *Oracle Access Manager* permite:

- ✓ Control de acceso escalable para los entornos heterogéneos con una solución integrada, basada en estándares para la autenticación, el inicio de sesión web único y la creación y cumplimiento de las políticas.
- ✓ Protección para las empresas y sus clientes a través de la sólida seguridad de autenticación de múltiples factores que puede ajustarse dinámicamente sobre la base del contexto de acción del usuario y de la prevención anticipada y en tiempo real ante cualquier situación fraudulenta.
- ✓ Aplicaciones *online* con sólida autenticación, también ofrece una calificación de riesgos en tiempo real para identificar cualquier posible situación de fraude en las distintas etapas de una transacción.
- ✓ A los usuarios, autenticación e inicio de sesión unificados para todos sus recursos empresariales. Los usuarios pueden autenticarse una sola vez con una credencial única y luego tener acceso seguro a todas sus aplicaciones empresariales sin tener que registrarse nuevamente.
- ✓ Para los Sistemas Operativos, entornos de *Linux* y *Unix*, una infraestructura de autenticación de usuarios centralizada, segura y sin defectos. Ahora, el acceso a los sistemas operativos puede administrarse, imponerse y auditarse centralmente.

Beneficios y características:

- ✓ La mejor suite integral de su clase.
- ✓ Con funcionamiento permanente.
- ✓ Administración de Identidad Centrada en Aplicaciones.
- ✓ Reduce los costos de desarrollo y administración.
- ✓ Reduce el riesgo.
- ✓ Permite el Cumplimiento Regulatorio.
- ✓ Brinda una mejor experiencia para el usuario final.

Oracle es ampliamente reconocido como líder en el entorno de gestión de identidad –este reconocimiento proviene de analistas del sector, la prensa y, lo más importante, de su creciente base de clientes. La gestión de identidad es un área estratégica de foco para *Oracle* –además de ofrecer las mejores tecnologías de su clase a sus clientes globales, *Oracle Identity Management* también sustenta la próxima generación de Aplicaciones *Oracle Fusion*. A medida que el mercado de gestión de identidad sigue desarrollándose, *Oracle* continúa ofreciendo innovaciones a través de su liderazgo en la comunidad de estándares, así como a través de la estrecha colaboración con sus clientes. (11)

Pluggable Authentication Module (PAM)

Pluggable Authentication Module se ha convertido en el estándar de facto para la autenticación de usuarios en los sistemas *UNIX.*

PAM no es un modelo en sí, sino un mecanismo flexible para la autenticación, que proporciona una interfaz entre las aplicaciones de usuario y diferentes métodos de autenticación, tratando así de dar solución a uno de los problemas más conocidos de la autenticación de usuarios: el de que una vez que se ha determinado e implantado un mecanismo en un medio, es difícil cambiarlo.

PAM permite el desarrollo de programas, independientes del mecanismo de autenticación a utilizar. Así es posible que un programa que aproveche las facilidades ofrecidas por PAM sea capaz de utilizar desde el sencilla contraseña hasta dispositivos *hardware* como lectores de huella digital, pasando por servidores LDAP o sistemas de gestión de bases de datos. Y, por supuesto, todo esto sin cambiar ni una sola línea de código.

Pero PAM va más allá todavía, permitiendo al administrador del sistema modelar e implementar diferentes políticas de autenticación para los distintos usuarios de forma individual para cada servicio.

Su misión no es, únicamente comprobar que un usuario es quien dice ser. Su alcance es mucho mayor y pueden dividirse sus tareas en cuatro grupos independientes de gestión, cada uno de los cuales se encarga de un aspecto diferente de los servicios restringidos.

- ✓ Cuenta: en este grupo se engloban tareas que no están relacionadas directamente con la autenticación. Algunos ejemplos son permitir/denegar el acceso en función de la hora, los recursos disponibles o, incluso, la localización. Ofrece verificación de cuentas de usuario.

- ✓ Autenticación: tareas encaminadas a comprobar que, efectivamente, el usuario es realmente quien dice ser. Estas tareas ofrecen incluso un sistema de credenciales que permiten al usuario ganar ciertos privilegios (fijados por el administrador).
- ✓ Contraseña: se encarga de mantener actualizado el elemento de autenticación asociado a cada usuario (por ejemplo, su contraseña). Acciones como comprobar la fortaleza de una clave son típicas de este grupo.
- ✓ Sesión: en este grupo se engloban tareas que se deben llevar a cabo antes de iniciarse el servicio y después de que este finalice. Es especialmente útil para mantener registros de acceso o hacer accesible el directorio *home* del usuario.

En resumen, podrían sintetizarse las ventajas más importantes de PAM en los siguientes puntos:

- ✓ Ofrece un esquema de autenticación común y centralizado.
- ✓ Permite a los desarrolladores abstraerse de las labores de autenticación.
- ✓ Facilita el mantenimiento de las aplicaciones.
- ✓ Ofrece flexibilidad y control tanto para el desarrollador como para el administrador de sistema. (12)

Este método se utiliza principalmente en aplicaciones de *Linux* para autenticarse en el sistema, pues una aplicación preparada para PAM puede cambiar el mecanismo de autenticación que se está usando sin tener que recopilar las fuentes. PAM fue más tarde estandarizado como parte de la *X/Open* proceso de normalización de *UNIX*, lo que resulta en la *X/Open Single Sign-On (XSSO)* estándar.

Como una de sus limitaciones es que es compatible actualmente solo con algunos sistemas operativos como: *Linux*, *Mac OS X*, *NetBSD* y *Solaris*, pero no con ninguna versión de *Windows*.

TrustedX Plataforma de Servicios de Confianza.

TrustedX es una plataforma de servicios web que aporta mecanismos de seguridad y confianza en Arquitecturas Orientadas a Servicios (SOA).

TrustedX está diseñado para: independizar los servicios de seguridad y confianza de los procesos de negocio; ofrecer un conjunto completo y uniforme de funciones de autenticación, firma electrónica y cifrado de datos; permitir la clasificación e interpretación del nivel de confianza de la información; ofrecer un marco común de interoperabilidad con los dominios de seguridad externos y mayor orientación a los procesos de negocio, en los procesos de toma de decisiones es clave conocer con exactitud tanto el nivel de confianza de la información como sus autores y atributos. *TrustedX* destaca por su orientación al

negocio, simplifica la complejidad de los procesos, aporta mayor fiabilidad y minimiza los cambios en las aplicaciones. De este modo, se alinea en la dinámica de reconocimiento de nuevos servicios de confianza o nuevos mecanismos de autenticación.

Las funciones de *TrustedX* permiten gestionar la seguridad y la confianza de una forma estándar y orientada a servicios. Dichas funciones se agrupan en diferentes servicios:

- ✓ Autenticación y autorización: permite el intercambio de información de autenticación y autorización entre las aplicaciones corporativas y los dominios de seguridad externos. Disponiendo así de un control de acceso único a nivel de web (SSO) mediante los estándares definidos por OASIS.
- ✓ Validación de certificados digitales: permite que se reconozcan múltiples prestadores de servicios de certificación, se uniformice la información asociada a los certificados y se soporten mecanismos estándares de validación de certificados y cualquier otro mecanismo personalizado.
- ✓ Firma electrónica: soporta la mayoría de los formatos de firma de documentos electrónicos, correo electrónico y mensajería web; estos formatos incluyen firmas múltiples, firmas con sello de tiempo y firmas longevas.
- ✓ Pasarela de integración: define y conecta sucesivas transformaciones de datos XML, posibilitando que la plataforma actúe como pasarela de confianza entre procesos, aplicaciones o redes.
- ✓ Gestión de objetos y entidades: permite que el sistema ofrezca una vista uniforme de los objetos y entidades gestionados por la plataforma; enmascaran los formatos de datos (XML, ASN.1, Texto) y las diferentes fuentes de información (LDAP, SQL, Ficheros) utilizados por la plataforma. (13)

Los sistemas estudiados constituyen la puesta en práctica de un conjunto de teorías e investigaciones sobre el tema, los cuales ocupan un papel importante en el progreso tecnológico e investigativo, la incorporación de sus funcionalidades constituiría un notable aumento en la flexibilidad y escalabilidad en el proceso de autenticación del Sistema de Administración de Identidades Gyes, pero sumado a los análisis particulares de cada sistema, algunos no cumplen las especificaciones de calidad de servicio (*Pluggable Authentication Module*), puesto que no brinda soporte para plataformas *Windows*, además para el país representaría una alta inversión asumir estos sistemas extranjeros (*Oracle Identity Manager, IBM Tivoli Access Manager for e-Business y TrustedX*), la cual no podría permitirse por los altos costos que implicaría el despliegue de los mismos. También se presenta el problema de que ninguno de estos

sistemas incluye su código fuente, siendo esto un riesgo para la seguridad del proceso de autenticación en el del Sistema de Administración de Identidades Gyes y en toda aquella organización que lo utilice.

1.5.2. A nivel nacional.

En Cuba el desarrollo informático se ha ido incrementado gradualmente, pues existe conciencia del valor que implica elevar la productividad, también la necesidad de nuevos sistemas que cumplan requisitos específicos en procesos empresariales determinados.

En la UCI existen algunas soluciones que incorporan un porciento alto de los escenarios propuestos por los estándares para sistemas de seguridad. Entre las soluciones estudiadas se encuentra la implementada por la antigua facultad 10 que desarrolló un software de autenticación y control centralizado para la corporación de PDVSA, capaz de mapear las credenciales de los usuarios hacia todas las aplicaciones de dicho sistema y otros sistemas heterogéneos, proporcionando una infraestructura para simular una autenticación única para el usuario corporativo frente a una plataforma tecnológica heterogénea dentro del ambiente de aplicativos de integración, en que los usuarios puedan acceder a diferentes aplicaciones con solo un conjunto de credenciales. Además el proyecto de salud de la facultad 7 que está implementado sobre tecnología *Java*, por lo que la integración con este sistema se reduce solo a nivel de servicios web, pero no es un componente o módulo independiente del sistema. Las aplicaciones de la facultad 7 y facultad 10 son soluciones hechas a la medida que solo incorporan las funcionalidades particulares de cada uno de sus problemas, además se basan fundamentalmente en reglas o flujos que en su mayoría hay que configurarlas de forma manual, a medida que aumenta el nivel de restricciones, aumenta la complejidad de las configuraciones. Este proceso se torna tedioso y complejo para administradores que gestionen la seguridad de varias aplicaciones ya que si un usuario tiene acceso a varias aplicaciones tendría que repetir la configuración en cada uno de estos. (14)

Se puede concluir que dichas soluciones no cubren los requerimientos identificados para el Sistema de Administración de Identidades Gyes por lo que no es recomendable reutilizarlas.

1.6. Tecnologías, herramientas y metodología a utilizar.

Para el desarrollo de la solución se utilizarán las tecnologías ya definidas en la línea de la arquitectura del Departamento de Seguridad Digital más específico en el del Sistema de Administración de Identidades Gyes.

1.6.1. Metodología de desarrollo de software.

Microsoft Solution Framework Agile (MSF).

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.

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.

Esta metodología soporta 17 flujos de trabajo básicos, en los cuales se agrupan diferentes actividades, e incluye además cinco fases para el desarrollo y seguimiento del producto, estas son: Visión y Alcance, Planificación, Desarrollo, Estabilización y Despliegue.

MSF Ágil dispone de los tipos de elementos de trabajo 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 el futuro. (15)

1.6.2. Lenguaje de modelado.

Lenguaje Unificado de Modelado (UML).

Lenguaje Unificado de Modelado es un lenguaje gráfico que especifica, construye, visualiza y documenta las partes o artefactos que constituyen la información utilizada y originada mediante un proceso de software, además es un lenguaje orientado a objetos.

Los principales beneficios de UML son:

- ✓ Modelar sistemas utilizando conceptos orientados a objetos.
- ✓ Establecer conceptos y artefactos ejecutables.
- ✓ Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- ✓ Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- ✓ Mejor soporte a la planeación y al control de proyectos.
- ✓ Alta reutilización y minimización de costos.

Se puede emplear de diferentes formas para dar soporte a una metodología de desarrollo de software pero no especifica en sí, qué metodología o proceso utilizar.

UML no se puede comparar con la programación estructurada, pues no es una programación, solo diagrama la realidad de un requerimiento. Mientras que, la programación estructurada es una forma de programar como lo es la orientada a objetos, sin embargo, la orientada a objetos es un complemento perfecto de UML, pero esto no quiere decir que UML se toma únicamente para lenguajes orientados a objetos. (16)

1.6.3. Herramienta de Modelado.

Altova UModel 2010.

Altova UModel 2010 genera códigos *Java*, C#, o *Visual Basic* .NET, además se utiliza para diseñar modelos de aplicaciones en UML y documenta el proyecto. Con esta herramienta se puede realizar una ingeniería inversa a los programas existentes pasándolos a diagramas UML, luego afina los diseños y termina generando el código a partir de estos. Además es la manera simple y asequible de dibujar diagramas en UML, pues incluye las mayores funcionalidades para proveer a los usuarios de las más complicadas ventajas del desarrollo de software y combina una interfaz visual agradable con funcionalidades de usabilidad superiores para ayudar a suavizar la curva de aprendizaje de UML.

Sus características para el desarrollo de software basado en las capacidades de modelado avanzado son:

- ✓ 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*. (17)

1.6.4. Lenguajes de Programación.

C-Sharp (C#).

C-Sharp es un lenguaje moderno y orientado a objetos, con una sintaxis muy similar a la de C++ y Java. Combina el poder y la flexibilidad de C++ con la productividad de *Visual Basic*. Además puede interactuar, tanto con otros componentes no gestionados fuera de la plataforma .NET como con componentes realizados en otros lenguajes en .NET. Este lenguaje evita los problemas de programación típicos en lenguajes como C o C++ debido a que, gestiona de forma automática la memoria.

Además, desde C# se puede acceder a una librería de clases completa y muy bien diseñada, permitiendo así a disminuir los tiempos de desarrollo en una gran medida.

La principal ventaja que tiene este lenguaje se encuentra en Visual Studio.*NET* ya que existe una estrecha relación entre dicho entorno y C#, mucho mayor que la que tiene Visual C++.NET y equivalente e incluso superior a la de Visual Basic .NET. (18)

JavaScript.

JavaScript 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 no sólo de pequeños *scripts*, sino también de programas más grandes, orientados a objetos, con funciones y estructuras de datos complejas. Es bastante sencillo y pensado para hacer las cosas con más rapidez, por lo que es fácil de aprender y utilizar en toda su fuerza. Con este lenguaje se puede definir interactividades con el usuario y se puede crear efectos especiales en las páginas, donde se tienen dos variantes: por un lado permite la ejecución de instrucciones como las respuestas a las acciones del usuario, con lo que se puede crear páginas interactivas y por otro 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*. (19)

CSS.

Las hojas de estilo se desplegaron para remediar los defectos de HTML 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.

Al utilizar las hojas de estilo, cuando sea necesario 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 sólo 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. (20)

1.6.5. Plataforma de Desarrollo.

.NET.

La plataforma .NET es el conjunto de nuevas tecnologías en las que *Microsoft* ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios, que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, el lenguaje de programación y el modelo de componentes con los que hayan sido desarrollados.

La plataforma .NET constituye un entorno para la construcción, desarrollo y ejecución de servicios *web* y otras aplicaciones. A continuación se resumen las ventajas más importantes que proporciona .NET:

- ✓ Código administrado: El CLR controla los recursos del sistema para que la aplicación se ejecute correctamente a través de un control automático del código.
- ✓ Interoperabilidad multilenguaje: El código puede ser escrito en cualquier lenguaje compatible con .NET.
- ✓ Compilación JIT: Compila el código intermedio a través del compilador JIT generando el código máquina, propio de la plataforma, aumentando así el rendimiento de la aplicación.
- ✓ Recolector de basura: Ocurre una vez que el CRL detecta cuando el programa deja de utilizar la memoria y la libera automáticamente, posibilitando que el programador no tenga que liberar la memoria de forma explícita aunque también es posible hacerlo manualmente.
- ✓ Seguridad de acceso al código: Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente de la *web* sin tener que preocuparse si esto va a estropear el sistema, además posibilita declarar que una pieza de código tenga permisos de lectura pero no de escritura. (21)

1.6.6. Herramienta de Desarrollo.

Visual Studio 2010 Ultimate.

Visual Studio 2010 Ultimate añade nuevas características mejoradas que hacen el proceso de desarrollo, desde el diseño hasta la implementación, sea más sencillo. Además, posibilita que los equipos puedan observar un mayor ahorro de costes y mayor productividad al hacer uso de características de colaboración avanzadas, así como herramientas de pruebas y depuración integradas que de una manera u otra ayudan a crear un código de gran calidad.

Visual Studio 2010 Ultimate toma como objetivo varias versiones de .NET *Framework* con la misma herramienta. A continuación se muestran otras características importantes:

- ✓ Administración del ciclo de vida de las aplicaciones (ALM): Contribuye a que las organizaciones colaboren y se comuniquen de forma efectiva en todos los niveles, y a que tengan una idea precisa del estado real del proyecto, garantizando que se ofrezcan soluciones de gran calidad al tiempo que se reducen los costos.

