Universidad de las Ciencias Informáticas
Facultad 1

Título:

Servicio de Autorización para el Sistema de Administración de Identidades.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Yusmisleidy Fernández Placeres

Angel Arias Baños

Tutor: Ing. Maikel de la Torre Luis

La Habana, Junio 24 de 2011

"Año 53 de la Revolución"


"El futuro de nuestra patria tiene que ser necesariamente,
un futuro de hombres de ciencia"

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 a hacer uso del mismo en su beneficio.

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

Autor: Yusmisleidy Fernández Placeres

Autor: Angel Arias Baños

Tutor: Ing. Maikel de la Torre Luis

DATOS DE CONTACTO

Maikel de la Torre Luis

Ingeniero Informático, Instructor Recién Graduado, un año de experiencia. Actualmente forma parte del grupo de desarrollo del Proyecto Seguridad Digital perteneciente al centro.

Correo electrónico: mdelatorre@uci.cu

La Habana, Cuba

Yusmisleidy Fernández Placeres

Correo electrónico: yfplaceres@estudiantes.uci.cu

La Habana, Cuba

Angel Arias Baños

Correo electrónico: abanos@estudiantes.uci.cu

La Habana, Cuba

AGRADECIMIENTOS

En primer lugar a mi madre por ser lo más especial de mi vida, por su apoyo incondicional, por estar conmigo en los momentos que más la necesito.

A mi hermana Anita que es mi tesoro y a mi padrino Rafael que ha estado siempre conmigo apoyándome.

A mi abuela Elsa. A mis tías Baby, Marlen y Ely, a mis primos que los quiero mucho. A mi papá.

A mi novio por estar siempre conmigo, apoyarme, comprenderme en los momentos de estrés y darme fuerzas para seguir adelante. A mis suegros que siempre estuvieron ahí, por acogerme y preocuparse por mí y a su familia por todo el cariño y apoyo que recibo.

A mis amigas Yare, Yirian, Lili, Lulu que las quiero mucho y fueron muy lindas conmigo estos cinco años. A mis amigos que forman parte de mí también, Jorge, Denier, Michel, Alberto, Reinier. A mi compañero de tesis por ser paciente. A mi tutor por ayudarme en todo momento, por ser tan entregado con lo que hace. A todos mis familiares que han dedicado empeño en mis estudios y me han apoyado con mucho amor. A todos mis amigos que desinteresadamente me han ayudado en toda mi carrera, Anabeisy, Yeni, Frank. Quisiera agradecer a mi abuela Nancy, que aunque hoy no se encuentra físicamente conmigo sé que donde quiera que esté me apoya y se enorgullece por mí.

A la Revolución, a Fidel, y a todos, gracias por permitir que se haga realidad mi sueño.

Yusmisledy Fernández Placeres

Agradecimientos

Después de tantos años de sacrificio y trabajo, por fin el sueño que tanto esperaba se hace realidad y se cumple la gran primera meta en mi vida, por lo que quiero agradecer especialmente:

A mi madre y mi padrastro por darme su apoyo en todo momento, y por ser los que más fuerzas me han dado para lograr mis objetivos.

A mi abuela por haber confiado en mí todos estos años.

En general a toda mi familia, a mi tíos José, Juan y Humberto, mis tías Zoila y Olga, y a todos mis primos.

A mi compañera de tesis por haberme soportado y coger tantos dolores de cabeza en este último año de la carrera.

A todos mi amigos de la UCI por compartir todos los momentos de alegría y tristeza durante los 5 años y a los que de una forma u otra me han ayudado y confiado en mí.

A todos los profesores que desde que ingrese en esta magnífica universidad han aportado su grano de arena para mi formación como profesional.

En general al grupo del proyecto, particularmente a mi tutor Maikel, por la preocupación constante que tenía por nosotros para que saliéramos bien, también a Roberto, Ruth, Yayneris, Diovis, en fin gracias a todos.

Angel Arias Baños

DEDICATORIA

- ✓ *A mis madre por ser mi mayor fuente de inspiración y porque es lo más grande en mi vida.*
- ✓ *A quienes dieron lo mejor de sí por hacer cumplir este día, a esas personas que sobrepusieron mis necesidades por encima de las suyas.*
- ✓ *A la Revolución cubana que me dio la posibilidad de ser la persona que soy, de permitirme estudiar y convertirme hoy en una profesional.*

Yusmisledy Fernández Placeres

- ✓ *A mi madre y mi padrastro por darme su apoyo incondicional durante todos estos años de carrera.*
- ✓ *A mí querida abuela que ha sido como una madre para mí, por darme su apoyado y haber confiado en mí.*
- ✓ *En especial a mi hermanita Lidi, de la cual espero siga mi ejemplo y se haga una profesional.*
- ✓ *En general a toda mi familia, particularmente a mi tía Zoila que se ha preocupado siempre por mí y brindarme su apoyo incondicional en todo.*

Angel Arias Baños

Resumen

RESUMEN

El presente trabajo de diploma se basa en el desarrollo de un Servicio de Autorización que permita controlar el acceso de los usuarios a los recursos de las aplicaciones conectadas al Sistema de Administración de Identidades, el cual se está desarrollando en el Centro de Identificación y Seguridad Digital (CISED). Actualmente este sistema no cuenta con un mecanismo que lleve a cabo este proceso de autorización.

La investigación estuvo enmarcada fundamentalmente en el estudio del proceso de autorización de usuarios en sistemas de administración de identidades existentes. Se realizó un estudio de las tendencias actuales y se utilizaron las herramientas y tecnologías establecidas por el proyecto, con el propósito de cumplir con los objetivos propuestos. Además, se realiza una descripción detallada de las características del servicio, así como la planificación, desarrollo y posteriormente las pruebas para la validación de la propuesta de solución.

PALABRAS CLAVE

Autorización, gestión de identidades, MSF ágil, RBAC, recurso, rol.

ÍNDICE

Índice

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Índice de tablas

Introducción

INTRODUCCIÓN

La seguridad en las aplicaciones informáticas es un tema que está muy vigente debido al crecimiento de los mercados, la piratería, delitos informáticos, el avance y enriquecimiento de la informática. Entre los mecanismos para garantizar los aspectos de la seguridad (confidencialidad, integridad y disponibilidad) se encuentra el control de acceso, donde se agrupan un conjunto de operaciones cuyo objetivo primordial consiste en controlar el acceso de los usuarios a los recursos. La tecnología utilizada para este control de acceso ha evolucionado con los propios sistemas de información protegidos. Hoy en día existen diversos modelos de seguridad donde se implementan políticas para el control de acceso en las aplicaciones, solucionando así los problemas de seguridad existentes.

El desarrollo informático en Cuba se incrementa progresivamente y con ello la necesidad de brindar una mayor seguridad a la información convirtiendo las debilidades en fortalezas. Es por esto que en la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Identificación y Seguridad Digital se encuentra el Departamento de Seguridad Digital, donde se está implementando un Sistema de Administración de Identidades que tiene como objetivo fundamental integrar la gestión de identidades, que se encarga de centralizar y administrar los procesos de la autenticación, autorización y aprovisionamiento de usuarios, en el cual se aglutinan las tareas requeridas para la creación, manejo y eliminación de identidades de usuarios en un ambiente computacional. En el proceso de autorización se verifican los permisos de acceso de los usuarios a los recursos basándose en mecanismos, modelos y políticas de control de acceso.

Actualmente en el subsistema de autorización del Sistema de Administración de Identidades no existe un mecanismo que realice este proceso, por lo que surge la necesidad de desarrollar un servicio de autorización para mantener un control adecuado del acceso de los usuarios sobre los recursos de las aplicaciones, evitando así que los usuarios accedan sin autorización y realicen operaciones que conlleven a la pérdida de información.

Introducción

Problema científico:

¿Cómo autorizar el acceso de los usuarios a los recursos de las aplicaciones controladas por el Sistema de Administración de Identidades?

Objeto de estudio:

Los procesos asociados con el control de acceso de los usuarios a los recursos.

Campo de acción:

Los procesos asociados con el control de acceso de los usuarios a los recursos en el Sistema de Administración de Identidades.

Para dar solución al problema existente, se ha determinado como **Objetivo general:**

Desarrollar un servicio de autorización para el Sistema de Administración de Identidades.
Además, se trazaron los siguientes **objetivos específicos:**

- ✓ Desarrollar el marco teórico de la investigación.
- ✓ Analizar las técnicas de programación, plataformas, herramientas y metodologías de desarrollo de software que faciliten la implementación del Servicio de Autorización.
- ✓ Realizar el diseño de la solución del Servicio de Autorización.
- ✓ Implementar el Servicio de Autorización.
- ✓ Realizar las pruebas para verificar el correcto funcionamiento del Servicio de Autorización.

Se plantea como **idea a defender:**

El desarrollo de un servicio de autorización para el Sistema de Administración de Identidades, permitirá controlar el acceso de los usuarios a los recursos de las aplicaciones.

Para cumplir con estos objetivos se trazaron las siguientes **tareas de investigación**:

- ✓ Análisis de los sistemas de administración de identidades existentes.

Introducción

- ✓ Análisis del proceso relacionado con la autorización en los sistemas de administración de identidades identificados.
- ✓ Revisión de modelos de control de acceso.
  - ❖ Modelo de control de acceso basado en roles: RBAC.
  - ❖ Modelo de control de acceso discrecional: DAC.
  - ❖ Modelo de control de acceso obligatorio: MAC
  - ❖ Modelo de control de acceso basado en tareas: TBAC
- ✓ Realización del estudio de la metodología ágil *Microsoft Solution Framework* (MSF).
- ✓ Revisión del *Framework Thrift* para el desarrollo de servicios.
- ✓ Realización del estudio de la tecnología *Windows Communication Foundation* (WCF).
- ✓ Realización del estudio de *Microsoft Enterprise Library* 5.0.
- ✓ Selección de la plataforma de desarrollo.
- ✓ Selección del servidor de base de datos.
- ✓ Elaboración de la propuesta de solución.
- ✓ Especificación y descripción de los escenarios.
- ✓ Realización del plan de iteraciones.
- ✓ Especificación de la arquitectura.
- ✓ Especificación de los patrones de diseño.
- ✓ Elaboración del diagrama de clases.
- ✓ Descripción de las clases.
- ✓ Modelado de la base de datos.
- ✓ Implementación de la librería de clases del Servicio de Autorización.
- ✓ Cifrado de la comunicación del Servicio de Autorización con los demás componentes o subsistemas del Sistema de Administración de Identidades.
- ✓ Ejecución de pruebas unitarias.
- ✓ Realización de pruebas de carga.

Introducción

Para llevar a cabo las tareas de investigación se utilizarán los siguientes **métodos de la investigación**:

Métodos teóricos: Histórico-lógico, Analítico-sintético y Modelado.

- ✓ **Histórico-lógico:** Para el estudio del estado del arte, para buscar y realizar un análisis previo de las ventajas y desventajas de cada una de las herramientas utilizadas para la elaboración del servicio de autorización.
- ✓ **Analítico-sintético:** Para realizar un análisis de teorías y documentos más significativos consultados durante el desarrollo de la investigación, permitiendo obtener una visión general del objeto de la investigación, partiendo de los elementos más importantes de cada uno de los aspectos esenciales, para buscar vías y solucionar los problemas que se generen.
- ✓ **Modelado:** Partiendo de todo lo investigado se realizan los modelos correspondientes a la metodología especificada para el servicio, que ayudarán a dar cumplimiento a las tareas de diseño de los procesos involucrados en la solución.

Métodos Empíricos: Observación.

- ✓ **Observación:** Permite analizar cada fase del proceso e ir observando cada tarea que se realice y tomar experiencia de esta para aplicarla en todas las demás, por lo que realizó una exhaustiva observación para ver que sucedía en el proceso de acceso de los usuarios a los recursos, identificando todos los problemas que se desean resolver.

Introducción

ESTRUCTURA DEL TRABAJO

Capítulo 1: Fundamentación teórica: Este capítulo es el respaldo teórico de los temas tratados en el informe, necesarios para el entendimiento correcto de la solución planteada. Se describen los conceptos fundamentales asociados al dominio del problema y el objeto de estudio, haciéndose un análisis de la situación actual. Además, se realizará el análisis de las herramientas, metodologías, lenguajes que existen en la actualidad y se definen las que serán utilizadas en el proceso de desarrollo del Servicio de Autorización.

Capítulo 2: Características del sistema: Se realizará la especificación de las funcionalidades del servicio de autorización. Además, se obtendrán los elementos necesarios, así como los requerimientos de calidad del servicio que se deben cumplir para la implementación del mismo.

Capítulo 3: Desarrollo y Pruebas de la solución propuesta: En este capítulo se establecerá la arquitectura y los patrones a utilizar para realizar la implementación del Servicio de Autorización. Se obtendrá el diseño del modelo de base de datos y el diagrama de clases. Además se ejecutará un conjunto de pruebas al mismo y se realizará un resumen de evaluación de estas.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

El acceso a la información ha cobrado una importancia inusitada en la actual sociedad. El tráfico global de información conlleva a generar soluciones y tecnologías que automaticen el acceso, mediante modelos y políticas de seguridad garantizando un ambiente seguro y confiable para las empresas. Las grandes organizaciones de hoy en día suelen tener procesos complejos y diseños poco satisfactorios en cuanto a la administración de las identidades, por lo que se hace necesaria la utilización de tecnologías y herramientas para obtener así un producto con calidad.

1.1. Definiciones

Se muestran a continuación las principales definiciones relacionadas con el dominio del problema que aportarán un breve conocimiento de todos los aspectos que se tratarán posteriormente.

1.1.1. Seguridad

La seguridad de sistemas o aplicaciones informáticas, son aquellas características que indiquen el índice de protección contra cualquier riesgo de agresión, resguardando la información y restringiendo el acceso al sistema de personal no autorizado.

La seguridad consta de tres aspectos fundamentales: la confidencialidad que se encarga de prevenir la divulgación de información a personas o sistemas que no estén autorizados a poseerla, la integridad que es la que mantiene los datos libres de modificaciones no autorizadas y la disponibilidad que es la que permite que la información se mantenga a disposición de quienes deben acceder a ella, ya sean personas, procesos o aplicaciones. (1) Estos tres aspectos permiten resguardar eficientemente la identidad digital de los usuarios de cualquier sistema informático.

1.1.2. Identidad digital

Identidad digital es aquel conjunto de rasgos propios de un individuo o colectividad que los caracterizan frente a los demás. La verificación de estos rasgos es lo que permite determinar que un individuo es quién dice ser. Algunos de estos rasgos son propios del individuo, otros son adquiridos con el tiempo. Por supuesto, no todos los rasgos son igualmente apreciables. Hay rasgos que son apreciables a simple vista, mientras que otros están ocultos y es necesario un conocimiento, en ocasiones herramientas para poder verificarlos. (2)

1.1.3. Gestión de identidades

La gestión de la identidad, es un área administrativa que se ocupa de la identificación de individuos en un sistema para cualquier organización, controlando el acceso a los recursos al imponer restricciones a las identidades establecidas.

Para cualquier empresa u organización, las soluciones de gestión de identidad constituyen la base para desarrollar nuevas oportunidades de negocio, disminuir los riesgos para la seguridad y reducir los costes de administración. (3)

En las empresas constantemente ocurren cambios con los empleados por diferentes circunstancias, lo que trae consigo actualizaciones de información continuamente, es aquí donde la gestión de identidades entra a jugar un papel significativo aportando capacidades a los usuarios, ya que se encarga de automatizar este proceso tan complejo.

1.1.4. Autenticación

La autenticación es el proceso de verificar de que el usuario que trata de identificarse es válido, usualmente se implementa con una contraseña en el momento de iniciar una sesión.

Existen cuatro tipos de técnicas que permiten realizar la autenticación de la identidad del usuario, las cuales pueden ser utilizadas individualmente o combinadas. (4)

Las cuatro técnicas son:

- ✓ Algo que solamente el individuo conoce: por ejemplo una contraseña.
- ✓ Algo que la persona posee: por ejemplo una tarjeta magnética.
- ✓ Algo que el individuo es y que lo identifica unívocamente: por ejemplo las huellas digitales.
- ✓ Algo que solo el individuo es capaz de hacer: por ejemplo los patrones de escritura.

1.1.5. Autorización

La autorización es el procedimiento para determinar si el usuario o proceso previamente identificado y autenticado tiene permitido el acceso a los recursos. La autenticación describe el proceso de verificar la identidad de una persona, mientras la autorización es el proceso de verificación de que una persona conocida tiene la autoridad para realizar una cierta operación. La autenticación, por lo tanto, debe preceder a la autorización usando atributos o derechos asociados con la identidad digital para determinar a qué recursos puede acceder dicha identidad digital. Se basa en mecanismos, modelos y políticas de control de acceso.

1.1.6. Control de acceso

El control de acceso es el proceso de conceder permisos a usuarios o grupos de acceder a los recursos. Son las políticas que se establecen dentro de un sistema de gestión de identidades que define que los usuarios autorizados accedan solo a los recursos que ellos requieren para realizar sus tareas. El control de acceso es necesario para proteger la confidencialidad, integridad y disponibilidad de los objetos. (1)

1.1.7. Auditoría

La auditoría es un examen crítico que se realiza con el fin de evaluar la eficiencia y eficacia de un sistema informático, consiste en recoger, agrupar y evaluar evidencias con el fin de proteger y detectar de forma sistemática el uso de los recursos y los flujos de la información dentro de una organización, verificando si los mismos cumplen eficientemente con las normas informáticas y generales existentes. (5)

1.2. Sistemas de administración de identidades

La realización de un minucioso estudio sobre las principales compañías que producen software de gestión de identidades y los más importantes sistemas a nivel mundial arrojó como resultado un conjunto de características que se exponen a continuación:

1.2.1. IBM Tivoli Identity Manager

IBM Tivoli Identity Manager interactúa directamente con los usuarios y con dos tipos de sistemas externos: orígenes de identidad y mecanismos de control de acceso. Los sistemas de identidad proporcionan información de autorización sobre los usuarios que necesitan cuentas. Mejora la gestión de acceso de los usuarios mediante el suministro integral basado en solicitudes para solicitar y aprobar los accesos de los usuario para los roles, las cuentas y las autorizaciones de acceso detalladas.

Facilita la administración y permite que las empresas amplíen la gestión de accesos e identidades a usuarios y servicios de otros proveedores, de forma que los clientes puedan controlar el acceso a las aplicaciones según el rol del usuario en la organización, entregando el control de acceso basado en roles a los nuevos servicios web.

- ✓ **Tivoli AccessManager**

A través de su servicio de autorización centralizado de alta disponibilidad, *IBM Tivoli Access Manager* permite una mejor gestión de la información crítica para la empresa. Proporciona un acceso simple y seguro a esta información crítica, mejorando las comunicaciones con clientes, socios comerciales, entre otros. (6)

Tivoli Access Manager tiene dos componentes principales: un registro de usuario y un servicio de autorización que consiste en una base de datos de autorización y un motor de autorización que realiza la acción de toma de decisiones sobre la solicitud. Otro componente que está muy cerca de los componentes básicos es el administrador de recursos, que es responsable de la aplicación de la política de seguridad a los recursos. El componente ejecutor de políticas

dirige la solicitud al servicio de autorizaciones para su evaluación, de acuerdo con el resultado del servicio de autorización (aprobación o rechazo), el gestor de recursos permite o deniega el acceso a los recursos protegidos. Las decisiones de autorización se basan en el certificado de atributos de privilegios (PAC), que se crea para cada usuario autenticado en un entorno de *Access Manager*, independientemente del mecanismo de autenticación utilizado. Esta herramienta, además de la gestión delegada de usuarios también permite la administración delegada de seguridad. (7)

1.2.2. Novell Identity Manager

Novell Identity Manager es una solución de administración de identidades que puede brindar un alto nivel de integración con las soluciones de autenticación, registro único y supervisión de eventos e información de seguridad. Además, ofrece herramientas eficaces así como capacidades avanzadas para modelado visual, flujo de trabajo y autoservicio, ayudando a reducir la carga administrativa de ingresar, actualizar y eliminar la información del usuario en todos los sistemas.