- ✓ Depuración y diagnóstico: Tiene una valiosa característica de depuración que hace que el argumento "no reproducible" sea cosa del pasado. Se pueden archivar errores enriquecidos y modificables. Además incluye análisis de código estático, métricas de código y creación de perfiles.
- ✓ Arquitectura y modelado: El Explorador de arquitectura ayuda a entender los activos de código existentes y otras interdependencias. Los diagramas por capas ayudan a garantizar el cumplimiento de la arquitectura y permiten validar artefactos de código con respecto al diagrama. Además, Visual Studio 2010 Ultimate admite los cinco diagramas de UML más comunes que conviven junto con su código.
- ✓ Desarrollo de bases de datos: Visual Studio 2010 Ultimate proporciona potentes herramientas de implementación y administración de cambios que garantizan que la base de datos y la aplicación estén siempre sincronizadas. (22)

1.6.7. Herramienta para el control de versiones.

Subversion.

Subversion es un sistema de control de versiones, es un software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser el nombre de la herramienta utilizada en la línea de órdenes. Además puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Una característica importante de Subversión es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente, en cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en un instante determinado. (23)

1.6.8. Gestor de Base de Datos.

PostgreSQL.

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD; además de ser un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Es más completo que

MySQL ya que permite métodos almacenados, restricciones de integridad y vistas, aunque en las últimas versiones de MySQL se han hecho grandes avances en ese sentido.

Entre sus principales características se tienen:

- ✓ Cuenta con una alta concurrencia mediante un sistema denominado MVCC.
- ✓ Permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- ✓ Cada usuario obtiene una visión consistente de lo último a lo que se le hizo al *commit.*
- ✓ Permite la creación de vistas, integridad transaccional, herencia de tablas, tipos de datos y operaciones geométricas, claves ajenas y disparadores.

Desventajas:

- ✓ Limitaciones al escribir funciones y procedimientos en comparación con Oracle's PL/SQL o Sybase's T-SQL.
- ✓ Las tablas espaciosas, tablas particionadas, y con bloqueo altamente complicado siguen siendo ofrecidas por los vendedores propietarios de bases de datos.
- ✓ Carencia de herramientas de desarrollo propio. (24)

MongoDB

MongoDB es una base de datos orientada a documentos, de código abierto, escrita en C++. A diferencia de otros sistemas de gestión de bases de datos, no sigue el modelo relacional. La base de datos gestiona colecciones de documentos en objetos JSON, posibilitando que se pueda inspeccionar la estructura de los documentos que contiene y propiciar así un sistema *query* genérico. Está orientada a la *web* 2.0 y a las configuraciones de varios servidores de bases de datos. Además es una solución escalable y de alto rendimiento.

Su principal limitante es que MongoDB tiene procesos de 32 bits que se limitan a alrededor de 2,5 GB de datos. La razón de esto es que su motor de almacenamiento utiliza archivos asignados en memoria para el rendimiento. Principalmente por esta limitación de capacidad de almacenamiento de datos es que hemos decidido no utilizarlo. (25)

CouchDB

CouchDB es una base de datos de documentos, de código abierto, mantenida por *Apache*, escrito en el lenguaje de programación *Erlang*.

Características:

- ✓ A diferencia de las bases de datos SQL que están diseñadas para almacenar e informar sobre las estructuras y los datos relacionados entre sí, CouchDB está diseñado para almacenar e informar sobre las grandes cantidades de documentos semi-estructurados y orientados. Además, simplifica enormemente el desarrollo de aplicaciones orientadas a documento, que constituyen la mayor parte de las aplicaciones *web* de colaboración.
- ✓ Proporciona un mecanismo de indexación muy potente que obtiene un control sin precedentes en comparación con la mayoría de bases de datos.
- ✓ Fue diseñado con replicación bi-dirección (o sincronización) y la operación *off-line* en mente. Esto significa que varias réplicas pueden tener sus propias copias de los mismos datos, modificarlo, y luego sincronizar los cambios en un momento posterior.
- ✓ Los documentos se ven como campo/valor expresados como JSON, los valores del campo pueden ser cosas simples, como cadenas, números o fechas. Pero también puede utilizar listas ordenadas y mapas asociativos. Todos los documentos en una base de datos CouchDB tiene un identificador único y no hay ningún esquema del documento requerido.
- ✓ CouchDB expone una API REST JSON que se puede acceder desde cualquier entorno, que permita a las peticiones HTTP y una arquitectura de *plugin* mediante la cual se puede crear las funciones en el lenguaje favorito como *JavaScript*. Además tiene miles de bibliotecas de cliente de terceros que hacen que este sea aún más fácil su lenguaje de programación de la elección. CouchDB construido en la administración de la consola *web* habla directamente a la base de datos mediante peticiones HTTP emitidas desde su navegador.
- ✓ Se adapta a los cambios en la estructura de los documentos que es necesario almacenar. De esta forma no es de preocupación lo que se va guardando en la base de datos.
- ✓ No ofrece un lenguaje tipo SQL para realizar consultas sino que nos ofrece un sistema basado en *MapReduce* para poder obtener los datos que se quieran.
- ✓ Permite hacer algo similar a vistas de datos tradicionales usando *JavaScript.*

- ✓ Está en uso en muchos proyectos de *software* y sitios *web*, incluyendo *Ubuntu*, donde se utiliza para sincronizar los datos de direcciones y marcador.
- ✓ Con CouchDB, el esquema no se aplica, los tipos de documentos con un nuevo significado puede ser añadido con seguridad junto a los antiguos. El motor de vista, utilizando *JavaScript*, está diseñado para manejar con facilidad los nuevos tipos de documentos y documentos diferentes pero similares. (26)

Se decidió utilizar este gestor de base de datos, primeramente porque las bases de datos documentales son sencillas, ligeras y flexibles, además la estructura de los esquema de autenticación con la que se trabajará es idéntica a la estructura que ofrece la bases de datos documentales, es una estructura llave/valor, por lo que se nos hace más fácil el trabajo con ella y no con una base de datos relacional donde existen tablas y relaciones entre ellas.

1.6.9. Tecnologías a utilizar.

ASP.NET.

ASP.NET es la parte más importante de la capa superior de la plataforma .NET, constituye más que una nueva versión de la tecnología ASP, ya que, supone una nueva idea y forma de programar aplicaciones *web*, donde el programador puede centrarse exclusivamente en la lógica de la aplicación sin preocuparse de los detalles de la interfaz.

En resumen ASP.NET es un *framework* para aplicaciones web, fue desarrollado y comercializado por *Microsoft*. Se usa para construir sitios *web* dinámicos, aplicaciones *web* y servicios *web* XML, incorporando un nuevo concepto en el desarrollo de tecnologías Internet: los servicios *web*. Estos servicios representan un paso más hacia la descentralización del *software* en la red y de hecho, son un factor clave para el desarrollo de una *web* orientada a objetos. Además está construido sobre el CLR, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .*NET Framework*, entre ellos *Microsoft Visual Basic* y C#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del CLR, seguridad de tipos y herencia. (27)

AJAX

AJAX son las siglas de **A**synchronous **J**avaScript **A**nd **X**ML. No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML) que permite crear aplicaciones interactivas usando diferentes tecnologías *web* que colaboran entre ellas.

La característica fundamental de AJAX es que posibilita actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar se puede enviar información al servidor. (28)

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, pues está basado en estándares abiertos como *JavaScript* y *Document Object Model* (DOM). JavaScript es el lenguaje interpretado, en el que normalmente se efectúan las funciones de llamada de AJAX mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*. (29)

Entre las características de AJAX se incluyen bibliotecas de *scripts* de cliente, que incorporan las tecnologías *script* y HTML dinámico para varios exploradores, e integración con la plataforma de desarrollo para servidores de ASP.NET. Las aplicaciones construidas con el mismo eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. Su nueva capa intermedia mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

Ventajas de AJAX:

- ✓ Soportada por la mayoría de los navegadores modernos.
- ✓ Interactividad: el usuario no tiene que esperar hasta que lleguen los datos del servidor.
- ✓ Portabilidad.
- ✓ Mayor velocidad, debido a que no hay que retornar la página nuevamente.
- ✓ La página se asemeja a una aplicación de escritorio.

Desventajas de AJAX:

- ✓ Se pierde el concepto de volver a la página anterior.
- ✓ La existencia de páginas con AJAX y otras sin esta tecnología hace que el usuario se desoriente.
- ✓ Problemas con navegadores antiguos que no implementan esta tecnología.
- ✓ No funciona si el usuario tiene desactivado el *JavaScript* en su navegador.
- ✓ Requiere programadores que conozcan todas las tecnologías que intervienen en AJAX.

- ✓ Dependiendo de la carga del servidor se puede experimentar tiempos tardíos de respuesta que desconciertan al visitante. (30)

ASP.NET MVC.

MVC es un *framework* que divide la implementación de una aplicación en tres componentes: Modelos, Vistas, Controladores.

Entre sus principales características se tienen:

- ✓ Permite mantener una separación limpia de detalles entre modelos, vistas y controladores de la aplicación posibilitando realizar un desarrollo basado en pruebas por defecto lo que permite implementar pruebas unitarias automáticas que verifican los requerimientos del nuevo código sin tener que ejecutar controles a un proceso ASP.NET.
- ✓ Se puede utilizar el *framework* de pruebas unitarias que se desee.
- ✓ Los contratos del *framework* se basan en interfaces y son fácilmente intercambiables.
- ✓ Está diseñado para que pueda ser personalizado fácilmente, por lo que es altamente extensible.
- ✓ Permite crear aplicaciones con URL limpias, pues incluye una herramienta muy potente para el mapeado de URL. A estas no les hace falta tener extensiones y están diseñadas para soportar un sistema de nombres amigable.
- ✓ No usa el modelo de interacción *post-back* para las comunicaciones con el servidor, sino que, se tiene que establecer una ruta para las interacciones del usuario a través de una clase controlador.
- ✓ Soporta todas las características de ASP.NET como la autenticación a través de formularios *Windows*, roles, cacheo de datos, administración del estado de la sesión, monitoreo del estado y sistemas de configuración. (31)

ORM Divan.

Divan es una librería de C# bajo la licencia MIT para usar CouchDB. Es más o menos completa, incluyendo las operaciones de API a accesorios, opiniones y documentos de diseño. Es bastante rápido y está diseñado para ser flexible.

Divan ha sido desarrollado internamente en *Foretagsplatsen* (una empresa sueca que lo utiliza en su sistema central).

Características:

- ✓ Tiene clases de documentos de diseño y definición de la vista para que se pueda crear y manipular sus puntos de vista de C#.
- ✓ Tiene soporte LINQ, por lo que se puede escribir un subconjunto de consultas en la sintaxis de LINQ.
- ✓ Tiene soporte básico y serialización JSON.
- ✓ Tiene integración básica con CouchDB-Lucene para la toma de índices de texto gratuitos.
- ✓ Ofrece varios "ganchos" para que sus propias subclases puedan añadir especialidades. (32)

1.7. Conclusiones.

Con el objetivo de desarrollar el Soporte Extensible para modos de Autenticación del Sistema de Administración de Identidades Gyes y que posea un alto grado de eficiencia en cuanto al proceso de autenticación de usuarios, se investigaron las características principales de los productos líderes en este campo, se analizaron los elementos fundamentales de las herramientas a utilizar para su implementación, reconociendo los beneficios que posibilita su uso en el desarrollo de la solución. La metodología que guiará su proceso de desarrollo será MSF Ágil, propiciando un marco de trabajo flexible adecuándose a las características del proyecto. La herramienta de modelado será Altova UModel 2010. Para la implementación de la solución se hará uso de la plataforma .NET lo que posibilita la creación de una aplicación robusta y escalable. Todo esto gestionado mediante el IDE de desarrollo Visual Studio 2010 Ultimate. Para lograr la persistencia de la información resultante de los procesos de negocio realizados, se utilizará CouchDB como gestor de base de datos. Todas las herramientas a utilizar tienen en común que son muy utilizadas a nivel mundial.

2. CARACTERÍSTICAS DEL SISTEMA.

2.1. Introducción.

En el presente capítulo se sentarán las bases necesarias para el desarrollo del sistema, abordándose las fases: Visión y Alcance y Planificación de la metodología MSF para el Desarrollo de Software Ágil. Para obtener una visión clara de la aplicación que se desea desarrollar se realizará la descripción del problema a resolver y la propuesta de solución planteada, incluyendo sus principales características, partiendo del análisis previo de la problemática y el objeto de estudio en cuestión; se efectuará además el levantamiento y descripción de los escenarios y los requerimientos de calidad de servicio que debe cumplir la aplicación. Conjuntamente se realizará un plan de iteraciones para la correcta planificación de las etapas de desarrollo del *software*.

2.2. Fase Visión y Alcance.

En el presente epígrafe se documentará la fase Visión y Alcance, proporcionando una visión clara de los objetivos que persigue el desarrollo del sistema, una idea de la lógica del negocio y de las alternativas propuestas para solucionar el problema planteado así como de la manera más óptima de hacerlo.

2.2.1. Descripción del Problema.

A pesar que actualmente el Sistema de Administración de Identidades Gyes se encuentra en etapa de desarrollo, alguno de sus elementos presentan un cierto grado de completamiento, en este caso se encuentra el Servicio de Autorización y la Aplicación de Administración. En el mismo caso está el Servidor y el Cliente de Autenticación que ofrecen las funcionalidades de *Single Sign-On* y *Single Logout*.

La autenticación se realiza a través de una aplicación web que utiliza como mecanismo de autenticación los parámetros usuario y contraseña, limitando la protección y el manejo correcto de los recursos existentes y frenando todo su potencial. Es decir: el Cliente de Autenticación intercepta las peticiones de acceso a recursos protegidos en los SPs y las transforma a peticiones SAML que son enviadas al Servidor de Autenticación o IdP que interpreta estas peticiones y ofrecer los mecanismos de autenticación predefinidos a partir de la interacción directa con el repositorio de identidades de la organización. Pero el IdP solo ofrece el mecanismo basado en los parámetros: Usuario y contraseña para determinar si una identidad es válida o no.

2.2.2. Propuesta Solución.

Se propone desarrollar un Soporte Extensible, que permita diferenciar los distintos esquemas existentes, a quienes pertenecen y cuando utilizarlos. La solución contará con dos módulos, el módulo para la gestión de los contratos de servicios donde: la comunicación entre los SPs y el IdP es posible gracias a una relación de confianza establecida a partir de un contrato de servicio y es precisamente esta aplicación la encargada de gestionarlos. En un contrato se definen los datos que identifican a un SP frente al IdP así como los esquemas de autenticación, los cuales se basa en reglas sencillas de combinación y estarán compuestos por uno o más parámetros como: número de activación de la computadora (PC), contraseña o certificado digital. Por su parte el módulo para la autenticación se encargará de todo el proceso de autenticación de usuarios en el Sistema de Administración de Identidades Gyes y cuenta con la política *Single Sign-On* (SSO), que permite iniciar sesión una vez y acceder a todos los recursos autorizados, centralizando el proceso en un solo punto.

De manera general el flujo en el proceso de autenticación coincidirá con el diagrama que se muestra en la siguiente figura y que se describe a continuación:

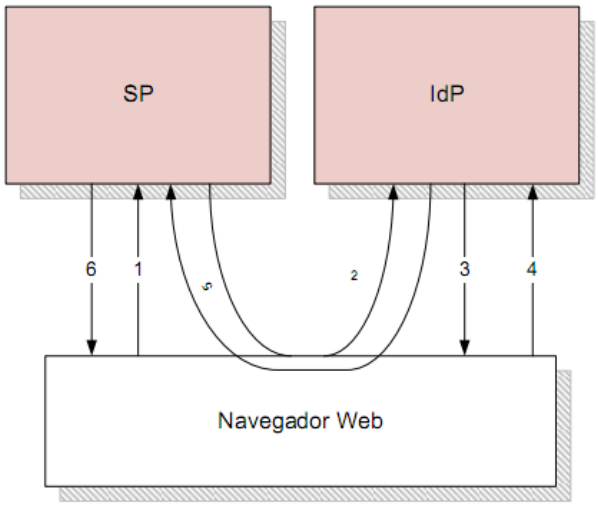

Figura 2: Flujo de Autenticación según el Modelo Web SSO.

1. El usuario, a través de su navegador web, solicita un recurso protegido en la aplicación.
2. SP redirige el pedido en formato SAML al IdP.
3. IdP solicita las credenciales según el esquema de autenticación definido en el contrato del SP.
4. El usuario envía las credenciales y el IdP verifica la validez de las mismas mediante la interacción con el repositorio de identidades.

5. IdP crea una sesión y redirige al recurso solicitado a través de una respuesta SAML.
6. SP interpreta la respuesta proveniente del IdP y devuelve el recurso solicitado.

En la siguiente tabla se muestra concretamente los módulos con que contará la aplicación y una breve descripción de los mismos.

Módulos del sistema	Descripción
Gestión de Contratos de Servicios.	Es el encargado de realizar todo el proceso de gestión de los contratos de servicios y los esquemas de autenticación para posibilitar la autenticación de usuarios en el Sistema de Administración de Identidades Gyes.
Autenticación.	Se encarga de gestionar la autenticación de usuarios, teniendo en cuenta los distintos esquemas definidos en el contrato de servicios establecido con anterioridad.

Tabla 1: Módulos del Sistema.

2.2.3. Vista Conceptual del sistema.

En la siguiente figura se puede apreciar la vista conceptual del sistema, la misma captura los objetos más importantes, los eventos que suceden en el entorno donde se desarrollan y las relaciones existente entre ellos.

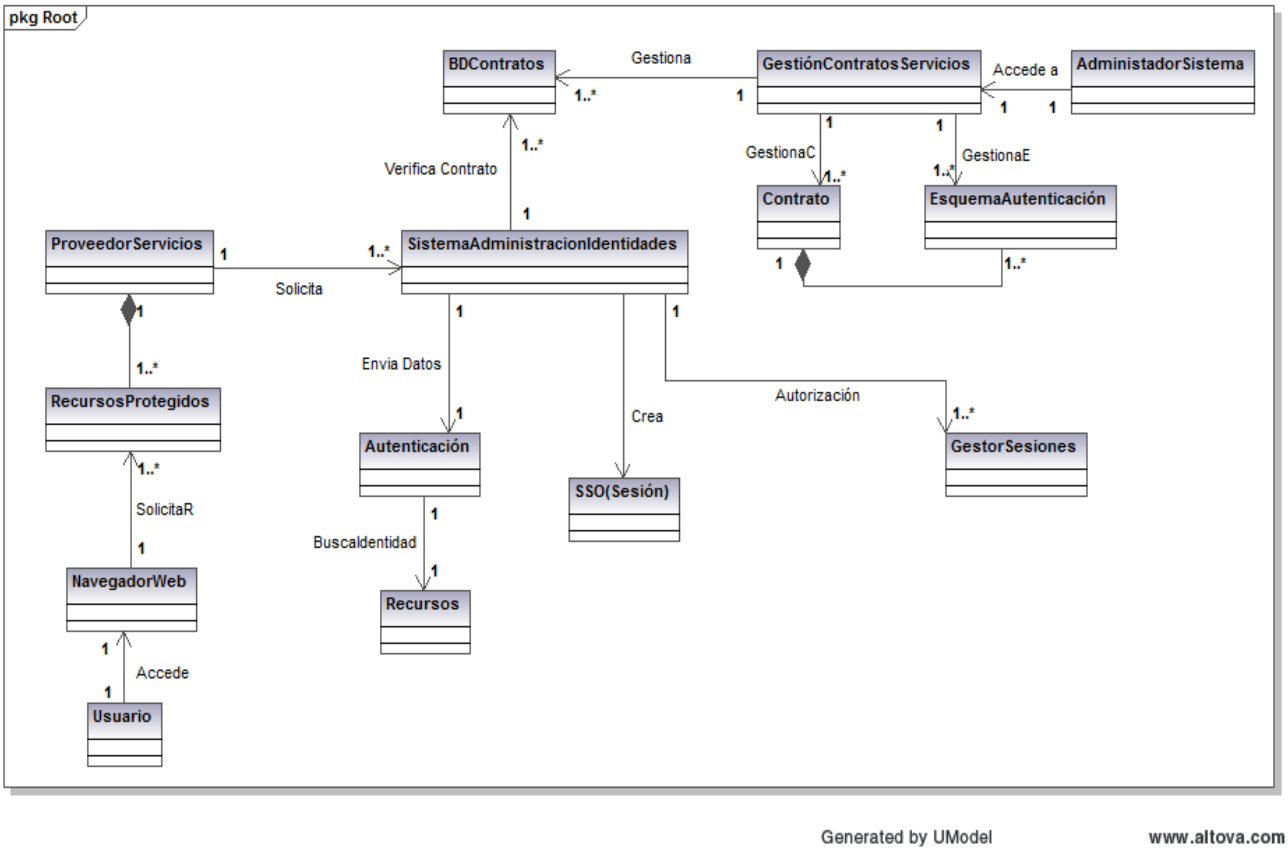

Figura 3: Modelo Conceptual del Sistema.

Descripción:

- ✓ **Gestión de Contratos de Servicios:** es la aplicación *web* encargada de todo el proceso de la gestión de los esquemas de autenticación y los contratos de servicios.
- ✓ **Contrato:** representa la entidad contratos de servicios.
- ✓ **Esquema de autenticación:** representa los esquemas a utilizar en la autenticación, están definidos en los contratos de servicios.
- ✓ **Proveedor de Servicios:** es el punto de comunicación entre las aplicaciones empresariales y el Servidor de Autenticación o IdP.
- ✓ **Aplicación Cliente:** representa la aplicación *web* que delega su seguridad al Sistema de Administración de Identidades Gyes.
- ✓ **Recursos Protegidos:** representa aquellos recursos protegidos a los que el usuario quiere acceder.

- ✓ **Usuario:** cualquier persona que quiera acceder a los recursos protegidos e iniciar una sesión global.
- ✓ **Administrador del Sistema:** es la persona encargada de gestionar los contratos de servicios y los esquemas de autenticación haciendo uso de la aplicación Gestión de Contratos de Servicios.
- ✓ **Base de Datos:** encargada de registrar todo el proceso de gestión de los contratos de servicios y los esquemas de autenticación.
- ✓ **Autenticación:** aplicación *web* que permite el proceso de autenticación de usuarios, interactúa con los lugares donde están guardados los parámetros de autenticación de cada identidad, validando las credenciales provistas por los usuarios de la organización.
- ✓ **Recursos:** lugares donde están guardados los parámetros de autenticación de cada identidad, puede ser: Ldap, Url, MySQL.
- ✓ **Gestor de Sesiones:** se encarga de manejar las sesiones *web* del Sistema de Administración de Identidades Gyes.

2.2.4. Definición de personas.

Los individuos o sistemas externos que interactúan con la aplicación se identificaron como personas, tal como se muestra a continuación.

Personas	Descripción
Administrador del Sistema.	Encargado de realizar todas las actividades relacionadas con la gestión de los contratos de servicios y maneja la base de datos.
Usuario de Sesión Global.	Usa la sesión global antes creada para acceder a los recursos protegidos.

Tabla 2: Descripción de las Personas.

2.3. Fase Planificación.

Es en esta fase en la que se determina la planificación del proyecto, el equipo prepara las especificaciones funcionales y los planes de trabajo. Los principales artefactos de esta fase son los escenarios y los requerimientos de calidad de servicios que sirven de guía para todo el proceso de desarrollo.

2.3.1. Lista de escenarios del sistema.

Para un mejor razonamiento en el momento de la implementación se especificarán y describirán los escenarios necesarios de la solución propuesta.

Un escenario es un medio por el cual se define una interacción entre una persona y el sistema para lograr cumplir una meta específica.

Módulo Gestión de Contratos de Servicios:

- ✓ GCS1. Gestionar Contrato de Servicios.
  - GCS1.1. Crear Contrato de Servicios.
  - GCS1.2. Editar Contrato de Servicios.
  - GCS1.3. Mostrar Contrato de Servicios.
  - GCS1.4. Eliminar Contrato de Servicios.
- ✓ GCS2. Gestionar Esquema de Autenticación.
  - GCS2.1. Crear Esquema de Autenticación.
  - GCS2.2. Mostrar Esquema de Autenticación.
  - GCS2.3. Eliminar Esquema de Autenticación.

Módulo Autenticación:

- ✓ A1 Solicitar Inicio de Sesión Global.
- ✓ A2. Solicitar Fin de Sesión Global.

2.3.2. Priorizar la lista de escenarios.

La priorización de los escenarios permite identificar los escenarios más importantes en el momento de implementar, para que los escenarios que tengan mayor prioridad se realicen en las primeras iteraciones del desarrollo del *software*. Según la metodología MSF para el desarrollo de *software* ágil los escenarios se clasifican con valores del 1 al 3; con 1 si tiene prioridad “Alta”, con 2 “Media” y con 3 “Baja”, por lo que la priorización de los escenarios identificados queda de la siguiente manera:

Escenarios	Prioridad
Gestionar Contrato de Servicios.	1
Gestionar Esquema de Autenticación.	1
Solicitar Inicio de Sesión Global.	2

Solicitar Fin de Sesión Global.	2

Tabla 3: Priorización de la Lista de Escenarios.

Para priorizar los escenarios se tuvo en cuenta que para desarrollar el escenario "Gestionar Contrato de Servicios", primeramente tiene que haberse desarrollado el escenario "Gestionar Esquema de Autenticación", pues uno depende del otro. Por otra parte los escenarios "Solicitar Inicio de Sesión Global" y "Solicitar Fin de Sesión Global" no son menos importantes sino que su implementación depende totalmente del desarrollo previo de los demás escenarios.

2.3.3. Requerimientos de calidad de servicio.

Los requerimientos de calidad de servicio representan los requisitos no funcionales en las categorías de plataforma, rendimiento, seguridad y apariencia. A continuación se listan los requerimientos de calidad de servicio que se identificaron para el desarrollo del sistema propuesto.

Plataforma que soporta.

- ✓ Las aplicaciones deben estar disponible las 24 horas del día.
- ✓ El sistema debe utilizar el .NET Framework en su versión 4.0.
- ✓ Debe ser soportado por los navegadores *Mozilla Firefox* e *Internet Explorer.*

Rendimiento.

- ✓ La conexión entre la base datos y las aplicaciones y las de la aplicaciones con los recursos donde están distribuidas las identidades debe ser de alta velocidad.

Apariencia o interfaz externa.

- ✓ Interfaz accesible e intuitiva, el manejo de las funcionalidades debe ser lo más intuitivo posible, de manera que sean muy claras las posibles acciones a llevar a cabo y la manera de hacerlas.
- ✓ Interfaz consistente con los prototipos de diseño definidos para los sistemas del proyecto.
- ✓ Permitir la paginación en los listados en la aplicación de Gestión de Contratos de Servicios.

Seguridad:

- ✓ El protocolo de seguridad para el *Login Page* del usuario debe ser HTTPS.

Accesibilidad:

- ✓ La aplicación de Gestión de Contratos de Servicios debe ser accesible nada mas para el administrador del sistema.

- ✓ La aplicación de Autenticación puede ser accesible para todos los usuarios que quieran iniciar una sesión global.

2.3.4. Plan de Iteraciones.

Para realizar de la manera más correcta y adecuada la planificación del desarrollo del sistema, es de gran necesidad hacer una estimación del tiempo de la codificación de cada uno de los escenarios. En relación con la prioridad que tengan 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.

La implementación de los escenarios estará dividida en las siguientes iteraciones:

- ✓ **Iteración #1:** se propone codificar los escenarios que proveen las funcionalidades pertenecientes al módulo "Gestión de Contratos de Servicios", es decir aquellos escenarios que tienen una alta prioridad.
- ✓ **Iteración #2:** se codificarán los escenarios correspondientes al módulo "Autenticación", específicamente aquellos que tienen una prioridad media.

En la siguiente tabla se evidencia la planificación realizada para el desarrollo de la solución propuesta.

Escenarios	Prioridad	Riesgo	Esfuerzo (días)	Iteración
Gestionar Contrato de Servicios.	Alta	Alto	17	1
Gestionar Esquema de Autenticación.	Alta	Alto	10	1
Solicitar Inicio de Sesión Global.	Media	Alto	17	2
Solicitar Fin de Sesión Global.	Media	Alto	10	2

Tabla 4: Planificación de los Escenarios.

2.3.5. Descripción de los escenarios.

Para el correcto desarrollo de la solución se realiza la descripción de los escenarios, a continuación se muestran las descripciones de los escenarios pertenecientes al módulo "Gestión de Contratos de Servicios", los cuales tienen una prioridad alta.

Nombre del Escenario	Gestionar Contrato de Servicios.
Identificador	GCS1

Objetivo del Escenario	Gestionar todos los procesos referentes al contrato de servicios.
Persona	Administrador del sistema.
Iteración.	1ra
Prioridad.	Alta.
Complejidad.	1
Descripción.	El administrador del sistema selecciona la opción Gestionar-Contrato de Servicios. El sistema muestra una lista con todos los contratos de servicios existentes en la base de datos y permite seleccionar las opciones Crear, Editar, Mostrar o Eliminar un contrato de servicios. Nota: Las opciones de Crear, Mostrar, Editar y Eliminar un contrato de servicios han sido omitidas debido a que estas se describen como tareas del escenario en cuestión.
Validaciones.	

Tabla 5: Descripción del Escenario "Gestionar Contrato de Servicios".

Nombre del Escenario	Gestionar Esquema de Autenticación.
Identificador	GCS2
Objetivo del Escenario	Gestionar todos los procesos referentes a los esquemas de autenticación.
Persona	Administrador del sistema.
Iteración.	1ra
Prioridad.	Alta.
Complejidad.	1
Descripción.	El administrador del sistema selecciona la opción Gestionar-Esquema de Autenticación. El sistema muestra una lista con todos los esquemas de autenticación existentes en la base de datos y permite seleccionar las opciones Crear, Editar, Mostrar o Eliminar un esquema de autenticación.

	Nota: Las opciones de Crear, Mostrar, Editar y Eliminar un esquema de autenticación han sido omitidas debido a que estas se describen como tareas del escenario en cuestión.
Validaciones.	

Tabla 6: Descripción del Escenario "Gestionar Esquema de Autenticación".

A continuación, se muestran las descripciones de los escenarios pertenecientes al módulo "Autenticación", los cuales tienen una prioridad Media.

Nombre del Escenario	Solicitar Inicio de Sesión Global.
Identificador	A1
Objetivo del Escenario	Lograr una sesión única de manera que el usuario no tenga que autenticarse en cada aplicación que tenga un contrato con el Servidor de Autenticación o IdP.
Persona	Usuario de sesión global.
Iteración.	2da
Prioridad.	Media
Complejidad.	1
Descripción.	El usuario solicita un recurso protegido. Automáticamente el sistema crea y envía una solicitud al Servidor de Autenticación, entonces, es ejecutado el esquema de autenticación especificado para el usuario, se realizan las validaciones correspondientes y se crea una respuesta al cliente con el resultado, que siendo positivo se crea una sesión global en dependencia de la solicitud. Finalmente, siempre que se cumpla con las condiciones establecidas, el sistema realiza una autenticación correcta para el usuario.
Validaciones.	✓ Se verifica que la petición sea válida. ✓ Se verifica que el sistema tenga un contrato válido con el Servidor de Autenticación.

Tabla 7: Descripción del Escenario "Solicitar Inicio de Sesión Global".

Nombre del Escenario	Solicitar Fin de Sesión Global.
Identificador	A2
Objetivo del Escenario	Eliminar la sesión única que se estableció en el escenario: "Solicitar Inicio de Sesión Global".
Persona	Usuario que tiene la sesión global.
Iteración.	2da
Prioridad.	Media.
Complejidad.	1
Descripción.	El escenario comienza cuando el usuario selecciona la opción *Logout* que a su vez envía la petición al Servidor de Autenticación y elimina la sesión global del usuario.
Validaciones.	✓ Se verifica la validez de la solicitud de fin de sesión.

Tabla 8: Descripción del escenario "Solicitar Fin de Sesión Global".

2.3.6. Especificación de tareas por escenarios.

El escenario "Gestionar Contrato de Servicios" se dividió en 4 tareas fundamentales, a continuación se describen las tareas Crear y Eliminar un contrato de servicios.

Nombre de la Tarea	Crear Contrato de Servicios.
Identificador	GCS1.1
Objetivo de la Tarea	Establecer una relación de confianza o contrato de servicios que dicte las reglas de autenticación entre el Servidor de Autenticación o IdP y un sujeto determinado o SP.
Persona	Administrador del sistema.
Iteración.	1ra
Prioridad.	Alta.
Complejidad.	1
Descripción.	El administrador accede al sistema y selecciona la opción Crear–Contrato de Servicios. El sistema muestra un formulario con los datos necesarios, el administrador introduce una identidad o sujeto y crea o selecciona un esquema de autenticación. El sistema crea el contrato de servicios con la información

	obtenida anteriormente haciéndose persistente. Finalmente queda establecida una relación de confianza entre el sujeto y el Servidor de Autenticación o IdP con la cual el sujeto debe poder autenticarse.
Validaciones.	

Tabla 9: Descripción de la Tarea "Crear Contrato de Servicios".

Nombre de la Tarea	Eliminar Contrato de Servicio.
Identificador	GCS1.4
Objetivo de la Tarea	Dar por terminada la relación de confianza entre el sujeto del contrato de servicios y Servidor de Autenticación o IdP.
Persona	Administrador del sistema.
Iteración.	1ra
Prioridad.	Alta.
Complejidad.	1
Descripción.	El administrador accede a la aplicación con la intención de eliminar un contrato de servicios. El sistema le muestra un listado de todos los contratos de servicios existentes y brinda la posibilidad de seleccionar el contrato de servicios que se desea eliminar. El administrador realiza la selección y oprime el opción Eliminar. Finalmente el sistema elimina el contrato de servicios de la base de datos y automáticamente deja de existir la relación de confianza entre el sujeto y el Servidor de Autenticación o IdP.
Validaciones.	