- ✓ **Novell Access Manager**

Novell Access Manager es una solución de gestión de acceso completa capaz de asignar los derechos de acceso pertinentes a todas las personas que acceden a la red de una empresa. *Novell Access Manager* permite centralizar el control de acceso a todos los recursos digitales, sin que sea necesario utilizar herramientas de software distintas para cada ubicación. Además, permite la entrada única, por lo que los socios y empleados sólo tendrán que recordar unos datos de inicio de sesión para poder acceder a todas las aplicaciones *web* de la empresa. Ayuda a las compañías a ganar inteligencia en las operaciones proporcionando visibilidad completa en relación al acceso de los usuarios. Con administración de roles y *workflow* integrados, reduce costos mientras potencia a los usuarios de negocios para definir y controlar quién tiene acceso a qué. Esto reduce fuertemente el tiempo que lleva tener a los usuarios aprovisionados y permite una vista más robusta de los derechos de los usuarios. La administración en los roles y derechos

de usuarios se establece mediante políticas establecidas utilizando procesos tales como la separación de cargas. (8)

1.2.3. CA Identity Manager

CA Identity Manager es un producto de gestión administrativa que brinda servicios automatizados de administración de identidades para la creación, modificación y eliminación eventual de cuentas y derechos según las relaciones del usuario. Administra el acceso y los derechos sobre una amplia gama de sistemas corporativos, desde *mainframe* hasta aplicaciones web. Ofrece varias funcionalidades entre las que se destacan la administración delegada del usuario, flujos de trabajo integrados, un modelo de administración estructurado y soporte integrado. (9)

CA Identity Manager utiliza dos métodos para privilegios administrativos seguros: la especificación y la delegación, ya que tiene por objeto garantizar que los usuarios privilegiados sólo se le concedan los privilegios mínimos necesarios para desempeñar sus responsabilidades de trabajo. *CA Identity Manager* ayuda a enlazar las identidades y los accesos con políticas de uso de la información, de modo que los datos críticos pueden protegerse según el rol de cada usuario.

- ✓ **CA Role &Compliance Manager**

CA Role & Compliance Manager proporciona una interfaz centralizada para que los administradores vean los privilegios de los usuarios e identifiquen cualquier asignación inadecuada. También se puede utilizar para establecer políticas de identidad de cumplimiento, tales como separación de carga, y automatizar los procesos para validar de manera eficiente los privilegios del usuario. Este componente necesita un mecanismo de administración de roles para poder controlar eficazmente el acceso a los recursos protegidos sobre la base de las responsabilidades del usuario.

✓ **CA Access Control**

CA Access Control brinda el nivel crítico de protección de seguridad para sus servidores. La administración detallada del acceso le permite limitar los privilegios del usuario al mínimo necesario según su cargo, incluyendo aquellos del súper usuario. Además, la información de auditoría segura puede ayudarlo a garantizar el cumplimiento al comprobar la interacción que cada usuario tiene con un sistema particular. Su organización depende de servidores para almacenar y acceder a sus recursos de información críticos. *CA Access Control* es un producto que centraliza el control y la ejecución distribuida del acceso a los recursos del servidor según los roles. (10)

1.2.4. 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, de aprobaciones y pedidos, administración de derechos basados en políticas, la integración de tecnologías y las auditorías.

✓ **Oracle Access Manager**

Oracle Access Manager define reglas de autorización, además expresiones de autorización, las cuales se pueden configurar mediante una o varias reglas de autorización para un dominio de la política. Una regla de autorización identifica quién puede acceder a un recurso y denegar explícitamente el acceso al recurso.

Cuando la información sobre un usuario que solicita el acceso a un recurso protegido por una regla de autorización se compara con las condiciones de una regla de autorización, el usuario puede acogerse a una de las condiciones de la regla, esa regla se evalúa para producir uno de los siguientes resultados:

- ❖ Autorización de éxito.

En este caso, el usuario consigue el acceso al recurso solicitado. Este resultado se asocia con la condición de permitir el acceso de la regla.

- ❖ La falta de autorización.

En este caso, el usuario no puede acceder al recurso solicitado. Este resultado se asocia con la condición de denegar acceso de la regla.

- ✓ **Oracle Entitlements Server**

Oracle Entitlements Server (OES) es un motor de autorización que unifica y simplifica la gestión de las políticas de acceso. OES asegura el acceso a los recursos de aplicaciones y componentes de *software* (entre las que se encuentran, direcciones URL[1] (*Uniform Resource Locator*), *Enterprise JavaBeans*, y *Java Server Pages*). OES ofrece un punto de administración centralizada para las políticas de acceso en una amplia gama de negocios y sistemas de Tecnologías de la Información, además brinda una administración centralizada de políticas basadas en estándares y cumplimiento de políticas distribuidas en todas las aplicaciones empresariales. Los roles son fundamentales para definir las políticas de autorización, para que el usuario según el rol que desempeña en cierta institución o empresa tenga acceso, esto satisface las necesidades de las empresas de estructuración compleja. (11)

1.2.5. Aportes obtenidos de los sistemas estudiados.

A partir de la investigación realizada sobre diferentes sistemas de administración de identidades se identificaron características importantes relacionadas con la administración de la autorización de usuarios que podrían ser utilizadas como referencia para la implementación del Servicio de Autorización que se desea construir. Una característica muy interesante que se destaca en los sistemas estudiados es la utilización del modelo de control de acceso basado en roles, es decir, estos sistemas controlan el acceso a los recursos según el rol que desempeña el usuario en la institución o empresa, estas políticas son establecidas utilizando procesos, tales como la separación de cargas. Además, conceden a los usuarios sólo los privilegios mínimos necesarios

[1] Localizador Uniforme de Recursos.

para desempeñar sus responsabilidades de trabajo. Los sistemas de administración de identidades estudiados en su mayoría son *software* propietario, por tanto no se utilizó ninguno, sino que se tomaron características que ayudarán al desarrollo de la propuesta de solución. Es relevante destacar que se estudiaron estos sistemas propietarios y no libres debido a que en el proyecto estaba definido desarrollar un *Identity Manager* que se guiara por estos sistemas que son líderes del mercado.

1.3. Modelos de control de acceso

Hoy en día existen diversos modelos de seguridad donde se implementan políticas para el control de acceso en las aplicaciones, solucionando así los problemas de seguridad existentes.

1.3.1. Modelo de control de acceso discrecional

En el modelo de control de acceso discrecional (*Discretionary Access Control*, DAC) es el usuario quien decide cómo proteger el sistema, mediante controles de acceso impuestos por el sistema, es decir, el propietario del objeto es autorizado a permitir u otorgar permisos para este objeto a otros usuarios. Los sujetos pueden ser usuarios, grupos o procesos.

El modelo DAC brinda al usuario el beneficio de la flexibilidad del modelo, debido a esto es demasiado débil para controlar el acceso a los recursos de forma efectiva. El DAC para realizar operaciones sobre los recursos utiliza las Listas de Control de Acceso (*ACLs*), que son listas de usuarios y grupos con sus permisos específicos. DAC es apropiado en ambientes donde la compartición de información es más importante que su protección. (12)

1.3.2. Modelo de control de acceso obligatorio

En el modelo de control de acceso obligatorio (*Mandatory Access Control,* MAC) es el sistema quién protege los recursos la autorización para que un sujeto acceda a un objeto depende de los niveles de seguridad que tengan, ya que el administrador es el que indica qué permiso de seguridad tiene el sujeto y el nivel de sensibilidad del objeto. (12)

El modelo de seguridad MAC describe niveles de seguridad a través de un conjunto de clasificaciones discretas que siguen el modelo de clasificación de la información militar donde la

confidencialidad de la información es lo más relevante (Desclasificado, Confidencial, Secreto y Sumamente Secreto); esto se conoce como política de seguridad multinivel. Un importante objetivo del modelo MAC es controlar el flujo de información en orden de asegurar su confidencialidad y su integridad, objetivo no alcanzado por los modelos DAC. Sin embargo las políticas que establece este modelo no proporcionan soluciones factibles dado que es demasiado rígido.

1.3.3. Modelo de control de acceso basado en tareas

El TBAC (Task Based Access Control) permite controlar el acceso en entornos representados por flujos de trabajo (*workflow*). Da soporte para el modelado de las tareas, la monitorización y gestión del proceso de autorización, garantizando el control de acceso mediante etapas de autorización. La activación y desactivación de los permisos de los estados de protección (son los que definen todos los permisos que pueden ser activados por la etapa de autorización) de forma dinámica en tiempo de ejecución, además permite monitorizar el uso de los permisos en tiempo de ejecución. En el modelo TBAC la especificación de políticas de seguridad es compleja, también la gestión, delegación y revocación de los privilegios son muy primitivas. (12)

1.3.4. Modelo de control de acceso basado en roles

El principal objetivo del modelo de control de acceso basado en roles (*Role Based Access Control, RBAC*) es prevenir que los usuarios tengan libre acceso a la información de la organización. (12)

La NIST[2] (*National Institute of Standard and Technology*) define de manera clara y organizada las características de la tecnología RBAC, no sólo reconociendo el beneficio potencial de ésta sino que ha propuesto un estándar. El modelo de referencia de RBAC propuesto consta de cuatro componentes: usuarios, roles, operaciones y recursos. En este modelo los usuarios son asignados a uno o varios roles mientras que los permisos y privilegios se asignan a estos roles,

[2] Instituto Nacional de Estándares y Tecnología

es decir, los permisos de acceso están asociados a los roles. Este modelo permite la construcción jerárquica de las políticas de acceso por herencia o especialización. RBAC de forma general describe un grupo de usuarios que pueden estar actuando bajo un conjunto de roles y realizando operaciones en las que utilizan un conjunto de objetos como recursos.

Este modelo incluye un grupo de sesiones donde cada sesión es la relación entre un usuario y un subconjunto de roles que son activados en dicha sesión. Las decisiones de acceso se realizan sobre el conjunto de permisos asociados con los roles activos de cada sesión. Para que un usuario pueda ejercer los permisos para los cuales está autorizado, es necesario que el sistema le haya activado los roles que tienen asociados tales permisos.

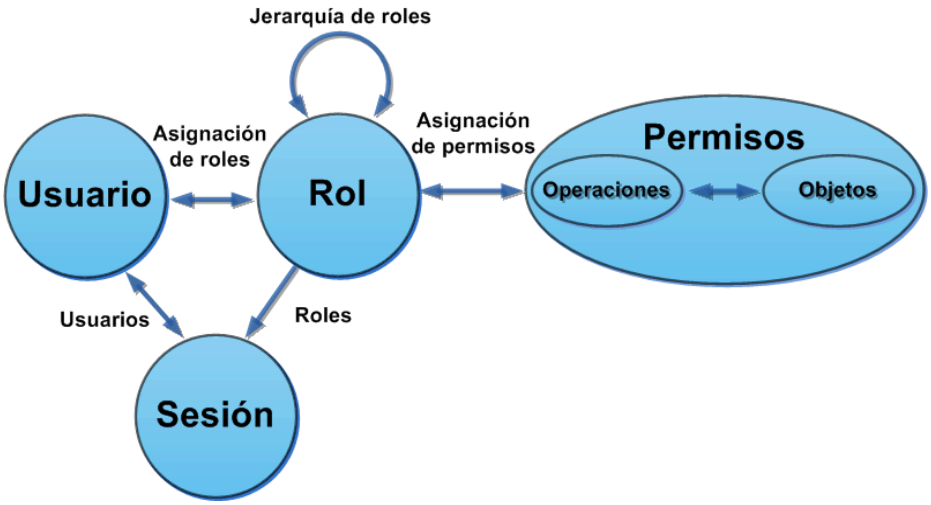


Figura 1. Modelo de control de acceso RBAC. (12)

El modelo RBAC especifica restricciones sobre la relación usuario/rol y sobre la activación de un conjunto de roles de usuario. Las restricciones pueden ser de dos tipos: estáticas o dinámicas.

La Separación Estática de Cargas (*SSD Static Separation of Duty*) previene los conflictos de intereses, esto sucede cuando dos roles son mutuamente exclusivos, ejemplo, un mismo usuario pide y otorga el permiso al mismo tiempo.

La Separación Dinámica de Cargas (*DSD Dynamic Separation of Duty*) tiene como objetivo principal permitir más flexibilidad en cuanto a operaciones se refiere. Las DSD permiten al

usuario asignar los roles en conflicto siempre y cuando no asuman ambos roles en una misma instancia de un proceso.

La diferencia entre SSD y DSD reside en el contexto en el cual la política es impuesta. Mientras que SSD restringe las asignaciones usuario/rol y de permiso/rol, DSD limita la disponibilidad de permisos, imponiendo restricciones en los roles que pueden ser activados simultáneamente en las sesiones.

El modelo RBAC ofrece una mayor eficiencia ya que trabaja de forma centralizada y elimina algunas abstracciones que tienen otros modelos. RBAC es reconocido como uno de los modelos que más ha revolucionado en los últimos años, ya que su fundamental aporte es que puede llegar a funcionar con grandes sistemas, y al mismo tiempo brindarle seguridad. Esto se debe principalmente a la forma jerárquica de funcionar que tiene el control de acceso basado en roles, ya que es más fácil administrar un sistema asignando roles a los usuarios, y así llevar una administración confiable y de bajo coste.

1.4. Metodología de desarrollo

Todo desarrollo de software implica riesgos y complejidad en la gestión del mismo, por tanto, si no se emplean metodologías adaptables, con certeza los clientes y desarrolladores quedarán insatisfechos con el resultado. Las metodologías son un conjunto de procedimientos y técnicas que ayudan a documentar el proceso de desarrollo del software. (13)

Las metodologías se clasifican en ágiles y tradicionales. Las tradicionales son metodologías que se centran fundamentalmente en el control del proceso, además de ser muy efectivas para proyectos de gran tamaño. Dentro de las metodologías tradicionales están OPEN, METRICA 3 y RUP[3] que es una de las más utilizadas actualmente. Las ágiles son aquellas donde el desarrollo de software se caracteriza por ser incremental, cooperativo, sencillo y adaptable; entre ellas se encuentran XP[4], SCRUM y MSF (*Microsoft Solution Framework*).

[3] *Rational Unified Process*, por sus siglas en inglés, significa proceso unificado del *Rational* en español.
[4] *Extreme Programming*, por sus siglas en inglés, significa Programación Extrema en español.

1.4.1. Microsoft Solution Framework ágil

MSF ágil es un proceso ágil y disciplinado de desarrollo de software y constituye un marco de trabajo extensible y adaptable. MSF ágil utiliza múltiples iteraciones y un enfoque basado en escenarios para la construcción de aplicaciones, proporcionando la automatización y la orientación necesaria para apoyar al equipo de desarrollo, además incorpora directamente, prácticas para el manejo de calidad del servicio como el rendimiento y la seguridad.

Define un conjunto de tareas a realizar durante las iteraciones, por los diversos roles que participan en el ciclo de desarrollo de software, incluyendo analistas de negocio, arquitectos, jefes de proyecto, administrador de versiones, desarrolladores y probadores.

Los requisitos no funcionales y funcionales del sistema se expresan en los escenarios y los requerimientos de calidad de servicio respectivamente. Esta metodología soporta 17 flujos de trabajo básicos, en los cuales se agrupan las diferentes actividades.

MSF ágil incluye además cinco ciclos y cinco fases para el desarrollo y seguimiento del producto. Los ciclos describen la frecuencia con que se llevan a cabo y son actualizados las actividades o productos de trabajo dentro de las fases y los grupos de desarrollo. Los ciclos describen la ejecución del proyecto y sus tareas.

Las fases 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. (14)

Las fases con las siguientes:

- ✓ **Visión y Alcance:** En esta fase se definen los objetivos que se persiguen y hasta dónde se quiere llegar con el proyecto. Este proceso se documenta en la declaración de visión que no es más que una descripción del valor del producto que será construido.
- ✓ **Planificación:** Es en esta fase cuando se termina la planificación del proyecto, el equipo prepara las especificaciones funcionales, se realiza el proceso de diseño de la solución y se

preparan los planes de trabajo, estimaciones de costo y cronogramas de los diferentes entregables del proyecto.

- ✓ **Desarrollo:** Durante esta fase el equipo realiza la mayor parte de la construcción de los componentes (tanto documentación como código) y se crea una solución de la arquitectura a establecer. Como resultado se obtiene una versión de la infraestructura del producto después de aplicarle revisiones de código que se va generando.
- ✓ **Estabilización:** En esta fase se llevan a cabo las pruebas sobre la solución, que enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en detectar los errores, solucionarlos y preparar la solución para el lanzamiento.
- ✓ **Implantación o Despliegue:** En esta fase se decide la tecnología base, sus componentes, se estabiliza la instalación y se traspasa el proyecto al cliente. (14)

1.5. Tecnologías y herramientas

Para la creación de cualquier software se hace imprescindible tener una visión de lo que se desea construir por lo que es de gran importancia contar con las herramientas y lenguajes necesarios para trazar la arquitectura a desarrollar garantizando el buen funcionamiento de la propuesta de solución así como la calidad del resultado final.

1.5.1. Framework Thrift

Thrift es una librería y un conjunto de herramientas de generación de código fuente para el desarrollo rápido y seguro de servicios escalables entre aplicaciones. Su principal objetivo es permitir comunicaciones eficientes y fiables entre diferentes lenguajes de programación mediante la abstracción de algunas porciones de cada uno de ellos en una librería común. Específicamente, *Thrift* permite a los desarrolladores definir tipos de datos e interfaces de servicio en un archivo único, utilizando un lenguaje neutral y generar el código necesario para construir clientes y servidores de llamada a procedimientos remotos (RPC). Actualmente soporta los siguientes lenguajes *C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk y OCaml*. Con *Thrift* es posible establecer una comunicación rápida con el sistema de

almacenamiento de datos. *Thrift* no presenta un mecanismo que ejecute una autenticación de forma segura. (15)

Algunas características de *Thrift*:

- ✓ **Tipos de datos primitivos:** *bool, byte*, i16 (entero de 16 *bits*), i32 (entero de 32 *bits*), i64 (entero de 64 bits), double, string, binary, list, map y set.
- ✓ **Formatos de salida:** protocolo binario, JSON.
- ✓ **Otras características:** enumerativos, constantes, excepciones tipadas, interfaces RPC.
- ✓ **Licencia:** *Apache*.
- ✓ **Implementa:** su propio lenguaje de descripción de interfaces. (16)

1.5.2. Windows Communication Foundation (WCF)

Windows Communication Foundation (WCF) es un conjunto de tecnologías .NET para la creación y puesta en marcha de sistemas interconectados. Es una nueva generación de infraestructura de comunicaciones que gira entorno a la arquitectura de servicios web. El soporte para servicios web avanzados en WCF proporciona una mensajería segura, fiable y organizada en transacciones, además de interoperabilidad. WCF unifica una gran variedad de funcionalidades de sistemas distribuidos en una arquitectura organizada y extensible, que abarca transportes, sistemas de seguridad, patrones de mensajería, sistemas de codificación, topologías de red y modelos de alojamiento.

WCF puede crear aplicaciones que funcionen como servicios y como clientes del servicio, creando y procesando mensajes a partir de un número ilimitado de otros servicios y clientes. WCF se basa en la noción de comunicación basada en mensajes y cualquier cosa que se pueda modelar como un mensaje (por ejemplo, una solicitud HTTP, HTTPs, TCP o un mensaje de MSMQ (*Message Queue Server*)), se puede representar de manera uniforme en el modelo de programación. *Windows Communication Foundation* administra y serializa los datos para crear aplicaciones eficaces y fáciles de mantener. WCF implementa mecanismos de seguridad

existentes, con la ventaja principal de utilizar SOAP[5] como protocolo, además de los protocolos existentes. WCF para programar su seguridad utiliza los certificados digitales de X.509 para autenticar clientes y servidores, codificar mensajes y firmar mensajes digitalmente. (17)

WCF se integra con dos modos de seguridad principales: modo de transporte y modo de mensajes. El modo de seguridad de transporte utiliza un protocolo de nivel de transporte, como HTTPS, para lograr la seguridad de la transferencia. El modo de transporte tiene la ventaja de estar ampliamente adoptado, disponible en muchas plataformas y ser menos complejo desde el punto de vista informático. Sin embargo el modo de mensajes se aplica directamente a los mensajes SOAP, y está incluida en envolturas SOAP junto con los datos de la aplicación. Tiene la ventaja de ser independiente del protocolo de transporte, más extensible y de garantizar la seguridad global (frente al protocolo punto a punto). La desventaja de este modo radica en que es mucho más lento que el modo de seguridad de transporte porque tiene que tratar con la naturaleza XML[6] (*Extensible Markup Language*) de los mensajes SOAP. (18)

1.6. Microsoft Enterprise Library 5.0

Enterprise Library, es un conjunto de componentes de *software* reutilizables diseñados para asistir a los desarrolladores de software con los retos comunes del desarrollo empresarial. Soporta el *Framework* .NET 4.0 y la integración con *Microsoft Visual Studio* 2010. Incluye una colección de bloques de aplicación funcional entre los que se encuentran:

- ✓ **Bloque de aplicaciones de almacenamiento en caché.** Se utiliza para incorporar una caché en sus aplicaciones. Proveedores de conexión en caché y tiendas de respaldo persistentes son compatibles.
- ✓ **Bloque de aplicación de la criptografía.** Los desarrolladores pueden utilizar este bloque de aplicación para incorporar cifrado simétrico en sus aplicaciones.

[5] *Simple Object Access Protocol* por sus siglas en inglés, significa Protocolo Simple de Acceso a Datos en español.
[6] Lenguaje de Marcas Extensible.

- ✓ **Bloque de aplicación de acceso a datos.** Este bloque de aplicación se puede utilizar para incorporar la funcionalidad de base de datos estándar en sus aplicaciones, incluyendo tanto el acceso sincrónico y asincrónico de datos y devolver los datos en una variedad de formatos.
- ✓ **Bloque de aplicación de control de excepciones.** Los desarrolladores y diseñadores de políticas pueden usar este bloque de aplicación para crear una estrategia coherente para el procesamiento de excepciones que se producen en las capas de arquitectura de aplicaciones empresariales.
- ✓ **Bloque de aplicación de registro.** Los desarrolladores pueden utilizar este bloque de aplicación para incluir la funcionalidad de registro para una amplia gama de objetivos de registro en sus aplicaciones.
- ✓ **Bloque de aplicación de inyección de política.** Impulsado por el mecanismo de intercepción construido en unidad, este bloque de aplicación puede utilizarse para aplicar políticas de interceptación para simplificar la implementación de características comunes, como el registro, almacenamiento en caché, control de excepciones y validación, a través de un sistema.
- ✓ **Bloque de aplicación de seguridad.** Los desarrolladores pueden utilizar este bloque de aplicación para incorporar la autorización y la funcionalidad de almacenamiento en caché de seguridad en sus aplicaciones. (19)

1.7. Visual Studio Team System 2010

Microsoft Visual Studio Team System 2010 mantiene los propósitos de *Microsoft* de facilitar a los desarrolladores y equipos de desarrollo la rápida creación de aplicaciones interconectadas. Presenta una versión renovada del servidor *Team Foundation Server* el cual ofrece un gestor de pruebas mejorado y nuevas herramientas para el servidor. Desarrolla aplicaciones para *Share Point* y la web, trabaja con múltiples versiones de .NET en un único entorno. Incluye también herramientas para el desarrollo de aplicaciones dirigidas a las últimas versiones de *Microsoft* tales como *Windows 7*, *Windows Server 2008 R2*, *Windows Phone 7* y *Windows Azure*. El

entorno de trabajo está desarrollado en *Windows Professional* dando la capacidad para utilizar múltiples monitores, además presenta muy buena compatibilidad con el código existente de versiones anteriores. *Visual Studio Team System 2010* proporciona nuevos beneficios significativos, como la capacidad de optimizar su entorno de desarrollo con compatibilidad para varios monitores y la capacidad de soportar varias versiones de *.NET Framework* con una sola herramienta. (20)

1.8. Lenguaje de programación

CSharp o *C#*, como también se le conoce, es un lenguaje de programación orientado a objetos desarrollado y estandarizado por la compañía *Microsoft* como parte de la plataforma .NET.

Entre sus características principales se destacan:

- ✓ Provee el beneficio de un ambiente elegante y unificado.
- ✓ El manejo de errores está basado en excepciones.
- ✓ Soporta los conceptos como herencia y polimorfismo de la programación orientada a objetos.
- ✓ No existen funciones globales, variables o constantes. Todo debe ser encapsulado dentro de la clase, como un miembro de la instancia o un miembro estático.
- ✓ Los métodos que se definen en las clases son por defecto no virtuales (no pueden ser sobrescritos al derivar clases).
- ✓ Solamente se permite una clase base, si se requiere herencia múltiple es posible implementar interfaces.
- ✓ No es posible utilizar variables no inicializadas. (21)

1.9. Plataforma de desarrollo

La plataforma .NET[7] constituye un entorno para la construcción, desarrollo y ejecución de servicios web y otras aplicaciones que consiste en tres partes fundamentales: el *Common Language Runtime* (entorno de ejecución, CLR), las *Framework Classes* (clases de la

[7] NET: Abreviación para "red" (*network*), ".net".

plataforma) y ASP.NET. La comunicación a través de la web se hace utilizando el protocolo SOAP, lo cual no supone ningún problema para el desarrollador, ya que, es la plataforma .NET la que se encarga de tratarlo.

Entre sus principales ventajas se encuentran:

- ✓ **Código administrado:** El CLR realiza un control automático del código controla los recursos del sistema para que la aplicación se ejecute correctamente.
- ✓ **Interoperabilidad multilenguaje:** El código puede ser escrito en cualquier lenguaje compatible con .Net.
- ✓ **Compilación *just-in-time*:** El compilador JIT[8] incluido en el *Framework* compila el código intermedio (MSIL[9]) generando el código máquina propio de la plataforma.
- ✓ **Seguridad de acceso al código:** Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del *web* sin tener que preocuparse si esto va a estropear el sistema. (22)

1.10. Gestor de base de datos

Un Sistema de Gestión de Base de Datos (SGBD) es un conjunto de programas que permite crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Constituye un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Tiene como principales objetivos manejar un conjunto de datos que constituyen información relevante para una organización, evitar la redundancia de los datos, mejorar los mecanismos de seguridad de los mismos, su privacidad e integridad realizando las validaciones necesarias para mejorar la eficacia en el acceso a datos.

- ✓ **Oracle Database 11g**

Oracle es un Sistema de Gestión de Base de Datos Relacional (RSGBD), elaborada por la compañía *Oracle Corporation*. En la actualidad el sistema se encuentra en la versión 11g,

[8] (*Just-In-Time*) por sus siglas en inglés, significa Justo a Tiempo, en español.
[9] *Microsoft Intermediate Lenguaje* por sus siglas en inglés, significa Lenguaje Intermedio de *Microsoft* en español.

liberada bajo una licencia privativa en el año 2007. Entre sus características principales, resaltan las siguientes:

- Soporte de transacciones: Capacidad de ejecutar un conjunto de órdenes formando una unidad de trabajo, es decir, en forma indivisible o atómica.
- Escalabilidad: Habilidad para manejar el continuo crecimiento del trabajo y estar preparado para hacerse más grande sin afectar su funcionamiento.
- Estabilidad: Posee una reducida tasa de fallos en su funcionamiento.
- Soporte multiplataforma: Compatibilidad con varios sistemas operativos explotados actualmente a nivel mundial (*Windows, Linux, Unix*).
- Soporte de *Triggers* (Disparadores): Es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (*INSERT*), actualización (*UPDATE*) o borrado (*DELETE*). (23)

1.11. Herramienta de modelado

✓ Altova UModel 2009 Enterprise Edition

Altova UModel 2009 Enterprise Edition pertenece al paquete de *Altova Mission Kit 2009 for Professional Software Architects*, permite el diseño visual de modelos de aplicaciones en *UML* y es capaz de generar código *Java*, *C#* o *Visual Basic.NET* y documentación del proyecto. *Altova UModel* 2009 te ofrece una aplicación que puede utilizarse para crear e interpretar los diseños de *software* a través del poder del estándar UML 2.2. Combina una rica interfaz visual, con funciones de usabilidad superiores para ayudar a nivelar la curva de aprendizaje de UML, además de incluir las más altas funcionalidades para potenciar a los usuarios con las más completas ventajas del desarrollo del producto. Las características de *UModel 2009* para el desarrollo de software basado en las capacidades de modelado avanzado son:

- Soporte para los catorce tipos de diagramas UML (*Unified Modeling Language*).
- Modelado de esquemas XML (*Extensible Markup Lenguaje*) en diagramas UML.
- Diagramas de Proceso de Negocio (BPD).

- Generación de código fuente en lenguajes *Java*, C#, y *VisualBasic*.NET.
- Ingeniería inversa de código fuente y ficheros binarios *Java*, C# y *VisualBasic*.NET.
- Sincronizado de modelo y código a través de ingeniería de ida y vuelta.
- Crea diagramas de secuencia desde el código fuente de la ingeniería inversa.
- Generación de documentación personalizable de proyecto.
- Compartir subproyectos para colaboración o reutilización.
- Capas de diagramas con visibilidad selectiva.
- *Hyperlinks* entre diagramas, documentos, o páginas web.
- Integración con sistemas de control de versiones.
- *Application Programming Interface* (API)[10] extendida para permitir manipulaciones externas.
- Estrecha integración con *Visual Studio* y *Eclipse*.

Se selecciona esta herramienta de modelado por las funcionalidades que brinda y la casi perfecta integración con la plataforma .NET. (24)

1.12. Herramienta de modelado de base de datos, Erwin Studio

Erwin Studio, herramienta líder para el modelado de datos, es usado para el diseño, la construcción lógica y física de base de datos permitiendo a los usuarios visualizar, analizar y documentar cómo fluyen los datos a través de una organización.

Erwin establece una conexión entre una base de datos diseñada y una base de datos, permitiendo transferencia entre ambas y la aplicación de ingeniería reversa. Usando esta conexión, *Erwin* genera automáticamente tablas, vistas, índices, reglas de integridad referencial (llaves primarias, llaves foráneas), valores por defecto, restricciones de campos y dominios.

Erwin soporta principalmente bases de datos relacionales SQL y bases de datos que incluyen *Oracle*, *Microsoft* SQL *Server*, *Sybase*. El mismo modelo puede ser usado para generar múltiples bases de datos, o convertir una aplicación de una plataforma de base de datos a otra. (25)

[10] Por sus siglas en inglés, significa Interfaz de Programación de Aplicaciones en español.

1.13. Entity Framework 4.0

Entity Framework es un ORM (*Object Relational Mapping*), herramienta que permite transformar representaciones de datos de los sistemas de bases de datos relacionales, a representaciones (modelos) de objetos. La existencia de un ORM es fundamental para el desarrollo de sistemas de software robusto y escalable. *Entity Framework* está diseñado para permitir a los programadores crear aplicaciones de acceso a datos programando con un modelo de la aplicación conceptual en lugar de programar directamente con un esquema de almacenamiento relacional. El objetivo es reducir la cantidad de código y mantenimiento que se necesita para las aplicaciones orientadas a datos.

El mismo admite un modelo *Entity Data Model (*EDM*)* para definir datos tanto en el nivel de almacenamiento como en el nivel conceptual y una asignación entre los dos. También permite a los programadores programar directamente con los tipos de datos definidos en el nivel conceptual como objetos de *Common Language Runtime (*CLR*)*. *Entity Framework* proporciona herramientas para generar un modelo EDM y los objetos de CLR relacionados basándose en una base de datos existente. Esto reduce en gran medida el código de acceso a datos que se solía necesitar para crear servicios y aplicaciones de datos basadas en objetos, y agiliza la creación de servicios y aplicaciones de datos orientadas a objetos a partir de una base de datos existente. (26)

1.14. Conclusiones del capítulo

Para analizar las tendencias actuales, en cuanto al control de acceso, se investigaron los principales sistemas de administración de identidades existentes en el mercado. El resultado de esta investigación aportó argumentos sólidos de cómo estos sistemas manejan el proceso de autorización y los modelos de control de acceso en que se basan.

Luego de revisar las principales características, ventajas y desventajas de los modelos de control de acceso, se decidió que el subsistema de autorización, específicamente el servicio que surge como parte de la problemática de esta investigación, implemente un modelo de control de acceso basado en roles.

Capítulo 1

Se define MSF ágil como la metodología que guiará el proceso de desarrollo del Servicio de Autorización, por las características y ventajas que plantea la misma, haciendo uso de la plataforma de .Net la cual se integra con la metodología antes mencionada. Para el desarrollo del servicio se utilizará *Windows Communication Foundation*, debido a que este brinda un mecanismo de seguridad en la autenticación del servicio, a diferencia de *Trhift,* que carece actualmente de algún mecanismo estable de seguridad, todo esto gestionado mediante el IDE de desarrollo *Visual Studio Team System* 2010 con el lenguaje CSharp y el gestor de base de datos *Oracle* 11g.

La combinación de todas estas herramientas y tecnologías, enfocadas a lograr un producto que cumpla con las exigencias tecnológicas actuales, dirigidas especialmente hacia un control de acceso que garantice los objetivos propuestos, se convierten en garantía para el futuro desarrollo del Servicio de Autorización.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

Para la realización de un software se hace imprescindible aclarar todos los conceptos y sus relaciones utilizando un lenguaje que sea comprendido por los involucrados, así como identificar los objetivos que constituyen el sostén fundamental para el desarrollo posterior del mismo.

Las fases más importantes de la metodología de desarrollo ágil MSF en este capítulo son Visión y Planificación. Respecto a este tema la metodología plantea que se debe tener una visión clara de lo que se quiere desarrollar así como realizar una planificación que guíe al equipo de trabajo hacia la construcción exitosa del producto, llevando una descripción adecuada de los escenarios y requerimientos de calidad de servicio que debe cumplir el mismo.

2.1. Fase Visión

Es primordial tener presente la visión y alcance del proyecto para mantenerse enfocado a las necesidades que se van a resolver. En esta fase se plantean cuáles son los objetivos que se persiguen con el trabajo y hasta donde se quiere llegar.

2.1.1. Descripción del servicio

El Centro de Identificación y Seguridad Digital cuenta con un Sistema de Administración de Identidades que actualmente está en desarrollo. Este sistema se encarga entre otros procesos y procedimientos de administrar la autorización de usuarios, específicamente, en el subsistema de autorización que consta de tres componentes, el componente Sistema Gestión de la Configuración de los parámetros del RBAC el cual permite la configuración de los elementos fundamentales del modelo de control de acceso: roles, recursos, operaciones y restricciones. Además de definir jerarquía de roles y recursos, y establecer diferentes tipos de restricciones temporales como las de cardinalidad en la asignación de roles y permisos a usuarios de la organización. Otro de los componentes es el Gestor de Sesiones que se encarga de administrar las sesiones por usuarios y el conjunto de roles activos para cada una de ellas. El tercer componente es el Servicio de Autorización, su desarrollo deviene de la necesidad de

complementar el subsistema de autorización, particularmente con la función de garantizar un adecuado control de acceso de los usuarios a los recursos de cada aplicación integrada al Sistema de Administración de Identidades, mediante el modelo RBAC ideal para organizaciones con una estructura bien definida donde los roles están bien identificados. En este modelo 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.

2.1.2. Vista Conceptual del servicio

Servicio de Autorización: Es el encargado de controlar el acceso de los usuarios a los recursos de las aplicaciones que delegaron su seguridad en el Sistema de Administración de Identidades.

Cliente de Autorización: Es un módulo en las aplicaciones que se encarga de capturar la petición de acceso del usuario a un recurso y enviarla al servicio.

Aplicación: Se refiere a una aplicación, o sistema informático que delega su seguridad al sistema de administración de identidades, y accede al Servicio de Autorización a través del cliente de autorización o directamente, para chequear el acceso de los usuarios a sus recursos.

Recurso: Un recurso puede ser cualquier objeto accesible en un sistema informático. Los recursos son tradicionalmente considerados como entidades pasivas que contienen o reciben información.

Operación: Es una acción o proceso activo invocado por un usuario.

Grupo: Es un lugar específico al que pertenece un usuario, por ejemplo una oficina.

Ámbito: Estructura organizacional lógica asociada a una institución, agrupando los usuarios, recursos y roles que se desarrollan dentro del mismo, con el fin de lograr un control desde el punto de vista administrativo.

Rol: Un rol es una función de trabajo en el contexto de una organización con una semántica asociada sobre la autoridad y la responsabilidad conferida en el usuario asignado a la función.

Usuario: Se refiere a quien utiliza determinado hardware y/o software, mediante el cual obtiene un servicio. Un usuario puede no ser un ser humano.

Petición de acceso: Es la acción que realiza el usuario en la aplicación al intentar acceder a un recurso de esta, la petición trae consigo usuario, recurso, aplicación, grupo, ámbito, sesión, operación entre otros elementos.

Sesión: Es el espacio que se le crea al usuario en una aplicación cuando accede por primera vez, además es donde el usuario desempeña determinados roles, que le dan ciertos privilegios de acceso en las diferentes aplicaciones.

Gestor de Sesiones: Se encarga de manejar las sesiones de los usuarios, el tiempo de expiración, entre otros elementos. Este gestor cuenta con el Servicio de Administración de Sesiones, que brinda un grupo de funcionalidades entre las que se encuentra la de activación de roles en sesión.

Decisión de acceso: Se corresponde con una petición de acceso específica y la respuesta correspondiente a esta petición, ya sea satisfactoria o no.

Caché de decisión de acceso: Caché en la que se almacena temporalmente en memoria un conjunto de decisiones de acceso.

Traza: Huella que deja un usuario en un sistema, a partir de las acciones que este realiza.

2.1.3. Modelo conceptual del servicio

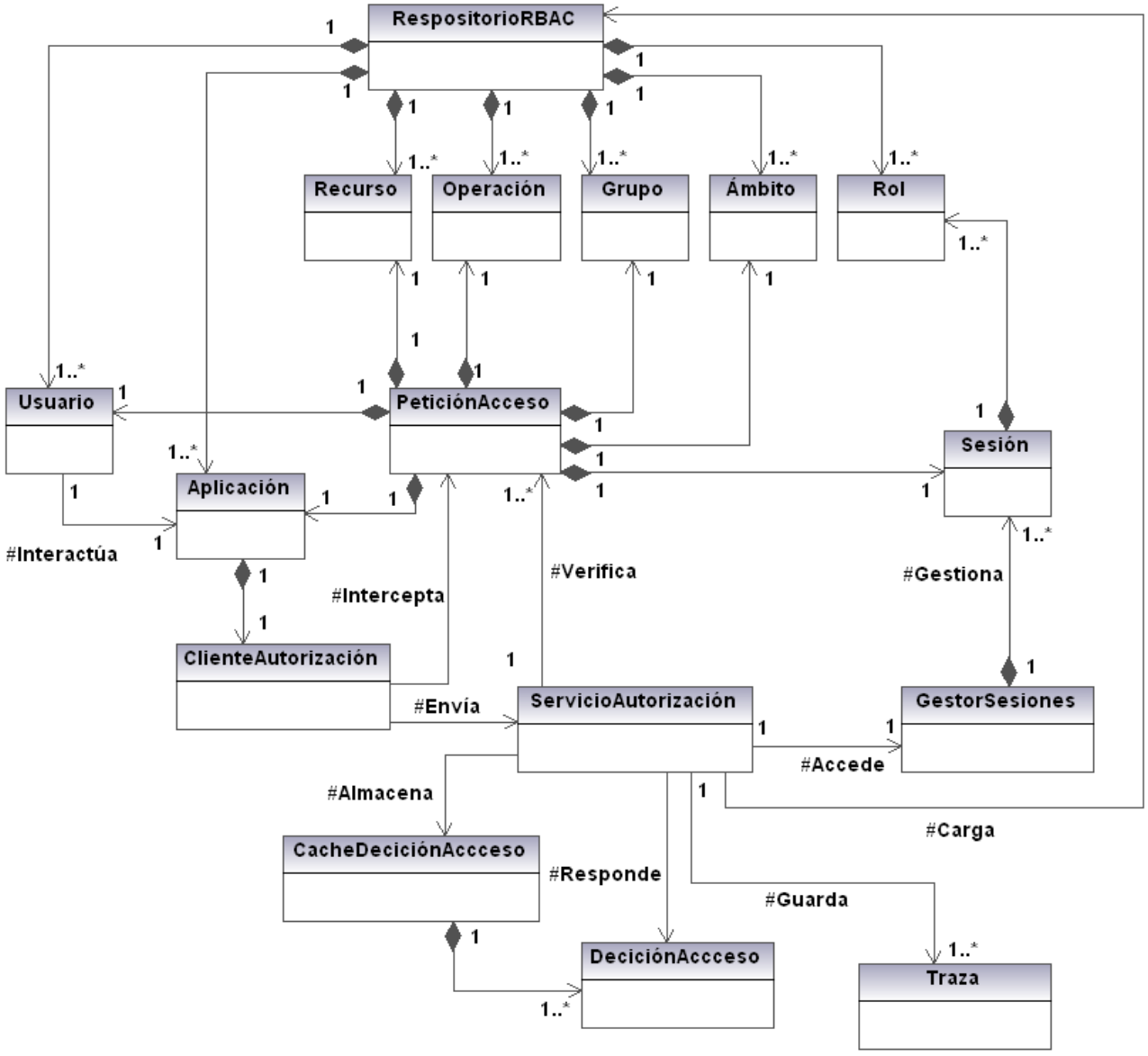


Figura 2. Modelo conceptual del servicio.