Tabla 10: Descripción de la Tarea "Eliminar Contrato de Servicios".

Las descripciones de las restantes tareas de los escenarios "Gestionar Contrato de Servicios" y "Gestionar Esquema de Autenticación" se pueden apreciar en el Anexo I.

2.4. Conclusiones.

En este capítulo se abordaron las fases Visión y Alcance y Planificación de la metodología MSF para el desarrollo de *software* ágil. Se realizó la descripción de la solución propuesta, definiéndose y

describiéndose los escenarios identificados en el problema a resolver y las tareas correspondientes a cada uno de ellos y se priorizaron dichos escenarios para hacer una correcta planificación por iteraciones y poder desarrollar de la forma más organizada posible el sistema propuesto. Además se especificaron los requerimientos de calidad de servicio que debe poseer el sistema.

3. DESARROLLO Y ESTABILIZACIÓN DEL SISTEMA.

3.1. Introducción.

En el presente capítulo se abordarán las fases: Desarrollo y Estabilización de la metodología MSF para el desarrollo de *software* ágil. Para un correcto desarrollo de la solución propuesta se definirá la arquitectura a utilizar, se especificarán los patrones de diseño a tener en cuenta para dar solución a los requisitos de calidad de servicio definidos, se modelarán y describirán las clases principales y para verificar el correcto funcionamiento de la solución se realizarán un conjunto de pruebas tanto a la interfaz como a las principales funcionalidades de la solución.

3.2. Fase de Desarrollo.

3.2.1. Especificación de la arquitectura a utilizar.

La Arquitectura de software es la organización fundamental de un sistema, encarnada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. Además es una vista estructural de alto nivel, que ocurre tempranamente en el ciclo de vida y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos de calidad del servicio. (33)

3.2.1.1. Modelo Vista Controlador (MVC).

Los estilos arquitectónicos se utilizan para sintetizar y tener un lenguaje que describa la estructura de las soluciones. Además definen los posibles patrones de las aplicaciones y permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos de calidad del servicio.

MVC es un estilo arquitectónico que separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres componentes diferentes y por tanto el Modelo, las Vistas y los Controladores se tratan como entidades independientes donde:

- ✓ La Vista: maneja la visualización de la información, o sea todo lo que el usuario ve en la pantalla.
- ✓ El Modelo: administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

- ✓ El Controlador: interpreta los eventos generados por las acciones del usuario, informando al modelo y/o a la vista que cambien según resulte apropiado. Tanto la vista como el controlador dependen del modelo, el cual no depende de estos. Esta separación permite construir y probar el modelo independientemente de la representación visual. (34)

Figura 4: Estilo arquitectónico Modelo-Vista-Controlador.

La solución divide la implementación del sistema en 3 componentes:

- ✓ Modelos (*Models):* Se encuentran constituidos por dos proyectos que agrupan los diferentes componentes: *GyesIdPReloaded.Lib,* es el encargado de gestionar los procesos del negocio y *GyesIdPReloaded.DAL,* es el encargado de mantener el estado del sistema para guardarlo en la base de datos.
- ✓ Vistas (*Views):* Guardan todas las interfaces de usuarios. Estas vistas son ficheros *aspx* que contienen todos los componentes *Java Script* y *CSS* del sistema. Algunos ejemplos son: *Contrato:* contiene a *Create.aspx, Details.aspx, Edit.aspx* e *Index.aspx* y *Esquema: c*ontiene a *Create.aspx, Edit.aspx, Index.aspx, Details.aspx*
- ✓ Controladores (*Controllers):* Administran y responden a las entradas de usuario y a las interacciones. Por ejemplo: *EsquemaController y ContratoController.*

3.2.1.2. Vista Lógica.

La vista lógica está representada por tres capas que se basan en la división del nivel de acceso a datos, nivel de lógica de negocio y nivel de presentación o aplicación como se muestra en la siguiente figura, lo que permite disminuir al máximo el acoplamiento, aumentar la reutilización entre las mismas, facilitar la modularidad, reusabilidad, el cambio y la portabilidad. La misma posibilita realizar grandes cambios sin tener que cambiar nada en las demás capas, permitiendo además trabajar de manera transparente una vez establecidas las conexiones entre las capas.

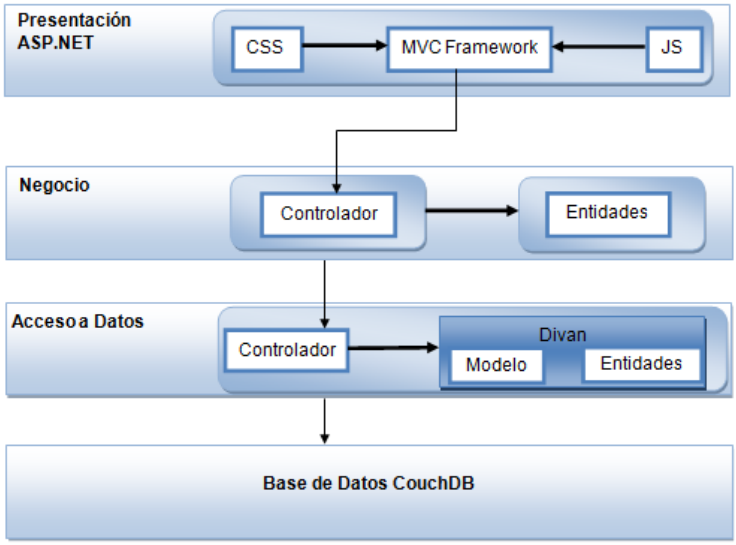

Figura 5: Vista Lógica de la Arquitectura.

Descripción de las capas:

- ✓ **Capa de presentación:** Está compuesta por todas las interfaces de usuario y los componentes necesarios para su correcto funcionamiento. Estos elementos pueden ser ficheros *JavaScript* y CSS. Esta capa se encuentra representada por un proyecto web y tiene interacción directa con la capa de negocio. Para el desarrollo de este proyecto web se hace uso del *framework* MCV ASP .NET el cual divide la implementación del proyecto en 3 componentes: modelos, vistas y controladores.
- ✓ **Capa de negocio:** En esta capa se recogen todas las funcionalidades necesarias para darle solución a los requerimientos de negocio. Las funcionalidades se encuentran definidas según el contexto en el que se desenvuelven. Tienen la responsabilidad de manejar todas las operaciones sobre una entidad de negocio en específico, así como todas las entidades que por conceptos de composición se encuentran relacionadas con esta. Por cada entidad de negocio se crea un controlador y una interfaz que debe ser implementada por el acceso a dato que le dará soporte. Además se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de acceso a datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. Se encuentra constituida por dos proyectos que agrupan los diferentes componentes: *GyesIdPReloadedCore.Lib y GyesIdPReloaded.Entities***.**

- ✓ **Capa de acceso a datos:** La capa de acceso a datos está directamente relacionada con las funcionalidades definidas en el negocio. Para establecer esta relación hace uso de las clases interfaces y controladoras que define la capa de negocio. De esta manera, es posible realizar cambios en esta capa sin que se vean afectadas las demás capas. Su principal función es realizar una implementación de las interfaces definidas en la capa de negocio y al mismo tiempo trabajar directamente con la fuente de datos establecida. En esta capa se encuentra incluido el framework Divan, herramienta utilizada para la generación del acceso a datos. La misma está constituida por el proyecto: *GyesIdPReloaded.DAL*
- ✓ **Base de Datos:** Está conformada por todo el conjunto de documentos y procedimientos que permiten el almacenamiento de la información recolectada y procesada utilizando como Sistema Gestor de Base de Datos *CouchDB.*

3.2.2. 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.

3.2.2.1. Patrones GoF.

Los patrones GoF se dividen en tres clasificaciones:

- ✓ Patrones Estructurales: describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.
- ✓ Patrones de Comportamiento: se utilizan para organizar, manejar y combinar comportamientos.
- ✓ Patrones Creacionales: muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones normalmente serán resultas dinámicamente decidiendo que clases instanciar o sobre que objetos delegará responsabilidades un objeto determinado. Su razón de ser es facilitar, ordenar, o ayudar en la creación de objetos. (35)

Los patrones GoF que se utiliza en el desarrollo de la solución son:

- ✓ *Factory Method* (Método de fabricación): está dentro de la categoría de los patrones creacionales. Define una interfaz para crear un objeto dejando a las subclases decidir el tipo específico al que

pertenecen, ocultando al usuario final del código la decisión de que subclase instanciar. Posibilita que el código sea mucho más legible y escalable, y promueve el encapsulamiento de las partes más variables del sistema. La ventaja al utilizarlo es que se tiene centralizada la creación de los objetos en una clase que ha pedido, y devuelve instancias de los mismos. Además es una forma de tener control de la creación de objetos. (36)

Se utiliza para crear objetos de la interfaz *Interpreter.*

- ✓ Patrón Fachada: está dentro de la categoría de los patrones estructurales, se utiliza para proporcionar una interfaz unificada de alto nivel para un conjunto de clases en un subsistema, haciendo más fácil su uso y simplificando el acceso a dicho conjunto de clases. Además se utiliza para estructurar un entorno de programación, reduciendo su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre éstos y para desacoplar un sistema de sus clientes y otros subsistemas, haciéndolo más independiente, portable y reutilizable.

  Este patrón trae como consecuencia la reducción del acoplamiento entre clientes y subsistemas y el aislamiento de cambios en la implantación. Además facilita la división en capas y reduce dependencias de compilación. Su principal ventaja consiste en que para modificar las clases de los subsistemas, solo hay que realizar cambios en la interfaz o fachada, y los clientes pueden permanecer ajenos a ellos. (37)

  Por ejemplo la clase *IContractManager* proporciona una interfaz unificada la interfaz *IDALContract.*

3.2.2.2. Patrones GRASP.

Los patrones GRASP son patrones generales de software para asignación de responsabilidades. Aunque se considera que más que patrones, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. Estos patrones se dividen 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.

  En la solución existen clases con responsabilidades estrechamente relacionadas que no realizan un trabajo enorme ellas son, *ContratoController, EsquemaController.*
- ✓ Bajo acoplamiento: el acoplamiento es una medida de la fuerza con la que las clases están conectadas entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas,

se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. El grado de acoplamiento no puede considerarse aisladamente de otros principios como experto y alta cohesión.

Entre las clases que contienen poca dependencia de otras clases se puede mencionar *ContractManager*, esta clase depende de las clases *Contract* y *DALContract*.

- ✓ Controlador: un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Un defecto frecuente al diseñar controladores consiste en asignarles demasiada responsabilidad. Normalmente, un controlador debería delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad.

  En la solución cualquier evento que se genere en la *Vista de Contratos* será atendido por la clase *ContratoController*.

  En la solución cualquier evento que se genere en la *Vista de Esquemas* será atendido por la clase *EsquemaController.*

3.2.3. 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 relacionen con la aplicación, como los servicios y aplicaciones web, aplicaciones *Windows*, aplicaciones de base de datos y de servicios externos. Además, muestra las conexiones entre estas aplicaciones reflejando la actual configuración de la solución.

La siguiente figura muestra todos los componentes que de una manera u otra se relacionan con las aplicaciones web: Gestión de Contratos de Servicios y Autenticación, las cuales cargan el componente *CoreLib*, el mismo representa la capa del negocio de la arquitectura, es donde se encuentran todas las clases controladoras que permiten realizar todas las funcionalidades sobre las entidades las cuales están ubicadas en el componente Entidades con el cual se relaciona, al igual que con el componente *DAL* que representa la capa de acceso a datos, la misma interactúa con la base de datos *CouchDB*. Por otra parte la aplicación web Autenticación se relaciona directamente con el OpenLdap que es uno de los recursos donde va a estar distribuida la información referente a las identidades.

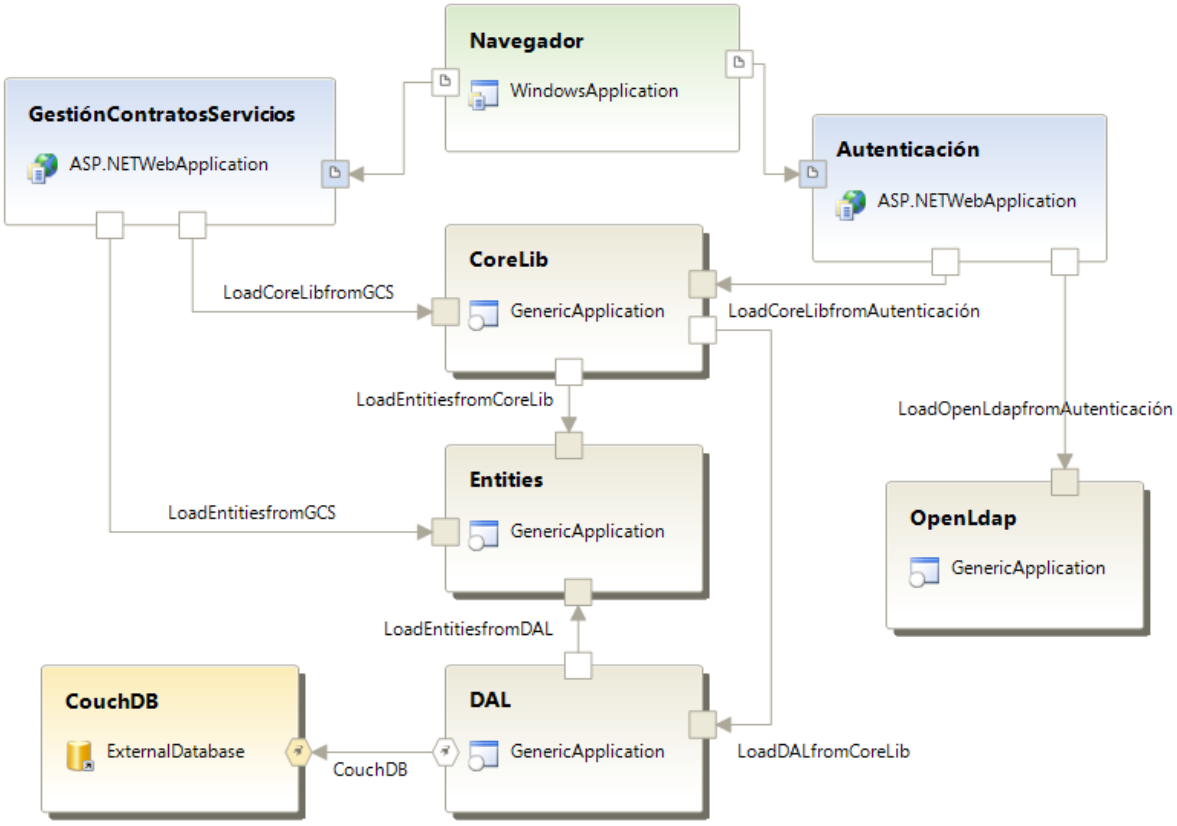

Figura 6: Diagrama de Aplicación.

3.2.4. Diagrama lógico de centro de datos.

El diagrama lógico de centro de datos define o documenta las configuraciones específicas de aplicaciones de tipo servidor, que tienen como propósito específico asegurar la conexión de entrada y salida del servidor web. El mismo modela como los servidores lógicos configurados están interconectados entre sí, los cuales pueden agrupar dentro de las zonas que definen los límites lógicos de comunicación.

En la figura que se muestra a continuación se puede observar como una Persona puede interactuar con la aplicación por medio del servidor web *Internet Information Server* a través del protocolo HTTPS, la aplicación web podrá a su vez comunicarse con la Base de Datos CouchDB usando el protocolo HTTP y con el OpenLdap por el TCP-IP.