2.1.4. Descripción del modelo conceptual

El proceso comienza cuando el usuario interactúa con la aplicación para realizar alguna operación sobre un recurso, entonces el Cliente de Autorización captura esta petición la cual está

formada por una serie de datos que son del usuario que realiza la petición, la sesión que mantiene activa, el grupo de donde proviene esa petición, el ámbito, la operación y el recurso al que trata de acceder y la envía al servicio. El servicio verifica si el usuario tiene acceso o no al recurso que intenta acceder. Para esto carga de la sesión del usuario los roles que ya mantiene activo, debido a peticiones anteriores, y en caso de no tener ninguno los carga del repositorio RBAC, donde se encuentra la información necesaria respecto a los permisos que el usuario tiene de acuerdo a los roles que le fueron asignados. En caso de tener los privilegios necesarios, el servicio accede al Gestor de Sesiones en donde activa el rol con el cual se le da el acceso al usuario. El servicio almacena en caché cada decisión de acceso que se realiza, para verificar si una nueva petición tiene las mismas características de alguna de las ya almacenadas y dar respuesta inmediatamente, mejorando así el rendimiento del servicio. Además guardará una traza de toda la información referente a la respuesta de acceso o no al usuario, fecha, la máquina de donde proviene la petición entre otros.

2.1.5. Propuesta de solución

A partir de un análisis de las tendencias actuales y teniendo en cuenta la problemática a resolver se propone el desarrollo de un Servicio de Autorización que englobará todo lo referente al control de acceso de los usuarios a los recursos de las aplicaciones que estén asociadas al Sistema de Administración de Identidades. El servicio atenderá las peticiones de autorización procedentes de las aplicaciones y luego de evaluar las políticas de acceso correspondientes según el modelo RBAC, enviará a la aplicación solicitante la decisión de acceso correspondiente.

Para verificar el acceso, el servicio evaluará la petición de autorización a través de un algoritmo, este debe comprobar en el repositorio de elementos de RBAC, las políticas de acceso definidas, y las características del modelo, roles, permisos que avalan que el usuario puede o no acceder a un determinado recurso. Uno de los elementos fundamentales que se debe tener en cuenta durante el desarrollo de este servicio es que el mismo debe ser capaz de activar en la sesión del usuario el rol con el cual se desempeña en cada instante, para esto el servicio deberá verificar un

conjunto de restricciones dinámicas que establecen que determinados roles no pueden ser activados simultáneamente en sesión.

Después de verificar las restricciones dinámicas y comprobar que no exista ninguna que involucre al rol con el que se pretende conceder el acceso al usuario, el servicio deberá activar en la sesión del usuario, el rol indicado siguiendo el principio del menor privilegio, de manera que se otorguen los permisos estrictamente necesarios reduciendo de esta manera los riegos en la seguridad, al disminuir las posibilidades de que un usuario se le concedan mayores privilegios de los que debe poseer para operar bajo un determinado rol en una aplicación.

El servicio debe ser capaz de almacenar en caché las decisiones de acceso, para verificar por cada petición de un usuario si existe ya una con las mismas características de las ya almacenadas, para mejorar el tiempo de respuesta a las peticiones que realiza el usuario. Esta caché de decisiones se le debe definir un tiempo de expiración. Además, el servicio brindará otras funcionalidades adicionales que facilitarán el trabajo a los desarrolladores de aplicaciones, de manera que las propias aplicaciones puedan acceder directamente al Servicio de Autorización para gestionar el acceso a determinados recursos de las mismas. Utilizando a la par el Cliente de Autorización como interceptor de las peticiones de acceso.

Por cada decisión de acceso, el servicio deberá guardar la traza correspondiente a esta, con la información referente a la petición que fue realizada por el usuario y la respuesta que le dio el servicio. En caso de que la decisión de acceso haya sido satisfactoria, se registrará además, el identificador del rol con el que se le dio acceso al usuario en ese momento, en caso contrario se almacena el mensaje de error que especifica el motivo por el cual no se le concedió el acceso al mismo.

El Servicio de Autorización se implementará usando la plataforma .net, el lenguaje de programación C# para la implementación de la librería de clases, WCF *Service* para el desarrollo del servicio utilizando una autenticación basada en certificados digitales, a nivel de transporte, para establecer una transferencia segura. Para el manejo de los datos se utilizará *Entity*

Framework y *Oracle* 11g y *Microsoft Enterprise Library* 5.0 para el almacenamiento en caché y de las trazas de acceso.

2.1.6. Definición de las personas

En esta actividad se definirán las personas o sistemas que van a interactuar con el servicio.

Tabla 1. Descripción de las personas.

Personas	Descripción
Cliente de Autorización	Es un módulo en las aplicaciones que se encarga de capturar la petición de acceso del usuario a un recurso y enviarla al servicio para que conceda o no el acceso al mismo.
Aplicaciones Web	Aplicaciones web que delegan su seguridad al Sistema de Administración de Identidades, y acceden al Servicio de Autorización para chequear directamente el acceso a los recursos por parte de los usuarios, y obtener los recursos de un tipo específico a los que estos tienes acceso.

2.2. Fase Planificación

En esta fase es donde se termina la planificación del proyecto, el equipo prepara las especificaciones funcionales, se realiza el proceso de diseño de la solución y se preparan los planes de trabajo cronogramas de los diferentes entregables del proyecto. Los principales artefactos que genera esta fase son la lista de escenarios y los requerimientos de calidad de servicios, los cuales son utilizados para especificar los requisitos de software que sirven de guía para todo el proceso de desarrollo.

2.2.1. Lista de escenarios

Los escenarios representan un camino único de la interacción del usuario a través del sistema. Los mismos se utilizan para definir los servicios que proporcionará el software para el usuario. Para escribir un escenario se debe ser específico. Se identificaron los siguientes escenarios:

- ✓ **Chequear acceso a un recurso.**
- ✓ **Chequear acceso a un recurso para roles activos en sesión.**
- ✓ **Chequear acceso a un recurso para roles asignados al usuario.**
- ✓ **Chequear acceso a un listado de recursos.**
- ✓ **Obtener listado de recursos de un tipo específico.**

2.2.2. Tareas de los escenarios

Las tareas de los escenarios ayudan a entender mejor como va a funcionar cada escenario. Se definieron las siguientes tareas:

- ✓ Verificar sesión del usuario.
- ✓ Obtener roles activos de una sesión.
- ✓ Obtener roles del grupo.
- ✓ Interceptar roles activos del usuario con los roles del grupo.
- ✓ Verificar permisos de los roles activos del grupo.
- ✓ Chequear restricciones dinámicas.
- ✓ Determinar el rol con menor privilegio.
- ✓ Obtener roles asignados al usuario.
- ✓ Interceptar roles asignados al usuario con los roles del grupo.
- ✓ Verificar permisos de los roles asignados al usuario con roles del grupo.
- ✓ Determinar el rol con menor privilegio.
- ✓ Activar rol en sesión con acceso.
- ✓ Crear decisión de acceso.
- ✓ Guardar decisión de acceso.

- ✓ Guardar traza de acceso.
- ✓ Dar respuesta de autorización.

2.2.3. Priorizar escenarios.

Los escenarios se priorizan según la importancia que tienen para la validez de la propuesta de solución. El proceso de priorizar cada escenario trae consigo identificar los escenarios más importantes a la hora de implementar, llevando a cabo su implementación en las primeras iteraciones. Los escenarios de prioridad uno son los de mayor relevancia.

Tabla 2. Priorizar lista de escenarios.

No.	Escenario	Prioridad
1	Chequear acceso a un recurso	1
2	Chequear acceso a un recurso para roles activos en sesión	1
3	Chequear acceso a un recurso para roles asignados al usuario	1
4	Chequear acceso a un listado de recursos	2
3	Obtener listado de recursos de un tipo específico	2

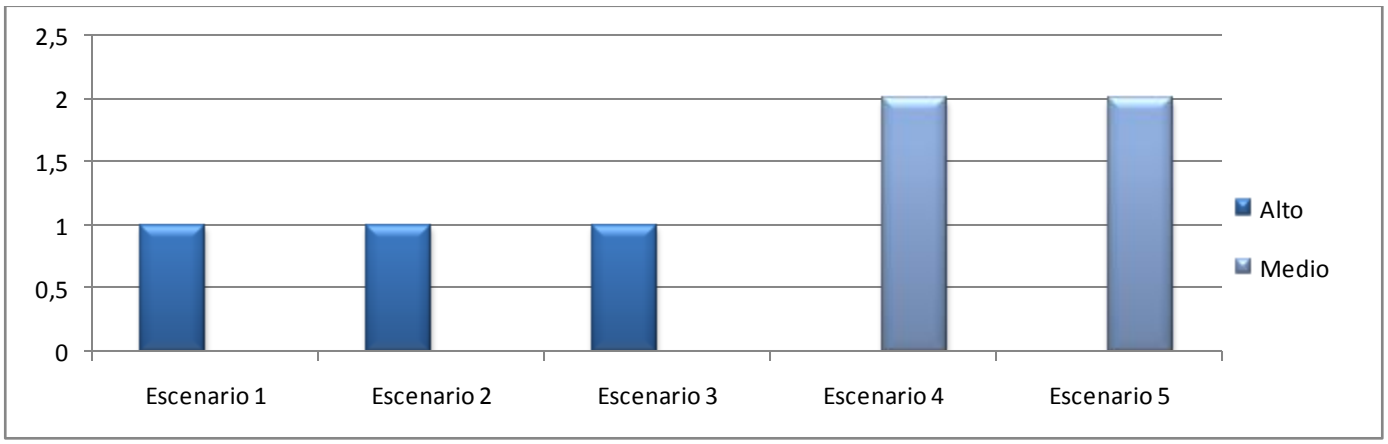


Figura 3. Gráfica evaluación escenarios (Escenario-Prioridad).

2.2.4. Requerimientos de calidad de servicio

Los requerimientos de calidad de servicio constituyen un material resultante que documenta las características del producto final. A continuación se listan:

- ✓ **Rendimiento**
  - ❖ La eficiencia del producto estará determinada en gran medida por la velocidad de las consultas de la base de datos.
  - ❖ El servicio propuesto debe ser capaz de soportar una cantidad escalable de peticiones y el tiempo de respuesta deberá ser rápido.
- ✓ **Soporte**
  - ❖ El servicio debe utilizar el .Net *Framework* en su versión 4.0.
  - ❖ La base de datos que debe usarse es *Oracle* 11g.
- ✓ **Disponibilidad**
  - ❖ Debe estar disponible el servicio en todo momento para los clientes de autorización o las aplicaciones correspondientes.
- ✓ **Accesibilidad**
  - ❖ El servicio sólo puede ser accesible por las aplicaciones que lo consumen, no por los usuarios de estas aplicaciones por problemas de seguridad.
- ✓ **Seguridad**
  - ❖ El servicio registrará todas las acciones que se realicen.
  - ❖ Se llevará el registro del tiempo de actividad y del lugar de acceso de cada usuario para cada acción.
  - ❖ El servicio tendrá un mecanismo de autenticación segura.

2.2.5. Plan de iteraciones

Para el desarrollo del servicio se hace necesario estimar el tiempo que se demorará el desarrollador en implementar cada uno de los escenarios que pertenecen a la propuesta de

solución, organizando así el trabajo. La prioridad de los escenarios es importante ya que mediante esta se decide cuáles de ellos se implementarán en las primeras iteraciones del ciclo de vida del software, su prioridad deviene por la importancia de cada escenario para la realización del servicio.

Iteración 1: Se propone la realización de los escenarios “Chequear acceso a un recurso”, ”Chequear acceso a un recurso para roles activos en sesión”, “Chequear acceso a un recurso para roles asignados al usuario” de prioridad uno.

Iteración 2: Se propone la realización de los escenarios “Chequear acceso a un listado de recursos” y “Obtener listado de recursos de un tipo específico”, de prioridad dos.

Tabla 3. Estimación de Iteraciones.

No.	Escenario	Prioridad	Riesgo	Estimación(horas)	Iteración
1	Chequear acceso a un recurso	Alta	Alto	10	1
2	Chequear acceso a un recurso para roles activos en sesión	Alta	Alto	15	1
3	Chequear acceso a un recurso para roles asignados al usuario	Alta	Alto	15	1
4	Chequear acceso a un listado de recursos	Media	Medio	20	2
5	Obtener listado de recursos de un tipo específico	Media	Medio	15	2

2.2.6. Descripción de los escenarios

Los escenarios proporcionan un medio para que los desarrolladores y usuarios finales lleguen a una compresión común de la propuesta de solución. Para la implementación del servicio se realiza la descripción de los escenarios según las necesidades del cliente. A continuación se presenta una breve descripción de los escenarios del servicio.

Tabla 4. Descripción del escenario "Chequear acceso a un recurso".

Nombre del Escenario: Chequear acceso a un recurso		***Identificador:*** 1
Objetivo del Escenario: Verificar si el usuario tiene sesión y verificar en caché.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza cuando el usuario envía una solicitud para acceder a un determinado recurso de una aplicación. Esta petición es capturada por el Cliente de Autorización que se encarga de enviarla al Servicio de Autorización, el cual verifica la existencia de la sesión del usuario **(Ver tarea Verificar sesión de usuario)** y verifica la petición en la caché decisiones de acceso **(Ver tarea Verificar decisión de acceso)**, guarda la traza de acceso (Ver tarea Guardar traza de acceso) y finalmente da una respuesta de autorización **(Ver tarea Dar respuesta de autorización)**. Si no existe ninguna decisión de acceso que involucre la petición del usuario el servicio pasa a chequear el acceso al recurso para los roles activos en sesión **(Ver escenario Chequear acceso a un recurso para los roles activos en sesión)**		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 5. Descripción del escenario "Chequear acceso a un recurso para roles activos en sesión".

Nombre del Escenario: Chequear acceso a un recurso para roles activos en sesión		***Identificador: 2***
Objetivo del Escenario: Chequear el acceso a un recurso pero sólo de los roles activos en la sesión del usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza cuando en la caché de decisiones de acceso no existe la petición que se realiza. Luego se obtienen los roles activos de la sesión **(Ver tarea Obtener los roles activos de una sesión)**. Una vez que se obtienen los roles se interceptan los roles activos con los roles del grupo de donde se		

realiza la petición **(Ver tarea Interceptar roles activos con roles del grupo)** y se verifican los permisos de los roles activos del grupo **(Ver tarea Verificar permisos de los roles activos del grupo)**, en el caso que ninguno de estos roles tenga el permiso solicitado (**Ver escenario Chequear acceso a un recurso para roles asignados al usuario)**. En caso positivo se determina el rol con menor privilegio **(Ver tarea Determinar rol con menor privilegio)** para luego crear **(Ver tarea Crear decisión de acceso)**, y guardar la decisión de acceso **(Ver tarea Guardar decisión de acceso)**, guardar la traza de acceso **(Ver tarea Guardar traza de acceso)** y dar una respuesta de autorización **(Ver tarea Dar respuesta de autorización)**.
Validaciones:
Prototipo de Interfaz de Usuario:

Tabla 6. Descripción del escenario "Chequear acceso a un recurso para roles asignados al usuario".

Nombre del Escenario: Chequear acceso a un recurso para roles asignados al usuario		***Identificador: 3***
Objetivo del Escenario: Chequear el acceso a un recurso pero sólo de los roles asignados al usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza cuando no se obtienen roles activos en sesión o ninguno de los roles activos en sesión posee el permiso que solicita el usuario. El servicio de autorización busca los roles del usuario **(Ver tarea Obtener roles asignados al usuario)** y los intercepta con los roles del grupo de donde se realiza la petición **(Ver tarea Interceptar roles asignados al usuario con roles del grupo)**, además verifica los permisos de los roles asignados al usuario en el grupo **(Ver tarea Verificar permisos de los roles asignados al usuario con roles del grupo)**, en caso que no tenga permiso se crea la decisión de acceso (**Ver tarea Crear decisión de acceso**). En caso positivo el servicio realiza las validaciones correspondientes **(Ver tarea Chequear restricciones dinámicas)**. Finalmente se determina el rol con menor privilegio **(Ver tarea Determinar el rol con menor privilegio**), activa el rol de menor privilegio con acceso en la sesión del usuario **(Ver tarea Activar rol en sesión con acceso)** para luego crear y guardar la decisión de acceso, guardar la traza de acceso y posteriormente dar una respuesta de autorización.		
***Validaciones*:** Se verifica que los roles activos en sesión no sean excluyentes con el rol con el que el usuario tiene acceso al recurso.		

Prototipo de Interfaz de Usuario:

Tabla 7. Descripción del escenario "Chequear acceso a un listado de recursos".

Nombre del Escenario: Chequear acceso a un listado de recursos		***Identificador: 4***
Objetivo del Escenario: Chequear el acceso de un usuario a un listado de recursos.		
Persona: Servicio de Autorización		
Iteración: 2	***Prioridad: 2***	***Complejidad: 2***
Descripción: El escenario comienza cuando la aplicación web envía al servicio un listado de recursos el identificador de un usuario, de la aplicación y el ámbito al que pertenecen estos recursos. Devolviendo una respuesta de acceso satisfactoria o no para cada uno de los recursos de la lista, de los cuales se pretende conocer si el usuario especificado tiene algún tipo de acceso en la aplicación y el ámbito también señalado. El servicio obtiene los roles que posee el usuario **(Ver tarea Obtener roles asignados al usuario)** y verifica los permisos que poseen estos roles **(Ver tarea Verificar permisos de roles asignados al usuario con roles del grupo)** para determinar si el usuario posee algún tipo de acceso sobre cada uno de los recursos que se quieren comprobar. El servicio envía un listado con valores de verdadero o falso que se refieren a cada recurso, y que indican si el usuario posee algún tipo de acceso o no a los mismos.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 8. Descripción del escenario "Obtener listado de recursos de un tipo específico".

Nombre del Escenario: Obtener listado de recursos de un tipo específico		***Identificador: 5***
Objetivo del Escenario: Obtener un listado de recursos de un tipo de recurso en específico sobre los que el usuario tiene algún tipo de acceso.		
Persona: Servicio de Autorización		
Iteración: 2	***Prioridad: 2***	***Complejidad: 2***
Descripción:		

El escenario comienza cuando la aplicación web envía al Servicio de Autorización. Una petición con la información referente a un usuario del que se quiere obtener un listado de recursos de un tipo determinado a los que este tiene algún tipo de acceso, para esto se debe especificar el identificador del usuario, el tipo de recursos, la aplicación y el ámbito de donde proviene la petición y el servicio devuelve un listado con los recursos del tipo solicitado a los cuales el usuario tiene acceso.
***Validaciones*:**
Prototipo de Interfaz de Usuario:

2.2.7. Especificación de tareas por escenarios

A continuación se describirán algunas de las tareas asociadas a cada escenario de la propuesta de solución. La descripción de las restantes tareas se encuentra en los anexos. (Ver anexo 1).

El escenario **"Chequear acceso a un recurso"** se dividió en dos tareas:

Tabla 9. Descripción de la tarea "Verificar decisión de acceso".

Nombre de la tarea: Verificar decisión de acceso		***Identificador: 1.1***
Objetivo del Escenario: Verificar la decisión para saber si el usuario tiene acceso o no a un recurso según la respuesta definida en la decisión de acceso almacenada en la caché de decisiones de acceso.		
Persona: Servicio de Autorización		
***Iteración:* 1ra**	***Prioridad:* 1**	***Complejidad:* 3**
Descripción: El escenario comienza luego de verificar que la sesión del usuario existe. Se verifica la petición en la caché de decisiones de acceso para comprobar si se realizó antes esta petición. Si se ha realizado, entonces se verifica la respuesta contenida en dicha decisión y se envía una respuesta que puede ser verdadero o falso, que indica si el usuario tiene acceso o no al recurso al cual intenta acceder.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

El escenario **"Chequear acceso a un recurso para roles activos"** se dividió en cinco tareas:

Tabla 10 . Descripción de la tarea "Obtener los roles activos de una sesión".