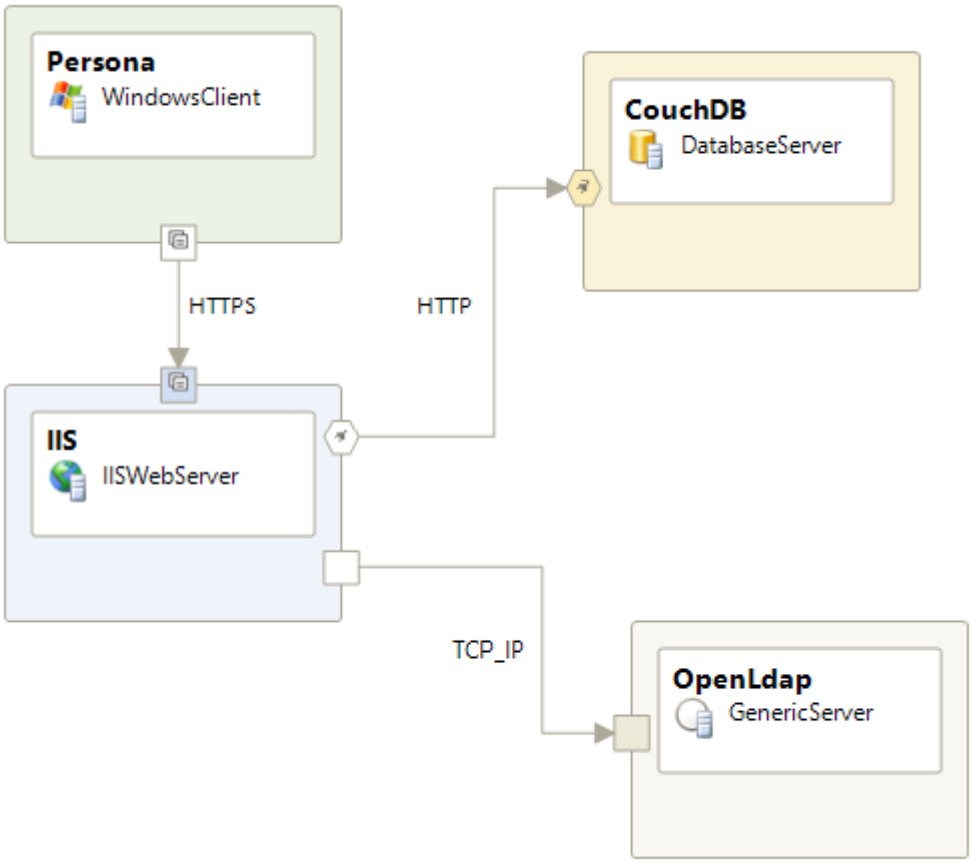

Figura 7: Diagrama de Centro de Datos Lógicos.

3.2.5. 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. Las clases del sistema están organizadas de una forma estructural adecuada según lo necesario en cada una de ellas, lo que las hace fáciles de manipular y permite que se interrelacionen entre ellas siempre que se necesite.

A continuación se representan las clases del módulo “Gestión de Contratos de Servicios” con sus atributos, sus métodos y sus relaciones.

Figura 8: Diagrama de Clases del Módulo "Gestión de Contratos de Servicios".

3.2.5.1. Descripción de clases.

Las clases controladoras son las encargadas de manipular las entidades del negocio y la información que se necesita que persista en la base de datos. Estas clases implementan interfaces que exponen las principales funcionalidades, posibilitando independencia entre las distintas capas y la escalabilidad del sistema.

Nombre	*ContractManager*.	
Tipo de Clase	Controladora.	
Descripción	Encargada de administrar los contrato de servicios.	
Atributo	**Tipo**	**Descripción**
idalcontract	IDALContract	Identificador del contrato de servicios.
Método	**Descripción**	
ContractManager()	Constructor de la Clase.	
ContractList():List<Contract>	Devuelve la lista de contratos.	
ContractById(string id):Contract	Devuelve el contrato que tiene como identificador el pasado por parámetro.	
ContractBySubject(string suject):Contract	Devuelve el contrato que tiene como sujeto el pasado por parámetro.	
ContractByScheme(string id): Contract	Devuelve el contrato que posee el esquema que tiene como identificador el pasado por parámetro.	
ContractLisbySuject(string subject): List<Contract>	Devuelve el listado de los contratos que tienen como sujeto el pasado por parámetro.	

Tabla 11: Descripción de la Clase Controladora "ContractManager".

Nombre	*Contract*.	
Tipo de Clase	Entidad.	
Descripción	Contiene todos los datos de los contratos de servicios.	
Atributo	**Tipo**	**Descripción**
Subject	string	Es el sujeto que establece el contrato de servicios.

Scheme	string	Esquema de autenticación.
_TYPE	const string	Constante para identificar el tipo de documento que es en la base de datos.
Método	**Descripción**	
Contract()	Constructor de la clase.	
Contract(string subject, string scheme)	Constructor de la clase.	
WriteJson(JsonWriter writer): void	Permite escribir en la base de datos.	
ReadJson(JObject obj): void	Permite leer de la base de datos.	

Tabla 12: Descripción de la Clase Entidad "Contract".

Las descripciones de las restantes clases pertenecientes al módulo "Gestión de Contratos de Servicios" y las del módulo "Autenticación" se puede ver en el Anexo II.

3.2.5.2. Descripción de clases persistentes.

Las clases persistentes en una aplicación implementan las entidades del problema de negocio. No todas las instancias de una clase persistente se considera que estén en el estado persistente, una instancia puede en cambio ser transitoria o estar separada. Las descripciones de las clases persistentes se pueden ver en el Anexo II.

3.2.6. Interfaz gráfica.

- ✓ Interfaz "Servicio de Autenticación": permite la autenticación del usuario teniendo en cuenta el esquema de autenticación definido en el contrato de servicios. En la siguiente figura se muestra un formulario para la autenticación con el esquema de compuesto por usuario, contraseña y email.

Figura 9: Interfaz "Servicio de Autenticación".

- ✓ Interfaz “Gestionar Contrato de Servicios”: permite administrar todas las acciones que se realizan sobre los contratos de servicios existentes en el sistema. Cuenta con las opciones crear, editar, eliminar, y mostrar contrato de servicios, además de permitir refrescar el listado.

Figura 10: Interfaz “Gestionar Contrato de Servicios”.

Las descripciones de las restantes interfaces se pueden ver en el Anexo III.

3.3. Fase de Estabilización.

La prueba del software es un conjunto de actividades que se lleva a cabo sistemáticamente, que puede planificarse por adelantado y ejecutarse una vez construido el código para la revisión final de las especificaciones, del diseño y de la codificación del *software*. Dentro de cada una de las etapas de desarrollo de un *software* las pruebas son fundamentales ya que a partir de ellas es posible controlar que los productos cumplan requisitos mínimos de operatividad además de garantizar la calidad de estos productos. Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.

3.3.1. Pruebas de Validación del Sistema.

3.3.1.1. Pruebas de Caja Negra.

Las pruebas de caja negra se refieren a las pruebas que se llevan a cabo sobre la interfaz del *software*, es decir que examinan algunos aspectos del modelo fundamental sin valorar demasiado la estructura lógica interna del sistema. (38)

3.3.1.2. Descripción de Casos de Prueba. Pruebas de Caja Negra.

Los casos de pruebas pretenden demostrar que las funciones del producto son operativas, que la entrada se realiza de manera adecuada, que se produce un resultado correcto y que la integridad de la información externa se mantiene.

Caso de Prueba: Gestionar Contrato de Servicios.

Escenario	Descripción	Texto	Respuesta del Sistema	Flujo Central
Esc.1: Crear nuevo contrato de servicio.	Se crea un nuevo contrato de servicio entre el sujeto y el Servidor de Autenticación o IdP.	Esquema (Válido) Nombre del Sujeto (Válido)	El sistema crea el contrato correctamente y actualiza el listado de los contratos.	Se crea o selecciona el esquema de autenticación y se construye un contrato con la información. Se establece una relación de confianza entre el sujeto y Servidor de Autenticación o IdP.
		Esquema (Inválido) Nombre del Sujeto (Inválido)	El sistema muestra un mensaje de aviso indicando que el sujeto ya existe y no activa el botón Aceptar.	
		Esquema (Inválidos) Nombre del Sujeto (Válido)	El sistema muestra un mensaje de aviso indicando que se introdujeron datos incorrectos del contrato.	
		Esquema (Válidos) Nombre del Sujeto (Inválido)	El sistema muestra un mensaje de aviso informando que el sujeto ya existe.	
Esc.2: Modificar	Se modifica el contrato de	Esquema (válido)	Se modifica el contrato.	Se selecciona el contrato, se modifica
		Esquema	Muestra un mensaje de aviso	

contrato de servicio.	servicio.	(Inválidos)	informando que la información no es válida.	con los nuevos parámetros y se hace persistente el contrato modificado.
Esc. 3: Eliminar contrato de servicio	Se elimina el contrato de servicio seleccionado.	Selección del contrato (Válido)	El sistema elimina el contrato seleccionado.	Del listado de contrato de servicio se selecciona el que se desea eliminar y se elimina.

Tabla 13: Descripción del caso de prueba "Gestionar Contrato de Servicios".

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Nombre del Esquema.	Lista desplegable.	No	Es el nombre que identifica al esquema de autenticación.
2	Nombre del Sujeto	Texto	No	Nombre de la parte que quiere establecer una relación de confianza con el Servidor de Autenticación o IdP.

Tabla 14: Descripción de las variables del caso de prueba "Gestionar Contrato de Servicios".

Las descripciones de los restantes casos de pruebas se pueden encontrar en el Anexo IV.

3.3.2. Pruebas Unitarias.

Las pruebas unitarias se utilizan para asegurar que cada módulo funcione, por separado y de manera correcta, fomentan el cambio y la refactorización. Se reducen drásticamente los problemas y el tiempo dedicado a la integración y se simulan las dependencias. Las pruebas unitarias se realizan a los servicios del sistema para validar que las salidas de los datos, le aseguran al programador que su solución no presenta errores lógicos de programación y ante una entrada de datos determinada por el probador los valores obtenidos son los esperados. (39)

3.3.2.1. Pruebas de Caja Blanca.

Las pruebas de caja blanca del software se basan en el examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de pruebas que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. (40)