Nombre de la tarea: Obtener los roles activos de una sesión.		***Identificador: 2.1***
Objetivo del Escenario: Obtener todos los roles activos en sesión de un usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza cuando en la caché de decisiones de acceso no existe la petición que se realiza. El Servicio de Autorización consulta al Servicio de Administración de Sesiones indicándole el ID de sesión y el ID de la aplicación, finalmente el servicio devuelve los roles activos del usuario en la sesión.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 11. Descripción de la tarea "Verificar permisos de los roles activos del grupo".

Nombre de la tarea: Verificar permisos de los roles activos del grupo		***Identificador: 2.2***
Objetivo del Escenario: Verifica los permisos de los roles activos según al grupo que pertenezca el usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza cuando se pasa el ID de rol, ID del recurso y el ID de la operación para comprobar si los permisos que tiene asociado ese rol le permite realizar la operación sobre el recurso especificado.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

El escenario **"Chequear acceso a un recurso para roles asignados al usuario"** se dividió en seis tareas:

Tabla 12. Descripción de la tarea "Chequear restricciones dinámicas".

Nombre de la tarea: Chequear restricciones dinámicas		***Identificador: 3.1***
Objetivo del Escenario: Chequear restricciones dinámicas para los roles que se van activar en sesión.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza se verifican las restricciones dinámicas, de manera que el rol de menor privilegio con el que el usuario tiene acceso al recurso, no sea mutuamente exclusivo con alguno de los roles que se encuentran activos en sesión según lo establecido en alguna restricción existente. En caso de que exista alguna restricción y se compruebe que algún rol de los activos en sesión es excluyente con el rol que se quiere activar, en este caso no se permite activar el rol. En caso contrario, es decir que no exista ninguna restricción que afecte al rol que se quiere activar, se da el visto bueno para que posteriormente se pueda activar el rol en la sesión del usuario.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 13. Descripción de la tarea "Verificar permisos de los roles asignados al usuario con roles del grupo".

Nombre de la tarea: Verificar permisos de los roles asignados al usuario con roles del grupo		***Identificador: 3.2***
Objetivo del Escenario: Verifica los permisos de los roles asignados al usuario que pertenezcan al grupo del cual se realizó la petición.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza cuando se pasa el ID de rol, ID del recurso y el ID de la operación para comprobar si los permisos que tiene asociado ese rol le permite realizar la operación sobre el recurso especificado.		
***Validaciones*:**		

Capítulo 2

Prototipo de Interfaz de Usuario:

2.3. Conclusiones del capítulo

Con la realización de este capítulo se logró una descripción detallada de las características del servicio, determinando un listado con los escenarios que resolverán las funcionalidades del mismo, y describiendo paso a paso cada una de las tareas y escenarios. Se identificaron las personas que van a interactuar con el servicio, así como los requerimientos necesarios para su desarrollo. También se facilitó un mejor entendimiento de la dinámica y estructura, sentando las bases para las futuras fases del proceso. Para obtener una visión clara sobre lo que se desea desarrollar, se escogieron los conceptos fundamentales para construir la vista conceptual del proceso a desarrollar.

El Servicio de Autorización tiene como propósito fundamental controlar el acceso de los usuarios a los recursos, es por esto que tiene como funcionalidades principales: chequear acceso a un recurso, determinar el rol con menor privilegio, activar rol en sesión con acceso, crear decisión de acceso, guardar traza de acceso, verificar permisos de los roles activos del grupo, entre otras, garantizando con ello el buen funcionamiento de este proceso.

Capítulo 3

CAPÍTULO 3: DESARROLLO Y PRUEBAS DE LA SOLUCIÓN PROPUESTA

Introducción

La metodología MSF para la construcción de software ágil provee las fases de desarrollo y pruebas las cuales son importantes para el buen desarrollo y acabado de cualquier proceso de software. Durante estas etapas se define la arquitectura y se seleccionan los patrones de diseño a utilizar con el objetivo de crear un sistema escalable, flexible y confiable. El desarrollo de un producto de software, es el mecanismo donde se ponen en práctica todas las descripciones y arquitecturas propuestas, donde se generan los artefactos correspondientes según establece la metodología utilizada. Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas. La fase de pruebas del sistema tiene como objetivo verificar el sistema para comprobar si este cumple con los requisitos planteados.

3.1. Fase desarrollo

Durante esta fase se realiza la mayor parte de la construcción de los componentes tanto de la documentación como del código, además se define una solución de la arquitectura a establecer. Se obtiene como resultado una versión de la infraestructura del producto después de aplicarle revisiones al código que se va generando.

3.1.1. Arquitectura a utilizar

La arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. (27) El servicio que se construye cuenta con una arquitectura basada en un modelo de tres capas.

3.1.1.1. Modelo de capas

El modelo de capas de una arquitectura organiza el sistema en capas, cada una de las cuales proporciona un conjunto de servicios. Este tipo de arquitectura soporta el desarrollo incremental de sistemas y evita que los cambios en una de las capas afecten directamente al resto. La

arquitectura del Servicio de Autorización se encuentra representada por tres capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades, permitiendo reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas.

Descripción de las capas.

- ✓ Capa de interfaz.

Esta capa contiene el servicio de *Windows Communication Foundation*, el cual es el equivalente a una interfaz de usuario en una aplicación de gestión. En esta capa se exponen distintas funcionalidades que podrán ser encuestadas por el cliente de autorización de las aplicaciones o por las aplicaciones directamente. El mismo estará disponible a través de dos protocolos de comunicación (Net TCP y HTTP).

- ✓ Capa de negocio

Esta capa está relacionada con la capa interfaz y con la capa de acceso a datos. La capa de negocio accede a la capa de acceso a datos para obtener información de la BD. Toda la lógica del negocio que está representada en esta capa, se expone a través de interfaces permitiendo, que independientemente de la implementación que se haga de la misma, el servicio *Windows Communication Fundation* no se afecte, pues este siempre accede al negocio a través de las interfaces, las que en ningún momento cambian, lo que cambia es la implementación que se hacen en las clases del negocio.

- ✓ Capa de acceso a datos.

En esta capa se encuentran todas las funcionalidades de acceso a datos. Para el manejo de los datos esta capa interactúa directamente con la base de datos, a través del *ORM Entity Framework*, el cual facilita el manejo de los datos a través de un mapeo que hace de las tablas de la base de datos a entidades, permitiendo a los programadores manejar los datos como objetos, de una manera muy sencilla.

Las funcionalidades de esta capa se exponen a través de interfaces, que son implementadas por las clases correspondientes, lo que posibilita que sin importar la implementación que se haga siempre los métodos que se exponen son los mismos, y así en caso de que ocurran cambios en las consultas por los motivos que sean, la capa de negocio que es la que interactúa con la de acceso a datos, no se afecte en lo absoluto, ya que la misma siempre estaría accediendo a las interfaces definidas en esta capa, las cuales nunca cambiarían.

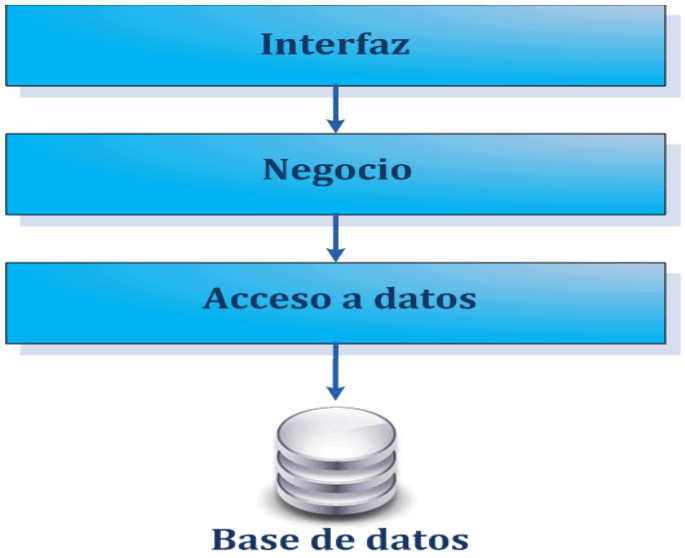


Figura 4. Arquitectura del Servicio de Autorización.

3.1.1.2. Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Los mismos identifican: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Los patrones brindan una manera más práctica de describir ciertos aspectos de la organización de un programa.

- ✓ Instancia única (*Singleton*):

Se asegura que solo se pueda crear una instancia de una clase y ofrece un punto global de acceso a esta instancia. Dicho patrón se utiliza en la clase *AccessManager* garantizando que se

haga una instancia única a la misma, obteniendo mejoras en rendimiento, evitando frecuentes accesos a la base de datos por cada petición de acceso de los usuarios. Esto se garantiza, ya que cuando se levante una instancia de esta clase se obtiene de la base de datos determinada información que pasa a estar disponible en memoria a partir de ese momento. De manera que cuando se quiera nuevamente obtener esa información no es necesario hacer una nueva instancia, sino con sólo obtener la ya existente se accede a los datos, agilizando este proceso considerablemente.

- ✓ Fachada(*Facade*):

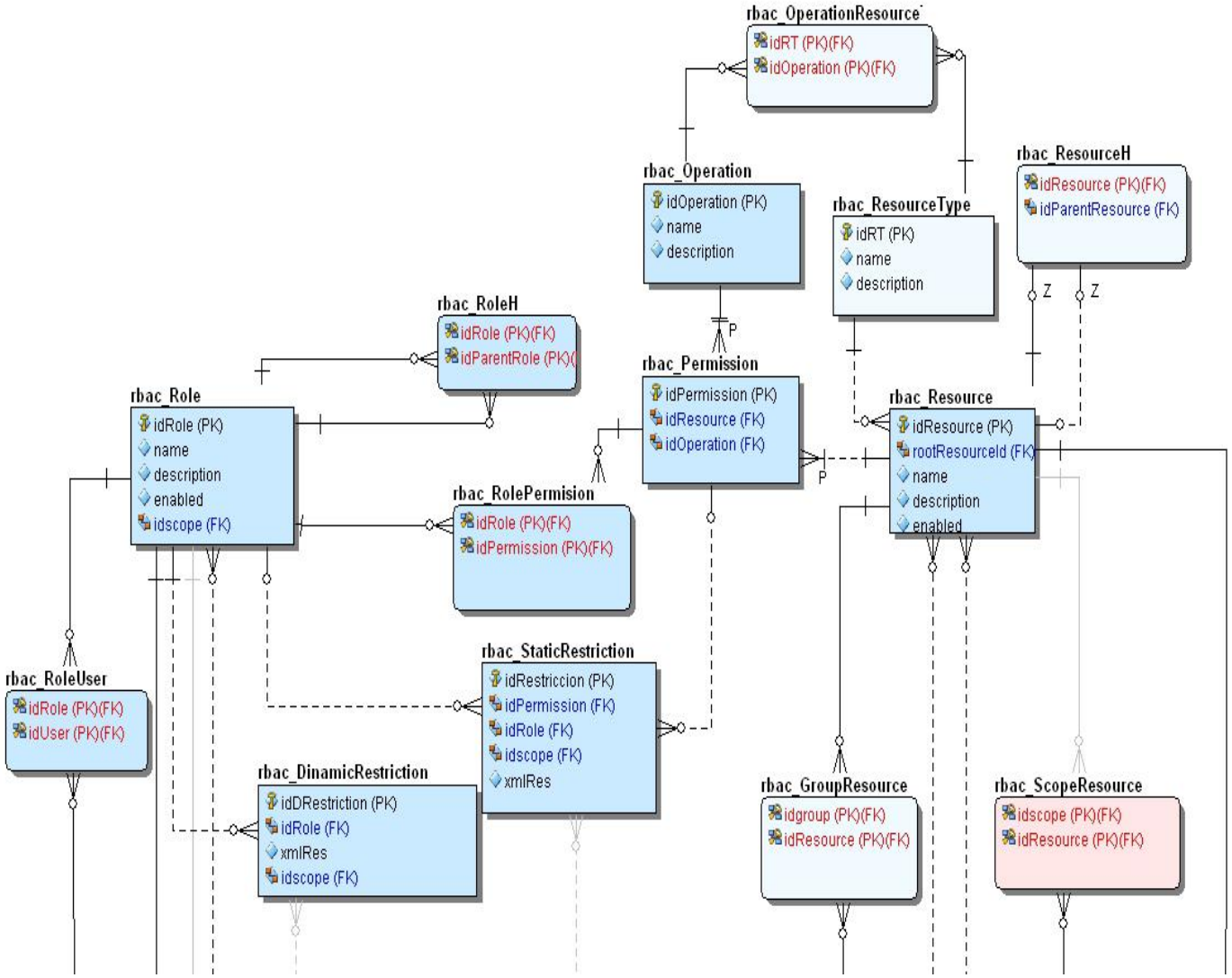
Es un patrón de diseño que sirve para proporcionar una interfaz unificada de alto nivel para un conjunto de clases en un subsistema, o cuando se quiera estructurar varios subsistemas en capas, ya que las fachadas serían el punto de entrada a cada nivel, además permite desacoplar un sistema de sus clientes y de otros subsistemas, haciéndolo más independiente, portable y reutilizable (esto es, reduciendo dependencias entre los subsistemas y los clientes). Se basa en una clase sencilla con un grupo de métodos, donde cada uno realiza internamente una operación específica, en este caso el cliente no se comunica directamente con las clases que implementan estos métodos, si no con las interfaces que se encargan de exponer estos métodos a los clientes, con el fin de llevar a cabo la correcta lógica del servicio. Este patrón se refleja en gran parte del proyecto, específicamente en la utilización de interfaces en las tres capas del mismo, permitiendo que cada capa sea independiente de las otras y la comunicación entre ellas sea a través de sus interfaces.

3.1.2. Modelo de datos del servicio

Un modelo de datos es una representación lógica y física de los datos persistentes usados por el servicio, permite describir:

- ✓ **Las estructuras de datos:** El tipo de datos que hay en la base de datos y la forma en que se relacionan.

- ✓ **Las restricciones de integridad:** Un conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
- ✓ **Operaciones de manipulación de los datos:** Operaciones de consulta de la base de datos.


rbac_OperationResource
idRT (PK)(FK)
idOperation (PK)(FK)
rbac_Operation
idOperation (PK)
name
description
rbac_ResourceType
idRT (PK)
name
description
rbac_ResourceH
idResource (PK)(FK)
idParentResource (FK)
rbac_RoleH
idRole (PK)(FK)
idParentRole (PK)(
rbac_Permission
idPermission (PK)
idResource (FK)
idOperation (FK)
rbac_Role
idRole (PK)
name
description
enabled
idscope (FK)
rbac_Resource
idResource (PK)
rootResourceId (FK)
name
description
enabled
rbac_RolePermision
idRole (PK)(FK)
idPermission (PK)(FK)
rbac_StaticRestriction
idRestriccion (PK)
idPermission (FK)
idRole (FK)
idscope (FK)
xmlRes
rbac_RoleUser
idRole (PK)(FK)
idUser (PK)(FK)
rbac_DinamicRestriction
idDRestriction (PK)
idRole (FK)
xmlRes
idscope (FK)
rbac_GroupResource
idgroup (PK)(FK)
idResource (PK)(FK)
rbac_ScopeResource
idscope (PK)(FK)
idResource (PK)(FK)

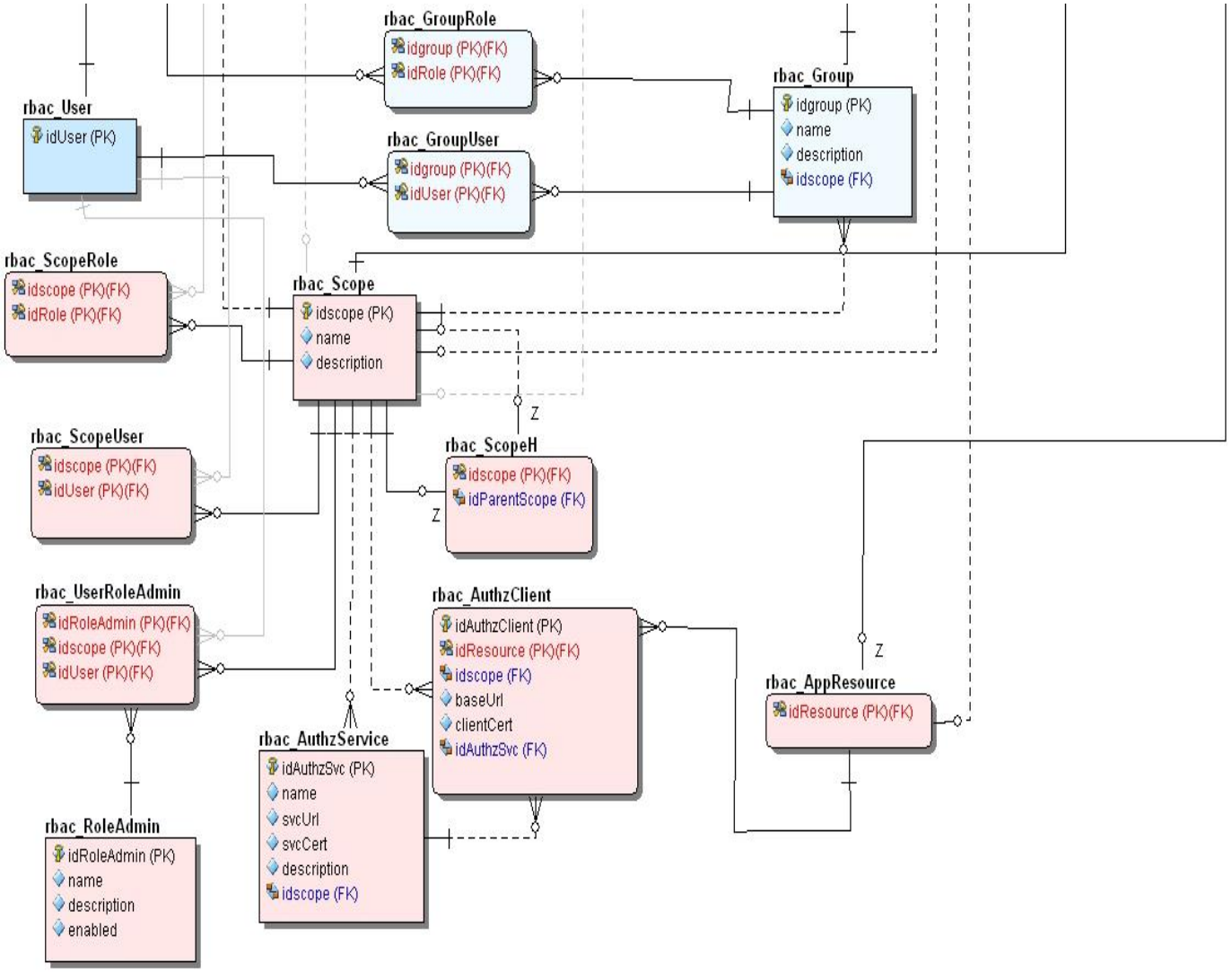


Figura 5. Diagrama del Modelo de datos.

3.1.3. Diagrama de aplicación

Según la metodología utilizada otro de los aspectos fundamentales que hay que tener en cuenta a la hora de definir la arquitectura del servicio, es realizar el diagrama de aplicación, el mismo se realiza con el objetivo de dar una perspectiva de todos los componentes que se relacionan con el servicio. Muestra todos los elementos desplegables tales como servicios y aplicaciones *web*.

Además, hace una referencia a las aplicaciones, bases de datos y servicios externos, reflejando de esta manera la actual configuración de la propuesta de solución.

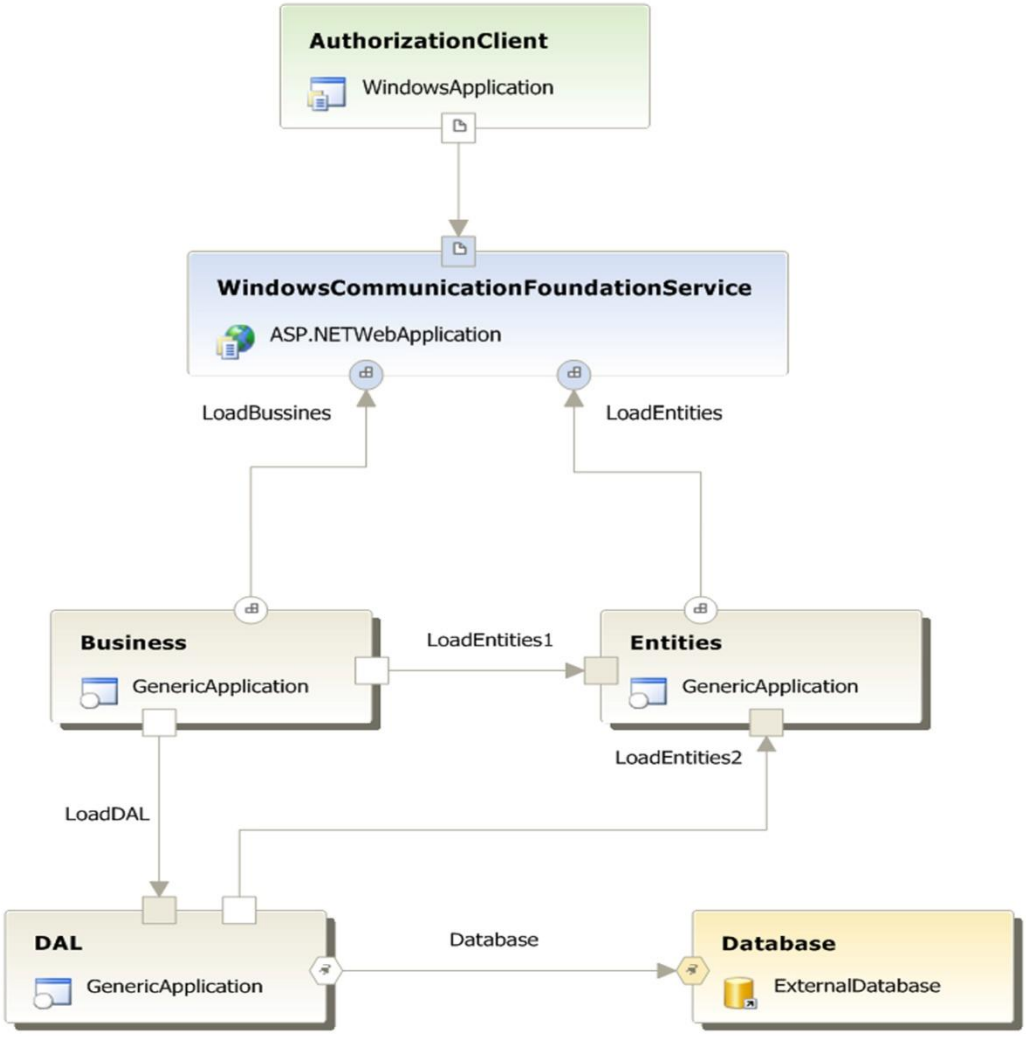


Figura 6. Diagrama de aplicación.

3.1.4. Diagrama lógico de centro de datos

El diagrama lógico de centro de datos define o documenta las configuraciones específicas de aplicaciones de tipo servidor, las cuales pueden ser *IIS*, *SQL Server* o *BizTalk Server.* Tiene el propósito específico de asegurar la conexión de entrada y salida del servidor web. El diagrama

muestra como estas aplicaciones configuradas lógicamente se interconectan. El diagrama de aplicaciones muestra los elementos de despliegue, como los servicios *web*, aplicaciones *web*, aplicaciones *Windows*, Bases de datos y servicios web externos mostrando las conexiones entre estas aplicaciones, reflejando la configuración actual de la solución.

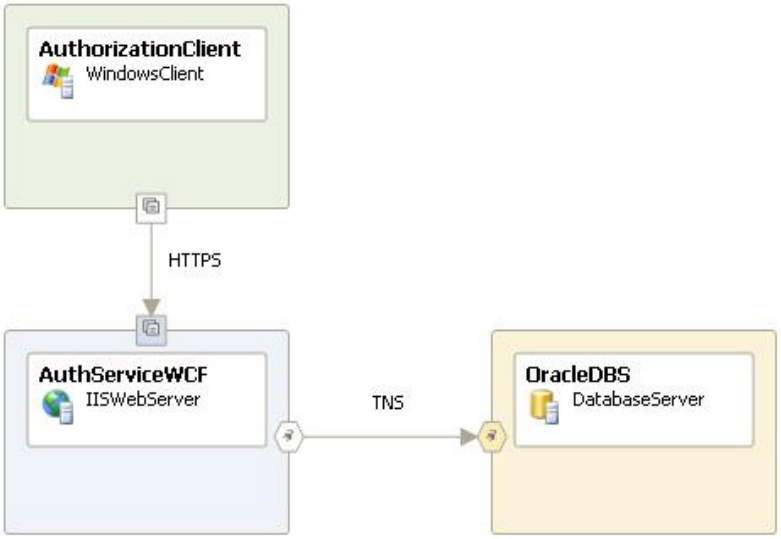


Figura 7. Diagrama lógico de centro de datos.

3.1.5. Diagrama de clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura del servicio mostrando sus clases, atributos y las relaciones entre ellos.

En el siguiente diagrama se muestran los atributos y métodos de cada clase identificada en la solución propuesta y se representa de una forma sencilla la colaboración y las responsabilidades de las distintas clases que conforman el servicio.

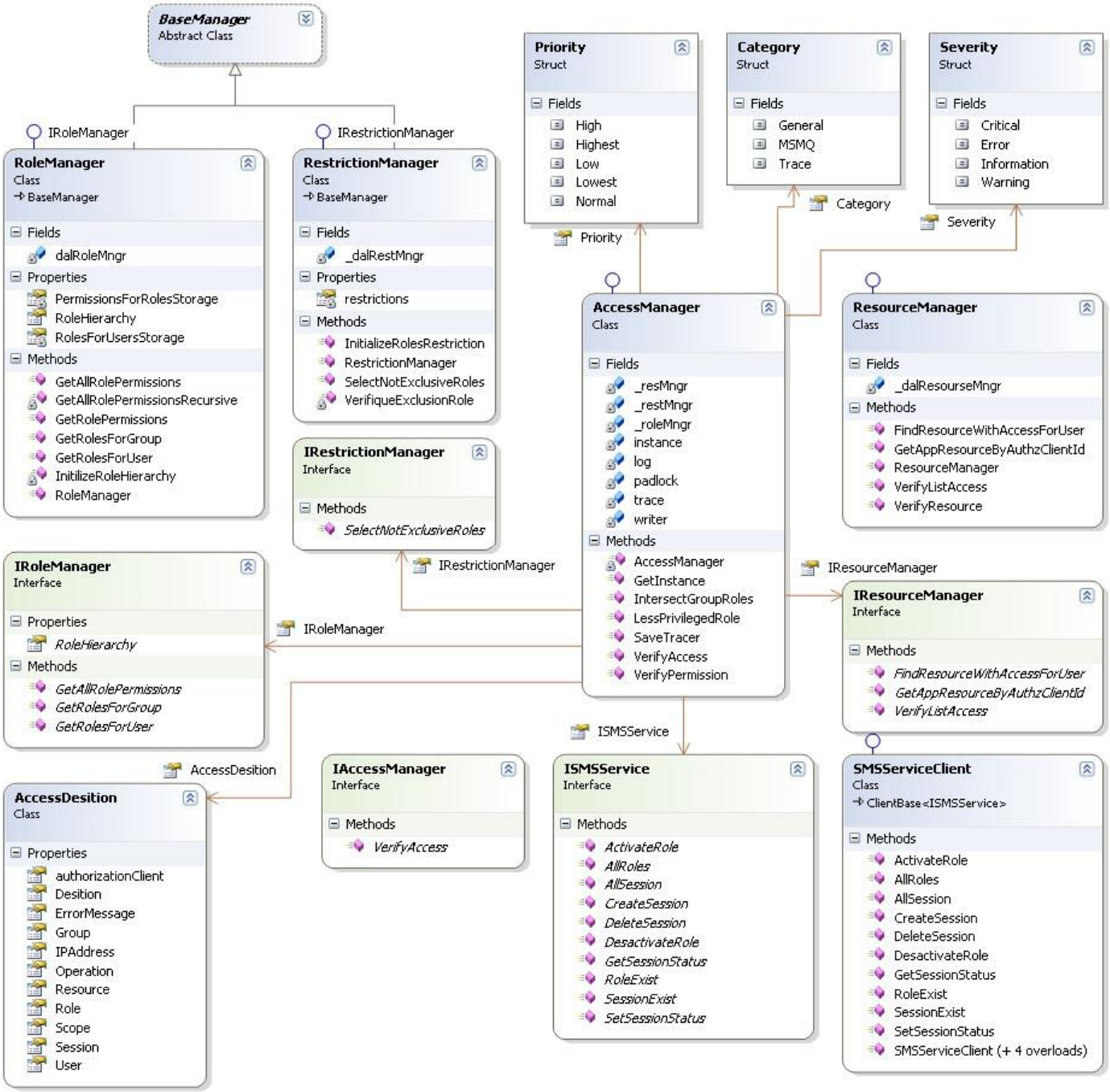


Figura 8. Diagrama de clases.

3.1.6. Descripción de las clases controladoras

Las clases controladoras tienen la responsabilidad de manejar las entidades del negocio y la información necesaria a persistir en la base de datos. En el diagrama propuesto estas clases implementan interfaces que exponen las funcionalidades del sistema, logrando independencia entre las diferentes capas y posibilitando la escalabilidad del sistema.

Tabla 14. Descripción de la clase RoleManager.

Nombre	RoleManager
Tipo de Clase	Controladora
Descripción	Esta clase implementa todos los métodos de la interfaz IRoleManager, donde se manejan todas las operaciones relacionadas con los roles.
Atributos	
Nombre	**Descripción**
dalRoleMngr:IDALRoleManager	Objeto de la clase interfaz de acceso a datos IDALRoleManager.
Métodos	
GetAllRolePermissions(Role role): HashSet<Permission>	Devuelve todos los permisos asignados ya sean directa o indirectamente al rol pasado por parámetro.
GetRolePermissions(String Id): HashSet<Permission>	Devuelve todos los permisos asignados directamente al rol pasado por parámetro.
GetRolesForGroup(Group group): HashSet<Role>	Retorna todos los roles de un grupo especificado.
GetRolesForUser(User ser): HashSet<Role>	Devuelve todos los roles para el usuario especificado.
Asociaciones	

Tabla 15. Descripción de la clase AccessManager.

Nombre	AccessManager
Tipo de Clase	Controladora
Descripción	Esta clase implementa todos los métodos de la interfaz IAccessManager, donde se manejan las acciones relacionadas con las peticiones de acceso que se realicen.
Atributos	
Nombre	**Descripción**
_resMngr: IResourceManager	Objeto de la clase interfaz IResourceManager.
_restMngr: IRestrictionManager	Objeto de la clase interfaz IRestrictionManager
_roleMngr: IRoleManager	Objeto de la clase interfaz IRoleManager
Métodos	
Nombre	**Descripción**
VerifyAccess(string scopeId, string groupId, string authorizationClientId, string idsesion, string userId, string operationId, string resourceId, string	Retorna una decisión de acceso a la petición realizada por un determinado usuario.
IntersectGroupRoles(string groupId, IEnumerable<Role>rolesForUser): HashSet<Role>	Retorna la intersección de los roles de un usuario con los que pertenezcan a un grupo determinado.
LessPrivilegedRole(List<Role>rolesWithAccess): Role	Devuelve el rol con menor privilegio de acceso a un recurso de una lista de roles pasada por parámetro.
VerifyPermission(Role role, string operationId, string pathHash): bool	Verifica si el rol pasado por parámetro tiene el permiso de realizar una operación determinada sobre un recurso dado.
SaveTracer(string title, string mensaje, string categ, int priority, System.Diagnostics.TraceEventType severity): void	Método encargado de guardar una traza de la petición anteriormente realizada con algunos parámetros de la misma.
Asociaciones	

3.1.7. Descripción de las clases persistentes

Las clases persistentes se definen para conocer la información real representada en las tablas de la base de datos. De forma general, las clases persistentes y sus atributos son análogos a las entidades lógicas y sus atributos. Las descripciones de las demás clases persistentes se encuentran en los anexos. (Ver anexo 2).

Tabla 16. Descripción de la clase entidad Role.

Nombre	Role
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
Id	Guid
Name	string
Description	string
Enable	bool
RoleParents	List<string>
Responsabilidad	
Nombre	**Descripción**
Role()	Constructor
Role(string name)	Constructor
Role(Guid id, string name)	Constructor
Role(Guid id, string name, string description, bool enabled)	Constructor
Role(Guid id, string name, string description, bool enabled, List<string> roleParents)	Constructor

Capítulo 3

3.2. Pruebas

Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad. Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del *software* y representa una revisión final de las especificaciones del diseño y de la codificación.

La creciente inclusión del *software* como un elemento más de muchos sistemas y la importancia de los costos asociados a un fallo del mismo, han motivado la creación de pruebas más minuciosas y bien planificadas.

3.2.1. Pruebas unitarias

Las pruebas unitarias son un punto básico para mejorar la calidad de la propuesta de solución, además son la mejor forma de detectar errores tempranamente en el desarrollo. Una prueba unitaria no es más que código que prueba otro código, chequeando el correcto funcionamiento del mismo.

Microsoft Visual Studio Team System 2010 permite generar pruebas específicas para métodos, facilitando los pasos necesarios para crear y personalizar pruebas unitarias, además de ejecutarlas y examinar sus resultados. A continuación se describen las pruebas realizadas con la herramienta del *Visual Studio* a las funcionalidades principales del servicio. Estas pruebas fueron realizadas al culminar el desarrollo, luego de la última iteración. Las restantes pruebas unitarias realizadas a otras funcionalidades se encuentran en los anexos. (Ver anexo 3).

Resultados de las pruebas unitarias

Para el desarrollo del servicio se tuvo en cuenta 2 iteraciones. En ellas se realizaron pruebas unitarias de cada uno de los escenarios del servicio. A continuación se muestra el código de uno

de los métodos al que se le realizó la prueba unitaria en la última iteración para verificar su correcto funcionamiento. Las pruebas se le aplican a cada una de las funcionalidades del servicio, se le generan los resultados esperados y se comparan con los resultados arrojados de la invocación del servicio.

A continuación se muestran las pruebas realizadas al método ***"VerifAccess":***

```
/// <summary>
///A test for VerifAccess
///</summary>
[TestMethod()]
public void VerifAccessTest()
{
    GyesAuthorizationService target = new GyesAuthorizationService(); // TODO: Initialize to an appropriate value
    string scopeId = "baf00e29-29fe-4fad-a425-831f4a429c64"; // TODO: Initialize to an appropriate value
    string groupId = "baf00e29-29fe-4fad-a425-831f4a429c64"; // TODO: Initialize to an appropriate value
    string authorizationClientId = "472a97a4-97a6-54f8-e040-007f01001eb5"; // TODO: Initialize to an appropriate value
    string sessionId = "472a97a4-97a6-54f8-e040-007f01001eb5"; // TODO: Initialize to an appropriate value
    string userId = "472a97a4-97a6-54f8-e040-007f01001eb5"; // TODO: Initialize to an appropriate value
    string operationId = "472a97a4-97a6-54f8-e040-007f01001eb5"; // TODO: Initialize to an appropriate value
    string resourceId = "472a97a4-97a6-54f8-e040-007f01001eb5"; // TODO: Initialize to an appropriate value
    string ipaddress = "192.168.103.17"; // TODO: Initialize to an appropriate value
    bool expected = false; // TODO: Initialize to an appropriate value
    bool actual;
    actual = target.VerifAccess(scopeId, groupId, authorizationClientId, sessionId, userId, operationId, resourceId, ipaddress);
    Assert.AreEqual(expected, actual);
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Las pruebas realizadas a la funcionalidad ***"VerifAccess"*** después de especificarle parámetros reales, arrojó como resultado la respuesta que se esperada. Los resultados se muestran en la siguiente tabla:

Tabla 17. Descripción de la prueba unitaria "VerifAccessTest".

Pruebas de Unidad		
Nombre de la Prueba: *VerifAccessTest*		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima Ejecución:** 05/06/2011

Ejecutado por: Yusmisleidy Fernández Placeres	**Verificado por:** Maikel de la Torre Luis

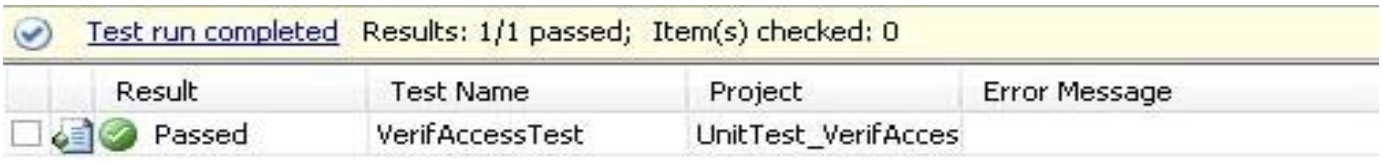
Descripción:

Para ejecutar la prueba se debe introducir un tipo de dato *string* correspondiente a una serie de parámetros que conforman la petición de acceso de un usuario. El método debe verificar cada parámetro y enviar una respuesta ya sea *verdadero o falso*.

Entrada: (*string scopeId, string groupId, string authorizationClientId, string sessionId, string userId, string operationId, string resourceId, string ipaddress*)

Criterio de aceptación: Muestra una respuesta de acceso en este caso *true*.

Resultado:



A continuación se muestran las pruebas realizadas al método ***"VerifyPermission"*:**

```
/// <summary>
///A test for VerifyPermission
///</summary>
[TestMethod()]
public void VerifyPermissionTest()
{
    // TODO: Initialize to an appropriate value
    string scopeId = "03ef403a-f895-4624-b3c0-1d62cc4b1821";
    // TODO: Initialize to an appropriate value
    AccessManager target = AccessManager.GetInstance(scopeId);
    // TODO: Initialize to an appropriate value
    Role role = new Role(new Guid("0e5f7129-da14-4638-9000-d2c37531ef9f"), "RolQ");
    // TODO: Initialize to an appropriate value
    string operationId = "8cdc0aa4-4720-6ab8-e040-007f01003565";
    // TODO: Initialize to an appropriate value
    string pathHash = "0sKd5uC6/pOh/X8zOVKxHg==";
    // TODO: Initialize to an appropriate value
    bool expected = true;
    bool actual;
    actual = target.VerifyPermission(role, operationId, pathHash);
    Assert.AreEqual(expected, actual);