En la siguiente figura y tabla se puede apreciar la prueba unitaria realizada al método *ContractById.*

```
/// <summary>
///A test for ContractById
///</summary>
[TestMethod()]
public void ContractByIdTest()
{
    string host = "seguridadserver.uci.cu"; // TODO: Initialize to an appropriate value
    int port = 5984; // TODO: Initialize to an appropriate value
    string dbName = "gyesbd_autenticacion"; // TODO: Initialize to an appropriate value
    DALContract target = new DALContract(host, port, dbName); // TODO: Initialize to an appropriate value
    string id = "5d26d2a8d84f526d1ccbed19a7065367"; // TODO: Initialize to an appropriate value
    Contract expected = new Contract() { Id = "5d26d2a8d84f526d1ccbed19a7065367" }; // TODO: Initialize to
    Contract actual;
    actual = target.ContractById(id);
    Assert.AreEqual(expected.Id.ToString(), actual.Id.ToString());
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 11: Prueba Unitaria al Método "ContractById".

Prueba de Unidad		
Nombre de la Prueba	*ContractById*.	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	03/06/2011
Ejecutado por	**Verificado por**	
Yenisey Calzada Rodríguez	Ernesto Miguel Muñoz.	
Descripción	Para poder ejecutar la prueba se debe pasar un tipo de dato *string* correspondiente al identificador del contrato, en caso de que exista un contrato con ese identificador, el método debe buscarlo y retornarlo, en caso contrario devuelve vacío.	
Entrada	*string* id	
Criterio de aceptación	Muestra un objeto del tipo contrato, con el identificador, el sujeto y el esquema del contrato.	

Resultado

Test Results

Alex@UCI-5FEF2795171 2011-06-03 | Run | Debug | Group

Test run completed Results: 1/1 passed; Item(s) checked: 0

Result	Test Name	Project	Error Message
Passed	ContractByIdTest	prueba	

Tabla 15: Prueba Unitaria al método "ContractById".

Las pruebas unitarias realizadas a los restantes métodos se pueden ver en el Anexos V.

Resultado de las pruebas.

Se realizaron tres iteraciones de pruebas con el objetivo de encontrar la mayor cantidad de no conformidades posibles y darle solución a las mismas. Durante la primera iteración fueron encontradas diez no conformidades, las cuales se corrigieron en esa etapa. Seguidamente se realizó la segunda iteración donde se comprobó que todos los errores de la iteración anterior habían sido rectificados, además esta iteración posibilitó encontrar 5 no conformidades que no habían sido detectadas. Posteriormente se llevó a cabo la tercera y última iteración en la cual fue encontrada otra no conformidad, con lo cual se evidencia la disminución de las mismas.

En la siguiente figura se muestra como se fueron corrigiendo las no conformidades detectadas en las iteraciones. Posteriormente se realizarán otras iteraciones para dar solución a las no conformidades aún presentes y garantizar el correcto funcionamiento de la aplicación.

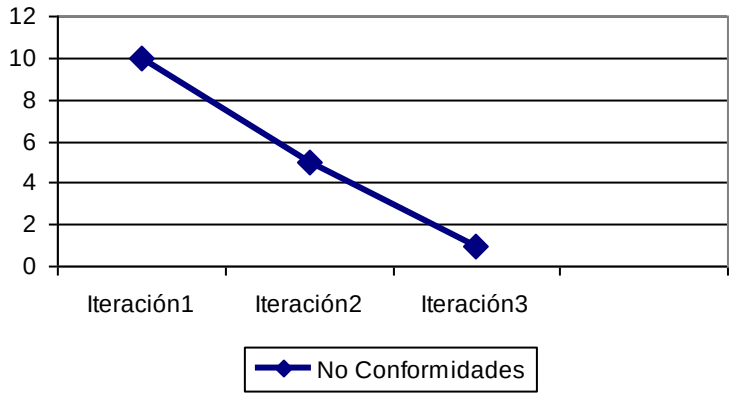

Figura 12: Resultado de las pruebas.

3.4. Conclusiones.

En este capítulo se definió la estructura básica del sistema que esta representada por una arquitectura de tres capas, que cuenta con el estilo arquitectónico Modelo-Vista-Controlador (MVC) en la capa de presentación, se identificaron los patrones de diseño que se utilizan en la implementación de la aplicación como son el Fachada, *Factory Method*, Bajo acoplamiento, Alta cohesión y el Creador posibilitando agilizar dicho proceso. Se modelaron y describieron las clases fundamentales, se identificaron todos los componentes que interactúan con la solución. Además se validó el sistema, realizándose algunas pruebas unitarias a las principales funcionalidades del mismo y se describieron y realizaron un conjunto de caso de prueba a la interfaz mediante el método de caja negra.

CONCLUSIONES GENERALES.

En el presente trabajo se le dio cumplimiento a los objetivos y tareas trazadas puesto que:

- ✓ Se seleccionaron los lenguajes, la metodología, herramientas y tecnologías más adecuados a utilizar en el producto.
- ✓ Se estudiaron los principales sistemas existentes similares tanto a nivel nacional como internacional.
- ✓ Se definió la arquitectura y los patrones de diseño presentes en la implementación.
- ✓ La aplicación se desarrolló en el período de tiempo establecido y cumple con todas las funcionalidades que se describieron.
- ✓ Se realizaron Pruebas de Caja Negra y Pruebas Unitarias para validar la propuesta.
- ✓ La solución obtenida, servirá de base para el desarrollo de nuevas aplicaciones que soporten la autenticación de usuarios por múltiples factores.
- ✓ La documentación generada servirá como base de estudio para futuros miembros del equipo de desarrollo del CISED.

El desarrollo de este trabajo permitió elaborar el Soporte extensible para modos de Autenticación del Sistema de Administración de Identidades Gyes del Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas.

RECOMENDACIONES.

Una vez concluido el presente trabajo de diploma se recomienda:

- ✓ Ampliar el modelo de negocio para que soporte múltiples esquemas en cada identidad.
- ✓ Hacer un instalador del Soporte Extensible para modos de Autenticación del Sistema de Administración de Identidades Gyes.

REFERENCIAS BIBLIOGRÁFICAS.

1. *Autenticación de usuarios.* 2008 Disponible en: http://www.rediris.es/cert/doc/unixsec/node14.html
2. **Nabeth, Thierry y Hildebrandt, Mireille.** *Inventory of topics and clusters.* 2010.
3. 2009. Disponible en: http://enciclopediavirus.com
4. *Hacia un meta-sistema para la gestión de la identidad y la privacidad.* 2009. Disponible en: https://www.cenitsegura.com/cenit/index.php?option=com_content&task=view&id=229&Itemid=124
5. **Roberto Quiñones Bondartchuk.** *Sistema de Administración de Identidades.* 2010
6. *Autenticación.* Disponible en: http://msdn.microsoft.com/es-es/library/syf5yeat.aspx
7. **Gustavo A. Gutiérrez.** *Autenticación, autorización y Ataque y contramedia.* 2006. Disponible en: http://www.fcca.umich.mx/Apuntes/Apuntes/Academia%20de%20Informatica/Redes%20de%20Computo%20%20G.A.G.C/Unidad-3-4-5.pdf
8. **Stéphane Vinsot.** *Los siete métodos de autenticación más utilizados.* 2009. Disponible en: http://www.evidian.com/evidian/contacts.php&c=lbstrauth
9. *Single Sing-On.* 2009. Disponible en: http://cetmar.com.ar/openid-y-cetmarweb/
10. *IBM Tivoli Access Manager for e-Business.* 2010.
11. 2010. Disponible en: www.oracle.com/identity
12. **Imobach González Sosa, Manolo Padrón Martínez.** *Pluggable Authentication Modules (PAM)*. 2007. Disponible en: http://sopa.dis.ulpgc.es/ii-aso/portal_aso/leclinux/seguridad/pam/pam_doc.pdf.
13. *TrustedX Plataforma de servicios de confianza.* 2006. Disponible en. http://www.safelayer.com/pdf/TrustedX_es.pdf.
14. **MSc. Oiner Gómez Baryolo.** ACAXIA: Solución Informática para la Gestión Centralizada de la Seguridad en Entornos Multientidad y Multisistemas. 2011.UCI
15. Presentación de metodología MSF(Microsoft Solution Framework). 2007. Disponible en: www.e-Gattaca.com
16. **Joaquín García.** *Desarrollo de Software Orientado a Objetos.* 2005. Disponible en: http://www.ingenierosoftware.com/analisisydiseno/uml.php
17. *Altova UModel2010*. Disponible en: http://mscerts.programming4.us/es/626432.aspx
18. *Visual C#.* 2009. Disponible en: http://msdn.microsoft.com/es-es/vcsharp/default.aspx.

19. **Damián Valdés Pérez.** *Maestros del Web.* 2005. Disponible en: http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/
20. **Javier Eguíluz Pérez.** *Introducción a CSS.* Año: 2008 Páginas: 223. Disponible en: http://www.librosweb.es/css/
21. *What is Microsoft .NET Framework?* 2009**.** Disponible en: http://www.microsoft.com/NET/.
22. Año: 2010. Disponible en: http://www.microsoft.com/spain/visualstudio/products/2010-editions/ultimate
23. **Alejandro Ramírez.** 2004. Disponible en: http://polaris.dit.upm.es/~rubentb/docs/subversion/TutorialSubversion/index.html#N10047
24. **John Brown.** *Características de PostgreSQL.* 2007. Disponible en: http://www.manualesdeayuda.com/manuales/bases-de-datos/postgresql/caracteristicas-de-postgresql-01844.html
25. **Manuel J. Carrascosa de la Blanca.** *MongoDB: Un paradigma alternativo para base de datos.* 2011. Disponible en: http://mjcarrascosa.com/mongodb-un-paradigma-alternativo-para-bases-de-datos/
26. **Jan Lebnardt.** *Couchdb Definitive Guide.*
27. *ASP.NET*. Disponible en: http://msdn.microsoft.com/es-es/asp.net/aa336522
28. 2009.Disponible en: http://www.ajaxya.com.ar/temarios/descripcion.php?cod=8&punto=1
29. 2008. Disponible en: http://www.programacionweb.net/cursos/curso.php?num=2
30. 2010. Disponible en:http://www.ajaxya.com.ar/temarios/descripcion.php?cod=34&punto=2
31. **Scott Guthrie.** 2007. Disponible en: http://weblogs.asp.net/scottgu/archive/2007/10/14/asp-net-mvc-framework.aspx.
32. 2010. Disponible en: https://github.com/foretagsplatsen/Divan#readme
33. *Arquitectura y Patrones de diseño*. 2010. Disponible en: http://eva.uci.cu.
34. **Ricardo Ánge Febe.** 2008.Disponible en: http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista2.shtml.
35. *WorldLingo.* 2004. Disponible en: http://www.worldlingo.com/ma/enwiki/es/Design_pattern_%28computer_science%29.
36. 2007. Disponible en: http://www.lawebdelprogramador.com/codigo/1719/Patron_Factory.html.
37. *Patrón Fachada.* 2006. Disponible en: http://my.opera.com/oscarmendez/blog/2006/10/15/el-patron-fachada.

38. **Johanna Rojas, Emilio Barrios.** *Métodos de prueba de caja negra.* 2007. Disponible en: http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html

39. *Trabajar con pruebas unitarias.* 2007.Disponible en: http://msdn.microsoft.com/es-es/library/ms182515%28VS.80%29.aspx.

40. *White and Black Box Testing*. 2006. Disponible en. https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/white-box/259-BSI.html.

BIBLIOGRAFÍA.

1. **Erich Gamma, R.H., Ralph Johnson, John Vlissedes.** *Patrones del diseño: Elementos del software orientado al objeto reutilizable.* 2002.
2. **Grady Booch, J.R, Ivar Jacobson**. *UML - El Lenguaje Unificado de Modelado.* 1999.
3. **Larman, C**. *UML y patrones.* 2003.
4. **Presman, R. S**. *Ingeniería del Software. Un enfoque práctico. (Sexta edición).* 2005.
5. **Dag-Erling Smørgrav.** *Pluggable Authentication Modules.*
6. **Jesse James Garrett**. *Ajax: A New Approach to Web Applications*. 2005
7. **Javier Eguíluz Pérez** *Introducción a AJAX*.
8. **Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato.** *Version Control with Subversion.* Disponible en: http://svnbook.red-bean.com/nightly/en/svn-book.pdf
9. **Tony Tomov.** *jqGrind.* 2009

GLOSARIO DE TÉRMINOS.

ASP: 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: Conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente. En una base de datos, la información se organiza en campos y registros.

BizTalk: Herramienta utilizada para conectar varios sistemas diferentes a través de un sistema de mensajes y orquestación basado en XML.

CISED: Centro de Identificación y Seguridad Digital.

CLR: Por sus siglas en inglés *Common Language Runtime*.

CVS: Por sus siglas en inglés *Concurrent Versions System*.

Dongle Criptográfico: *Pen Drive Security,* llave electrónica, candado electrónico o seguro electrónico es un pequeño dispositivo de hardware que se puede integrar a un programa y se conecta a un ordenador, normalmente, para autenticar un fragmento de software.

Las fases de MSF Ágil son grupos de actividades que llevan a cada uno de los puntos de control que guían el desarrollo, es decir, abordar las cuestiones que se refieren a gastos de tiempo y dinero ya que representan el proceso en tiempo. Las fases no se refieren necesariamente a la ejecución de las tareas o la división del trabajo que se manejan en los ciclos. Tenga en cuenta que las fases se superponen considerablemente y que existe una retroalimentación continua entre las actividades de diferentes fases.

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.

GRASP: Patrones de asignación de responsabilidad.

GoF: Por sus siglas en inglés *Gang of Four*.

HTML: Lenguaje de Marcado de Hipertexto.

HTTP: Por sus siglas en inglés *Hypertext Transfer Protocol.*

JSON: acrónimo de *JavaScript Object Notation*, es un formato para el intercambio de datos. Subconjunto de la notación literal de objetos de *JavaScript* que no requiere el uso de XML.

JQuery: Es una librería basada en *JavaScript* para acceder a los objetos del DOM de un modo simplificado.

JIT: Por sus siglas en inglés *Just In Time*.

LDAP: Protocolo Ligero de Acceso a Directorios es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

MSF: *Microsoft Solution Framework* es un conjunto de ingeniería de *software* procesos, principios y prácticas probadas por objeto permitir a los desarrolladores para lograr el éxito en el desarrollo del ciclo de vida del *software*.

MVC: Estilo arquitectónico modelo-vista-controlador.

MVCC: Por sus siglas en inglés *Multiversion Concurrency Control.*

OASIS: *Organization for the Advancement of Structured Information Standards* por sus siglas en inglés, es un consorcio internacional sin fines de lucro que orienta el desarrollo, la convergencia y la adopción de los estándares de comercio electrónico y servicios web.

PDVSA: Petróleos de Venezuela S.A. es la corporación estatal de la República Bolivariana de Venezuela que se encarga de la exploración, producción, manufactura, transporte y mercadeo de los hidrocarburos.

RBAC: Control de Acceso basado en Roles.

SQL: *Structured Query Language* por sus siglas en inglés, en español lenguaje de consulta estructurado, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas.

***Single Sign-On* (SSO):** Es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación.

Servidor: Un servidor es una computadora que maneja peticiones de data, email, servicios de redes y transferencia de archivos de otras computadoras (clientes). También puede referirse a un software específico. Una computadora puede tener distintos software de servidor, proporcionando muchos servidores a clientes en la red.