}
```

Luego de realizarle pruebas unitarias a la funcionalidad se pudo comprobar que el objeto de tipo *Role* que se especificó por parámetro contenía entre sus permisos al permiso que igualmente se especificó a través de los parámetros "*operationId*" y "*pathHash*". El resultado de la prueba se puede observar en la siguiente tabla:

Tabla 18. Descripción de la prueba unitaria "VerifyPermissionTest".

Pruebas de Unidad		
Nombre de la Prueba: VerifyPermissionTest		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima Ejecución:** 05/06/2011
Ejecutado por: Angel Arias Baños	**Verificado por:** Maikel de la Torre Luis	
Descripción: Para la ejecución de la prueba se debe introducir una serie de parámetros los cuales el método utiliza para verificar si un rol tiene permisos para realizar una operación sobre un recurso y posteriormente enviar una respuesta ya sea *verdadero o falso*.		

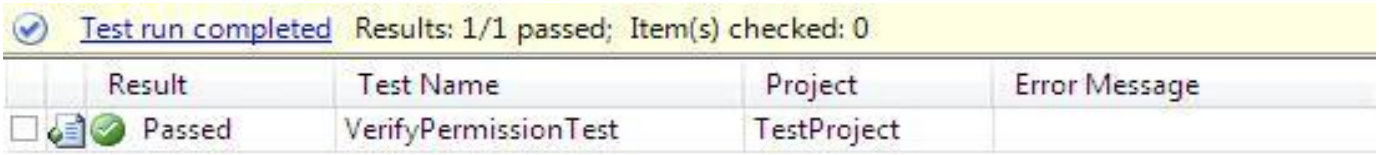
Entrada: (*Role role, string operationId, string pathHash*)
Criterio de aceptación: Se muestra una respuesta de acceso en este caso *true*.
Resultado: 

Tabla 19. Pruebas a los escenarios por iteración del servicio.

Escenarios	Iteración 1	Iteración 2
Chequear acceso a un recurso.	No Pasó	Pasó
Chequear acceso a un recurso para roles activos en sesión.	Pasó	Pasó
Chequear acceso a un recurso para roles asignados al usuario.	No Pasó	Pasó
Chequear acceso a un listado de recursos.	-	Pasó
Obtener listado de recursos de un tipo específico.	-	Pasó

3.2.2. Pruebas de carga

En la última iteración del desarrollo del Servicio de Autorización, se le realizaron pruebas de carga, específicamente para determinar rendimiento bajo condiciones extremas de alta concurrencia. Se hicieron peticiones simultáneas al servicio a través de varios hilos.

Las pruebas al servicio se realizaron en una máquina con las siguientes características:

Tabla 20. Descripción del hardware de la máquina de prueba.

Hardware	Máquina
Microprocesador	Intel(R) Celeron(R) CPU 2.50GHz
Memoria RAM	1 GB
Ancho de Banda	100 Mbps
Disco Duro	150 GB

Para realizar las pruebas de carga se utilizó la herramienta del *Visual Studio* 2010 (*Load Test*) a la cual se le definió realizar 25 hilos de peticiones constantes, estas peticiones fueron persistentes por 10 minutos al servicio. Los resultados de las pruebas realizadas están comprendidos en una tabla que contiene el total de pruebas, los errores y el promedio de tiempo por pruebas. También se muestran gráficos de comportamiento del promedio de pruebas por tiempo.

Tests

Test	Scenario	Total	Passed	Failed	Tests/Sec	Test Time	95% Test Time
FindResourceWithAccessForUserT...	Scenario1	1.409	1.409	0	2,39	0,53	1,80
VerifAccessTest	Scenario1	1.464	1.461	3,00	2,61	0,63	0,01
VerifyListAccessTest	Scenario1	1.398	1.398	0	2,37	0,96	2,63

Figura 9. Resultados de la prueba de carga.

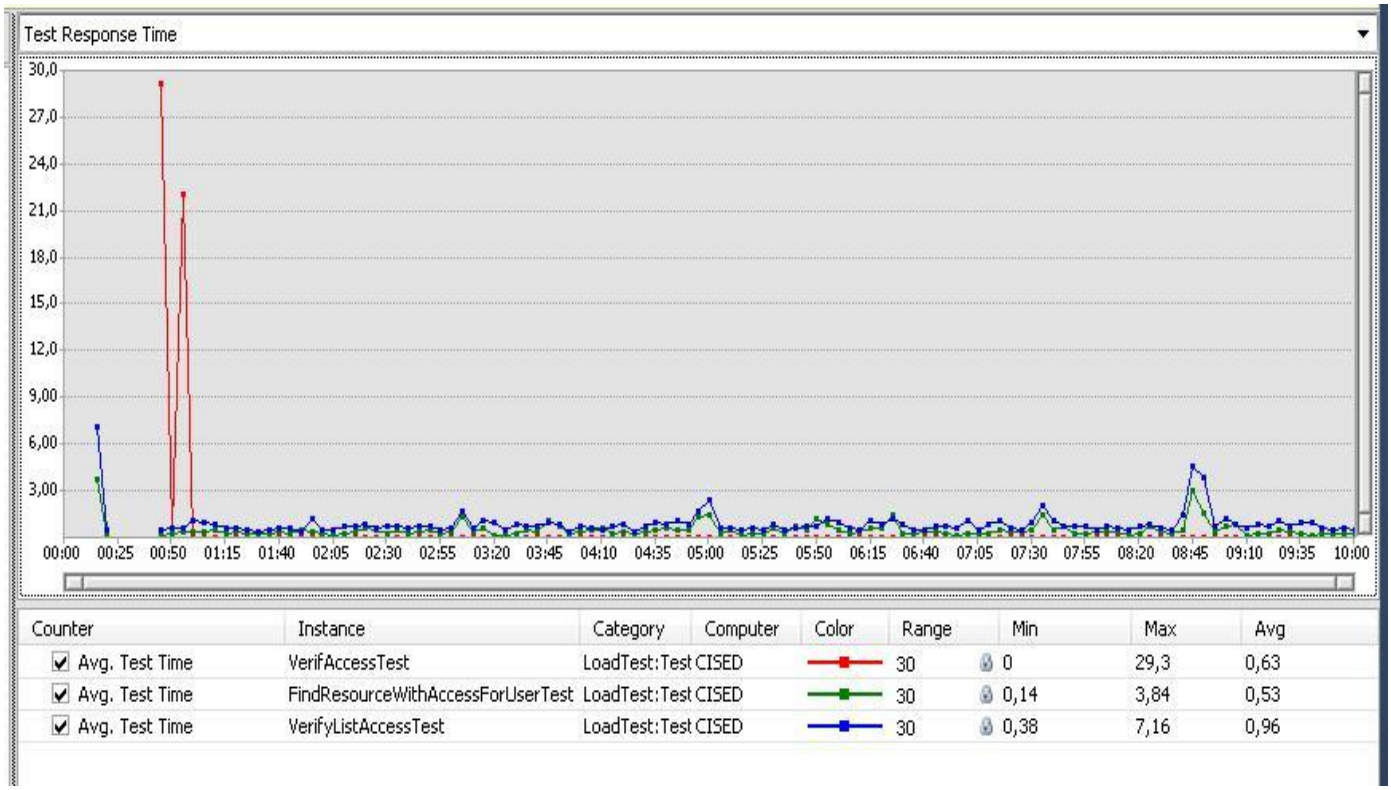


Figura 10. Gráfico de tiempo de respuesta de las funcionalidades respecto al tiempo de pruebas total.

✓ Conclusiones de las pruebas de carga

Luego de realizar las pruebas en una máquina con las características planteadas anteriormente (Ver Tabla 20), se pudo observar que los tiempos de respuesta estaban acorde con los requerimientos de calidad de servicio definidos en la fase planificación, cumpliendo a su vez con las necesidades y expectativas de rendimiento del mismo.

Capítulo 3

3.3. Conclusiones del capítulo

El diseño del servicio se realizó a partir de una arquitectura basada en tres capas, que garantizó lograr altos niveles de reutilización de código. Durante la fase de desarrollo se utilizaron diferentes patrones como el de instancia única y el de fachada, que permitieron mejorar el rendimiento del servicio. Las descripciones de las clases, sirvieron para un mayor entendimiento del proceso, de manera que quedaron bien detalladas todas las funcionalidades de cada una de las clases y sus atributos.

Con el objetivo de validar la solución propuesta se condujeron un conjunto de pruebas, enfocándose fundamentalmente en las pruebas unitarias, sin descartar las pruebas de carga que son necesarias para poder medir sus capacidades y las condiciones en las cuales trabaja realizando una cantidad definida de peticiones. Se anotaron y evaluaron los resultados obtenidos.

CONCLUSIONES GENERALES

El estudio del estado del arte sobre diferentes temas, permitió centrar la investigación en cómo administran los sistemas de administración de identidades el proceso de autorización de usuarios, facilitando la elaboración de la propuesta de solución.

Para realizar un correcto diseño e implementación del servicio, se hace uso de lenguajes, herramientas y tecnologías de avanzada, entre las que se encuentran *Windows Communication Foundation*, *Microsoft Enterprise Library 5.0*, *Visual Studio* 2010 entre otras, ya que se acoplan al ambiente de desarrollo del proyecto y a las necesidades del servicio.

La solución propuesta se ajusta con la metodología especificada, MSF ágil. La arquitectura del servicio de autorización se conformó en tres capas bien estructuradas, teniendo en cuenta en el acceso a datos, la utilización de *Entity Framework* de .NET como ORM.

Partiendo de la aplicación de pruebas al servicio, se comprobó que las funcionalidades descritas satisfacen las necesidades del mismo. El estudio realizado permitió probar la idea a defender planteada teniéndose los resultados esperados en cuanto a tiempo de desarrollo.

Recomendaciones

RECOMENDACIONES

Se recomienda que en futuras versiones del servicio, se optimice aun más el mismo teniendo en cuenta los siguientes aspectos:

- ✓ Buscar un mecanismo que permita que cuando se hagan cambios en el repositorio de políticas de acceso del RBAC, mediante la consola de administración que forma parte del componente de autorización que se desarrolla en el proyecto, se actualice la información que mantiene en caché el servicio para su correcto funcionamiento.
- ✓ Implementar una mejor estrategia para la selección del rol de menor privilegio, teniendo en cuenta niveles de acceso establecidos a nivel de diseño.
- ✓ Mejorar el rendimiento del servicio a través de mejoras en las consultas de acceso a datos.

Referencias Bibliográficas

REFERENCIAS BIBLIOGRÁFICAS

1. UCI. *Dpto de Seguridad Digital.* [En línea] Introducción a la Seguridad Informática. http://eva.uci.cu..

2. Evolucy. [En línea] diciembre de 2010. http://www.evolucy.com/esp/digital_identity.html.

3. *Borrmart*. [En línea] 13 de enero de 2011. http://www.borrmart.es/articulo_redseguridad.php?id=819&numero=20..

4. **Microsoft.** MSDN. [En línea] 2010. http://msdn.microsoft.com/en-us/library/syf5yeat.aspx.

5. *Documento Word.* [En línea] concepto. www.soeduc.cl/apuntes/concepto%20de%20auditoria.doc.

6. **IBM Corporation.** [En línea] http://www.infovision.info/clientes/appscan/IBM_TAM_ESP.pdf. .

7. **IBM.** [En línea] http://publib.boulder.ibm.com/tividd/td/ITAME/SC32-1132-01/en_US/HTML/am410_admin33.htm.

8. **Novell.** Novell. [En línea] http://www.novell.com/es-es/products/accessmanager/.

9. **CA.** CA. [En línea] http://www.ca.com/ar/products/overview.aspx?id={8371D4FB-85CB-4474-8E91-AD51D3E6082D}.

10. *Computernet*. [En línea] 2010. http://www.computernet.co.cr/index.php?option=com_content&view=category&layout=blog&id=4&Itemid=9.

11. **Oracle.** [En línea] http://download.oracle.com/docs/cd/E12530_01/oam.1014/b32420/v2authz.htm#BABJCFDE.

12. Sistedes. *Sociedad de Ingienería de Software y Tecnología de Desarrollo de Software.* [En línea] www.sistedes.es/TJISBD/Vol-1/No-1/.../SCHA-07-Sanchez-Acceso.pdf.

13. Scribd. [En línea] http://www.scribd.com/doc/12983329/Metodologia-de-Desarrollo-de-Software.

14. **Microsoft Corporation.** *MSF for Agile Software Development Process Guidance.* [En línea]

15. **Apache.** [En línea] http://thrift.apache.org/.

Referencias Bibliográficas

16. [En línea] http://upcommons.upc.edu/pfc/bitstream/2099.1/9465/1/memoria.pdf.

17. **Microsoft.** *MSDN. ¿Qué es Windows Communication Foundation?* . [En línea] 16 de octubre de 2010. HTTP://MSDN.MICROSOFT.COM/ES-ES/LIBRARY/MS731082.ASPX..

18. —. MSDN Library . *MSDN Library.* [En línea] http://www.msdn.microsoft.com/en-us..

19. Blogs.msdn. [En línea] [Citado el: 16 de enero de 2011.] http://blogs.msdn.com/b/mvplead/archive/2010/04/21/microsoft-enterprise-library-5-0.aspx.

20. *Blog de Soma en español. Nuevas ofertas para Visual Studio 2010.* [En línea] 15 de diciembre de 2010. http://blogs.msdn.com/b/somaespanol/archive/2010/03/02/nuevas-ofertas-para-visual-studio-2010.aspx..

21. *Canal Visual basic .net*. . [En línea] http://www.canalvisualbasic.net/manual-net/c-sharp/#cSharp..

22. *FlupsNet*. *FlupsNet*. [En línea] http://flups.net/e-books-y-tutoriales-f11/c-sharp-el-lenguaje-del-futuro-t515029.html...

23. **Oracle Corporation.** [En línea] http://www.oracle.com/technology/global/lad-es/documentation/database.html#11g..

24. Danysoft. Altova UModel-Danysoft. [En línea] 22 de julio de 2010. http://shop.danysoft.com/epages/danyshop_com.sf?ObjectPath=/Shops/danyshop_com/Products/%22Altova%20UModel%22/SubProducts/%22Altova%20UModel-0001%22...

25. Soluciones. [En línea] 2009. www.soluciones-ag.com/.../Spanish_ER-Studio_Datasheet_2009.pdf.

26. **Microsoft.** *Developers code with Microsoft. .NET Framework 4.* . [En línea] http://msdn.microsoft.com/es-es/library/w0x726c2.aspx..

27. Desarrollo web. [En línea] 9 de septiembre de 2004. http://www.desarrolloweb.com/articulos/1622.php. IEEE Std 1471-2000.

Bibliografía

BIBLIOGRAFÍA

Aguirre, J.R., *Seguridad Informática y Criptografía*. p. Universidad Politécnica de Madrid. Disponible en: http://www.scribd.com/doc/4395536/Seguridad-Informatica-Introduccion-a-Seguridad-Informatica

F.Ferrarolo, D. y. K., D. Richard Role Based Access Control. **Segunda Edición**. Diponible en:

www.kernelchina.org/secpattern/RBAC.pdf.

MSDN Patrones de diseño. Disponible en:

http://msdn.microsoft.com/es-es/library/bb972240.aspx

Murcia, U. d. (2010). Sistema de Gestión de Bases de Datos. Sistema de Gestión de Bases de Datos. Universidad de Murcia. Disponible en:

www.um.es/geograf/sigmur/sigpdf/temario_9.pdf

NetWorkWorld (2011). Gestión de identidades: el escenario global IDC, en un reciente informe, muestra el escenario global de la gestión de identidades. Disponible en:

http://www.networkworld.es/Gestion-de-identidades:-el-escenario-global_IDC,-en-un-recie/seccion-tendencias/articulo-151151

Oracle (2011). Productos de Oracle. Disponible en:

www.oracle.com/es/products/database/index.html

Sandhu, R., Ferraiolo, David y Kuhn, Richard The NIST Model for Role Based Access Control: Towards A Unified Standard. Disponible en:

http://www.csrc.nist.gov/staff/Kuhn/towards-std.pdf.

GLOSARIO

Base de datos: Una base de datos es un "almacén" que nos permite guardar grandes cantidades de información relacionadas entre sí de forma organizada. Cada base de datos se compone de una o más tablas, cada tabla tiene una o más columnas y filas.

BizTalk Server: 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.

Caché: Es un conjunto de datos duplicados de otros originales, con la propiedad de que los datos originales son costosos de acceder, normalmente en tiempo, respecto a la copia en la caché.

DAC: Control de Acceso Discrecional.

DSD: Separación Dinámica de Cargas.

E-business: Se refiere al conjunto de actividades y prácticas de gestión empresariales resultantes de la incorporación a los negocios de las Tecnologías de la Información y la Comunicación (TIC).

Hyperlinks: Crea vínculos en una página web que permiten a los usuarios moverse por las páginas de una aplicación.

IIS: *Internet Information Services* es un servidor web y un conjunto de servicios para el sistema operativo *Microsoft Windows*.

Just-In-Time: Herramienta denominada compilador JIT (*Just-In-Time*) genera el código máquina real que se ejecuta en la plataforma que tenga la computadora.

J2ME: es una especificación de un subconjunto de la plataforma Java orientada a proveer una colección certificada de APIs de desarrollo de software para dispositivos con recursos restringidos.

MAC: Control de Acceso Obligatorio.

Glosario

MSF: *Microsoft Solution Framework.*

Mainframe: Una computadora central, potente y costosa, usada principalmente por una gran compañía para el procesamiento de una gran cantidad de datos.

Message Queue Server: es una infraestructura de mensajería y una herramienta de desarrollo para crear aplicaciones de mensajería distribuida para los sistemas operativos *Microsoft Windows*.

Microsoft Intermediate Lenguaje: Es una herramienta de desarrollo que compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio.

RSA: En criptografía, RSA(Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública, RSA es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

RBAC: Control de Acceso Basado en Roles.

Rol: Un rol es una función de trabajo en el contexto de una organización asociado a la autoridad y responsabilidad conferida en el usuario asignado a la función.

Servidor: Un servidor es una computadora que maneja peticiones de datos, email, servicios de redes y transferencia de archivos de otras computadoras (clientes). También puede referirse a un software específico.

SSD: Separación Estática de Cargas.

SSO: Inicio de sesión único.

Share point: Herramientas administradoras de contenidos que nos permite compartir información dentro de una organización creando páginas web de forma muy rápida y sencilla.*"SharePoint"* es una abreviatura que algunas personas usan para referirse a uno o varios productos o tecnologías de Microsoft SharePoint.

Glosario

SOAP: Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

TBAC: Control de Acceso Basado en Tareas.

Triggers: (o disparador) en una Base de datos , es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación. Un *trigger* es un bloque PL/SQL (*Procedural Language/Structured Query Language*) es un lenguaje de programación incrustado en Oracle). Asociado a una tabla, que se ejecuta cuando una determinada instrucción en SQL se va a ejecutar sobre dicha tabla.

TI: Tecnología de la Información.

URL: Localizador uniforme de recursos, más comúnmente denominado URL (sigla en inglés de *Uniform Resource Locator*), es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en *Internet* para su localización o identificación.

ANEXOS

Anexo 1 Especificación de tareas por escenarios.

El escenario "**Chequear acceso a un recurso"** se dividió en dos tareas fundamentales, que proveerán esta funcionalidad.

Tabla 21. Descripción de la tarea "Verificar sesión del usuario".

Nombre de la tarea: Verificar sesión del usuario		***Identificador: 1.2***
Objetivo del Escenario: Verificar que el usuario tenga una sesión creada.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad:*** 1	***Complejidad:*** 1
Descripción: El escenario comienza cuando el Cliente de Autorización envía la petición de acceso al Servicio de Autorización, que se encarga de verificar si el usuario tiene o no una sesión creada en el Servicio de Administración de Sesiones. Finalmente devuelve una respuesta (verdadera o falso).		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

El escenario **"Chequear acceso a un recurso para roles activos"** está conformado por cinco tareas:

Tabla 22. Descripción de la tarea "Obtener los roles del grupo".

Nombre de la tarea: Obtener roles del grupo		***Identificador: 2.2***
Objetivo del Escenario: Obtener todos los roles del grupo al que pertenece el usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción:		

El escenario comienza cuando se realiza una petición al Servicio de Autorización que contiene el ID del grupo de donde se realiza la petición. Se realizan las verificaciones correspondientes y se devuelve un listado que contiene los roles del grupo.
***Validaciones*:** Debe verificarse que el grupo exista en la BD.
Prototipo de Interfaz de Usuario:

Tabla 23. Descripción de la tarea "Interceptar roles activos del usuario con roles del grupo".

Nombre de la tarea: Interceptar roles activos del usuario con roles del grupo		***Identificador: 2.3***
Objetivo del Escenario: Interceptar los roles del usuario con los roles del grupo al que pertenezcan.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: Se interceptan los roles que tiene activo el usuario en sesión con los roles del grupo de donde se realiza la petición, finalmente se obtienen los roles activos en sesión que se pueden desempeñar en el grupo.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 24. Descripción de la tarea "Verificar permisos de los roles activos del grupo."

Nombre de la tarea: Verificar permisos de los roles activos del grupo		***Identificador: 2.4***
Objetivo del Escenario: Verificar entre los roles activos los permisos.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza cuando se pasa el ID de rol, ID del recurso y el ID de la operación para comprobar		

si los permisos que tiene asociado ese rol le permite realizar la operación sobre el recurso especificado.
***Validaciones*:**
Prototipo de Interfaz de Usuario:

El escenario **"Chequear acceso a un recurso para roles asignados al usuario"** está conformado por seis tareas:

Tabla 25. Descripción de la tarea "Obtener roles del usuario."

Nombre de la tarea: Obtener roles del usuario		***Identificador: 3.3***
Objetivo del Escenario: Obtener todos los roles del usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza cuando se introduce el ID del usuario al Servicio de Autorización devolviendo el listado de roles del usuario ya sean asignados o heredados.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 26. Descripción de la tarea "Interceptar roles asignados al usuario con los roles del grupo".

Nombre de la tarea: Interceptar roles asignados al usuario con roles del grupo		***Identificador: 3.4***
Objetivo del Escenario: Interceptar los roles asignados al usuario con los roles del grupo al que pertenece el usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: Se interceptan los roles que tiene asignado el usuario con los roles del grupo de donde se realiza la		

petición, finalmente se obtienen los roles asignados o heredados que se pueden desempeñar en el grupo.
***Validaciones*:**
Prototipo de Interfaz de Usuario:

Tabla 27. Descripción de la tarea "Determinar el rol con menor privilegio."

Nombre de la tarea: Determinar el rol con menor privilegio		***Identificador: 3.5***
Objetivo del Escenario: Determinar el rol que tenga menos privilegios de acceso.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: Comienza cuando se le pasa una lista con los roles con los que el usuario tiene acceso para determinar el rol que menor cantidad de permisos tenga, siendo el de menor privilegio.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 28. Descripción de la tarea "Activar rol en sesión con acceso".

Nombre de la tarea: Activar rol en sesión con acceso		***Identificador: 3.6***
Objetivo del Escenario: Ativar el rol en sesión si tiene acceso al recurso que intenta acceder.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: El escenario comienza cuando se crea la decisión de acceso a través del servicio de administración de sesiones, que se encarga de activar en sesión el rol con acceso.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 29. Descripción de la tarea "Crear decisión de acceso".

Nombre de la tarea: Crear decisión de acceso		***Identificador: 6***
Objetivo del Escenario: Crear una decisión de acceso por cada usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: Cada vez que un usuario realice una petición se va a crear una decisión de acceso que va a tener la información de la petición y la respuesta de la misma.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 30. Descripción de la tarea "Guardar decisión de acceso".

Nombre de la tarea: Guardar decisión de acceso		***Identificador: 7***
Objetivo del Escenario: Guardar una decisión de acceso de cada usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: Dado una petición de acceso y la respuesta a esa petición se guarda en una caché de decisiones de acceso la decisión de acceso.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 31. Descripción de la tarea "Guardar traza de acceso".

Nombre de la tarea: Guardar traza de acceso	***Identificador: 8***

Objetivo del Escenario: Guardar una traza de acceso del usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: Se guarda en un directorio que va estar en el servidor donde se va a desplegar el servicio una traza con la información referente al acceso o no del usuario, la fecha en que se realiza la solicitud, la máquina de donde proviene la misma, el rol si la respuesta de acceso fue positiva, el usuario, el tipo de evento y una descripción.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Tabla 32. Descripción de la tarea "Dar respuesta de autorización".

Nombre de la tarea: Dar respuesta de autorización		***Identificador: 9***
Objetivo del Escenario: Dar una respuesta de acceso al usuario.		
Persona: Servicio de Autorización		
Iteración: 1	***Prioridad: 1***	***Complejidad: 3***
Descripción: Se envía la respuesta cifrada al Servidor de Seguridad que determina si el usuario tiene acceso o no a realizar el permiso que solicitó previamente.		
***Validaciones*:**		
Prototipo de Interfaz de Usuario:		

Anexo 2 Descripción de las clases.

Descripción de las clases controladoras *RestrictionManager*, *ResourceManager*, *AccessDesition.*

Tabla 33. Descripción de la clase RestrictionManager.

Nombre	RestrictionManager
Tipo de Clase	Controladora
Descripción	Esta clase implementa todos los métodos de la interfaz IRestrictionManager, donde se manejan todas las operaciones relacionadas con las restricciones.
Atributos	
Nombre	**Descripción**
_dalRestMngr: IDALRestrictionManager	Objeto de la clase interfaz de acceso a datos IDALRestrictionManager.
Métodos	
Nombre	**Descripción**
SelectNotExclusiveRoles(List<Role>rolesInSesion, List<Role>rolesWithAcc): List<Role>	Devuelve una lista de roles que no son excluyentes con los roles que el usuario tiene activos en sesión.
VerifiqueExclusionRole(List<Role>RolesInSess, List<Role>ListRoleExcluy): bool	Verifica si en dos listas de roles hay al menos un rol de una lista que sea excluyente con cualquier rol de la otra lista.
InitializeRolesRestriction():void	Inicializa las restricciones entre roles.
Asociaciones	

Tabla 34.Descripción de la clase ResourceManager.

Nombre	ResourceManager

Tipo de Clase	Controladora
Descripción	Esta clase implementa todos los métodos de la interfaz IResourceManager, donde se manejan todas las operaciones relacionadas con los recursos.
Atributos	
Nombre	**Descripción**
dalResourceMngr: IDALResourceManager	Objeto de la clase interfaz de acceso a datos IDALResourceManager.
Métodos	
Nombre	**Descripción**
GetAppResourceByAuthzClientId(string authorizationClientId): string	Devuelve dado un id del cliente de autorización, la aplicación de donde se realizó la petición.
FindResourceWithAccessForUser(string type, string scop, string appid, string userId): List<Resource>	Permite obtener el listado de recursos de un tipo a los que un usuario tiene algún tipo de acceso.
VerifyListAccess(string iduser, List<string> resources, string idscope, string idapp): List<bool>	Obtiene el listado con la respuesta referente a si un usuario tiene algún tipo de acceso sobre un listado de recursos que se pasa por parámetro.
VerifyResource(List<Role> roles, Resource resource, Scope scope): bool	Verifica si algún rol de los que se pasan por parámetro tiene algún permiso sobre un recurso que se especifica.
Asociaciones	

Descripción de las estructuras *Category, Severity, Priority*.

Tabla 35. Descripción de la estructura Category.

Nombre	Category
Tipo de Clase	Estructura
Descripción	Esta estructura representa de que categoría va a ser la traza, si es de tipo general, trace o MSQM
Atributos	
Nombre	**Descripción**
General: const string	Representa al tipo de Categoría General
Trace: const string	Representa al tipo de Categoría Trace
MSQM: const string	Representa al tipo de Categoría MSQM

Tabla 36. Descripción de la estructura Severity.

Nombre	Severity
Tipo de Clase	Estructura.
Descripción	Esta estructura representa el tipo de traza, ya sea de error, información, alerta o crítica.
Atributos	
Nombre	**Descripción**
Error: const System.Diagnostics.TraceEventType	Representa al tipo de traza de error
Crítical: const System.Diagnostics.TraceEventType	Representa al tipo de traza crítical

Information: const System.Diagnostics.TraceEventType	Representa al tipo de traza de información
Warning: const System.Diagnostics.TraceEventType	Representa al tipo de traza *warning*

Tabla 37. Descripción de la estructura Priority.

Nombre	Priority.
Tipo de Clase	Estructura.
Descripción	Esta estructura representa la prioridad de cada una de las trazas guardadas.
Atributos	
Nombre	**Descripción**
Lowest: const int	Representa al tipo de prioridad muy baja
Low: const int	Representa al tipo de prioridad baja
Normal: const int	Representa al tipo de prioridad normal
High: const int	Representa al tipo de prioridad alta
Highest: const int	Representa al tipo de prioridad muy alta

Descripción de las clases entidad *AccessDesition, User, Scope, Restriction, ResourceType, Resource, Permission, Operation, Group.*

Tabla 38. Descripción de la clase AccessDesition.

Nombre	AccessDesition
Tipo de Clase	Entidad

Atributos	
Nombre	**Tipo**
Role	string
Group	string
Operation	string
User	string
Resource	string
Session	string
Desition	bool
ErrorMessage	string
Scope	string
authorizationClient	string
IPAddress	string

Tabla 39. Descripción de la clase User.

Nombre	User
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
Id	Guid
UserName	string
Roles	List<Role>
Responsabilidad	
Nombre	**Descripción**

User()	Constructor
User(GuididUser)	Constructor
User(GuididUser, string username)	Constructor

Tabla 40. Descripción de la clase Scope.

Nombre	Scope
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
Id	string
Name	string
Responsabilidad	
Nombre	**Descripción**
Scope()	Constructor
Scope(stringid)	Constructor
Scope(stringid, string name)	Constructor

Tabla 41. Descripción de la clase Restriction.

Nombre	Restriction
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
Target	Role
ExclusiveRoles	List<Role>

Responsabilidad	
Nombre	**Descripción**
Restriction()	Constructor
Restriction(Role target, List<Role> roles)	Constructor

Tabla 42. Descripción de la clase ResourceType.

Nombre	ResourceType
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
Id	Guid
Name	string
Description	string
Responsabilidad	
Nombre	**Descripción**
ResourceType()	Constructor

Tabla 43. Descripción de la clase Resource.

Nombre	Resource
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**

Id	Guid
Name	string
Description	string
MD5	string
ResourceType	ResourceType
Enable	bool
ResourceParent	Resource
Responsabilidad	
Nombre	**Descripción**
Resource()	Constructor
Resource(string name)	Constructor
Resource(string name, string description, ResourceType type)	Constructor
Resource(Guid id, string name, string description, ResourceType type)	Constructor
Resource(Guid id, string name, string description)	Constructor
Resource(Guid id, string name, string description, string mnd5)	Constructor
Resource(Guid id, string name, string description, ResourceType type, string md5)	Constructor
Resource(Guid id, string name, bool enable ,string description, ResourceType type, string md5)	Constructor

Tabla 44. Descripción de la clase Permission.

Nombre	Permission
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
Id	Guid
Operation	Operation
Resource	Resource
Responsabilidad	
Nombre	**Descripción**
Permission()	Constructor
Permission(Guid id, Resource resource, Operation operation)	Constructor

Tabla 45. Descripción de la clase Operation.

Nombre	Operation
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
Id	Guid
Name	string
Description	string

Responsabilidad	
Nombre	**Descripción**
Operation()	Constructor
Operation(Guid id, string name)	Constructor
Operation(Guid id, string name, string description)	Constructor

Tabla 46. Descripción de la clase Operation.

Nombre	Group
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
Id	string
Responsabilidad	
Nombre	**Descripción**

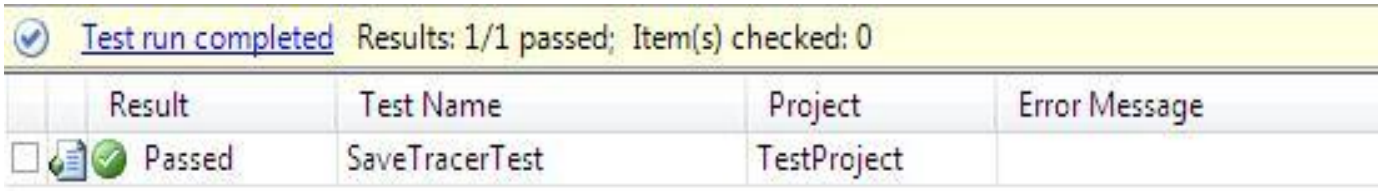
Anexo 3 Pruebas unitarias.

A continuación se muestran las pruebas realizadas al método ***"SaveTracer"*:**