SOAP: Protocolo simple de acceso a objetos. Se trata de un protocolo que permite la comunicación entre aplicaciones, a través de mensajes, por medio de Internet. Está basado en XML y es la base de los servicios web. Es independiente de la plataforma y del lenguaje.

SAML: por sus siglas en inglés *Security Assertion Markup Language.*

SAI: Sistema de Administración de Identidades.

SPML: por sus siglas en ingles *Services Provisioning Markup Language,* es un lenguaje basado en XML.

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

UML: Lenguaje Unificado de Modelado es el lenguaje de modelado de sistemas de software que se utiliza para especificar, visualizar, modificar, construir y documentar los artefactos que se obtienen durante el desarrollo.

Usuario: Persona que tiene una cuenta en una determinada computadora por medio de la cual puede acceder a los recursos y servicios que ofrece una red.

XML: Lenguaje de marcado extensible.

ANEXOS.

Anexo I: Descripción de las Tareas.

- ✓ **Especificación de tareas del Escenario: "Gestionar Contrato de Servicios".**

Nombre de la Tarea	Mostrar Contrato de Servicios.
Identificador	GCS1.3
Objetivo de la Tarea	Mostrar toda la información referente a un contrato de servicios determinado.
Persona	Administrador del sistema.
Iteración.	1ra
Prioridad.	Alta.
Complejidad.	1
Descripción.	El administrador accede a la aplicación. El sistema muestra un listado de todos los contratos de servicios existentes y brinda la posibilidad de seleccionar el contrato de servicios que se desea que se muestren los datos. El administrador realiza la selección y oprime el botón Detalles. Finalmente el sistema muestra todos los datos referentes al contrato de servicios seleccionado.
Validaciones.	

Tabla 16: Descripción de la Tarea "Mostrar Contrato de Servicios".

Nombre de la Tarea	Editar Contrato de Servicios.
Identificador	GCS1.2
Objetivo de la Tarea	Editar un contrato de servicios previamente creado.
Persona	Administrador del sistema.
Iteración.	1ra

Prioridad.	Alta.
Complejidad.	1
Descripción.	El administrador accede a la aplicación y esta le muestra en pantalla un listado de los contratos de servicios existentes. El administrador selecciona el contrato de servicios que desea editar y el sistema le permite realizar las modificaciones al contrato de servicios con los nuevos parámetros. Finalmente se hace persistente el contrato de servicios modificado.

Tabla 17: Descripción de la Tarea "Editar Contrato de Servicios".

- ✓ **Especificación de tareas del Escenario: "Gestionar Esquema de Autenticación".**

El escenario "Gestionar Esquema de Autenticación" se dividió en 4 tareas fundamentales, que proveerán las funcionalidades de crear, editar, mostrar y eliminar un esquema.

Nombre de la Tarea	Crear Esquema de Autenticación.
Identificador	CS2.1
Objetivo de la Tarea	Empaquetar y estandarizar las reglas de autenticación o los medios necesarios para que se pueda lograr una autenticación con éxito.
Persona	Administrador del sistema.
Iteración.	1ra
Prioridad.	Alta.
Complejidad.	1
Descripción.	El administrador accede al sistema con la intención de crear un esquema de autenticación el sistema le permite seleccionar los nombres de los parámetros que contendrá el esquema (usuario, contraseña, certificado, huella, entre otros) así como el lugar donde se encuentran los valores de estos parámetros (OpenLdap, MySQL, Ficheros, XML, etc.). El administrador realiza la selección e inserta estos datos en el esquema. Finalmente se hace el esquema persistente.
Validaciones.	✓ El esquema deberá poder ser usado en un contrato.

Tabla 18: Descripción de la Tarea "Crear Esquema de Autenticación".

Nombre de la Tarea	Mostrar Esquema de Autenticación.
Identificador	CS2.3
Objetivo de la Tarea	Mostrar toda la información referente a un esquema de autenticación determinado.
Persona	Administrador del sistema.
Iteración.	1ra
Prioridad.	Alta.
Complejidad.	1
Descripción.	El administrador accede a la aplicación. El sistema muestra un listado de todos los esquemas de autenticación existentes y brinda la posibilidad de seleccionar el esquema que se desea que se muestren los datos. El administrador realiza la selección y oprime el botón Detalles. Finalmente el sistema muestra todos los datos referentes al esquema de autenticación seleccionado.
Validaciones.	

Tabla 19: Descripción de la Tarea "Mostrar Esquema de Autenticación".

Nombre de la Tarea	Eliminar Esquema de Autenticación.
Identificador	CS2.3
Objetivo de la Tarea	Eliminar el esquema de autenticación de la base de datos.
Persona	Administrador del sistema.
Iteración.	1ra
Prioridad.	Alta.
Complejidad.	1
Descripción.	El administrador accede a la aplicación con la intención de eliminar un esquema de autenticación. El sistema le muestra un listado de todos los esquemas de autenticación existentes y brinda la posibilidad de seleccionar el esquema de

	autenticación que se desea eliminar. El administrador realiza la selección y oprime el botón Eliminar. Finalmente el sistema elimina el esquema de autenticación de la base de datos.
Validaciones.	✓ El esquema no puede pertenecer a ningún contrato para ser eliminado.

Tabla 20: Descripción de la Tarea "Eliminar Esquema de Autenticación".

Anexo II: Descripción de clases.

En las tablas que se muestran a continuación, se describen las clases controladoras pertenecientes al módulo "Gestión de Contratos de Servicios".

Nombre	*SchemeManager*	
Tipo de Clase	Controladora.	
Descripción	Encargada de administrar los esquemas de autenticación.	
Atributo	**Tipo**	**Descripción**
IdalEschema	IDALScheme	Identificador del esquema de autenticación.
Método	**Descripción**	
SchemeManager()	Constructor de la Clase.	
SchemeList(): List<Scheme>	Devuelve una lista de los esquemas de autenticación.	
SchemeByName(string name): Scheme	Devuelve el esquema que tiene como nombre el pasado por parámetro.	
SchemeById(string id): Scheme	Devuelve el esquema que tiene como identificador el pasado por parámetro.	

Tabla 21: Descripción de la Clase Controladora "SchemeManager".

Nombre	*Scheme*.	
Tipo de Clase	Entidad.	
Descripción	Contiene todos los datos de los esquemas de autenticación.	
Atributo	**Tipo**	**Descripción**

name	string	Nombre del esquema de autenticación.
parameters	NameValueCollection	Colección de valores de tipo llave-valor. Almacena la colección de parámetros de autenticación.
_TYPE	const string	Constante para identificar el tipo de documento que es en la base de datos.
Métodos	**Descripción**	
Scheme()	Constructor de la clase.	
Scheme(string name, NameValueCollection parameters)	Constructor de la clase.	
WriteJson(JsonWriter writer): void	Permite escribir en la base de datos.	
ReadJson(JObject obj): void	Permite leer de la base de datos.	

Tabla 22: Descripción de la Clase Entidad "Scheme".

En las tablas que se muestran a continuación, se describen las clases pertenecientes al módulo "Autenticación".

Nombre	*Interpreter.*	
Tipo de Clase	Interface.	
Descripción	Interpreta la cadena donde se guarda toda la información referente a la dirección donde esta guardada dicha información.	
Atributo	**Tipo**	**Descripción**
Type	string	Tipo de intérprete que se va a utilizar, puede ser OpenLdapInterpreter.
Value	string	Es el valor del Número de Activación de la PC que quiere autenticarse en el Subsistema de Autenticación.
Métodos	**Descripción**	
Valid(string value): bool	Valida que el valor del Número de Activación de la PC.	

Tabla 23: Descripción de la Clase "Interpreter".

Nombre	OpenLdapInterpreter.	
Descripción	Encargada de separar todos los parámetros que se encuentran en la dirección.	
Atributo	**Tipo**	**Descripción**
str	string	Dirección completa donde se encuentran todos los valores de los parámetros definidos en el Contrato de Servicios.
userName	string	Parámetro usuario.
password	string	Parámetro contraseña.
server	string	Servidor.
port	int	Puerto.
Métodos	**Descripción**	
OpenLdapInterpreter(string srtStrip)	Constructor de la clase.	

Tabla 24: Descripción de la Clase "OpenLdapInterpreter".

En las tablas que se muestran a continuación, se describen las clases persistentes.

Nombre	*Scheme.*	
Descripción	Contiene todos los datos correspondientes a los esquemas de autenticación.	
Atributo	**Tipo**	**Descripción**
id	string	Identificador del esquema de autenticación, se genera dinámicamente.
rev	string	Versión del documento.
name	string	Nombre del esquema de autenticación.
parameters	NameValueCollection	Colección de parámetros del esquema de autenticación.
type	string	Constante para identificar el tipo de documento en la base de datos

Tabla 25: Descripción de la Clase Persistente "Scheme".

Nombre	*Contract*.	
Descripción	Contiene todos los datos correspondientes a los contratos de servicios.	
Atributo	**Tipo**	**Descripción**
id	string	Identificador del contrato de servicios. Se genera dinámicamente.
rev	string	Versión del documento.
subject	string	Nombre del sujeto que desea establecer un contrato de servicios.
scheme	string	Nombre del esquema de autenticación que tiene el contrato de servicios.
type	string	Constante para identificar el tipo de documento en la base de datos

Tabla 26: Descripción de la Clase Persistente "Contract".

Anexo III: Descripción de las interfaces.

✓ **Descripción de las opciones de la interfaz: "Gestionar Contrato de Servicios".**

- Opción "Editar" de la interfaz "Gestionar Contrato de Servicios": permite modificar los datos del contrato seleccionado en el listado, actualizando la información del contrato de servicios.

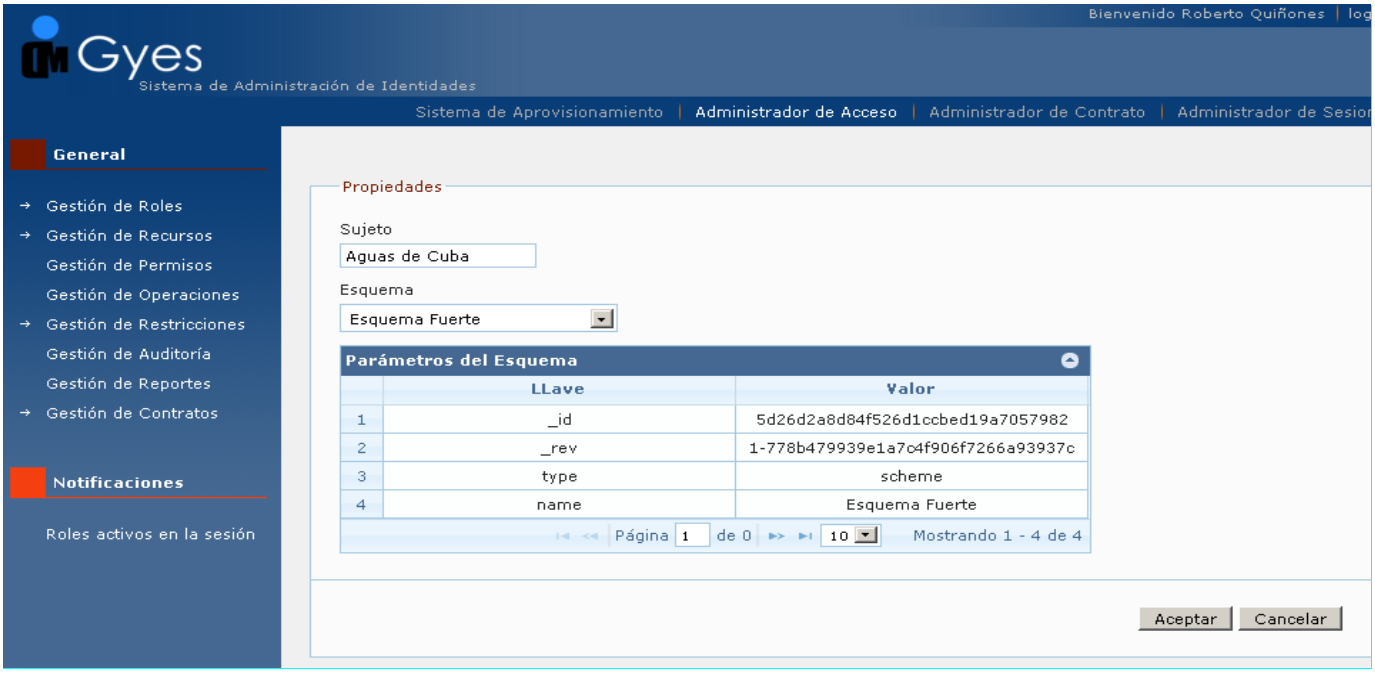

Figura 13: Interfaz de la opción "Editar Contrato de Servicios".

- Opción "Detalles" de la interfaz "Gestionar Contrato de Servicios": permite mostrar todos los datos del contrato seleccionado en el listado.

Figura 14: Interfaz de la opción "Detalles".

- Opción “Eliminar” de la interfaz “Gestionar Contrato de Servicios”: permite eliminar de la base de datos el contrato seleccionado en el listado y muestra la lista actualizada.

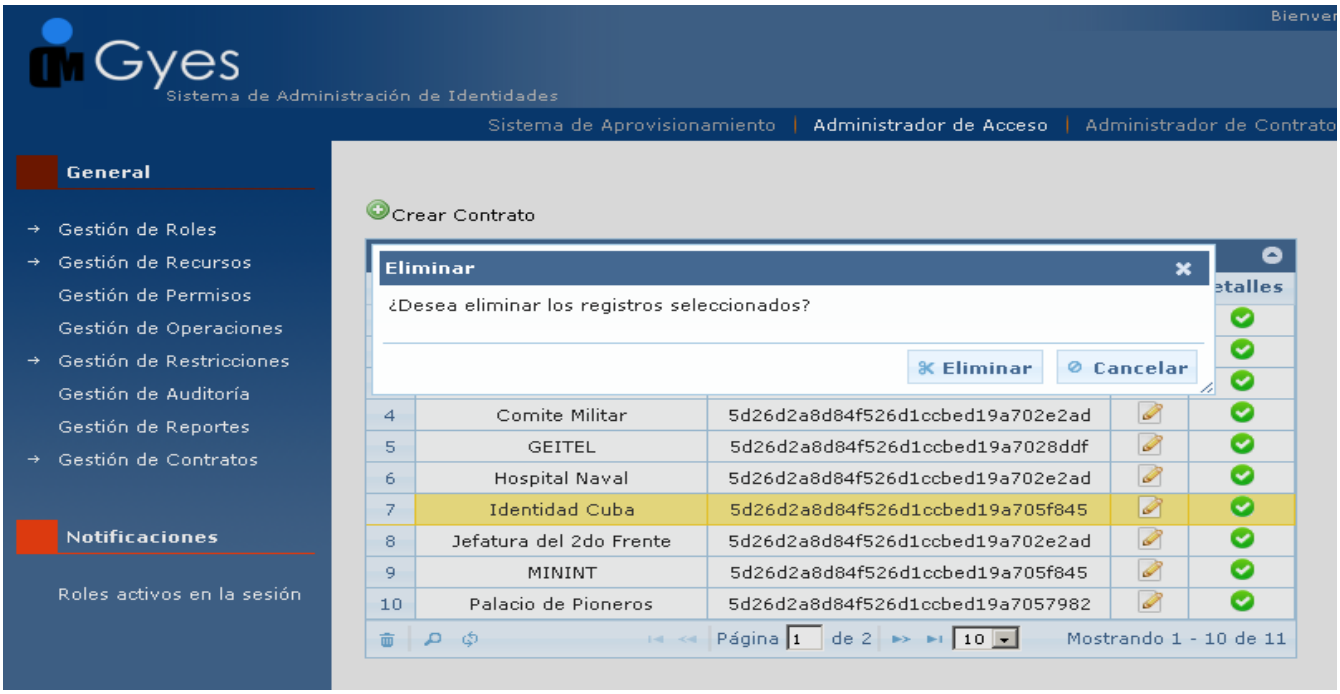

Figura 15: Interfaz de la opción “Eliminar”.

- Opción “Crear” de la interfaz “Gestionar Contrato de Servicios”.

La Opción “Crear” de la interfaz “Gestionar Contrato de Servicios” permite crear un nuevo contrato de servicios en la base de datos y muestra la lista actualizada.

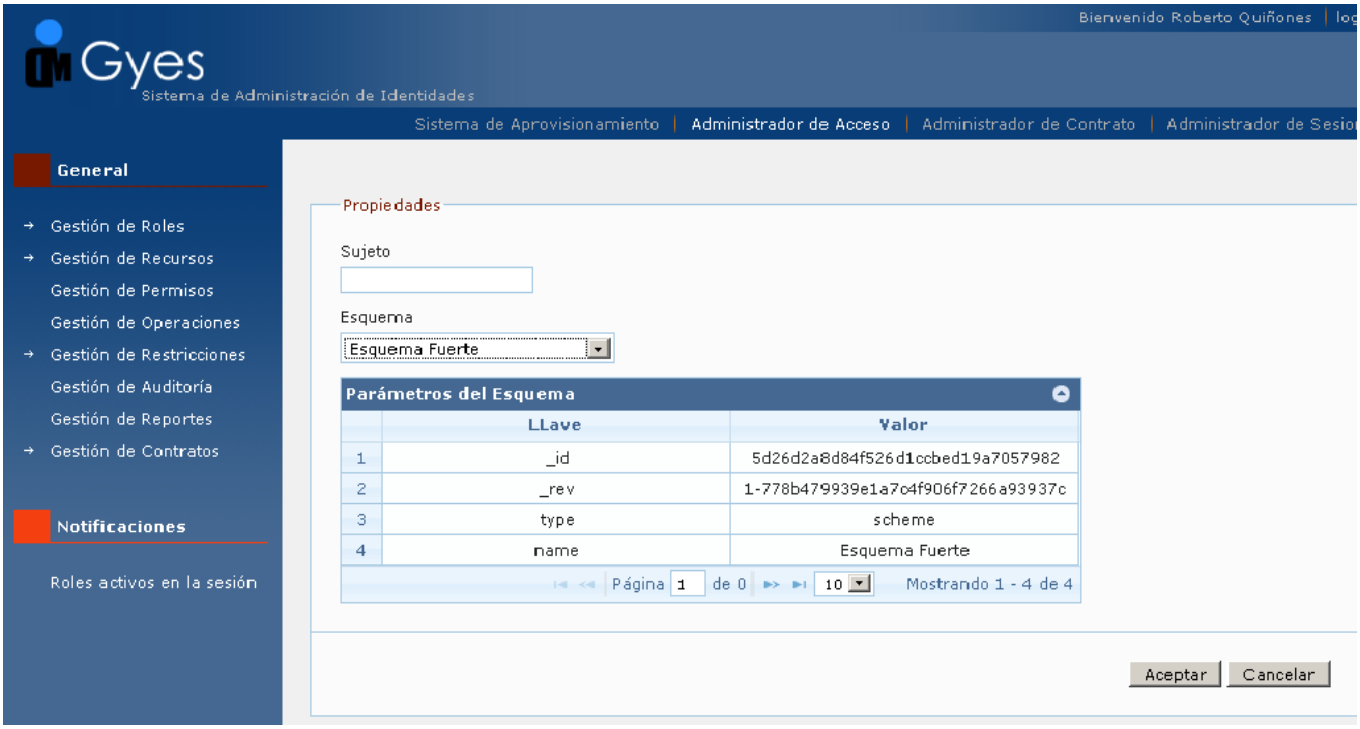

Figura 16: Interfaz de la opción "Crear Contrato de Servicios".

- ✓ **Descripción de las opciones de la interfaz: "Gestionar Esquema de Autenticación".**

La interfaz "Gestionar Esquema de Autenticación" permite administrar todas las acciones que se realizan sobre los esquemas de autenticación existentes en el sistema. Cuenta con las opciones crear, editar, eliminar, y mostrar esquema de autenticación, además de permitir refrescar el listado.

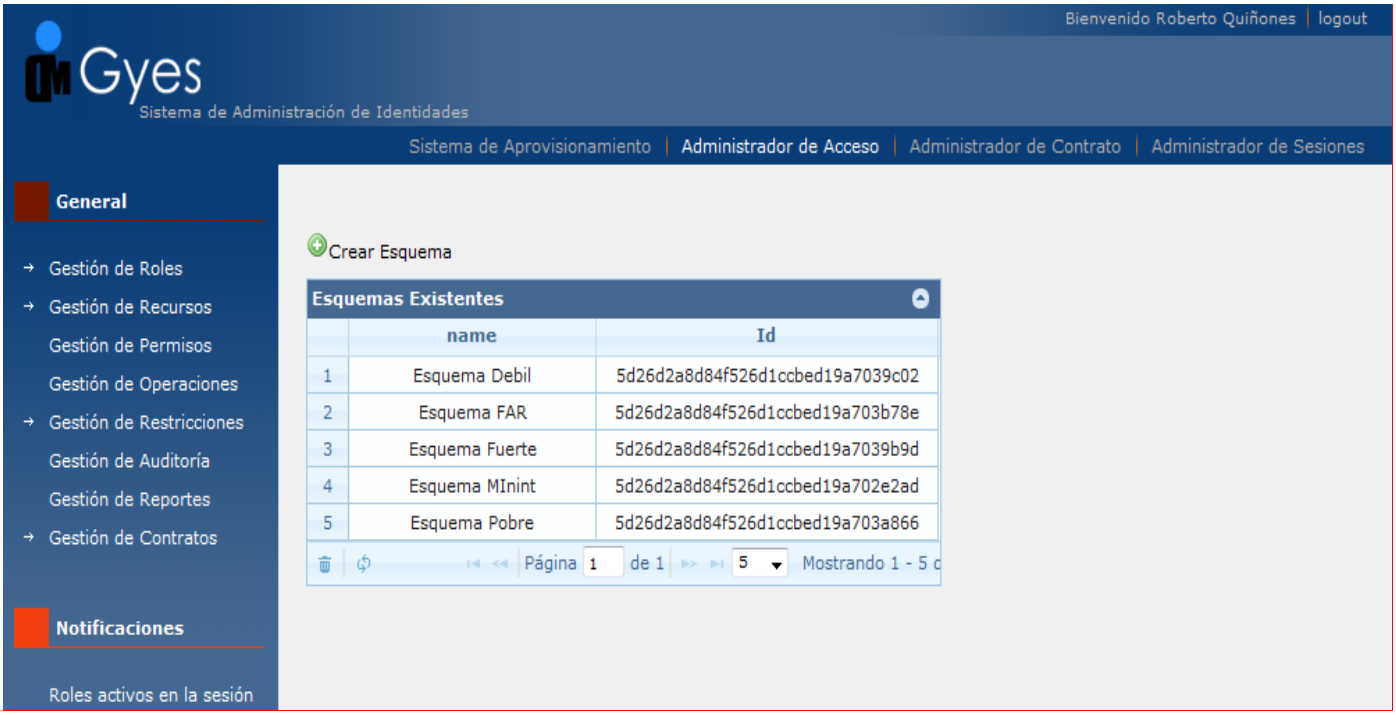

Figura 17: Interfaz "Gestionar Esquema de Autenticación".

- Opción "Eliminar" de la interfaz "Gestionar Esquema de Autenticación": permite eliminar de la base de datos el esquema seleccionado en el listado y muestra la lista actualizada.

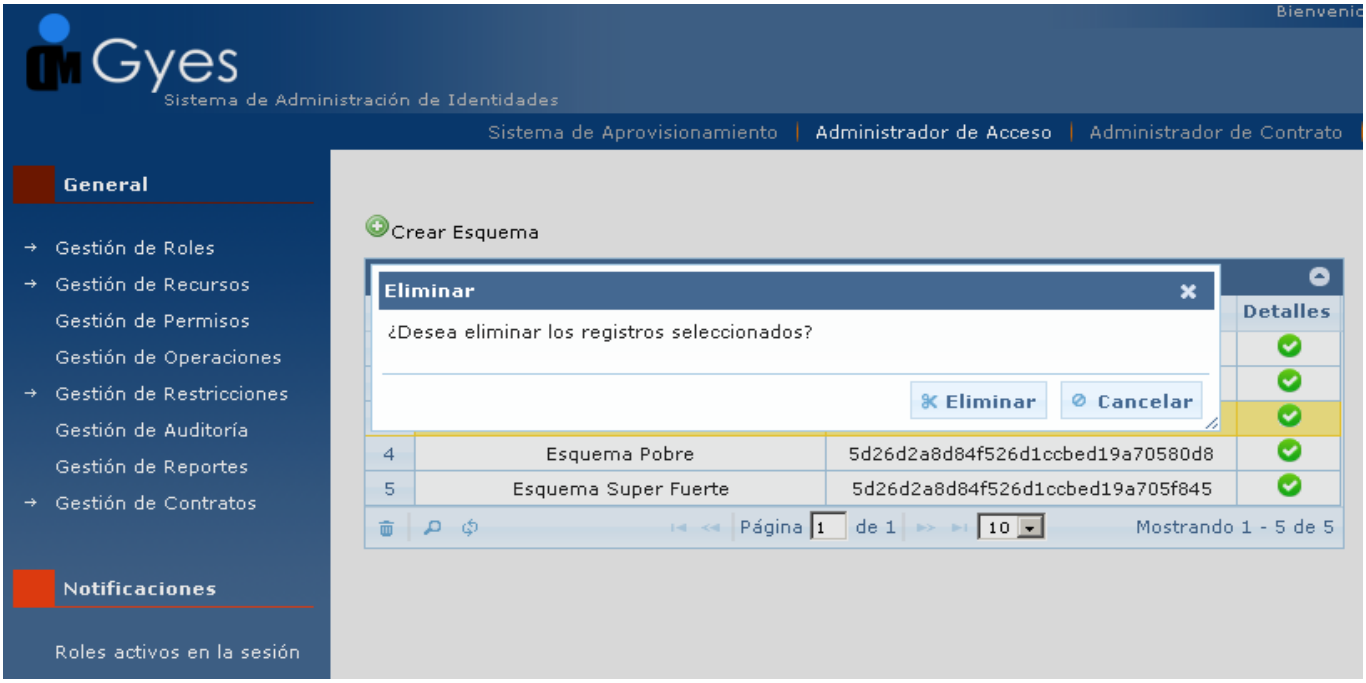

Figura 18: Interfaz de la opción "Eliminar".

- Opción "Crear" de la interfaz "Gestionar Esquema de Autenticación": permite crear un nuevo esquema de autenticación en la base de datos y muestra la lista actualizada.

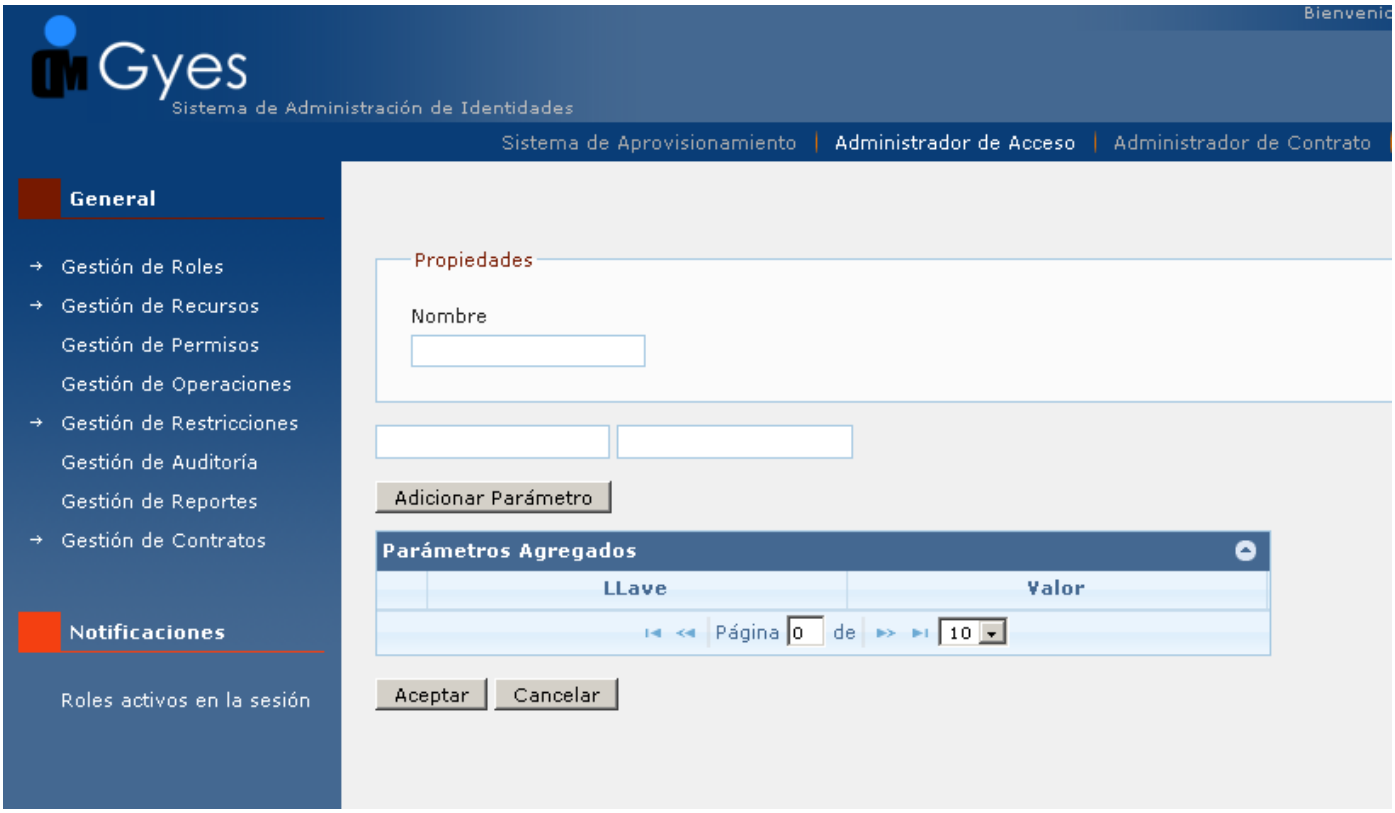

Figura 19: Interfaz de la opción "Crear Esquema de Autenticación".

Anexo IV: Descripción de Casos de Prueba.

Caso de Prueba: "Solicitar Inicio y Fin de Sesión Global."

Escenario	Descripción	Texto	Respuesta del Sistema	Flujo Central
Esc. 1: Solicitar inicio de sesión global.	Se da inicio a la sesión global una vez que se	Contrato de usuario (Válido)	Autenticación correcta para el usuario.	El sistema crea y envía una solicitud de autenticación al

	autentica el usuario.	Contrato de usuario (Inválido)	Muestra un mensaje de aviso informando que se entraron datos incorrectos.	servidor, es ejecutado el esquema de autenticación y se crea la sesión global.
Esc. 2: Solicitar fin de sesión global.	Se finaliza la sesión global del usuario.	Oprime el botón Logout (Válido)	Finalización correcta de la sesión global.	Se envía una solicitud al servidor y se elimina la sesión global.
		Se cierra el Internet Explorer (Inválido)	Se cierra el Internet Explorer pero no se finaliza la sesión global.	

Tabla 27: Descripción del caso de prueba "Solicitar Inicio y Fin de Sesión Global".

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Contrato de usuario	Puede ser cualquier combinación de modos de autenticación (usuario y contraseña o correo y contraseña)	No	Constituye una relación de confianza entre las partes que usen el servidor de autenticación.

Tabla 28: Descripción de las variables del caso de prueba "Solicitar Inicio y Fin de Sesión Global".

Caso de Prueba: "Crear y Eliminar Esquema de Autenticación."

Escenario	Descripción	Texto	Respuesta del Sistema	Flujo Central
Esc.1: Crear esquema de autenticación.	Se crea un esquema de autenticación.	Datos del esquema (Válido)	Se crea el esquema correctamente.	Se selecciona el esquema de autenticación y se

		Datos del esquema (Inválido)	Se muestra un mensaje de aviso informando que se han introducido datos incorrectos.	insertan los datos correspondientes.
Esc 2. Eliminar esquema de autenticación.	Se elimina un esquema de autenticación.	Esquema (Válido)	Se elimina el esquema seleccionado.	Se selecciona el esquema de autenticación y se elimina el esquema.
		Esquema (Invalido)	No se puede eliminar el esquema.	

Tabla 29: Descripción del caso de prueba "Crear y Eliminar Esquema de Autenticación".

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Datos del esquema	Colección de tipo llaves-valor el cual va a describir el modo de autenticación que tiene el esquema.	No	

Tabla 30: Descripción las variables del caso de prueba "Crear y Eliminar Esquema de Autenticación".

Anexo V: Pruebas Unitarias.

```
/// <summary>
///A test for ContractBySubject
///</summary>
[TestMethod()]
public void ContractBySubjectTest()
{
    string host = "seguridadserver.uci.cu"; // TODO: Initialize to an appropriate value
    int port = 5984; // TODO: Initialize to an appropriate value
    string dbName = "gyesbd_autenticacion"; // TODO: Initialize to an appropriate value
    DALContract target = new DALContract(host, port, dbName); // TODO: Initialize to an appropriate value
    string subject = "Poder Popular"; // TODO: Initialize to an appropriate value
    Contract expected = new Contract("Poder Popular","5d26d2a8d84f526d1ccbed19a70580d8"); // TODO: Initiali
    Contract actual;
    actual = target.ContractBySubject(subject);
    Assert.AreEqual(expected.Subject.ToString(), actual.Subject.ToString());
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 20: Prueba Unitaria al Método "ContractBySubject".

Prueba de Unidad		
Nombre de la Prueba	*ContractBySubject*.	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	03/06/2011
Ejecutado por	**Verificado por**	
Yenisey Calzada Rodríguez	Ernesto Miguel Muñoz.	
Descripción	Para poder ejecutar la prueba se debe pasar un tipo de dato *string* correspondiente al sujeto del contrato, en caso de que exista un contrato con ese sujeto, el método debe buscarlo y retornarlo, en caso contrario devuelve vacío.	
Entrada	*string* subject	
Criterio de aceptación	Muestra un objeto del tipo contrato, con el identificador, el sujeto y el esquema del contrato.	
Resultado		

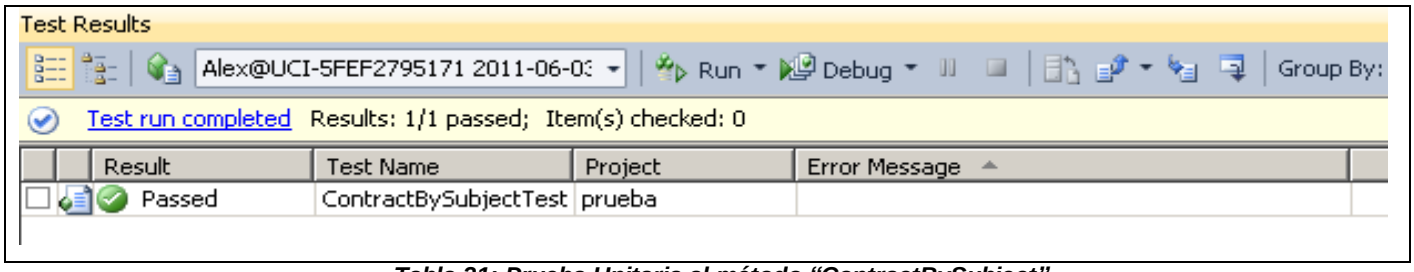

Tabla 31: Prueba Unitaria al método "ContractBySubject".

```
/// <summary>
///A test for ContractByScheme
///</summary>
[TestMethod()]
public void ContractBySchemeTest()
{
    string host = "seguridadserver.uci.cu"; // TODO: Initialize to an appropriate value
    int port = 5984; // TODO: Initialize to an appropriate value
    string dbName = "gyesbd_autenticacion"; // TODO: Initialize to an appropriate value
    DALContract target = new DALContract(host, port, dbName); // TODO: Initialize to an appropriate value
    string id = "5d26d2a8d84f526d1ccbed19a70580d8"; // TODO: Initialize to an appropriate value
    Contract expected = new Contract() { Scheme = "5d26d2a8d84f526d1ccbed19a70580d8" }; // TODO: Initializ
    Contract actual;
    actual = target.ContractByScheme(id);
    Assert.AreEqual(expected.Scheme.ToString(), actual.Scheme.ToString());
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 21: Prueba Unitaria al método "SchemeByName".

Prueba de Unidad		
Nombre de la Prueba	*ContractByScheme*.	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	03/06/2011
Ejecutado por	**Verificado por**	
Yenisey Calzada Rodríguez	Ernesto Miguel Muñoz.	
Descripción	Para poder ejecutar la prueba se debe pasar un tipo de dato string correspondiente al identificador del esquema que tiene un contrato, en caso de que exista un contrato con el esquema con ese identificador, el método debe buscarlo y retornarlo, en caso contrario devuelve vacío.	

Entrada	*String id.*
Criterio de aceptación	Muestra un objeto del tipo contrato, con el identificador, el esquema y el sujeto.

Resultado

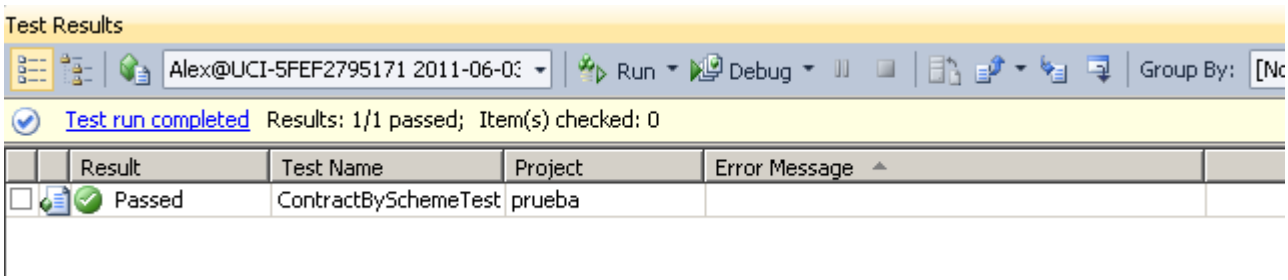

Tabla 32: Prueba Unitaria al método "ContractByScheme".

```
/// <summary>
///A test for SchemeById
///</summary>
[TestMethod()]
public void SchemeByIdTest()
{
    string host = "seguridadserver.uci.cu"; // TODO: Initialize to an appropriate value
    int port = 5984; // TODO: Initialize to an appropriate value
    string dbName = "gyesbd_autenticacion"; // TODO: Initialize to an appropriate value
    DALScheme target = new DALScheme(host, port, dbName); // TODO: Initialize to an appropriate value
    string id = "5d26d2a8d84f526d1ccbed19a70580d8"; // TODO: Initialize to an appropriate value
    Scheme expected = new Scheme() { Id = "5d26d2a8d84f526d1ccbed19a70580d8" }; // TODO: Initialize to a
    Scheme actual;
    actual = target.SchemeById(id);
    Assert.AreEqual(expected.Id.ToString(), actual.Id.ToString());
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 22: Prueba Unitaria al método "SchemeById".

Prueba de Unidad		
Nombre de la Prueba	*SchemeById*.	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	03/06/2011

Ejecutado por	Verificado por
Yenisey Calzada Rodríguez	Ernesto Miguel Muñoz.
Descripción	Para poder ejecutar la prueba se debe pasar un tipo de dato string correspondiente al identificador del esquema, en caso de que exista un esquema con ese identificador, el método debe buscarlo y retornarlo, en caso contrario devuelve vacío.
Entrada	*String id.*
Criterio de aceptación	Muestra un objeto del tipo esquema, con el identificador, el nombre y los parámetros.
Resultado	

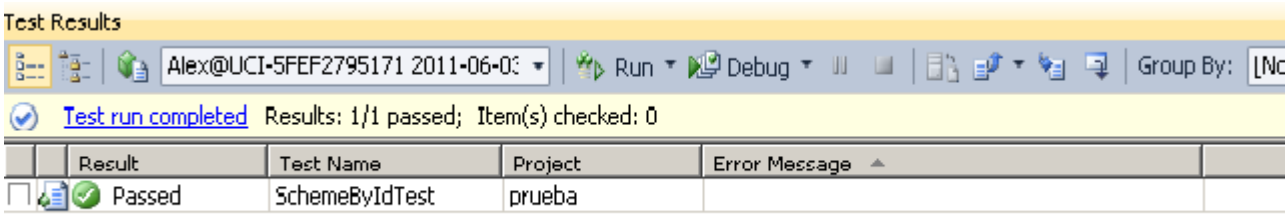

Tabla 33: Prueba Unitaria al método "SchemeById".

```
/// <summary>
///A test for SchemeByName
///</summary>
[TestMethod()]
public void SchemeByNameTest()
{
    string host = "seguridadserver.uci.cu"; // TODO: Initialize to an appropriate value
    int port = 5984; // TODO: Initialize to an appropriate value
    string dbName = "gyesbd_autenticacion"; // TODO: Initialize to an appropriate value
    DALScheme target = new DALScheme(host, port, dbName); // TODO: Initialize to an appropriate value
    string name = "Esquema"; // TODO: Initialize to an appropriate value
    Scheme expected = new Scheme() { name = "Esquema" }; // TODO: Initialize to an appropriate value
    Scheme actual;
    actual = target.SchemeByName(name);
    Assert.AreEqual(expected.name.ToString(), actual.name.ToString());
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 23: Prueba Unitaria al método "SchemeByName".

Prueba de Unidad		
Nombre de la Prueba	*SchemeByName*.	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	03/06/2011
Ejecutado por	**Verificado por**	
Yenisey Calzada Rodríguez	Ernesto Miguel Muñoz.	
Descripción	Para poder ejecutar la prueba se debe pasar un tipo de dato string correspondiente al nombre del contrato, en caso de que exista un esquema con ese nombre, el método debe buscarlo y retornarlo, en caso contrario devuelve vacío.	
Entrada	*String name.*	
Criterio de aceptación	Muestra un objeto del tipo esquema, con el identificador, el nombre y los parámetros.	

Resultado

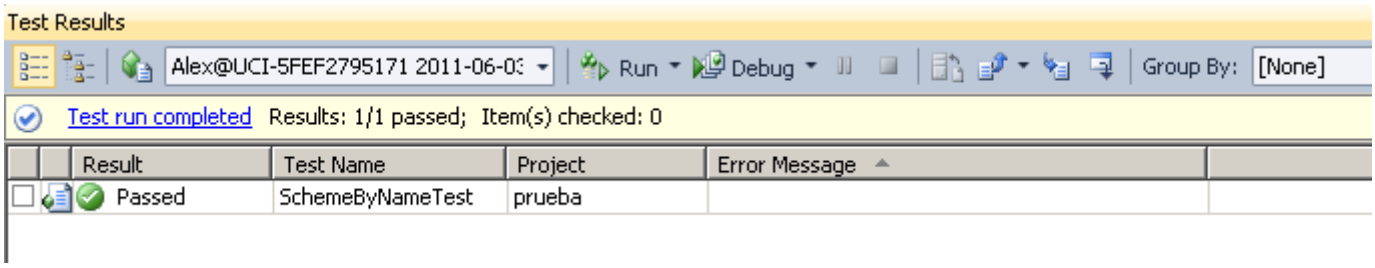

Tabla 34: Prueba Unitaria al método "SchemeByName".