```
/// <summary>
///A test for SaveTracer
///</summary>
[TestMethod()]
public void SaveTracerTest()
{
    // TODO: Initialize to an appropriate value
    string scopeId = "03ef403a-f895-4624-b3c0-1d62cc4b1821";
    // TODO: Initialize to an appropriate value
    AccessManager target = AccessManager.GetInstance(scopeId);
    string title = "Access granted"; // TODO: Initialize to an appropriate value
    // TODO: Initialize to an appropriate value
    string userid = "f4ceecf4-3b35-415c-93e8-b709bdf4e1ed";
    string sesionid = "c3f13283-f0a6-485b-b38d-b07f2710b966";
    string groupid = "6cd122d0-27a7-4005-a71d-fd8300cd2637";
    string authzclientid = "48c15aa5-d446-5763-e040-007f010043e3";
    string appid = "36ca7b84-1117-4511-9dc4-0cbccddb280c";
    string operationid = "8cdc0aa4-4720-6ab8-e040-007f01003565";
    string resourcemd5 = "0sKd5uC6/pOh/X8zOVKxHg==";
    bool desition = true;
    string roleid = "0e5f7129-da14-4638-9000-d2c37531ef9f";
    string ipaddress = "192.168.103.18";
    string mensaje = "\nUser: " + userid + "\nSesion: " + sesionid +
                     "\nGroup: " + groupid + "\nAuthzClient: " + authzclientid +
                     "\nApplication: " + appid + "\nOperation: " + operationid +
                     "\nResourses: " + resourcemd5 + "\nDesition: " + desition +
                     "\nRole: " + roleid + "\nIpAdress: " + ipaddress;
    string categ = Category.General;
    int priority = Priority.Normal;
    target.SaveTracer(title, mensaje, categ, priority, Severity.Information);
}
```

Luego de realizarle pruebas unitarias a la funcionalidad se pudo comprobar que a partir de los parámetros especificados se almacena correctamente una traza que contiene la información referente a una petición de acceso y la decisión que el servicio dio a la misma. En el fichero "*AuthzLogs.log"* fue insertado el nuevo registro. El resultado de la prueba se puede observar en la siguiente tabla:

Tabla 47.Descripción de la prueba unitaria "SaveTracerTest".

Pruebas de Unidad		
Nombre de la Prueba: VerifyPermissionTest		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima Ejecución:** 05/06/2011
Ejecutado por: Angel Arias Baños	**Verificado por:** Maikel de la Torre Luis	
Descripción: Para la realización de la prueba se introducen una serie de parámetros que son la información relacionada con petición de acceso más la decisión que el servicio concedió a esa petición, guardo toda esta información en un fichero correctamente.		
Entrada: (*string title, string mensaje, string categ, int priority, System.Diagnostics.TraceEventType severity*)		
Criterio de aceptación: Se guardó en un fichero toda la información deseada de forma correcta.		
Resultado:  Test run completed Results: 1/1 passed; Item(s) checked: 0 Result | Test Name | Project | Error Message Passed | SaveTracerTest | TestProject		

A continuación se muestran las pruebas realizadas al método ***"FindResourceAccessForUser"*:**

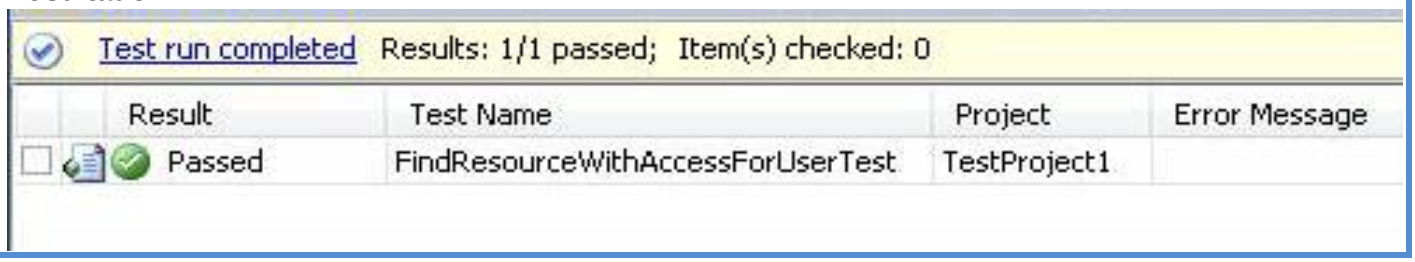
```
/// <summary>
///A test for FindResourceWithAccessForUser
///</summary>
[TestMethod()]
public void FindResourceWithAccessForUserTest()
{
    IAuthorizationService target = new GyesAuthorizationService();
    string type = "Módulo";
    string scop = "03ef403a-f895-4624-b3c0-1d62cc4b1821";
    string appid = "36ca7b84-1117-4511-9dc4-0cbccddb280c";
    string userId = "f4ceecf4-3b35-415c-93e8-b709bdf4e1ed";
    List<Resource> expected = new List<Resource>();
    Resource parent = new Resource(new Guid("36ca7b84-1117-4511-9dc4-0cbccddb280c"), "App1", "Descripción de App1");
    parent.Enable = true;
    Resource rec1 = new Resource(new Guid("08ab60b8-5478-45a1-a0af-48efbb33613c"), "Modulo1", "Modulo1");
    Resource rec2 = new Resource(new Guid("768c3fde-9a16-4377-aa94-c7054b7e0345"), "Modulo3", "Modulo3");
    Resource rec3 = new Resource(new Guid("dd9549f5-d133-4509-9124-a751cdc00ed3"), "Modulo5", "Modulo5");
    rec1.ResourceParent = parent;
    rec2.ResourceParent = parent;
    rec3.ResourceParent = parent;
    expected.Add(rec1);
    expected.Add(rec2);
    expected.Add(rec3);
    List<Resource> actual;
    actual = target.FindResourceWithAccessForUser(type, scop, appid, userId);
    if (expected.Count == actual.Count)
    {
        for (int i = 0; i < expected.Count; i++)
        {
            Assert.AreEqual(expected[i].Id, actual[i].Id);
        }
    }
    else
        Assert.Fail("La lista que se espera no tiene la misma cantidad de elementos que la que se obtiene");
}
```

Luego de realizarle pruebas unitarias a la funcionalidad se pudo comprobar que a partir de los parámetros especificados el método retorna lo que se esperaba como resultado. Esto se puede comprobar en la siguiente tabla:

Tabla 48.Descripción de la prueba unitaria "FindResourceAccessForUserTest".

Pruebas de Unidad		
Nombre de la Prueba: FindResourceAccessForUserTest		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima Ejecución:** 05/06/2011
Ejecutado por: Yusmisleidy Fernández Placeres	**Verificado por:** Maikel de la Torre Luis	
Descripción:		

Para llevar a cabo la realización de la prueba se introducen los parámetros correspondientes que permite obtener un listado de recursos a los que el usuario tiene algún tipo de acceso, y que estos sean de un tipo especificado.
Entrada: *(string type, string scopid, string appid, string userId)*
Criterio de aceptación: Devolvió un listado de recursos de un tipo específico al que el usuario tiene acceso.
Resultado:



A continuación se muestran las pruebas realizadas al método ***"VerifyListAccess"*:**

```
///A test for VerifyListAccess
///</summary>
[TestMethod()]
public void VerifyListAccessTest()
{
    // TODO: Initialize to an appropriate value
    IAuthorizationService target = new GyesAuthorizationService();
    string iduser = "f4ceecf4-3b35-415c-93e8-b709bdf4e1ed"; // TODO: Initialize to an appropriate value
    // TODO: Initialize to an appropriate value
    List<string> resources = new List<string>();
    resources.Add("Modulo1");
    resources.Add("Modulo3");
    resources.Add("Modulo5");
    resources.Add("Modulo20");
    // TODO: Initialize to an appropriate value
    string idscope = "03ef403a-f895-4624-b3c0-1d62cc4b1821";
    // TODO: Initialize to an appropriate value
    string idapp = "36ca7b84-1117-4511-9dc4-0cbccddb280c";
    // TODO: Initialize to an appropriate value
    List<bool> expected = new List<bool>();
    expected.Add(true);
    expected.Add(true);
    expected.Add(true);
    expected.Add(false);
    List<bool> actual;
    actual = target.VerifyListAccess(iduser, resources, idscope, idapp);
    if (expected.Count == actual.Count)
    {
        for (int i = 0; i < expected.Count; i++)
        {
            Assert.AreEqual(expected[i], actual[i]);
        }
    }
    else
        Assert.Fail("La lista que se espera no tiene la misma cantidad de elementos que la que se obtiene");
}
```

Con la realización de pruebas unitarias a la funcionalidad se comprobó que a partir de una serie de parámetros especificados el método retorna una lista con el usuario que se especifica y si tiene acceso o no a los recursos correspondientes, siendo este el resultado que se esperaba. El resultado de la prueba se puede comprobar en la siguiente tabla:

Tabla 49.Descripción de la prueba unitaria "VerifyListAccess".

Pruebas de Unidad		
Nombre de la Prueba: VerifyListAccessTest		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima Ejecución:** 05/06/2011
Ejecutado por: Yusmisleidy Fernández Placeres	**Verificado por:** Maikel de la Torre Luis	
Descripción: La prueba se realiza con la entrada de los parámetros correspondientes a esta funcionalidad la cual permite a partir de un listado de recursos, devuelve otro listado de booleanos, que indican si el usuario que se especifica, tiene algún tipo de acceso a los recursos respectivamente.		
Entrada: *(string iduser, List<string> res, string idscope, string idapp)*		
Criterio de aceptación:		
Resultado: