Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Aplicación para la gestión de la configuración de los parámetros para el control de acceso basado en roles en el Sistema de Administración de Identidades. Versión 2.0

Autor(es): Alejandro García Fernández

Yaimi Manso Machado

Tutor(es): Ruth Yurina Vega Cutiño

Maikel de la Torre Luis

La Habana, Junio de 2011

"Año 53 de la Revolución"

Frase

La única manera de hacer grandes trabajos es amar lo que uno hace. Si no lo encontraron todavía, sigan buscando.

Steve Jobs.

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Identificación y Seguridad Digital 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 _____.

Alejandro García Fernández

_____

Firma del autor

Yaimi Manso Machado

_____

Firma del autor

Ing. Ruth Yurina Vega Cutiño

_____

Firma del tutor

Ing. Maikel de la Torre Luis

_____

Firma del tutor

Datos de contacto

Ing. Ruth Yurina Vega Cutiño: Graduada de Ingeniería Informática en el Instituto Superior Politécnico José Antonio Echeverría. Es profesor con categoría docente Asistente y 9 años de experiencia. Se desempeña como asesora técnico-metodológica del Departamento Central de Sistemas Digitales hace cinco años en la Universidad de las Ciencias Informáticas (UCI). Ha impartido en pregrado las asignaturas: Teleinformática, Sistemas Operativos y Seguridad Informática. Ha formado parte de tribunales de tesis de pregrado como tutor, oponente y miembro del tribunal. Ha participado en conferencias y talleres sobre el tema de la seguridad de sistemas en el marco de varios eventos. Ha impartido cursos de posgrado sobre servicios telemáticos, transmisión de datos, seguridad en redes y administración de servidores Windows y Linux; tanto a profesionales de empresas foráneas como de la UCI.

ruth@uci.cu

Ing. Maikel de la Torre Luis: Ingeniero Informático, UCI 2009, Instructor recién graduado, 1 año de experiencia.

mdelatorre@uci.cu

Dedicatoria

A mis padres Fermin y Betty por ser un ejemplo para mí. A ustedes le debo lo que soy y los logros que he alcanzado no solo durante estos cinco años sino también durante toda mi vida.

Yaimi Manso Machado.

A mi familia y a mis amigos.

Alejandro García Fernández.

Agradecimientos

A mi mamá: por ser mi guía, por quererme tanto y por darme el apoyo que siempre he necesitado.

A mi papá: por ser mi inspiración y por guiarme en los caminos difíciles de la vida.

A mi familia: por confiar en mí.

A mi novio: por amarme tanto y estar conmigo en las buenas y en las malas. Gracias por existir.

A la familia de mi novio: por quererme como una hija más y demostrarme que soy muy importante para ellos.

A mis tutores: por enseñarme a trabajar con calidad.

A mi compañero de tesis: por su entrega y dedicación.

A mis profesores: por ayudarme en estos cinco años de sacrificio.

A mis amigos: por estar a mi lado y compartir momentos juntos.

Yaimi Manso Machado.

A mi familia: por darme el apoyo necesario durante estos cinco años.

A mis amigos: por haberme acompañado en este largo camino.

A mi compañera de tesis: por todo su empeño.

A mis tutores, sin ustedes este trabajo no hubiese podido salir adelante.

Alejandro García Fernández.

Resumen

En la Universidad de las Ciencias Informáticas *(UCI),* específicamente en el Centro de Identificación y Seguridad Digital *(CISED)* se trabaja en el desarrollo de un Sistema de Administración de Identidades; el que incluye entre otros, un módulo de autorización basado en el modelo de control de acceso basado en roles *(RBAC)* estandarizado por la NIST *(Instituto Nacional de Estándares y Tecnologías)*. Dicho módulo, está compuesto por el servicio que evalúa los permisos y restricciones para decidir si se da acceso al recurso que el usuario solicita y un gestor, desarrollado con tecnología web, para la configuración de los parámetros del modelo y del cual ya se tenía una primera versión funcional.

Como objetivo de este trabajo se planteó la actualización del gestor a una segunda versión teniendo en cuenta las funcionalidades que quedaron fuera de su alcance en la primera, para lo que fue necesario realizar una investigación previa del estado del arte de sistemas que implementan el proceso de autorización empleando RBAC.

Como resultado se obtuvo una aplicación web con una interfaz sencilla que garantiza la gestión de la configuración del *RBAC* en el Sistema de Administración de Identidades.. Para la implementación de la propuesta de solución, se trabajó sobre la plataforma *.NET*, por las ventajas que esta brinda. Se utilizó *Entity Framework* como tecnología para generar el acceso a datos, el *ASP .NET MVC* como *framework* para la arquitectura, y *MSF Agile* como metodología de desarrollo. Luego de realizadas las pruebas, se validó la aplicación para aumentar la calidad de la solución obtenida.

Palabras claves: administración de identidades, control de acceso, roles, permisos, seguridad.

Índice general

Índice general

Índice general

Índice de figuras

Introducción

La seguridad es uno de los aspectos más importantes que debe tenerse en cuenta a la hora de implementar un sistema. Esta es garantizada a través de tres elementos fundamentales: la confidencialidad, que es el aseguramiento de que la información es accesible solo para aquellos autorizados a tener acceso, la integridad, que es la que garantiza la exactitud y completitud de la información y los métodos de su procesamiento y la disponibilidad, que es la que permite que los usuarios autorizados tienen acceso a la información y a sus activos asociados en cualquier momento que lo necesiten.

Desarrollar un sistema seguro y que sea capaz de proteger, los recursos de las aplicaciones de la red corporativa, de accesos no deseados, es lo ideal. No son muchas las productoras de software que se dedican a productos de seguridad y específicamente a la gestión del control de acceso, lo más común, es que cada sistema informático cuente con su propio módulo de seguridad y esto trae como consecuencia complejidad en la administración e información duplicada.

En la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Identificación y Seguridad Digital (CISED), se está desarrollando un Sistema de Administración de Identidades que incluye un módulo de autorización basado en un modelo de control de acceso estandarizado. Este módulo de autorización está formado por un servicio que intercepta las peticiones de acceso a los recursos digitales y toma las decisiones de autorización basándose en la configuración de los elementos que forman la política de control de acceso y un sistema que gestiona la configuración de los elementos que forman parte del modelo de control de acceso.

Luego de una primera versión del sistema de gestión de la configuración del RBAC (*Control de acceso basado en roles*), aún existen funcionalidades que quedaron fuera de su alcance:

- ✓ El módulo de autorización actualmente está implementado con una estructura centralizada para las funciones de administración, existe un único rol administrativo que controla todos los elementos del modelo lo que no permite la descentralización de la administración.
- ✓ No tiene una estructura que permita separar los elementos por áreas administrativas.
- ✓ No tiene una interfaz gráfica para modelar y gestionar la jerarquía de recursos.
- ✓ No cuenta con un mecanismo para prevenir que usuarios accedan a módulos no autorizados.

- ✓ El funcionamiento del módulo de autorización se subordina a la alternativa de autenticación que emplea el Sistema de Administración de Identidades lo que no permite la opción de que sea empleado de manera independiente.

- ✓ El espectro de validaciones existente para el manejo de herencia de roles es limitado de acuerdo a lo que define este estándar.

Por todas las consecuencias planteadas con anterioridad surge el siguiente **problema científico**: ¿Cómo garantizar el crecimiento de la aplicación que gestiona la configuración de los parámetros del modelo de control de acceso basado en roles del Sistema de Administración de Identidades, teniendo en cuenta los requerimientos del modelo previstos para el alcance de la segunda versión de la aplicación?

El **objeto de estudio** será: El modelo de control de acceso basado en roles *(RBAC)*. El **campo de acción** se centra en: La gestión de los elementos del modelo de control de acceso basado en roles en el módulo de autorización del Sistema de Administración de Identidades.

Se definió como **objetivo general,** diseñar e implementar nuevas funcionalidades para la aplicación que gestiona la configuración de los parámetros del modelo de control de acceso basado en roles del Sistema de Administración de Identidades.

De donde se derivan los siguientes **objetivos específicos**:

- Evaluar ventajas y desventajas de otras soluciones existentes.
- Identificar las tecnologías y estándares a emplear.
- Realizar análisis, diseño e implementación de las nuevas funcionalidades.
- Rediseñar e implementar las actuales funcionalidades a partir de la incorporación de los nuevos elementos.
- Validar el sistema.

La **idea a defender** que se plantea es: La implementación de nuevas funcionalidades para el sistema de gestión de la configuración de los parámetros del modelo de control de acceso basado en roles garantizará el crecimiento de la aplicación a una segunda versión.

Para lograr que se cumplan los objetivos propuestos se trazaron las siguientes **tareas de investigación.**

- Estudio de los requerimientos del modelo de control de acceso basado en roles.
- Análisis de propuestas de solución existentes.
- Análisis de las tecnologías posibles a emplear para la implementación del software.
- Especificación de los escenarios y los requisitos de calidad de servicio de las nuevas funcionalidades a desarrollar.
- Elaboración del modelo conceptual y la vista global del sistema.
- Definición de la arquitectura del sistema.
- Especificación de los patrones de diseño y los artefactos propuestos por la metodología de desarrollo utilizada.
- Implementación de la propuesta de solución con el objetivo de dar cumplimiento a los escenarios definidos.
- Diseño e implementación del módulo Gestión de Ámbitos.
- Diseño e implementación del módulo Gestión de Grupos.
- Diseño e implementación de nuevas interfaces gráficas para el modelado y gestión de la jerarquía de recursos.
- Diseño e implementación de una alternativa de autenticación integrada o no al subsistema de autenticación del Sistema de Administración de Identidades.
- Ampliación del espectro de validaciones para el manejo de herencia de roles según lo definido en el estándar.
- Realización de pruebas unitarias a las nuevas funcionalidades implementadas.
- Realización de pruebas funcionales a las nuevas funcionalidades implementadas.

Para llevar a cabo estas tareas se utilizaron los siguientes **métodos de la investigación.**

Métodos Teóricos:

- **Analítico – Sintético:** Con el objetivo de analizar la información y la documentación relevante para el desarrollo de la aplicación enfatizando en los elementos más importantes que se relacionan con el objeto de estudio.

- **Histórico – Lógico:** Para comprobar teóricamente cómo ha evolucionado el desarrollo de la administración del RBAC y así deducir la necesidad de un sistema que facilite este proceso.

Métodos Empíricos:

- **Observación:** Se realizó una íntegra observación para ver que sucedía en el proceso de gestión del control de acceso basado en roles en diferentes sistemas y así identificar todos los problemas que se desean resolver.

Este trabajo está compuesto por **3 capítulos** que contienen la siguiente información:

Capítulo 1: Fundamentación Teórica: Se efectuará un estudio del estado del arte analizando las principales aplicaciones dedicadas a la gestión del control de acceso basado en roles, exponiendo las características esenciales que sirvan de base para el sistema propuesto. Además, se realizará el análisis de las herramientas, metodología, lenguajes y tecnologías que existen en la actualidad y que se pretenden utilizar en el proceso de desarrollo de la aplicación.

Capítulo 2: Características del Sistema: Se realizará la especificación de los escenarios y los requisitos de calidad de servicio del sistema. Además, se describirá la propuesta de solución para la futura implementación de la aplicación partiendo del análisis de la problemática en cuestión.

Capítulo 3: Desarrollo y Prueba de la solución propuesta**:** En este capítulo se modelará el diagrama de clases y se diseñará el modelo de base de datos, se presentará la interacción entre los diferentes módulos de la aplicación y se generará el diagrama de centro de datos lógicos y el diagrama de aplicación. Además, se ejecutará el plan de pruebas y se valorará el resultado de las mismas

Capítulo 1: Fundamentación teórica

1.1. Introducción

La gestión de identidades, proporciona una combinación de procesos y tecnologías para administrar y garantizar el acceso a la información y los recursos de una organización, a la vez que se protegen los perfiles de los usuarios.

Existen tres conceptos que están muy relacionados con la gestión de identidades: identificación, autenticación y autorización. La identificación es la acción de un usuario de presentar su identidad a un sistema, por su parte la autenticación es la verificación de que el usuario que trata de identificarse es válido. Las técnicas de autenticación van desde una simple entrada de identificador de usuario y contraseña, hasta potentes mecanismos como los certificados de clave pública y sistemas biométricos. La autorización es el procedimiento para determinar si el usuario o proceso previamente identificado y autenticado tiene permitido el acceso a los recursos.

El control de acceso es un conjunto más amplio de políticas dentro de un sistema de gestión de identidades que definen las normas en todo lo que a un usuario se le permite hacer en el ámbito de aplicación de dicho sistema. Diversos modelos de control de acceso especifican como se otorgan los permisos para llevar a cabo el proceso de autorización, entre ellos se encuentran el DAC *(Control de Acceso Discrecional)*, MAC *(Control de Acceso Obligatorio)*, TBAC *(Control de Acceso basado en tareas)*, considerando el control de acceso basado en roles *(RBAC)* uno de los modelos más generales por su flexibilidad y neutralidad, principal razón por lo que se ha convertido en el modelo más predominante.

El control de acceso basado en roles permite expresar de forma sencilla la política de acceso a los recursos de una organización, facilitando las tareas administrativas al separar la asignación de individuos a funciones o perfiles de trabajo y la definición de políticas de acceso (definición de roles en términos de lo que pueden hacer en el sistema). [1]

1.2. RBAC de la NIST

El Instituto Nacional de Estándares y Tecnología *(National Institute of Standard and Technology)* es una agencia de la Administración de Tecnología para promover la innovación y la competencia industrial mediante avances tecnológicos actuales, metrología y normas o estándares de forma que mejoren la estabilidad económica y la calidad de vida. El proceso de creación del estándar RBAC fue iniciado por este instituto en reconocimiento de una necesidad entre los compradores del gobierno y la industria de productos de tecnología de información.

RBAC está basado en la definición de un conjunto de elementos y de relaciones entre ellos (Ver Figura 1). A nivel 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 recursos.

En una organización, un rol puede ser definido como una función que describe la autoridad y responsabilidad dada a un usuario en un instante determinado. RBAC añade la posibilidad de modelar una jerarquía de roles de forma que se puedan realizar generalizaciones y especializaciones en los controles de acceso y se facilite el modelado de la seguridad en sistemas complejos.

Entre sus atributos incluye el concepto de sesión que no es más que la relación entre un usuario y un subconjunto de roles que son activados en el momento de establecer dicha sesión. Cada sesión está asociada con un único usuario, mientras que un usuario puede tener una o más sesiones asociadas. Los permisos disponibles para un usuario son el conjunto de permisos asignados a los roles que están activados en todas las sesiones del usuario, sin tener en cuenta las sesiones establecidas por otros usuarios en el sistema.

Otro aspecto importante en el modelo RBAC es la posibilidad de especificar restricciones. Estas restricciones son un fuerte mecanismo para establecer políticas organizacionales de alto nivel. Las restricciones pueden ser de dos tipos: estáticas o dinámicas. Las restricciones estáticas nos permiten solucionar conflictos de intereses sobre relaciones usuario/rol *(Static Separation of Duty, SSD)* y reglas de cardinalidad de roles (*Role Cardinality),* desde una perspectiva de política de seguridad. Por otro lado, las restricciones dinámicas *(Dynamic Separation of Duty, DSD)* al igual que las SSD, limitan los permisos que son disponibles para un usuario. Sin embargo DSD difieren de las SSD por el contexto en el cual estas limitaciones son impuestas. Las DSD limitan la disponibilidad de los permisos aplicando las restricciones sobre los roles que pueden ser activados durante una sesión de usuario. [2]

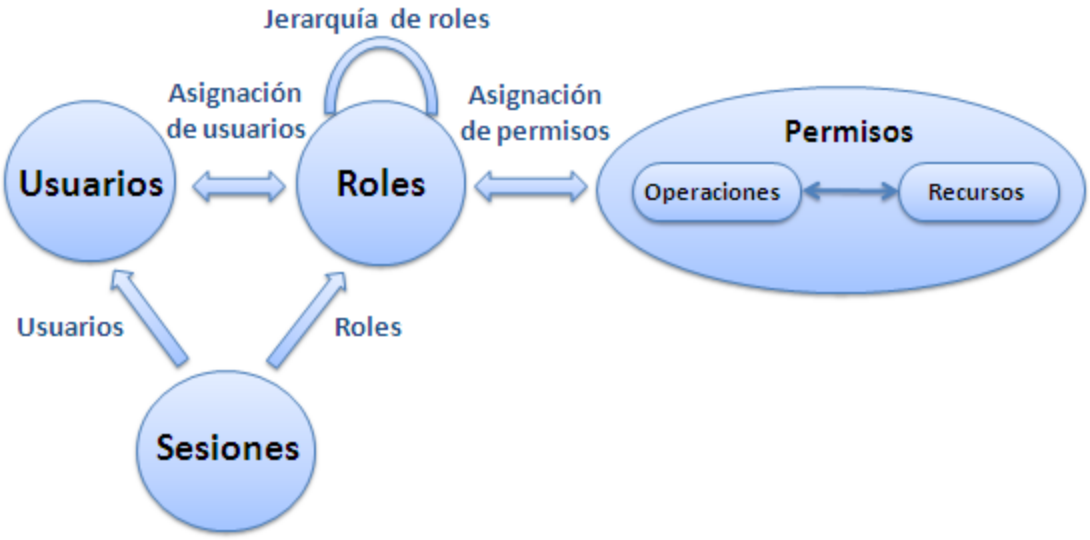


Figura 1: Modelo de control de acceso RBAC [2]

1.2.1. Niveles de RBAC

El Instituto Nacional de Estándares y Tecnología define para el control de acceso basado en roles los siguientes niveles: RBAC_0*(usuarios, roles y permisos)*, RBAC_1 *(usuarios, roles, permisos y jerarquía de roles)*, RBAC_2 *(usuarios, roles, permisos y restricciones)* y RBAC_3 *(usuarios, roles, permisos, jerarquía de roles, y restricciones)*. [3].

RBAC_0

El modelo RBAC_0, es el concepto de RBAC en donde se detalla que un usuario tiene diferentes roles y cada rol lleva asociado algún permiso.

RBAC_1

Está basado en el modelo RBAC_0, e introduce el concepto de jerarquía de roles. La jerarquía de roles básicamente es la autoridad que cada usuario tiene para cumplir uno o varios roles. Los roles son las funciones que cada usuario tiene que cumplir, es decir un rol con mayor jerarquía adquiere los derechos del rol con menor jerarquía, heredando así los privilegios del rol anterior. Por ejemplo, el usuario con mayor jerarquía dentro de una organización puede tener acceso a cualquier tipo de información, incluso a la información que pueda tener el usuario con menor jerarquía, demostrando así que un rol puede ser miembro de otro rol.

RBAC_2

Al igual que el modelo anterior, éste también está basado en el modelo de RBAC_0, pero introduce el concepto de restricciones. El uso más frecuente que se tiene con las restricciones es la separación de cargas dentro de una organización. Por ejemplo, una restricción sería que un usuario en particular debe de cumplir solamente con un rol especificado, pero que no puede efectuar funciones de otros roles.

RBAC_3

Este es el modelo más complejo de los que se han presentado con anterioridad, ya que incluye tanto las jerarquías de roles como las restricciones. En RBAC_3, las restricciones pueden ser impuestas sobre los roles jerárquicos dentro de una organización.

1.2.2. Otros atributos del RBAC

Existen atributos en la definición del RBAC que no están explícitos completamente, pues este modelo es de composición muy abierta y no sería conveniente precisar todos los aspectos de RBAC en un modelo estándar. Algunas particularidades se encuentran explicadas a continuación. [4]

- ✓ **Escalabilidad**

Un software escalable es aquel que tiene la capacidad de incrementar el rendimiento del mismo. En el modelo RBAC se puede hablar de escalabilidad cuando se refiere al número de roles y de permisos, tamaño de la jerarquía de roles, limitaciones en la asignación de roles a usuarios, entre otros. Este estándar de la NIST no incorpora el atributo de escalabilidad sin embargo es un tema muy importante que debe ser tratado para el buen funcionamiento de una aplicación.

- ✓ **Autenticación**

El estándar de la NIST no establece como implementar un sistema de autenticación específico ya que los usuarios son individualmente autenticados y asociados con los roles a que pertenecen. Este atributo esta fuera del modelo y es parte de la arquitectura del sistema, el mismo debe ser considerado y creado según las necesidades del sistema.

- ✓ **Permisos Negativos**

El modelo RBAC está basado en permisos positivos que controlan la realización de acciones en los determinados recursos, lo cual no prohíbe el uso de permisos negativos en una aplicación

determinada, estos permisos pueden ser bastante confusos especialmente en presencia de jerarquías generales.

- ✓ **Naturaleza de los permisos**

La naturaleza de los permisos no está especificada en el modelo RBAC de la NIST. Los permisos pueden ser tanto para objetos individuales como para subsistemas enteros. También pueden ser definidos para funciones primitivas u operaciones abstractas. Los mismos pueden encontrarse a la vez personalizados. La exacta naturaleza de los permisos es determinada por la naturaleza del producto. Los grandes sistemas también pueden soportar diferentes tipos de permisos, la estandarización de permisos está más allá del ámbito del objetivo general del modelo de control de acceso.

- ✓ **Activación de roles**

Este modelo no especifica la habilidad de un usuario de seleccionar que roles serán activados en una sesión en particular. Solo permite la activación de múltiples roles simultáneamente para un usuario. En algunos sistemas se permite el uso de un solo rol al mismo tiempo en una sesión, otros por su parte activan un grupo de roles por defecto y permiten a la sesión del usuario agregar o substraer roles de esta lista. El modelo RBAC de la NIST no implanta ningún requerimiento en esta área y estas reglas se deciden según las necesidades del sistema a implementar.

- ✓ **Ingeniería de roles**

La NIST no provee ninguna guía a la hora de crear roles, asignar permisos a los roles y usuarios a los roles. Esta actividad es llamada ingeniería de roles. El uso efectivo del RBAC a gran escala es fuertemente dependiente de la efectividad de la ingeniería de roles. De todas maneras esta actividad se encuentra fuera del ámbito del modelo.

- ✓ **Restricciones**

El estándar reconoce restricciones de separación de deberes, esto es requerido en el tercer nivel de la jerarquía RBAC. Existen restricciones dinámicas y estáticas, no obstante estas no son las únicas. El concepto de restricción de obligación es considerado demasiado nuevo para introducirlo en el modelo.

- ✓ **Administración del RBAC**

La autorización para la asignación de usuarios a los roles, permisos a los roles, roles a roles en caso de jerarquía y la revocación de estas asignaciones no están específicas para administrar el RBAC. Debido a esto la NIST incorpora un componente administrativo.

- ✓ **Revocación de roles**

La semántica de la revocación de roles no está definida en el modelo RBAC. El principal problema es la inmediación de la revocación. El modelo no declara cómo debe ser el comportamiento en una revocación. Este es un problema donde los desarrolladores deben poner cuidadosa atención.

1.2.3. Análisis de soluciones existentes para la gestión del RBAC

Como parte de la investigación se realiza un análisis de los sistemas informáticos existentes, que pudieran contribuir a la solución del problema planteado.

1.2.3.1. Soluciones existentes en el mundo

Dentro de la comunidad del software se han desarrollado diferentes sistemas que implementan el control de acceso a través del modelo RBAC, o que han añadido esta funcionalidad después de creados.

- ➢ **Solaris Management Console**

El modelo RBAC en el entorno operativo Solaris está basado en el inicio de sesión de los usuarios como ellos mismos y asumiendo identidades especiales que les permiten ejecutar restringidas herramientas de administración y servicios públicos.
Las herramientas proporcionadas por el entorno operativo Solaris operan con RBAC por 4 vías:

- Provee una interfaz para crear roles y mostrar los roles creados en una ventana.
- Administra los elementos de la infraestructura de RBAC.
- Restringe el acceso a la consola administrativa de las herramientas de Solaris basada en la autorización con el ámbito del servidor actual.
- Ejecuta aplicaciones heredadas con los atributos de seguridad.

El modelo RBAC introduce tres elementos en el entorno operativo Solaris:

- Roles: Una identidad especial que solo puede ser asumida por los usuarios asignados solamente.
- Autorización: Un permiso que puede ser asignado a un rol o usuario para realizar una clase de acción prohibida por las políticas de seguridad.
- Derechos de perfil: Un paquete que se puede asignar a un rol o usuario, puede consistir en: autorizaciones, comandos con atributos de seguridad y complementarios perfiles de derechos. [5]

➢ **Oracle Entitlements Management Tool**

Esta herramienta de gestión puede ser utilizada por los usuarios de negocio para gestionar los roles y permisos. Con su interfaz de usuario permite gestionar los permisos de los usuarios a través del conocido modelo de control de acceso RBAC. Además implementa la jerarquía del modelo RBAC. En este modelo, hay una relación jerárquica entre los roles padres que heredan los permisos de sus hijos y los roles hijos heredan los usuarios de sus padres. De esta manera, los datos se heredan en ambas direcciones. Utilizando un modelo jerárquico se permite agregar permisos asociados a los usuarios.

La Oracle Entitlements Management Tool proporciona una forma eficaz y flexible de gestión de roles y permisos. Esta herramienta es una aplicación web que consta de tres partes principales:

- Capa de presentación: Archivos JSP usando Java Script y etiquetas que determinan cómo se muestra la interfaz de usuario.
- Lógica del negocio: Proporciona el estado y funcionalidad de almacenamiento en caché para la gestión de derechos de usuario.
- RBAC_API: Un API de administración, construido en la parte superior de la actual BLM_API que proporciona una interfaz para administrar componentes RBAC (roles, permisos y conjuntos de permisos).

Entitlements Management Tool implementa un modelo RBAC jerárquico. En este modelo, existe una relación entre los roles. De esta manera, los datos se heredan en ambas direcciones, con la membrecía moviéndose hacia abajo y los permisos subiendo el árbol. El uso de un modelo jerárquico permite a los agregar permisos asociados a los usuarios. [6]

- **Visual Guard .Net**

Visual Guard para .Net, es una solución dirigida a cubrir los temas de seguridad de las aplicaciones .Net, no importando lo complejo de su entorno.

Es un sistema de control de acceso basado en roles RBAC. Esta herramienta añade elementos de seguridad a sus proyectos, como:

- Gestión de usuarios, membership, roles y política de contraseña.
- Definición de los permisos de los usuarios.
- Autentificación de los usuarios con el Directorio Activo *(Active Directory)* o con las cuentas de una base de datos.
- Implementación de un mecanismo de autentificación única *(Single Sing-On).*
- Registro de un historial de las transacciones sensibles (Implementación de un sistema de registro y de auditoría) [7]

✓ **Autenticación única**

Una vez que un usuario entra en una sesión, él podrá acceder a cualquier otra aplicación sin necesidad de identificarse de nuevo.

✓ **Autorización**

Las autorizaciones definen lo que un usuario puede hacer en la aplicación. Básicamente, se define lo que un usuario puede ver, hacer y modificar en la aplicación basándose en su rol. Visual Guard .Net no necesita escribir código en la aplicación para poder definir sus permisos. Su código se modifica dinámicamente en tiempo de ejecución.

✓ **Tipos de permisos que se pueden gestionar**

No existe limitación alguna en lo que concierne a los permisos que se pueden implementar al utilizar Visual Guard. Los permisos se definen de manera totalmente independiente del código de la aplicación.

✓ **Auditorías y reportes**

Se pueden generar de forma automática reportes detallados de las aplicaciones existentes, cuentas de usuario, roles, permisos, entre otros. También se pueden registrar cualquier evento de una

aplicación y revisar este registro en cualquier momento. Todos los registros de eventos son centralizados en un documento PDF.

- ✓ **Roles compartidos**

Visual Guard .Net permite compartir roles en diferentes aplicaciones. Por lo tanto no se deben gestionar roles diferentes dependiendo de la aplicación.

1.2.3.2 Selección del sistema para la gestión del RBAC

Los sistemas mencionados anteriormente poseen una alta calidad y eficiencia pero también tienen un costo elevado, el cual nuestro país no está en condiciones de afrontar, además tienen características que no se adaptan al sistema que se desea desarrollar, por lo que no constituyen una solución factible para la aplicación que gestiona los parámetros del RBAC del Sistema de Administración de Identidades.

1.3. Antecedentes

El Centro de Identificación y Seguridad Digital *(CISED)* de la Universidad de las Ciencias Informáticas *(UCI)* está desarrollando un Sistema de Administración de Identidades, el cual debe poder ser empleado por los programadores de aplicaciones web sin que estos tengan que enfrentarse a los detalles de la seguridad en cuanto a la autenticación del usuario y a la autorización de acceso a los recursos.

Este Sistema de Administración de Identidades está compuesto por cuatro módulos fundamentales: Servidor de Seguridad, Cliente de Seguridad, Gestor de Sesiones y Administrador de Acceso, que funcionan de manera conjunta para centralizar la seguridad digital de las aplicaciones empresariales en una organización.

El módulo encargado de autenticar y autorizar a los usuarios de las aplicaciones es el Administrador de Acceso, el cual debe utilizar un servicio de autorización para su funcionamiento y un sistema para la gestión de la configuración de los parámetros para el control de acceso.

1.3.1. Sistema para la gestión de la configuración de los parámetros para el control de acceso

Actualmente el CISED cuenta con una primera versión del sistema para la gestión de la configuración de los parámetros para el control de acceso basado en el modelo RBAC el cual da cumplimiento a las reglas que definen quien tiene acceso a determinado recurso sobre la base de su

rol (Ver Figura 2). Este mantiene dichos parámetros de forma centralizada, evitando así la dispersión y la falta de control sobre las configuraciones.

Figura 2: Elementos del RBAC que son gestionados [1]

El sistema de administración de la configuración del RBAC está fragmentado en siete módulos (Ver tabla 1) que abarcan todas las funcionalidades referentes a cada uno de los elementos del modelo así como otras funcionalidades que son necesarias por su vital importancia para la gestión del acceso.

Tabla 1: Módulos del Sistema de Administración de la configuración del RBAC v1.0

No.	Módulos del Sistema
1	Gestión de Roles
2	Gestión de Recursos
3	Gestión de Operaciones
4	Gestión de Permisos
5	Gestión de Restricciones
6	Gestión de Reportes
7	Gestión de Auditoría

En su vista más abstracta el Sistema de gestión de la configuración del RBAC es una solución Cliente-Servidor. La arquitectura se encuentra representada por cuatro capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades. (Ver Figura 3)

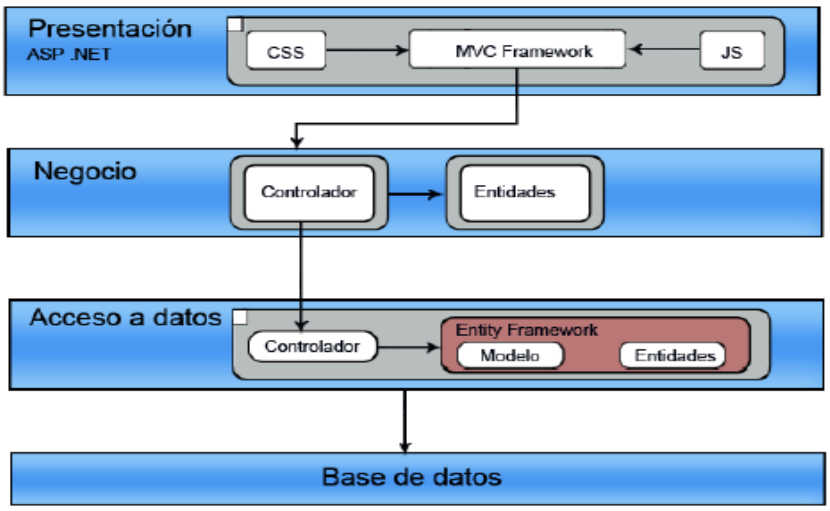


Figura 3: Vista lógica de la arquitectura del sistema v1.0 [1]

La interfaz que presenta este sistema aporta información al usuario y le permiten a su vez interactuar con esta información. Un aspecto importante que brinda es la posibilidad de que un usuario pueda escoger entre diversos enlaces para acceder a los diferentes módulos presentes en el sistema. La misma reúne los aspectos referentes a sencillez, claridad, originalidad, adaptabilidad, versatilidad e interactividad y posee una gran ventaja, agiliza las tareas de administración de cada uno de los componentes que integran este sistema.

1.4. Análisis de tecnologías existentes para el desarrollo de sistemas de gestión sobre web

Para lograr una aplicación con la calidad requerida y que cumpla con los requisitos deseados por los clientes se realiza una investigación sobre las herramientas, lenguajes y metodología que se pudieran utilizar en la creación de la solución. Estas tecnologías son las definidas en la línea de la arquitectura del Sistema de Administración de Identidades, debido a que posteriormente esta aplicación será integrada al propio sistema, además de que fueron las tecnologías utilizadas en la implementación de la primera versión del sistema para la gestión de la configuración de los parámetros para el control de acceso basado en el modelo RBAC; no obstante se realiza una investigación para mostrar las principales características de estas herramientas.

1.4.1. Metodología de desarrollo

Las metodologías de desarrollo de software van indicando paso a paso todas las actividades a realizar para lograr el producto deseado.

Las metodologías ágiles están orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

1.4.1.1 MSF Agile

MSF Agile *(Microsoft Solution Framework Agile)* es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

El proceso definido por MSF Agile incorpora ideas claves del movimiento de software ágil, junto con los principios y prácticas de MSF. MSF Agile proporciona la automatización y la orientación necesaria para apoyar el equipo de desarrollo, incluida la gestión de configuración, gestión de proyectos y seguimiento de elementos de trabajo. 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, desarrolladores y probadores.

Esta metodología tiene la ventaja de estar integrada con el Visual Studio Team System y cuenta con una serie de plantillas y guías adaptadas y orientadas a los roles definidos en cada una.

MSF Agile dispone de los tipos de elementos de trabajo siguientes:

- Escenario: Descripción de la necesidad o solicitud del usuario.
- Error: Defecto o desviación entre el comportamiento esperado y el comportamiento observado en el producto.
- Requisito de calidad de servicio: Material resultante esperado del producto final. Éste puede ser un resultado, un problema resuelto o una característica.
- Tarea: Acción independiente que debe realizar una persona o un grupo de personas.
- Riesgo: Evento o condición probable que puede dar resultados potencialmente negativos en el proyecto en el futuro. [8]

1.4.2. Lenguaje para el modelado

- **UML**

Es una especificación de notación orientada a objetos. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.

UML*(Unified Modeling Languaje)* intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que, UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama. [9]

1.4.3. Herramienta para el modelado

- **Altova UModel 2010**

Constituye el punto de salida para el desarrollo de software de éxito. Diseña visualmente modelos de aplicaciones en UML, genera código Java, C#, o Visual Basic .NET y documentación del proyecto. Realiza ingeniería inversa de los programas existentes pasándolos a diagramas UML 2, luego afina los diseños y termina con la generación de código. UModel es la herramienta UML que hace el diseño visual de software práctico para cualquier proyecto. Es una manera simple y asequible de dibujar en UML. UModel 2010 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 de software UML. [10]

1.4.3.1 Herramienta para el modelado de la base de datos

- **Embarcadero ER/Studio 8.0**

Embarcadero ER/Studio se considera el líder en modelado y diseño de base de datos, pues permite documentar y reutilizar los activos en datos. Facilita la ingeniería inversa y la optimización de las bases de datos existentes, además, con su uso se gana en productividad y fuerza el cumplimiento de los estándares de la organización ya que soporta el ciclo de vida completo de la base de datos.

Brinda a los usuarios la capacidad de visualizar, analizar y documentar cómo fluyen los datos a través de la organización. [11]

1.4.4. Lenguajes para el desarrollo

- **C#**

C# es un lenguaje moderno y orientado a objetos, con una sintaxis muy similar a la de C++ y Java. Combina la alta productividad de Visual Basic con el poder y la flexibilidad de C++. Se puede crear una gran variedad de aplicaciones en C#: aplicaciones de consola, aplicaciones para Windows con ventanas y controles, aplicaciones para la web, etc. C# gestiona automáticamente la memoria, y de este modo evita los problemas de programación tan típicos en lenguajes como C o C++. Mediante la plataforma .NET, desde la cual se ejecuta, es posible interactuar con otros componentes realizados en otros lenguajes en .NET de manera muy sencilla. También es posible interactuar con componentes no gestionados fuera de la plataforma .NET. Por ello, puede ser integrado con facilidad en sistemas ya creados. Desde C# se puede acceder a una librería de clases muy completa y muy bien diseñada, que permita disminuir en gran medida los tiempos de desarrollo. Sin embargo, la mayor potencia de este lenguaje se encuentra en Visual Studio .NET debido a la estrecha integración entre dicho entorno y C#, mayor que la que tiene Visual C++ .NET y equivalente e incluso superior a la de Visual Basic .NET. [12]

- **Java Script**

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es uno de los lenguajes de programación del lado del cliente más utilizado. Con Java Script se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Java Script y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. Es un lenguaje bastante sencillo y pensado para hacer las cosas con rapidez. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica. Entre las acciones típicas que se pueden realizar en Java Script existen dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, Java Script permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo

que se logra crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo. Java Script es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además, Java Script pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente. Con Java Script el programador, se convierte en el verdadero dueño y controlador de cada acción o evento que ocurre en la página cuando la está visualizando el cliente. [13]

- **CSS**

El principio de las hojas de estilo consiste en la utilización de un solo documento para almacenar las características de presentación de las páginas asociadas a grupos de elementos. Esto implica nombrar un conjunto de definiciones y características de presentación de las páginas, y activar esos nombres para aplicarlos a una parte del texto. Las hojas de estilo se desarrollaron para compensar los defectos de HTML con respecto a la presentación y al diseño de las páginas. HTML tiene varias etiquetas para modificar la presentación y definir los estilos del texto, pero cada elemento tiene su propio estilo, independientemente de los elementos que lo rodean. Al utilizar hojas de estilo, cuando se necesite cambiar la apariencia de un sitio que tiene cientos de páginas web todo lo que hay que hacer es editar las definiciones de la hoja de estilo en un solo lugar para cambiar la apariencia del sitio completo. Se denominan "hojas de estilo en cascada" porque se pueden definir múltiples hojas y los estilos pueden aplicarse a todas las páginas. [14]

1.4.5. Plataforma para el desarrollo

- **Plataforma .NET**

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

A continuación se resumen las ventajas más importantes que proporciona .NET:

1. Código administrado: El CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.
2. Interoperabilidad multilenguaje: El código puede ser escrito en cualquier lenguaje compatible con .NET, ya que, siempre se compila en código intermedio.
3. Compilación just-in-time: El compilador JIT incluido en el framework compila el código intermedio generando el código máquina, propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.
4. Recolector de basura: El CLR proporciona un sistema automático de administración de memoria denominado recolector de basura *(Garbage collector)*. El CLR detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. De esta forma, el programador no tiene por qué liberar la memoria de forma explícita aunque también sea posible hacerlo manualmente.
5. Seguridad de acceso al código: Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente de la web sin tener que preocuparse si esto va a estropear el sistema.
6. Despliegue: Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones. [15]

1.4.6. Herramienta para el desarrollo

- **Visual Studio 2010 Ultimate**

Microsoft Visual Studio 2010 Ultimate incluye potentes herramientas que simplifican todo el proceso de desarrollo de aplicaciones, de principio a fin. Los equipos pueden observar una mayor productividad y ahorro de costes al utilizar características de colaboración avanzadas, así como herramientas de pruebas y depuración integradas que le ayudarán a crear siempre un código de gran calidad.

- ✓ **Administración del ciclo de vida de las aplicaciones**

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

- ✓ **Depuración y diagnóstico**

Visual Studio 2010 Ultimate presenta IntelliTrace, una valiosa característica de depuración que hace que el argumento “no reproducible” sea cosa del pasado. Los evaluadores pueden archivar errores enriquecidos y modificables para que los desarrolladores puedan reproducir siempre el error del que se informe en el estado en el que se encontró. Otras características incluyen análisis de código estático, métricas de código y creación de perfiles.

- ✓ **Herramientas de prueba**

Visual Studio 2010 Ultimate incorpora todas las herramientas avanzadas de pruebas para ayudarle a garantizar la calidad del código en todo momento. Aprovéchese de las pruebas de IU codificadas, que automatizan la realización de pruebas de la interfaz de usuario en aplicaciones basadas en web y en Windows®, así como de pruebas manuales, Test Professional, pruebas de rendimiento de web, pruebas de carga, cobertura de código y otras características completas que no se encuentran en otras ediciones de Visual Studio.

- ✓ **Arquitectura y modelado**

El explorador de arquitectura de Visual Studio 2010 Ultimate ayuda a entender los activos de código existentes y otras interdependencias. Los diagramas por capas ayudan a garantizar el cumplimiento de la arquitectura y permiten validar artefactos de código con respecto al diagrama. Además, Visual Studio 2010 Ultimate admite los cinco diagramas de UML más comunes que conviven junto con su código.

- ✓ **Desarrollo de bases de datos**

El desarrollo de bases de datos requiere el mismo cuidado y atención que el desarrollo de aplicaciones. Visual Studio 2010 Ultimate es consciente de ello y proporciona potentes herramientas

de implementación y administración de cambios que garantizan que la base de datos y la aplicación estén siempre sincronizadas.

- ✓ **Entorno de desarrollo integrado**

Visual Studio 2010 Ultimate le permite ponerse al mando. Aprovéchese de las características personalizables como, por ejemplo, compatibilidad con varios monitores, de modo que pueda organizar y administrar su trabajo como quiera. También puede dar rienda suelta a su creatividad utilizando los diseñadores visuales para mejorar las últimas plataformas, incluido Windows 7.

- ✓ **Compatibilidad con la plataforma de desarrollo**

Tanto si crea soluciones nuevas como si quiere mejorar las aplicaciones ya existentes, Visual Studio 2010 Ultimate le permite hacer realidad su idea en una gran variedad de plataformas, entre las que se incluyen Windows, Windows Server, Web, Cloud, Office y SharePoint, entre otras, todo en un único entorno de desarrollo integrado. [16]

1.4.7. Herramienta para la gestión de base de datos

- ➢ **Oracle Database 11g**

Oracle Database 11g Enterprise Edition ofrece confiabilidad, escalabilidad y desempeño de primer nivel para configuraciones en clúster y en un solo servidor. Ofrece las más completas características para soportar el procesamiento de transacciones, inteligencia de negocios y aplicaciones para la administración de contenido. Brinda además protección ante las fallas del servidor, fallas del sitio, errores humanos, reducción del tiempo programado, protección de datos con seguridad única en el nivel de filas, auditorías detalladas y encriptación transparente de datos. Incluye almacenamiento de datos *(data warehousing)* de alto desempeño, procesamiento analítico online, y características de extracción de datos. Constituye un sistema gestor de bases de datos con características objeto-relacionales.

Sus principales características son las siguientes:

- Entorno cliente/servidor.
- Gestión de grandes bases de datos.
- Usuarios concurrentes.
- Alto rendimiento en transacciones.

- Sistemas de alta disponibilidad.
- Disponibilidad controlada de los datos de las aplicaciones.
- Adaptación a estándares de la industria, como SQL-92.
- Gestión de la seguridad.
- Autogestión de la integridad de los datos.
- Opción distribuida.
- Portabilidad.
- Compatibilidad.
- Replicación de entornos.

Provee un control de acceso discrecional, es decir, acceso restringido a la información basado en privilegios. Gestiona la seguridad de la base de datos usando:

- Usuarios y esquemas de la base de datos.
- Privilegios.
- Roles.
- Ajustes de rendimiento y cuotas.
- Límites sobre los recursos.
- Auditoría.

Cada usuario posee un dominio de seguridad, que determina:

- Acciones (privilegios y roles) disponibles para el usuario.
- Cuotas sobre tablespaces.
- Límites en los recursos del sistema.

Posee varias estructuras y mecanismos de software para proveer:

- Recuperación de la base de datos ante distintos tipos de fallos.
- Operaciones de recuperación flexibles.
- Disponibilidad de los datos durante las operaciones de reserva (backup) y recuperación (recovery).

Utiliza varias estructuras para proveer la recuperación completa de la instancia:

- Redo Log.
- Segmentos de retroceso (rollback).

- Fichero de control.
- Copias necesarias de la base de datos. [17]

1.4.8. Herramienta para el control de versiones

➢ **SVN**

Subversion se creó para igualar y mejorar la funcionalidad de CVS *(Concurrent Versions System)*, preservando su filosofía de desarrollo. Este sistema comenzó en el año 2000 como proyecto de código abierto. Es un software libre bajo una licencia de tipo Apache/BSD y se le conoce también como SVN por ser el nombre de la herramienta utilizada en la línea de órdenes.

Permite acceder al repositorio a través de redes, lo que facilita que pueda ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. [18]

1.4.9. Otras tecnologías

➢ **ASP.NET**

Es la parte más importante de la capa superior de la plataforma .NET. Constituye mucho más que una nueva versión de la tecnología ASP, ya que, supone una nueva idea y forma de programar aplicaciones web, donde el programador puede centrarse exclusivamente en la lógica de la aplicación sin preocuparse de los detalles de la interfaz. Además, incorpora un nuevo concepto en el desarrollo de tecnologías Internet: los servicios web. Estos servicios representan un paso más hacia la descentralización del software en la red y de hecho, son un factor clave para el desarrollo de una web orientada a objetos. Los servicios Web permiten a los desarrolladores construir aplicaciones combinando recursos locales y remotos para una solución distribuida e integrada. Las características de AJAX en ASP.NET permiten crear rápidamente páginas web para que la experiencia del usuario sea más satisfactoria, gracias a elementos, de interfaz de usuario, más familiares y receptivos. Entre las características de AJAX se incluyen bibliotecas de scripts de cliente, que incorporan las tecnologías script *(Java Script)* y HTML dinámico *(DHTML)* para varios exploradores, e integración con la plataforma de desarrollo para servidores de ASP.NET. Las características de AJAX en ASP.NET permiten generar aplicaciones web enriquecidas que tienen muchas ventajas frente a las aplicaciones web basadas completamente en servidor. [19]

- **Entity Framework**

Una aplicación de Entity Framework requiere crear un modelo conceptual que defina las entidades y las relaciones, un modelo lógico que represente el modelo relacional subyacente y las asignaciones entre los dos. A continuación, se genera un modelo de objetos programable a partir del modelo conceptual.
Las características y componentes siguientes del Entity Framework trabajan conjuntamente para proporcionar un entorno de programación de un extremo a otro.

- El Entity Data Model *(EDM)* es la pieza central de Entity Framework. Especifica el esquema de diseño, que se usa para generar las clases programables que usa el código de la aplicación. Las estructuras de almacenamiento de los datos conservados se representan en un esquema de almacenamiento y una especificación de asignación conecta el esquema de diseño con el esquema de almacenamiento. Las entidades conceptuales se pueden materializar como objetos o se pueden leer en un formato serializado mediante un lector de datos. Los desarrolladores pueden extender estos objetos cuando sea necesario para la compatibilidad con diferentes necesidades de la aplicación.

- El componente objeto de servicio *(Object Service)* permite a los programadores trabajar con las clases de CLR generadas a partir del modelo conceptual. También proporcionan compatibilidad de infraestructura con Entity Framework, con servicios como la administración de estados, el seguimiento de cambios, la resolución de identidad, las relaciones de carga, la navegación, la propagación de cambios de objeto a modificaciones de base de datos y la compatibilidad con consultas para Entity SQL.

- LINQ to Entities proporciona compatibilidad con LINQ para consultar las entidades. LINQ to Entities permite a los programadores escribir consultas con la base de datos utilizando uno de los lenguajes de programación de .NET admitidos, como Visual Basic o Visual C#.

- Entity SQL es un lenguaje independiente del almacenamiento que es similar a SQL y que se ha diseñado para la consulta y manipulación de gráficos enriquecidos de objetos basados en el modelo Entity Data Model.

- El proveedor Entity Client extiende el modelo de proveedor de ADO.NET teniendo acceso a los datos en lo que respecta a las entidades conceptuales y relaciones, además, ejecuta

consultas que usan Entity SQL. Entity SQL proporciona el lenguaje de consulta subyacente que permite a Entity Client comunicarse con la base de datos.

- El componente de metadatos de ADO.NET administra los metadatos en cuanto a las necesidades de tiempo de ejecución y tiempo de diseño de Entity Framework. Todos los metadatos asociados a los modelos y asignaciones se exponen a través de las interfaces de metadatos que son independientes de los mecanismos usados para el almacenamiento de los metadatos. El mecanismo de almacenamiento actual utiliza el archivo que se basa en tres dialectos XML: el lenguaje de definición de esquemas conceptuales *(CSDL),* el lenguaje de definición de esquemas de almacenamiento (*SSDL)* y el lenguaje de especificación de asignaciones *(MSL).*

- Entity Framework incluye un conjunto de herramientas en evolución que generan asignaciones y clases parciales que representan las entidades en el modelo conceptual. [20]

➢ **ASP .NET MVC Framework**

ASP .NET MVC es un framework que divide la implementación de una aplicación en tres componentes: modelos, vistas y controladores.

- Los “modelos” de una aplicación basada en MVC son los componentes responsables de mantener el estado. Normalmente, el estado se guarda en una base de datos.
- Las “vistas” son los componentes responsables de mostrar la interfaz de usuario de la aplicación. Normalmente, esta interfaz de usuario se crea a través del modelo de datos.
- Los “controladores” son los encargados de administrar la interacción con el usuario final, manipular el modelo, y en último lugar elegir una vista para construir la interfaz de usuario. En una aplicación MVC la vista solo muestra la información, el controlador es el que administra y responde a las entradas de usuario y a las interacciones.

Uno de los beneficios de usar este framework, es que ayuda a mantener una separación limpia entre modelos, vistas y controladores en la aplicación. Manteniendo una separación clara de conceptos hace que las pruebas a las aplicaciones sean más fáciles, ya que, los contratos entre los diferentes componentes de la aplicación están mejor definidos y articulados.

Este framework también ayuda a realizar un desarrollo basado en pruebas *(Test Driven Development, TDD)*, donde se implementan pruebas unitarias automáticas, que definen y verifican los requerimientos del nuevo código, antes de escribir el código. [21]

Conclusiones

En este capítulo se realizó un análisis de los diferentes sistemas existentes para la gestión de la configuración del modelo de control de acceso *RBAC*, llegando a la conclusión de que a pesar de las ventajas que tienen cada uno de estos, ninguno cumple con las características del sistema que se desea implementar. Después de un análisis de las herramientas que se podían utilizar y teniendo en cuenta las políticas para el desarrollo de aplicaciones web se seleccionaron entre otras, como lenguajes de programación *C#, Java Script* y *CSS,* como gestor de bases de datos *Oracle*, *Altova Umodel* como herramienta de modelado y *MSF Agile* como metodología para llevar a cabo el proceso de desarrollo del software.

Capítulo 2: Características del sistema

2.1. Introducción

En este capítulo se le presta atención a las fases Visión y Planificación que propone la metodología para el desarrollo de software *MSF Agile.* Esta metodología plantea alcanzar una visión de lo que se desea desarrollar y propone realizar una planificación que guíe al equipo de trabajo hacia la construcción favorable del sistema. Para ello se realizan las actividades: capturar la visión del proyecto, crear los escenarios y crear los requerimientos de calidad de servicios.

2.2. Objeto de informatización

En el entorno existe una intranet corporativa poblada de numerosas aplicaciones. Para cada una de estas aplicaciones se ha tenido que implementar un módulo de seguridad que entre otras funciones sea capaz de controlar el acceso a los recursos, por lo que existen numerosos módulos implementados, dificultando la compartición de recursos y la autorización de acceso a estos.

En el Centro de Identificación y Seguridad Digital existe una primera versión del sistema de gestión de la configuración de los parámetros del modelo de control de acceso RBAC *(de este sistema se mencionan las principales características en el capítulo anterior)*, el objetivo de esta aplicación es garantizar que la gestión de las políticas de control de acceso se encuentren de manera centralizada. Para complementar el correcto funcionamiento del sistema de gestión se necesitan incluir nuevos conceptos que ampliarán el mismo a una segunda versión de manera que se alcanzará un mejor funcionamiento logrando el mayor desempeño posible para el sistema.

Entre los nuevos conceptos asociados se encuentra el de ámbito y el de grupo, los ámbitos no son más que una estructura organizacional lógica asociada a una institución y que son capaces agrupar a otros ámbitos, permitiendo así la asignación de usuarios para controlar las funciones administrativas del sistema, por su parte los grupos se refieren a una estructura física organizacional también destinada a agrupar elementos, estos grupos van a estar contenidos dentro de los ámbitos y se encargarán de gestionar los controles de accesos. Es importante destacar que estos conceptos no están definidos en el estándar de la NIST, pero son muy útiles pues los mismos facilitan las funciones administrativas del sistema y permiten mantener los elementos separados por áreas dependientes.

2.3. Descripción de la propuesta de solución

Se propone desarrollar una aplicación web para la gestión de la configuración de los parámetros para el control de acceso basado en roles del Sistema de Administración de Identidades, versión 2.0 con una interfaz sencilla que abarque todo lo referente a la gestión de ámbitos, grupos, roles, roles administrativos, restricciones y recursos así como otras funcionalidades que son de gran importancia para la gestión del control de acceso.

El sistema estará compuesto por diversos módulos encargados de concentrar todas las funcionalidades referentes a cada uno de los elementos del modelo RBAC, así como otras que no están concebidas en este estándar.

Tabla 2: Módulos del sistema de gestión de los parámetros del modelo RBAC v2.0

Módulo	Descripción
Gestión de Ámbitos	Es el encargado de gestionar las funciones administrativas del sistema permitiendo la creación, modificación o eliminación de los mismos. Este a su vez controla la asignación de usuarios.
Gestión de Grupos	Incluye la creación, modificación o eliminación de grupos. Además permite asignar roles, recursos y usuarios para facilitar la gestión del control de acceso siguiendo las especificaciones del modelo RBAC.
Gestión de Roles	Aborda la creación de roles organizacionales y administrativos, así como su posterior eliminación o modificación. Además, controla la asignación de los usuarios a determinados roles, dependiendo de las responsabilidades que estos usuarios posean.
Gestión de Recursos	Engloba la creación, eliminación y modificación de los recursos.
Gestión de Restricciones	Incluye todo lo referente a las diferentes restricciones que plantea el modelo RBAC. Este módulo es el encargado de crear las restricciones, imponiendo diferentes reglas que se deben tener en cuenta para la realización de determinadas funcionalidades.

2.4. Modelo conceptual del sistema

Un modelo conceptual no es más que la representación visual de las clases conceptuales u objetos del mundo real que reportan interés para un producto software. Este modelo utiliza el estándar UML para su representación y para su construcción se modela un diagrama de clase compuesto por objetos del dominio o clases conceptuales y relaciones entre estas clases.

La determinación de realizar este modelo conceptual (Ver Figura 4) para el sistema se toma a partir de no tenerse una claridad suficiente en los procesos como para modelar un negocio o no existir una claridad de quiénes son los que inician el negocio y quiénes se benefician con este.

Conceptos asociados al modelo conceptual

Ámbito: Estructura organizacional lógica asociada a una institución, agrupando los usuarios que se desarrollan dentro del mismo para facilitar las tareas administrativas.

Grupo: Estructura organizacional física perteneciente a una institución o ámbito, agrupando los usuarios, recursos, y roles que se desarrollan dentro del mismo, facilitando los controles de accesos en el ámbito al que pertenece.

Restricciones: Son políticas que restringen determinados elementos dentro del modelo RBAC.

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.

Recurso: Un recurso puede ser cualquier objeto accesible en un sistema informático, incluidos archivos, periféricos, tales como impresoras, bases de datos, y entidades tales como campos individuales en los registros de base de datos. Los recursos son tradicionalmente considerados como entidades pasivas que contienen o reciben información.

Permiso: Accesos que se otorga al usuario para acceder a una funcionalidad dentro del sistema. Representa la asociación de una operación sobre un recurso.

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

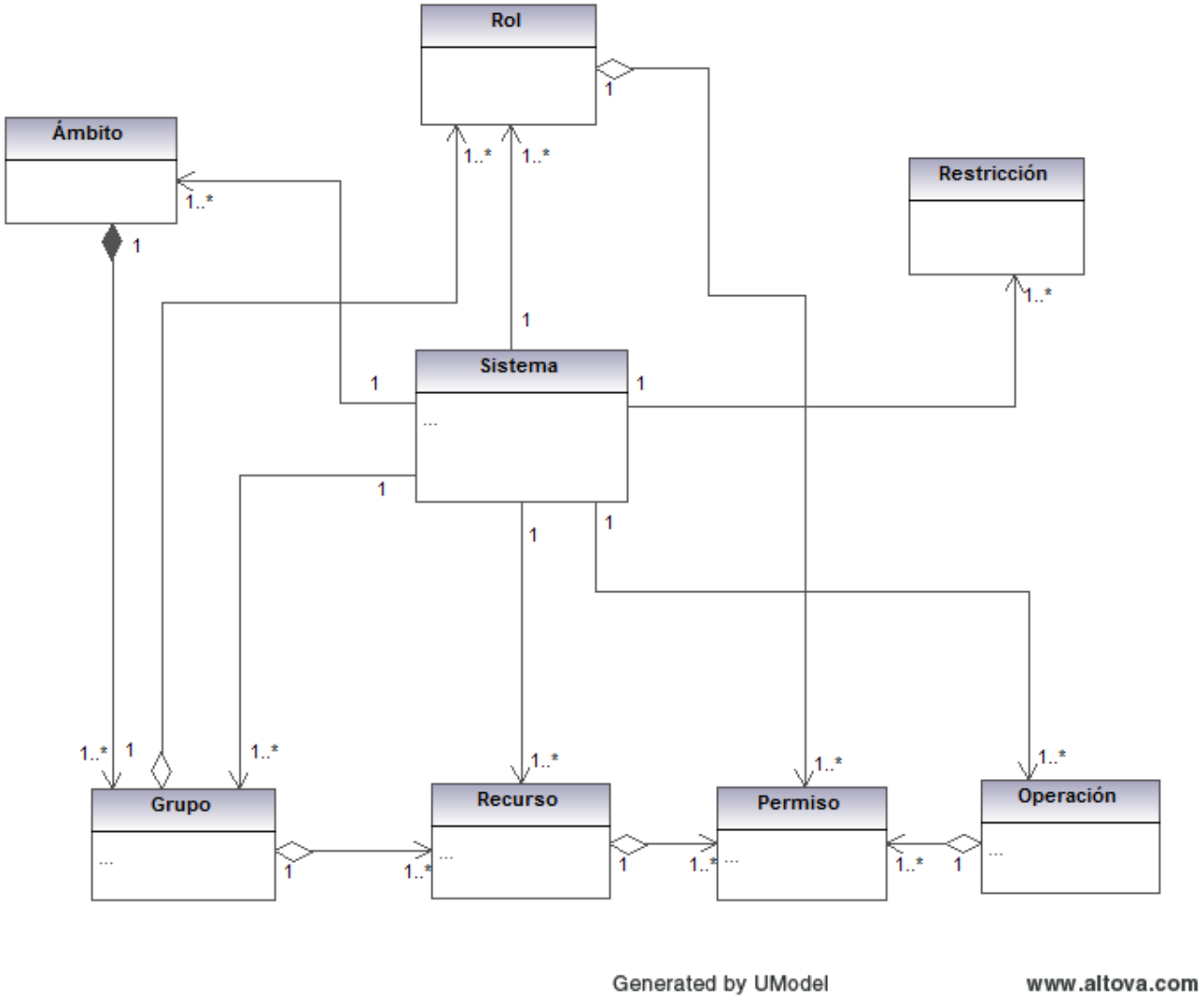


Figura 4: Modelo conceptual del sistema

En el caso del sistema de gestión de la configuración del modelo RBAC esta vista conceptual brinda la posibilidad de conocer como están organizados los parámetros que serán gestionados para garantizar el control de acceso a los recursos.

2.5. Fase visión y alcance

En esta fase se proporcionan los factores y la justificación para la construcción del sistema. Es importante tener bien definida la visión y alcance, pues ayuda a mantener el sistema enfocado en las necesidades del mismo. Las actividades que se contemplan durante este período son: capturar la visión y definir personas.

2.5.1. Visión y alcance de la propuesta de solución.

Con esta aplicación se persigue lograr que el módulo de autorización del Sistema de Administración de Identidades de una solución final, para automatizar la gestión de la configuración de los elementos del modelo RBAC incluyendo algunos que no se definen en el estándar.

2.5.2. Definición de personas

Las personas representan individuos o sistemas externos que interactúan con la aplicación.

Tabla 3: Personas

Personas	Descripción
Administrador de Ámbito	Podrá realizar operaciones de administración y tendrá todos los privilegios que existan sobre los ámbitos del sistema.
Administrador de Grupo	Podrá realizar operaciones de administración y tendrá todos los privilegios que existan sobre los grupos contenidos dentro de los ámbitos del sistema
Administrador de Roles	Podrá realizar operaciones de administración y tendrá todos los privilegios que existan sobre los roles del sistema
Administrador de Recursos	Podrá realizar operaciones de administración y tendrá todos los privilegios que existan sobre los recursos del sistema
Administrador de Restricciones	Podrá realizar operaciones de administración y tendrá todos los privilegios que existan sobre las restricciones del sistema

2.6. Fase planificación

Los principales artefactos que se crean durante la fase de planificación son los escenarios y los requerimientos de calidad de servicios que sirven de guía para todo el proceso de desarrollo.

2.6.1 Lista de escenarios del sistema

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

En este sistema los escenarios se encuentran en los diferentes módulos que conforman la aplicación.

Módulo Gestión de Ámbitos

1. Gestionar ámbito
2. Asignar usuario al ámbito
3. Revocar asignación de usuario al ámbito

Módulo Gestión de Grupos

4. Asignar usuario a grupo
5. Asignar recurso a grupo
6. Asignar rol organizacional a grupo
7. Revocar asignación de rol organizacional a grupo
8. Revocar asignación de recurso a grupo
9. Revocar asignación de usuario a grupo
10. Gestionar grupo

Módulo Gestión de Roles

11. Asignar usuario a rol organizacional
12. Asignar permiso a rol organizacional
13. Asignar usuario a rol administrativo
14. Asignar permisos a un rol organizacional a través de la herencia de roles
15. Revocar la asignación de usuarios a rol organizacional
16. Revocar la asignación de permisos a rol organizacional
17. Revocar la asignación de usuarios a rol administrativo
18. Habilitar/Deshabilitar rol organizacional
19. Habilitar/Deshabilitar rol administrativo
20. Gestionar rol organizacional

Módulo Gestión de Recursos

21. Gestionar recursos
22. Verificar permisos sobre recursos.

Módulo Gestión de Restricciones

23. Gestionar restricciones de cardinalidad de roles
24. Gestionar restricciones de cardinalidad de permisos

25. Gestionar restricciones de exclusión de roles
26. Gestionar restricciones de exclusión de permisos
27. Gestionar restricciones de precondición de roles
28. Gestionar restricciones de asignación.

2.6.1.1 Prioridad de los escenarios del sistema

Los escenarios se priorizan en dependencia de la importancia que tienen para los usuarios y para la aplicación. El proceso de priorizar la lista de escenarios trae consigo identificar aquellos que se consideran los más importantes a la hora de implementar.

Tabla 4: Prioridad de los escenarios

Escenarios	Prioridad
Gestionar ámbito	Alta
Gestionar rol administrativo	Alta
Gestionar rol organizacional	Alta
Gestionar grupos	Alta
Gestionar restricciones	Alta
Gestionar recursos	Alta
Resto de los escenarios	Media

2.6.2 Lista de requerimientos de calidad de servicios

Los requerimientos de calidad de servicios se utilizan para capturar los requisitos no funcionales del sistema. La metodología MSF define varios tipos de requerimientos de calidad de servicios. Estos requerimientos serán las bases sobre como el sistema debe operar.

- ✓ **Requerimientos de diseño e implementación**

El sistema permitirá la reutilización del código así como la facilidad de actualización del mismo, para lo cual se utilizarán un conjunto de patrones de diseño. El código cumplirá con los estándares de codificación establecidos por el cliente.

- ✓ **Requerimientos de seguridad**

La seguridad estará implementada a través de los diferentes módulos del sistema integrando la aplicación con un subsistema de identificación que permita que sólo los usuarios con derechos de administrador puedan acceder a las funciones administrativas garantizando así la confiabilidad.

- ✓ **Requerimientos de disponibilidad**

La disponibilidad del sistema estará dada solo para aquellos usuarios autorizados a registrarse en la aplicación. Se garantizará que el sistema esté disponible las 24 horas del día mediante el uso de un servidor explícitamente para estas funciones.

- ✓ **Requerimientos de usabilidad**

El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de las computadoras, el Navegador Mozilla Firefox y el modelo RBAC, mediante una interfaz sencilla y clara que facilitará las tareas administrativas u otras actividades.

- ✓ **Requerimientos de soporte**

Se debe realizar la aplicación de forma versionable que permita darle mantenimientos al sistema a fin de aumentar las funcionalidades y/o corregir los errores del mismo a través de versiones posteriores.

- ✓ **Requerimientos de confiabilidad**

Las actividades de mantenimiento, supervisión y edición del sistema no deben interferir en el correcto desempeño del resto del sistema, ni afectar la ejecución de la aplicación, esto se logrará mediante el uso de servidores alternativos durante la realización de estas tareas.

- ✓ **Requerimientos de rendimiento**

El sistema deberá ser rápido y el tiempo de respuesta será el mínimo posible para lo cual se implementarán funcionalidades de caché y optimización del código.

2.6.3 Plan de iteraciones

Para planificar adecuadamente el desarrollo de la aplicación, se estima el tiempo que les tomará a los programadores la codificación de cada uno de los escenarios. A partir de la prioridad se tiene en cuenta cuáles de ellos se implementarán en iteraciones tempranas del ciclo.

Tabla 5: Planificación de los escenarios

No. Iteración	Escenarios	Prioridad	Estimación(horas)
1	Gestionar ámbitos	Alta	228
1	Gestionar grupos	Alta	198
1	Gestionar roles	Alta	116
2	Gestionar recursos	Alta	200
2	Gestionar restricciones	Alta	116
3	Resto de los escenarios	Media	72

2.6.4. Descripción de los escenarios del sistema

Para la implementación del sistema se realiza la descripción de los escenarios según las necesidades del cliente. En las tablas 6, 7, 8, 9 y 10 se muestran las descripciones de algunos escenarios pertenecientes al módulo "Gestión de Ámbitos". El resto de las descripciones se pueden ver en el **Anexo I**.

Tabla 6: Descripción del escenario "Gestionar Ámbito"

Nombre del Escenario: Gestionar ámbito		**Identificador:** RBAC1
Objetivo del Escenario: Brindar la posibilidad de crear, editar, mostrar y eliminar uno o varios ámbitos en el sistema.		
Persona: Administrador de ámbito		
Iteración: 1ra	**Prioridad:** 1	**Complejidad:** 3
Descripción: El administrador de ámbito accede a la aplicación con la intención de Gestionar ámbito. Se autentica en el sistema y este le muestra en pantalla la jerarquía de ámbitos existentes dentro del nivel institucional en los cuales el administrador tiene los permisos asignados para realizar operaciones. Entre las operaciones que puede gestionar el administrador de ámbito se encuentran la de Crear un ámbito cumpliendo con la estructura jerárquica establecida; la de Editar y Eliminar ámbito, teniendo siempre como precondición que sea seleccionado el ámbito para realizar dichas acciones sobre él; y por último Mostrar las características generales de un determinado ámbito. Para consultar las descripciones se recomienda ver escenarios: • Crear ámbito (RBAC1.1) • Editar ámbito (RBAC1.2) • Eliminar ámbito (RBAC1.3)		

• Mostrar ámbito (RBAC1.4)
Validaciones: • El sistema debe verificar los permisos del administrador.

Tabla 7: Descripción del escenario "Crear ámbito"

Nombre del Escenario: Crear ámbito		**Identificador:** RBAC1.1
Objetivo del Escenario: Crear uno o varios ámbitos.		
Persona: Administrador de ámbito		
Iteración: 1ra	**Prioridad:** 1	**Complejidad:** 2
Descripción: El administrador de ámbito accede a la aplicación con la intención de crear un nuevo ámbito. Una vez autenticado en el sistema, éste muestra en pantalla una estructura jerárquica de los ámbitos existentes dentro del ambito institucional y le permite seleccionar un ámbito dentro del cual desea poder crear uno nuevo. El administrador selecciona el ámbito padre y crea el nuevo ámbito dentro de este, e introduce los datos correspondientes como son el nombre y la descripción del mismo. Finalmente el administrador selecciona el botón Crear Ámbito y el sistema crea el nuevo ámbito dentro del sistema.		
Validaciones: • El sistema debe verificar los permisos del administrador.		

Tabla 8: Descripción del escenario "Mostrar ámbito"

Nombre del Escenario: Mostrar ámbito		**Identificador:** RBAC1.4
Objetivo del Escenario: Mostrar ámbitos del sistema.		
Persona: Administrador de ámbito		
Iteración: 1ra	**Prioridad:** 1	**Complejidad:** 2
Descripción: El Administrador de ámbito desea poder consultar la información relacionada con un determinado ámbito. Después de haberse autenticado en la aplicación, esta muestra en pantalla la jerarquía de todos los ámbitos existentes, y le permite al administrador seleccionar el ámbito que desea consultar. Una vez seleccionado el ámbito, el sistema muestra automáticamente la información referente al		

mismo, como lo es el nombre y la descripción del ámbito.
Validaciones: • El sistema debe verificar los permisos del administrador.

Tabla 9: Descripción del escenario "Editar ámbito"

Nombre del Escenario: Editar ámbito		**Identificador:** RBAC1.2
Objetivo del Escenario: Editar los valores de los campos de los ámbitos.		
Persona: Administrador de ámbito		
Iteración: 1ra	**Prioridad:** 1	**Complejidad:** 2
Descripción: El administrador de ámbito desea poder editar un ámbito existente dentro de la estructura jerárquica definida, y una vez autenticado en el sistema, éste muestra en pantalla la opción de Editar ámbito. El administrador selecciona el ámbito que desea editar, y el sistema le permite modificar los datos nombre y descripción correspondientes al ámbito seleccionado. El administrador introduce los nuevos datos y selecciona el botón Guardar cambios. Finalmente el sistema guarda los cambios realizados sobre el ámbito.		
Validaciones: • El sistema debe verificar los permisos del administrador.		

Tabla 10: Descripción del escenario "Eliminar ámbito"

Nombre del Escenario: Eliminar ámbito		**Identificador:** RBAC1.3
Objetivo del Escenario: Eliminar ámbitos del sistema.		
Persona: Administrador de ámbito		
Iteración: 1ra	**Prioridad:** 1	**Complejidad:** 2
Descripción: El administrador de ámbito accede a la aplicación con la intención de eliminar uno o varios ámbitos dentro de la estructura jerárquica existente en el sistema. Una vez autenticado en la aplicación, se muestra en pantalla la opción de eliminar ámbito. El administrador selecciona el ámbito que desea eliminar y oprime el botón Eliminar ámbito. El sistema muestra una ventana con un mensaje de aviso ´´ Al eliminar el ámbito seleccionado, se		

eliminan también los ámbitos que heredan de este. Está usted seguro que desea Eliminar el ámbito ´´, y le da al administrador la opción de seleccionar el botón Eliminar o Cancelar. El administrador de ámbito selecciona el botón Eliminar, y el sistema elimina automáticamente el ámbito que ha sido seleccionado y con él sus datos y ámbitos hijos. El administrador de ámbito también tiene la opción de no eliminar el ámbito seleccionando el botón Cancelar. Finalmente el sistema realiza las operaciones correspondientes en dependencia de la opción seleccionada por el administrador.
Validaciones: • El sistema debe verificar los permisos del administrador.

Conclusiones

En el transcurso de este capítulo se hizo una propuesta para dar solución a la situación problémica planteada, proponiendo también las personas que se vinculan al sistema y las responsabilidades que debe cumplir cada una de ellas. Se recoge además una especificación de los escenarios que resolverán las funcionalidades de los módulos propuestos.

Capítulo 3: Desarrollo y prueba de la solución propuesta

3.1. Introducción

Este capítulo es el encargado de formular el diseño de la solución propuesta mediante la descripción de la arquitectura, los patrones de diseño y los diagramas que propone la metodología de desarrollo. En este también se describe el modelado de la base de datos, se especifica la implementación de los escenarios a través de las diferentes iteraciones y se definen los casos de pruebas para verificar el funcionamiento del sistema.

3.2. Fase de desarrollo

La fase de desarrollo iniciará la implementación del sistema. En esta se obtendrán entregas parciales del producto, desarrollados por partes para medir su progreso y así asegurarse que puedan integrarse para obtener el producto final.

3.2.1 Especificación de la arquitectura

La arquitectura de software es el diseño de más alto nivel de la estructura de un sistema, donde se establecen los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema, cubriendo todas las necesidades.

3.2.1.1 Patrón de arquitectura MVC

El patrón de arquitectura Modelo-Vista-Controlador (MVC) (Ver Figura 5) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

Modelo: El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Maneja la visualización de la información, o sea todo lo que el usuario ve en la pantalla.

Controlador: Interpreta los eventos generados por las acciones del usuario, informando al modelo y/o a la vista que cambien según resulte apropiado. Tanto la vista como el controlador dependen del modelo, el cual no depende de estos. Esta separación permite construir y probar el modelo independientemente de la representación visual. [22]

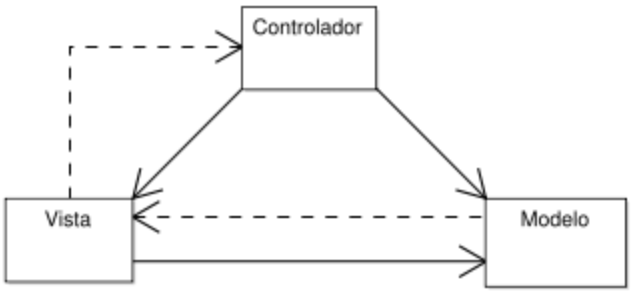


Figura 5: Patrón de arquitectura MVC

El sistema de gestión de la configuración de los parámetros para el modelo de control de acceso RBAC divide la implementación del sistema en 3 componentes:

Modelos (*Models):* Se encuentran constituidos por dos proyectos que agrupan los diferentes componentes: *Gyes.RBAC.CoreLib,* es el encargado de gestionar los procesos del negocio y *Gyes.RBAC.DAL,* es el encargado de mantener el estado del sistema para guardarlo en la base de datos.

Vistas (*Views):* Guardan todas las interfaces de usuarios. Estas vistas son ficheros *aspx* que contienen todos los componentes *Java Script* y *CSS* del sistema. Algunos ejemplos son:

Group ⇨ Contiene a *Create.aspx, Details.aspx, Edit.aspx* e *Index.aspx.*

Role ⇨ Contiene a *Create.aspx, Delete.aspx, Edit.aspx, Index.aspx, Details.aspx e IndexRoleAdmin.aspx*

Controladores (*Controllers):* Administran y responden a las entradas de usuario y a las interacciones. Por ejemplo *RoleController, ScopeController, GroupController, ResourceController,* entre otros.

3.2.1.2 Vista lógica

La vista lógica del sistema se encuentra representada por tres capas que se basan en la división en el nivel de acceso a datos, nivel de lógica de negocio y nivel de presentación o aplicación. Esto permite distribuir el trabajo de creación de una aplicación evitando que los cambios en una de las capas afecten directamente al resto. (Ver Figura 6)

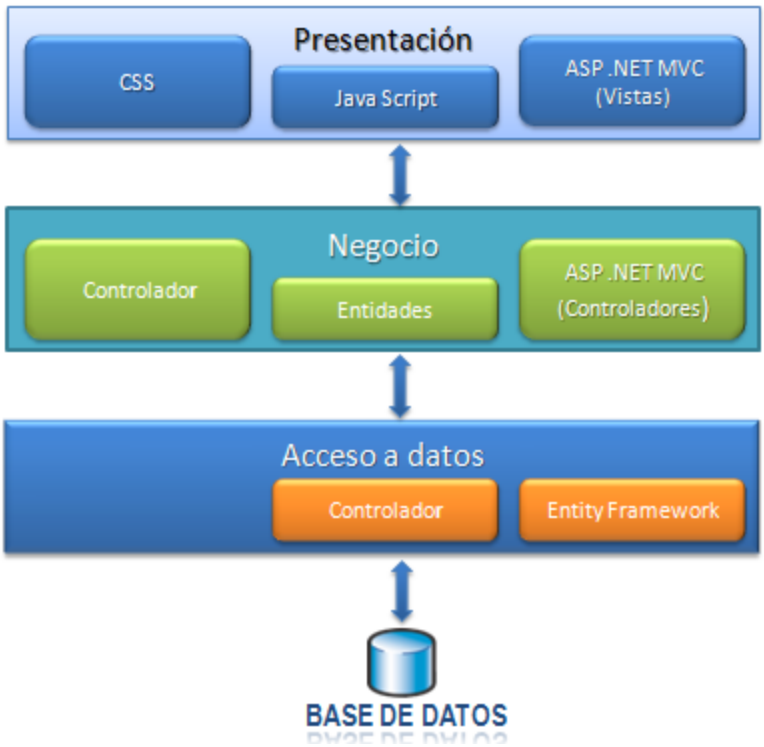


Figura 6: Vista lógica en 3 capas

Descripción de las capas

Capa de presentación:

Está compuesta por todas las interfaces de usuario y los componentes necesarios para su correcto funcionamiento. Estos elementos pueden ser ficheros *Java Script* y CSS. Esta capa se encuentra representada por un proyecto web y tiene interacción directa con la capa de negocio. Para el desarrollo de este proyecto web se hace uso del *framework* ASP .NET MVC el cual divide la implementación del proyecto en 3 componentes: modelos, vistas y controladores.

Capa de negocio:

En esta capa se recogen todas las funcionalidades necesarias para darle solución a los requerimientos de negocio. Las funcionalidades se encuentran definidas según el contexto en el que se desenvuelven. Tienen la responsabilidad de manejar todas las operaciones sobre una entidad de negocio en específico, así como todas las entidades que por conceptos de composición se encuentran relacionadas con esta.

Por cada entidad de negocio se crea un controlador y una interfaz que debe ser implementada por el acceso a datos que le dará soporte. Se encuentra constituida por dos proyectos que agrupan los diferentes componentes: ***Project.Entities***, ***Project.CoreLib.***

Capa de acceso a datos:

La capa de acceso a datos está directamente relacionada con las funcionalidades definidas en el negocio. Para establecer esta relación hace uso de las clases interfaces y controladoras que define la capa de negocio. De esta manera, es posible realizar cambios en esta capa sin que se vean afectadas las demás capas. Su principal función es realizar una implementación de las interfaces definidas en la capa de negocio y al mismo tiempo trabajar directamente con la fuente de datos establecida.

En esta capa se encuentra incluido el *Entity Framework*, herramienta utilizada para la generación del acceso a datos. *Entity Framework* es un nuevo *framework* de modelado que permite a los desarrolladores definir un modelo conceptual a partir de un esquema de base de datos que está alineado a una vista del mundo real de la información. Uno de sus beneficios es la facilidad de entendimiento y de mantenimiento del código de la aplicación, ya que, está preparado para los cambios en el esquema del modelo de datos que la soporta.

Base de datos:

Está constituida por todo el conjunto de tablas y procedimientos que permiten el almacenamiento de la información recolectada y procesada, posibilitando que la aplicación pueda manipular dichos datos dinámicamente.

3.2.2. Patrones de diseño

Los patrones de diseño son una solución estándar para un problema común de programación, una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios entre otras.

Para el desarrollo del sistema de gestión del RBAC v2.0 se han tenido en cuenta un conjunto de patrones que permiten darle la flexibilidad necesaria a la aplicación. A continuación se definen estos patrones.

3.2.2.1. Patrones GRASP

Los patrones GRASP (*General Responsability Assignment Software Patterns)* describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. El nombre se eligió para indicar la importancia de captar *(grasping)* estos principios, si se quiere diseñar eficazmente el *software* orientado a objetos. [23]

Experto: A través del uso de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo

que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase “sencillas” y más cohesivas que son fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Por ejemplo, en la clase *Scope* se evidencia este patrón, ya que los ámbitos van a contener grupos y si se desea obtener información acerca de estos grupos la clase *Group* se encarga de gestionar su información y no *Scope*.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Acoplamiento bajo significa que una clase no depende de muchas clases. Acoplamiento alto significa que una clase recurre a muchas otras clases. El grado de acoplamiento no puede considerarse aisladamente de otros principios como experto y alta cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño.

Entre las clases que contienen poca dependencia de otras clases se puede mencionar *ScopeManager*, esta clase depende de las clases *Scope* y *DALScopeManager*.

Alta cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo.

Existen clases con responsabilidades estrechamente relacionadas que no realizan un trabajo enorme ellas son, *RoleController, ResourceController* y *ScopeController.*

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

Vista de roles ⇒ *RoleController*: Cualquier evento que se genere en la vista de roles será atendido por *RoleController*.

Vista de Recursos ⇒ *ResourceController*: Cualquier evento que se genere en la vista de recursos será atendido por *ResourceController*.

3.2.2.2 Patrones GoF

Fachada: El patrón fachada trata de simplificar la interfaz entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase. La idea principal es la de ocultar todo lo posible la complejidad de un sistema, el conjunto de clases o componentes que lo forman, de forma que solo se ofrezca un (o unos pocos) punto de entrada al sistema tapado por la fachada. [24]

Por ejemplo la clase *IPermissionManager* proporciona una interfaz unificada para el conjunto de interfaces *IDALPermissionManager, IDALResourceManager e IDALOperationManager.*

3.2.3. Diagrama de clases

Un diagrama de clases es un diagrama que representa una agrupación lógica o física de las clases y sus relaciones. Las clases del sistema están organizadas de una forma estructural adecuada según lo necesario en cada una de ellas, lo que las hace fáciles de manipular y permite que se interrelacionen entre ellas siempre que se necesite**.**

Para una mejor comprensión del diagrama de clases, se dividió el diagrama de clases general en los diagramas correspondientes a cada uno de los módulos del sistema. (Ver Figura 7 y 8). El resto de los diagramas de clases se encuentran en el **Anexo III**.

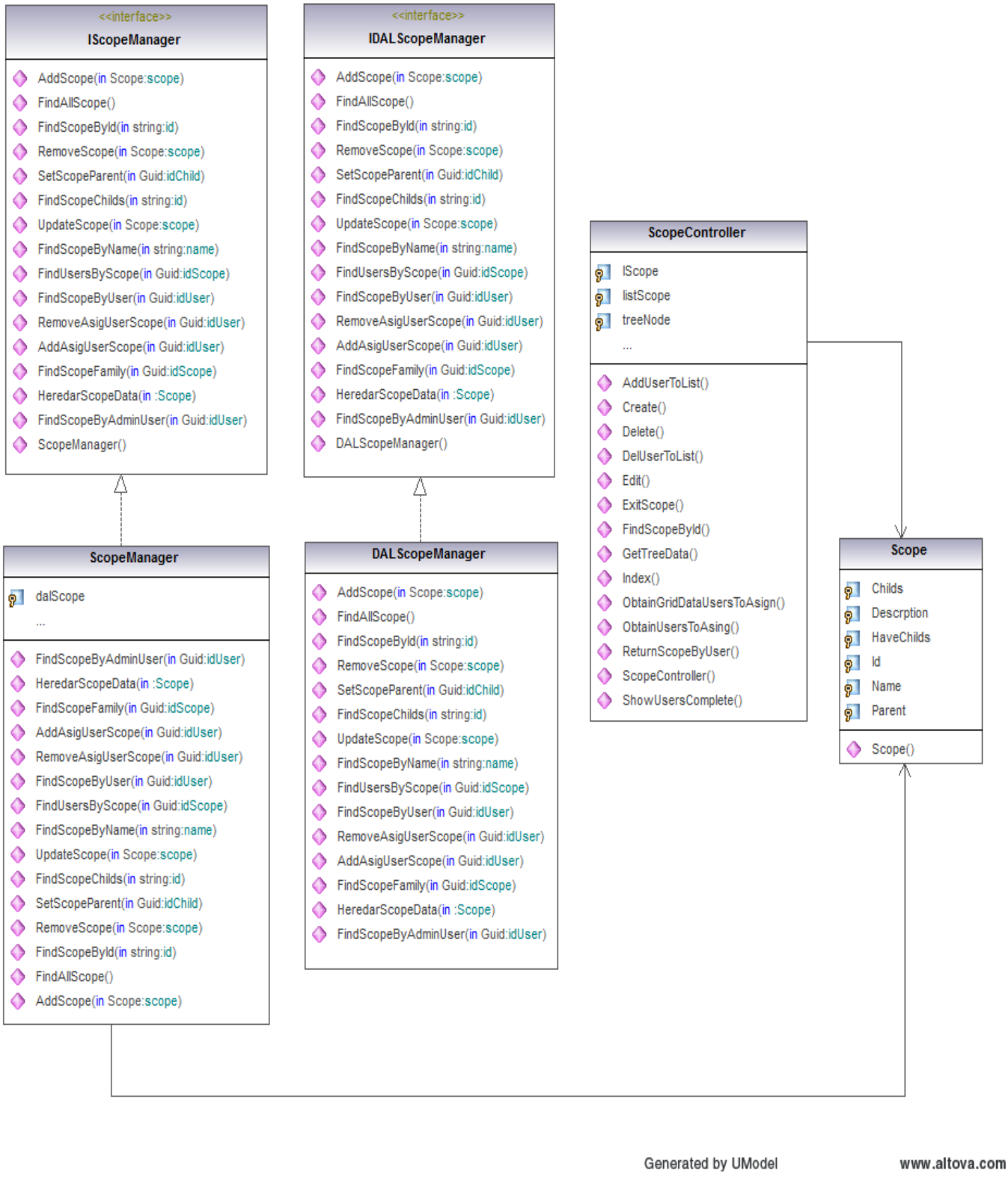


Figura 7: Diagrama de clases del módulo “Gestión de Ámbitos”

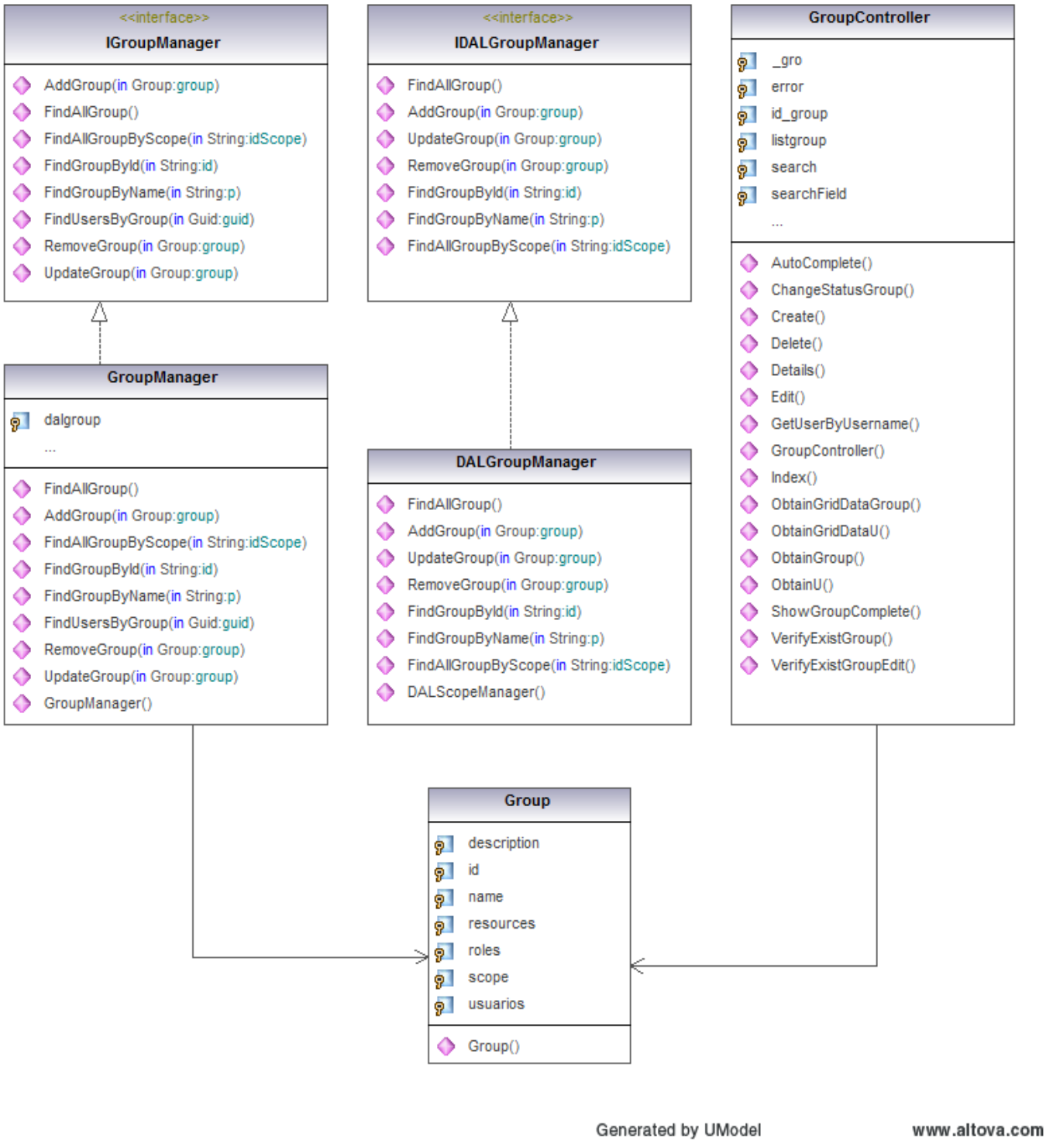


Figura 8: Diagrama de clases del módulo “Gestión de Grupos”

3.2.4. Descripción de clases

Tabla 11: Descripción de la clase *"Scope Manager"*

Nombre:	*Scope Manager*
Tipo de Clase:	Controladora
Descripción:	Esta clase es la encargada de implementar todos los métodos de la interfaz *IScopeManager*. Maneja todas las acciones que se deseen realizar sobre los ámbitos.
Atributos	
Nombre	**Descripción**
_dalScope: IDALScopeManager	Objeto de la clase interfaz de acceso a datos *IDALScopeManager*.
Métodos	
Nombre	**Descripción**
AddScope (Scope scope): bool	Permite adicionar un nuevo ámbito.
FindAllScope(): List<Scope>	Devuelve todos los ámbitos existentes.
FindScopeById(string id):Scope	Retorna el ámbito que posea el identificador especificado por parámetro
FindScopeByName(string name):Scope	Retorna el ámbito que posea el nombre especificado por parámetro
RemoveScope(Scope scope):bool	Elimina el ámbito que se especifica por parámetro.
FindScopeChilds (Guid id): List<Scope>	Devuelve todos los ámbitos hijos del ámbito que se especifica por parámetro.
UpdateScope (in scope: Scope): bool	Actualiza el ámbito especificado por parámetro.
FindUsersByScope(Guid idScope) : List<User>	Devuelve todos los usuarios del ámbito que se especifica por parámetro.
FindScopeByUser(Guid idUser): List<Scope>	Devuelve los ámbitos del usuario que se especifica por parámetro.
RemoveAsigUserScope(Guid idUser, Guid idScope): bool	Elimina la asignación del usuario especificado por parámetro sobre el ámbito especificado por parámetro.

SetScopeParent(Guid idChild, Guid idParent):bool	Asigna un padre al ámbito especificado por parámetro.
AddAsigUserScope(Guid idUser, Guid idScope): bool	Asigna el usuario especificado por parámetro al ámbito especificado por parámetro.
FindScopeFamily(Guid idScope): List<Scope>	Devuelve los subámbitos del ámbito especificado por parámetro.
HeredarScopeData(Scope child, Scope parent): bool	Copia los datos del ámbito especificado por parámetro en los subámbitos de él.
FindScopeByAdminUser(Guid idUser): List<Scope>	Devuelve los ámbitos del usuario administrativo que se especifica por parámetro.
Asociaciones	*Scope, ScopeController*

Tabla 12: Descripción de la clase "Group Manager"

Nombre:	*GroupManager*
Tipo de Clase:	Controladora
Descripción:	Esta clase es la encargada de implementar todos los métodos de la interfaz *IGroupManager*. Maneja todas las acciones que se deseen realizar sobre los grupos.
Atributos	
Nombre	**Descripción**
_dalGroup: IDALGroupManager	Objeto de la clase interfaz de acceso a datos *IDALGroupManager*.
Métodos	
Nombre	**Descripción**
AddGroup(Group group): bool	Permite adicionar un nuevo grupo.
FindAllGroup():List<Group>	Devuelve todos los grupos existentes.
FindGroupById(string id): Group	Retorna el grupo que posea el identificador especificado por parámetro
FindGroupByName(string p, Guid idScope): Group	Retorna el grupo que posea el nombre especificado por parámetro

RemoveGroup(Group group): bool	Elimina el grupo que se especifica por parámetro.
FindAllGroupByScope(string idScope): List<Group>	Devuelve todos los grupos del ámbito especificado por parámetro.
UpdateGroup(Group group):bool	Actualiza el grupo especificado por parámetro.
Asociaciones	*Group, GroupController*

Las demás descripciones de clases se encuentran especificadas en el **Anexo IV**.

3.2.5. Clases persistentes

Las clases persistentes se definen para conocer la información real representada en las tablas de la base de datos.

3.2.5.1 Descripción de las clases persistentes

Tabla 13: Descripción de la clase "Scope"

Nombre de la Clase:	*Scope*
Tipo de Clase:	Entidad
Atributo	**Tipo**
id	*Guid*
name	*string*
description	*string*
childs	*Scope*
parents	*Scope*
Responsabilidad	
Nombre	**Descripción**
Scope ()	Constructor

Tabla 14: Descripción de la clase "Group"

Nombre de la Clase:	*Group*
Tipo de Clase:	Entidad
Atributo	**Tipo**
id	*Guid*
name	*string*
description	*string*
resources	*Resource*

role	*Role*
scope	*Scope*
usuarios	*User*
Responsabilidad	
Nombre	**Descripción**
Group ()	Constructor

Las otras descripciones de clases persistentes se encuentran especificadas en el **Anexo V**.

3.2.6. Modelo de datos

Los modelos de datos que se muestran en las figuras 9 y 10 están basado en las necesidades de persistencia del sistema, su principal objetivo es representar relaciones coherentes y optimizadas entre cada una de las entidades de la base de datos, logrando así un uso eficiente del espacio de almacenamiento, evitando la persistencia de datos redundadas y agilizando la ejecución de consultas para la recuperación de información. Los restantes modelos de datos pertenecientes a cada uno de los módulos del sistema están evidenciados en el **Anexo II**.

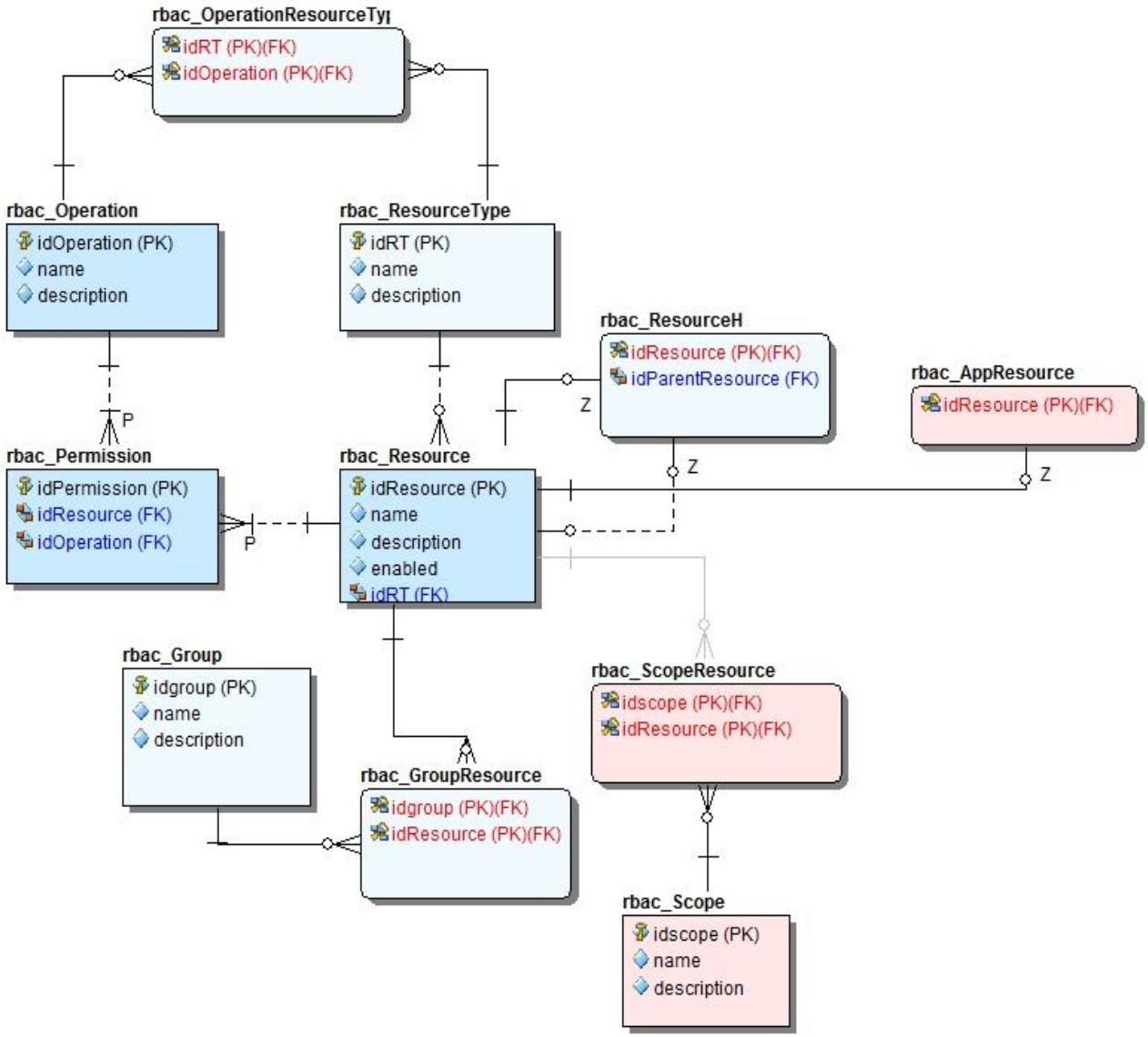


Figura 9: Modelo de datos del módulo “Gestión de Recursos”

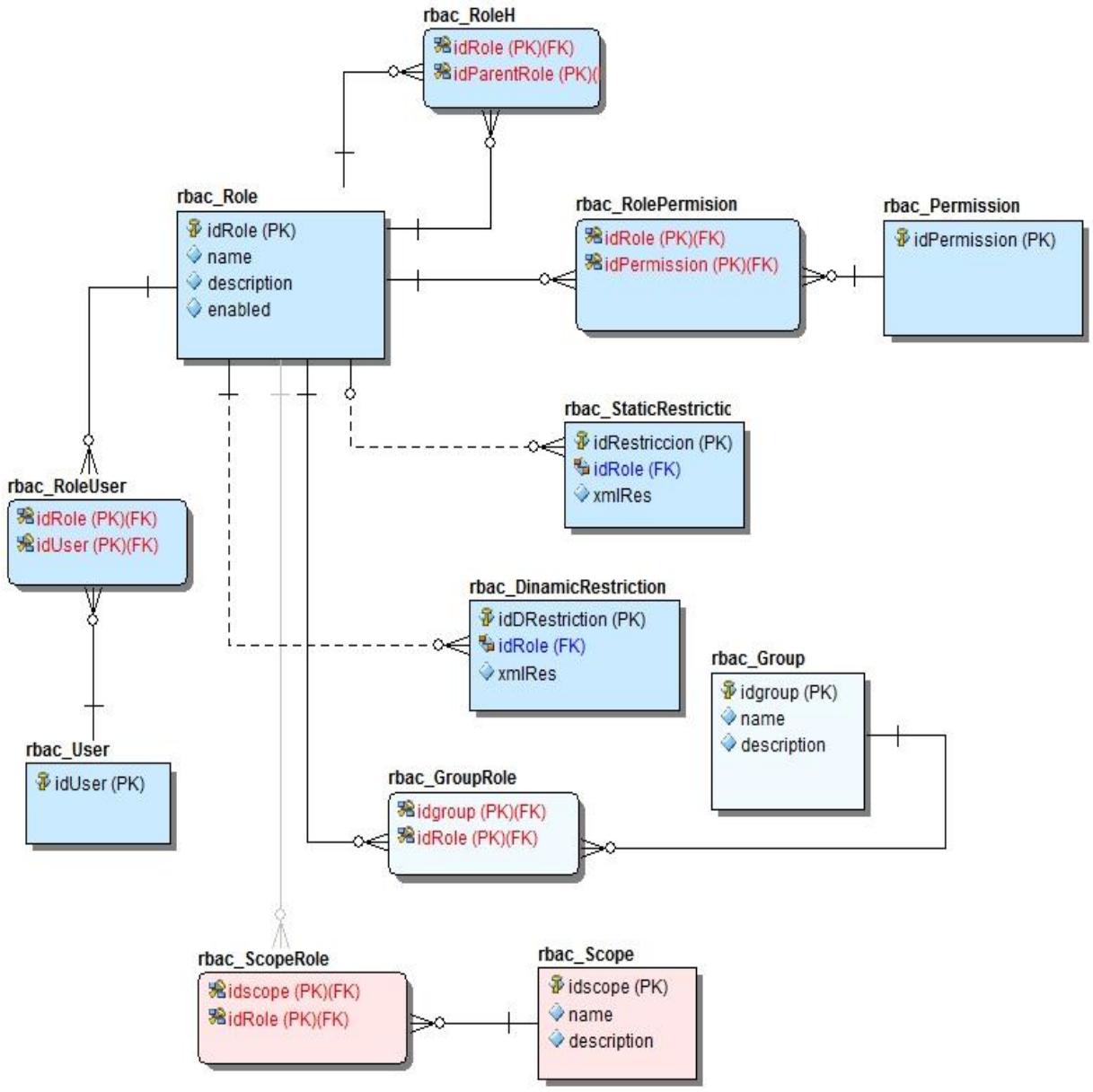


Figura 10: Modelo de datos del módulo “Gestión de Roles”

3.2.7. Diagrama de aplicación

Un diagrama de aplicación muestra los elementos de despliegue, como los servicios web, aplicaciones web, aplicaciones Windows y las aplicaciones a las que se hace referencia como bases de datos externas, servicios web externos y los servicios externos de *BizTalk*.

En el caso del sistema para la gestión de la configuración de los parámetros del modelo RBAC v2.0 este diagrama muestra las conexiones entre las aplicaciones antes mencionadas reflejando la configuración actual de la solución.

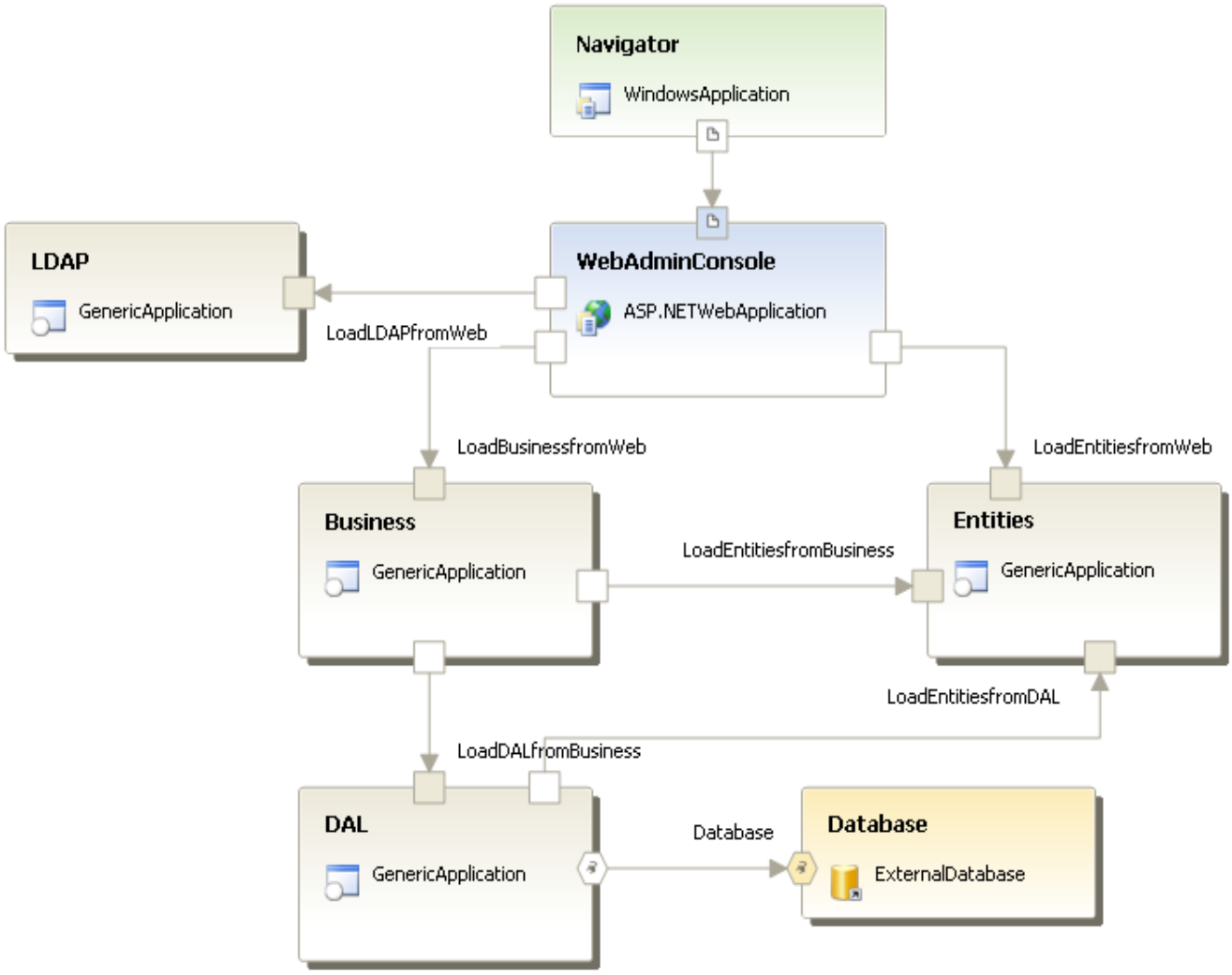


Figura 11: Diagrama de aplicación

3.2.8. Diagrama de centro de datos lógicos

Un diagrama de centros de datos lógicos define o documenta las configuraciones específicas de *software* de los servidores de las aplicaciones, como *Internet Information Server*, *SQL Server* o *BizTalk Server*, que tienen un propósito específico, como un servidor web seguro. El diagrama de centro de datos lógicos del sistema de gestión del RBAC muestra cómo estos servidores lógicos configurados están interconectados entre sí. Los servidores lógicos se pueden agrupar dentro de las

zonas que definen los límites lógicos de comunicación. Las zonas se pueden configurar para restringir los tipos de servidores lógicos que pueden contener y la dirección y tipo de comunicación que puede fluir dentro y fuera de la zona.

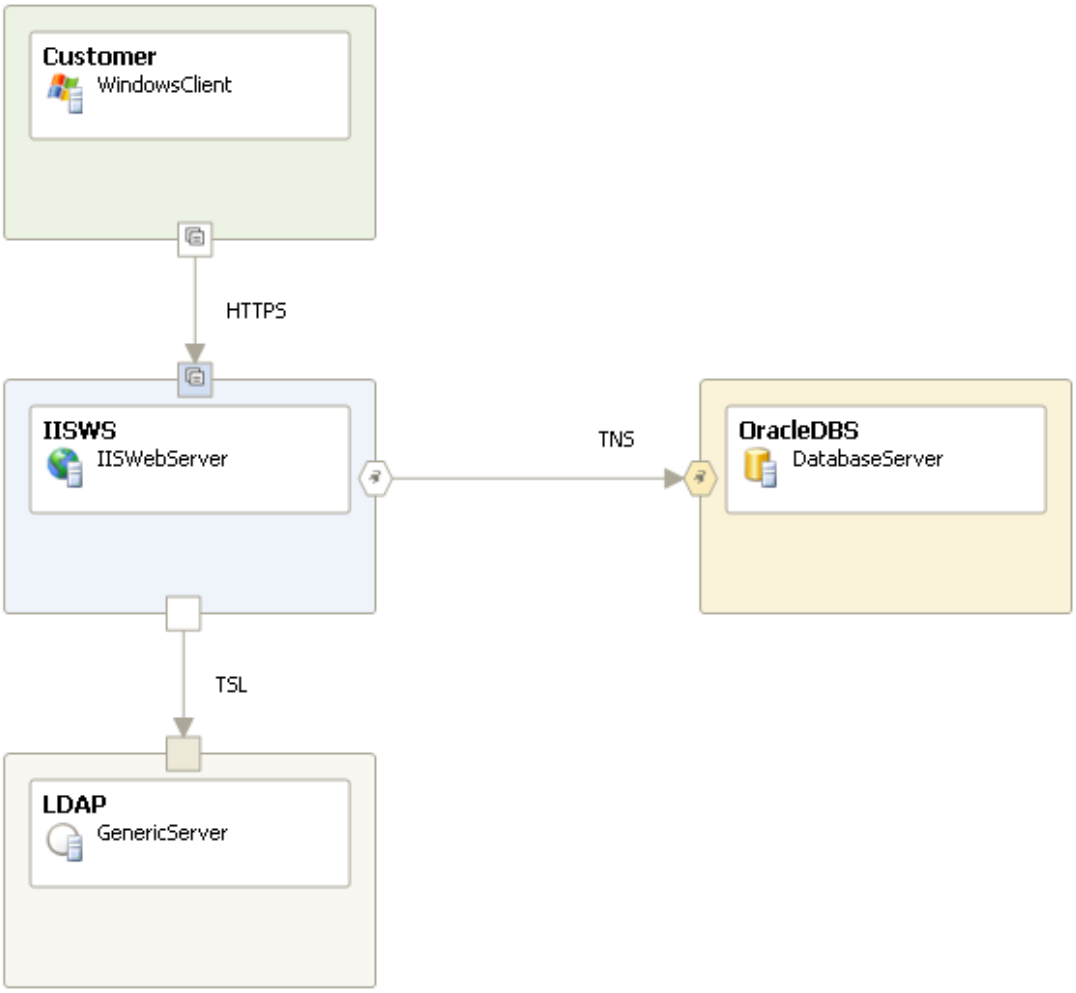


Figura 12: Diagrama de centro de datos lógicos

3.2.9. Implementación de escenarios

Teniendo en cuenta la planificación realizada en el capítulo anterior, se llevaron a cabo tres iteraciones de desarrollo sobre el sistema, obteniéndose un producto con funcionalidad en cada una de ellas. A continuación se detallan cada una de las iteraciones.

3.2.9.1 Iteración 1

En esta iteración se implementaron los escenarios correspondientes a los módulos especificados para la primera iteración, con el objetivo de obtener una versión del producto con algunas de las funcionalidades más importantes.

Tabla 15: Implementación en la iteración 1

Módulos	Tiempo de Implementación (horas)	
	Estimación	Tiempo real
Gestión de Ámbitos	228	220
Gestión de Grupos	198	198
Gestión de Roles	116	110

3.2.9.2 Iteración 2

En esta iteración se implementaron los escenarios correspondientes al módulo “Gestión de recursos” y “Gestión de restricciones” con una prioridad alta para el usuario.

Tabla 16: Implementación en la iteración 2

Módulos	Tiempo de Implementación (horas)	
	Estimación	Tiempo real
Gestión de Recursos	200	189
Gestión de Restricciones	116	110

3.2.9.3 Iteración 3

En esta iteración se implementaron los escenarios de prioridad media. Al finalizar se cuenta con un producto listo para ser probado y puesto en funcionamiento.

Tabla 17: Implementación en la iteración 3

Módulos	Tiempo de Implementación (horas)	
	Estimación	Tiempo real
Resto de los escenarios	72	72

3.2.10. Interfaz de usuario

La interfaz que presentará la aplicación web destinada al control de acceso basado en roles del Sistema de Administración de Identidades no será la misma que se utilizó en la primera versión pues se incluirán nuevos componentes para visualizar la información lo que le facilitará al usuario la interacción con los elementos del sistema. Todas las páginas deben tener colores claros basados fundamentalmente en el blanco y azul y el tamaño de la letra no debe exceder de los 11 px. Los aspectos mencionados garantizan que el sistema sea agradable al usuario y muy fácil de usar, pues le permite adaptarse cómodamente al área de trabajo que ante él se despliega.

Gestión de roles: Interfaz mediante la cual los administradores de roles podrán ver los roles administrativos existentes y los usuarios asignados a cada rol administrativo.

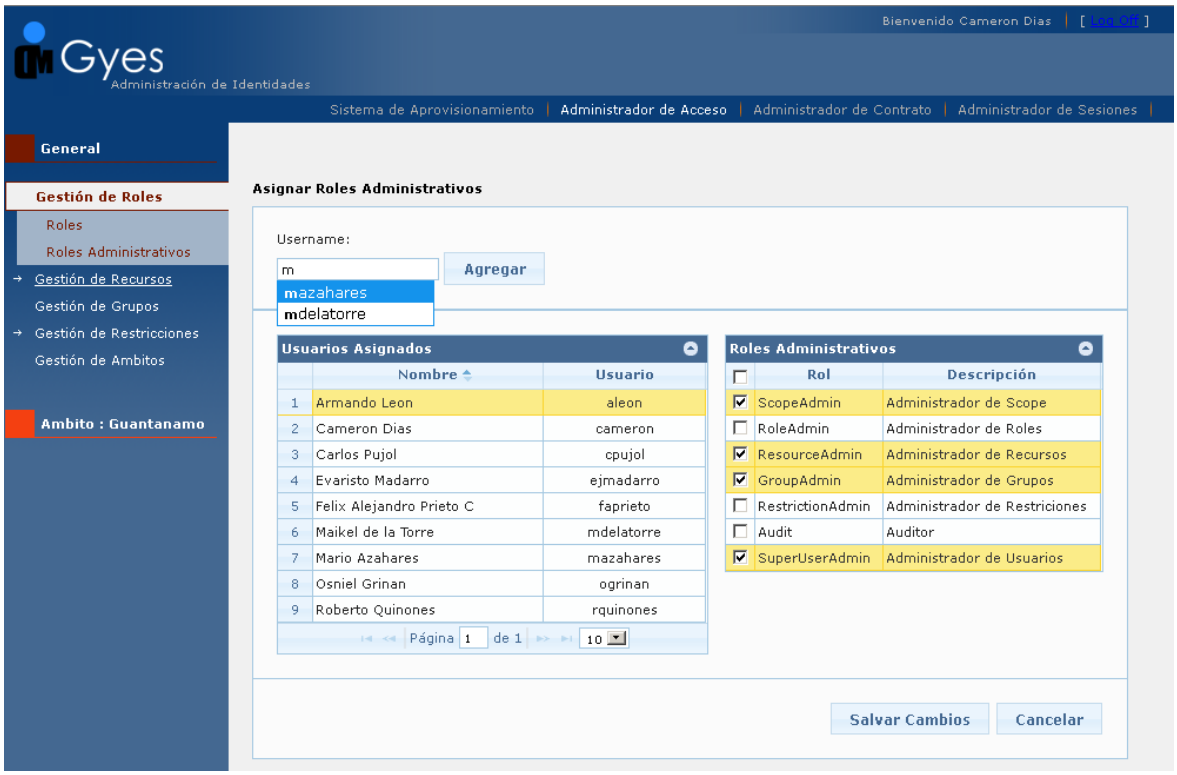


Figura 13: Interfaz Gráfica del Módulo “Gestión de Roles”

3.3. Pruebas

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requisitos especificados, los resultados son observados y registrados y se hace una evaluación del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del sistema y representa una revisión final de las especificaciones del diseño y de la codificación.

3.3.1 Pruebas funcionales

Para la validación de las funcionalidades se diseñaron y aplicaron pruebas funcionales para cada escenario del sistema. Esta sección cubre el conjunto de pruebas funcionales relacionadas con los escenarios: Gestionar ámbito y Gestionar grupos, pues el resto de las pruebas funcionales se encuentran descritas en el **Anexo VI**.

Descripción de la funcionalidad Gestionar Ámbito: En este escenario hay que comprobar la creación de un ámbito en la base de datos. Si los datos del ámbito son correctos se debe crear satisfactoriamente. Si existen errores mostrar mensajes indicando los mismos. Se debe además comprobar la modificación de un ámbito previamente seleccionado. Si los datos modificados son correctos, la modificación fue satisfactoria, en caso contrario mostrar mensajes indicando los errores que existen. También se comprueba la eliminación de un ámbito y los efectos que puede producir esta acción en el sistema.

Condiciones de ejecución:

- ✓ El administrador de ámbito debe estar previamente autenticado y autorizado.
- ✓ Debe estar disponible la base de datos que utiliza el sistema de gestión.

Tabla 18: Caso de prueba escenario: Gestionar ámbito.

Escenario	Descripción	Texto	Respuesta del Sistema	Flujo Central
Crear ámbito	Permite crear un ámbito en el sistema.	Nombre, descripción (Válido)	Se crea el ámbito correctamente.	Se oprime clic derecho sobre el ámbito que se desea crear un nuevo ámbito. Se muestra la opción “Crear”. Se introducen los datos del nuevo ámbito y luego se pulsa “aceptar”.
		Nombre, descripción (Inválido)	Muestra un mensaje de aviso informando que se entraron datos incorrectos.	
Editar ámbito	Permite modificar los datos de los	Nombre, descripción (Válido)	Se modifican los datos de los ámbitos correctamente.	Se oprime clic derecho sobre el ámbito que se

	ámbitos del sistema.	Nombre, descripción (Inválido)	Muestra un mensaje de aviso informando que se entraron datos incorrectos.	desea modificar. Se muestra la opción "Editar". Se introducen los nuevos datos y luego se pulsa "aceptar".
Eliminar ámbito	Permite eliminar un ámbito del sistema		Se elimina el ámbito correctamente.	Se oprime clic derecho sobre el ámbito que se desea eliminar. Se muestra la opción "Eliminar". Se muestra un mensaje de aviso. Finalmente se pulsa "aceptar".
			Muestra un mensaje de error informando que no se pudo eliminar el ámbito.	

Descripción de la funcionalidad Gestionar grupo: En este escenario hay que comprobar la creación de un grupo en la base de datos. Si los datos del grupo son correctos se debe crear satisfactoriamente. Si existen errores mostrar mensajes indicando los mismos. Se debe además comprobar la modificación de un grupo previamente seleccionado. Si los datos modificados son correctos, la modificación fue satisfactoria, en caso contrario mostrar mensajes indicando los errores que existen. También se comprueba la eliminación de un grupo y los efectos que puede producir esta acción en el sistema.

Condiciones de ejecución:

- ✓ El administrador de grupo debe estar previamente autenticado y autorizado.
- ✓ Debe estar disponible la base de datos que utiliza el sistema de gestión.

Tabla 19: Caso de prueba escenario: Gestionar grupo.

Escenario	Descripción	Texto	Respuesta del Sistema	Flujo Central
Crear grupo	Permite crear un grupo en el sistema.	Nombre, descripción (Válido)	Se crea el grupo correctamente	Se selecciona la opción “Gestión de grupos”. Se marca la opción “Crear”. Se introducen los datos del nuevo grupo y luego se pulsa “aceptar”.
		Nombre, descripción (Inválido)	Muestra un mensaje de aviso informando que se entraron datos incorrectos.	
Editar grupo	Permite modificar los datos de los grupos del sistema.	Nombre, descripción (Válido)	Se modifican los datos de los grupos correctamente.	Se selecciona la opción “Gestión de grupos”. Se oprime la opción “Editar”. Se introducen los nuevos datos y luego se pulsa “aceptar”.
		Nombre, descripción (Inválido)	Muestra un mensaje de aviso informando que se entraron datos incorrectos.	
Eliminar grupo	Permite eliminar un grupo del sistema		Se elimina el grupo correctamente.	Se selecciona la opción “Gestión de grupos”. Se oprime la opción “Eliminar”. Se muestra un mensaje de aviso. Finalmente se pulsa “aceptar”.
			Muestra un mensaje de error informando que no se pudo eliminar el grupo.	

3.3.2. Pruebas unitarias

Las pruebas unitarias son una de las formas que existen para probar pequeñas e individuales porciones de código. A través de ellas se verifica que cierto módulo o funcionalidad se ejecuta dentro de los parámetros especificados. Estas pruebas son una referencia muy importante para los desarrolladores, pues al ir probando pequeñas porciones de códigos posibilitan que luego cuando se

integren todos los componentes en un solo sistema, esta tarea se realice de la forma exitosa y con el menor número de errores posibles.

3.3.2.1 Pruebas de caja blanca

Otro de los métodos de prueba utilizados para detectar defectos al sistema desarrollado fue el de caja blanca. Este tipo de prueba evalúa distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados.

Ventajas de utilizar Visual Studio Team System para realizar pruebas unitarias.

- ✓ Proporcionan acciones preventivas sobre fallos.
- ✓ Son fáciles de automatizar
- ✓ Ideal para pruebas de aplicación en diferentes máquinas.
- ✓ Proporcionan modelo para la configuración, implementación y ejecución de código de aplicación real.

A continuación se muestran las pruebas realizadas a los métodos "*AddRoleAdmin*","*FindScopeById*" y *"UpdateScope":* Otros ejemplos de este tipo de pruebas se pueden consultar en el **Anexo VI**.

```
/// <summary>
///A test for _AddRoleAdmin
///</summary>
[TestMethod()]
public void _AddRoleAdminTest()
{
    DALRoleManager target = new DALRoleManager(); // TODO: Initialize to an appropriate value
    RoleAdmin _ra = new RoleAdmin("Admin","Administrador del Sistema",true); // TODO: Initialize to
    bool expected = true; // TODO: Initialize to an appropriate value
    bool actual;
    actual = target._AddRoleAdmin(_ra);
    Assert.AreEqual(expected, actual);
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 14: Prueba al método *"AddRoleAdmin"*

```
/// <summary>
///A test for FindScopeById
///</summary>
[TestMethod()]
public void FindScopeByIdTest()
{
    DALScopeManager target = new DALScopeManager(); // TODO: Initialize to an appropriate value
    string id = "569e80a9-2509-42ab-a51e-9b7318b1914d"; // TODO: Initialize to an appropriate value
    Scope expected = new Scope(new Guid("569e80a9-2509-42ab-a51e-9b7318b1914d"),"Guantanamo","Guant");
    Scope actual;
    actual = target.FindScopeById(id);
    Assert.AreEqual(expected.Id, actual.Id);
    Assert.AreEqual(expected.Name, actual.Name);
    Assert.AreEqual(expected.Description, actual.Description);
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 15: Prueba al método *"FindScopeById"*

```
/// <summary>
///A test for UpdateScope
///</summary>
[TestMethod()]
public void UpdateScopeTest()
{
    DALScopeManager target = new DALScopeManager(); // TODO: Initialize to an appropriate value
    Scope scope = new Scope(new Guid("5cf3423c-e260-45d4-b926-2316e8822c91"), "Yateras", "Yateras");
    bool expected = true; // TODO: Initialize to an appropriate value
    bool actual;
    actual = target.UpdateScope(scope);
    Assert.AreEqual(expected, actual);
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 16: Prueba al método *"UpdateScope"*

3.3.3 Resultado de las pruebas

Al realizar las pruebas unitarias en el sistema de gestión de la configuración de los parámetros del modelo RBAC los resultados obtenidos son los siguientes:

Tabla 20: Resultado de prueba al método *"AddRoleAdmin"*

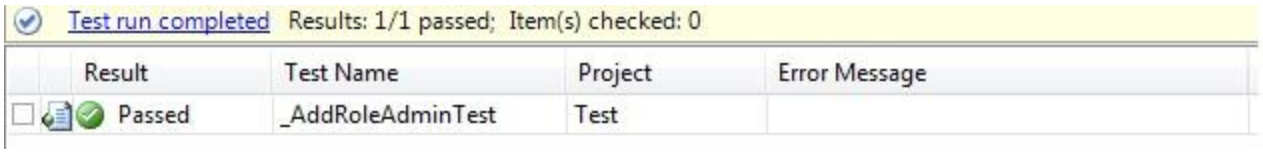
Prueba de unidad		
Nombre de prueba: *AddRoleAdminTest*		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima ejecución:** 7/06/2011
Ejecutado por: Yaimi Manso Machado	**Verificado por:** Maikel de la Torre Luis	
Descripción: Para poder ejecutar la prueba se deben pasar los datos correspondiente al rol administrativo que se desea adicionar, en caso de que se inserte correctamente se devuelve *"true"*, en caso contrario se devuelve *"false".*		
Entrada: *RoleAdmin ra*		
Criterio de aceptación: Adiciona los campos del rol administrativo en cuestión		
Resultado: 		

Tabla 21: Resultado de prueba al método *"FindScopeById"*

Prueba de unidad		
Nombre de prueba: *FindScopeById*		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima ejecución:** 7/06/2011
Ejecutado por: Yaimi Manso Machado	**Verificado por:** Maikel de la Torre Luis	
Descripción: Para poder ejecutar la prueba se debe pasar un tipo de dato *string* correspondiente al identificador del ámbito, en caso de que exista un ámbito con este identificador, el método debe buscarlo y retornarlo, en caso contrario devuelve vacío.		
Entrada: *string id*		
Criterio de aceptación: Muestra un objeto de tipo ámbito, con el *id* del ámbito, el *nombre* del ámbito y la *descripción*.		
Resultado:		

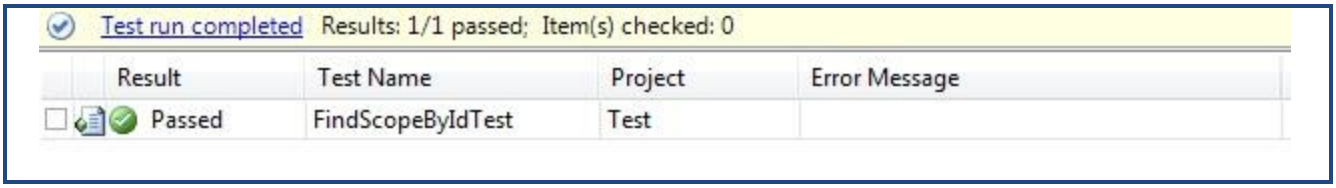
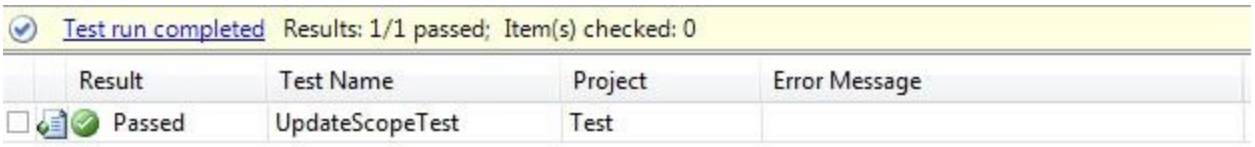


Tabla 22: Resultado de prueba al método *"UpdateScope"*

Prueba de unidad		
Nombre de prueba: *UpdateScope*		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima ejecución:** 7/06/2011
Ejecutado por: Yaimi Manso Machado	**Verificado por:** Maikel de la Torre Luis	
Descripción: Para poder ejecutar la prueba se deben pasar los datos correspondientes a los campos que se quieren modificar del ámbito en caso que se actualice correctamente se devuelve "*true*", en caso contrario devuelve "*false*".		
Entrada: *Scope scope*		
Criterio de aceptación: Actualiza los campos del ámbito en cuestión.		
Resultado:		



Conclusiones

En el presente capítulo se detalló la arquitectura del sistema basando la misma en una arquitectura 3 capas (presentación, negocio y acceso a datos) donde se le asignó una función específica a cada una de ellas. Para el desarrollo de la propuesta de solución se tuvieron en cuenta diferentes patrones de diseño, tales como, fachada, experto, controlador, permitiendo agilizar el proceso de implementación. Finalmente se realizaron pruebas para verificar que las funcionalidades se ejecutan dentro de los parámetros especificados.

Conclusiones generales

Luego de aplicar los métodos y herramientas en el proceso de investigación y desarrollo, se alcanzaron las siguientes conclusiones:

- ✓ La implementación de nuevas funcionalidades para el sistema de gestión de la configuración de los parámetros del modelo de control de acceso basado en roles, garantizó el crecimiento de la aplicación a una segunda versión mejorando considerablemente el desempeño del sistema.

- ✓ La metodología ágil MSF, empleada para guiar el proceso de desarrollo, se ajustó perfectamente a la aplicación, optimizando el tiempo de implementación y aumentando la calidad del producto final.

- ✓ La especificación de la arquitectura en 3 capas y la utilización del patrón MVC garantizaron una mayor flexibilidad en la ejecución de la propuesta de solución.

- ✓ La validación del sistema demostró que el mismo cumple con las expectativas del cliente.

Dadas las características del sistema de gestión del RBAC, se concluye que podrá cumplir satisfactoriamente su labor dentro del Sistema de Administración de Identidades.

Recomendaciones

Una vez concluido el presente trabajo de diploma se recomienda:

- ✓ Graficar la ingeniería de roles para facilitar la interacción del usuario con la aplicación.
- ✓ Ampliar lo concerniente a las restricciones para establecer un mejor control sobre los parámetros que se gestionan.

Referencias bibliográficas

1. *Implementación de la definición estandarizada por la NIST del modelo de Control de Acceso Basado en Roles, para un sistema de administración de identidades en intranets corporativas*. Vega Cutiño, Ruth Yurina, de la Torre Luis, Maikel, Madarro Capó, Evaristo José, Griñán Rodriguez, Osniel. Ciudad Habana.

2. **Sánchez, Miguel, Jiménez, Beatriz y Gutiérrez, Francisco L**. [En línea] [Citado el: 20 de Noviembre de 2010.] http://www.sistedes.es/sistedes/pdf/2007/SCHA-07-Sanchez-Acceso.pdf.

3. **Naranjo, Ibarra M Hyldeé y Argemi, Mañas, A José**. *RBAC: Alternativa actual para la realización de Control de Acceso a gran escala.pdf* [Citado el: 12 de noviembre de 2010]

4. **Ferraiolo David, Kuhn, Richard y Sandhu, David**. *The NIST Model for Role-Based Access Control__Towards A Unified Standard.pdf* [Citado el: 12 de noviembre de 2010]

5. **Solaris Corporation**. [En línea] [Citado el: 19 de Noviembre de 2010.] http://www.sun.com/software/whitepapers/wp-rbac/wp-rbac.pdf.

6. **Oracle Corporation.** [En línea] 2010. [Citado el: 2 de Diciembre de 2010.] http://www.oracle.com/products/middleware/identity-management/docs/intro-oracle-idm-whitepaper.pdf.

7. **Novalys Corporation.** [En línea] [Citado el: 25 de Noviembre de 2010.] http://www.visual-guard.com/download/VisualGuard-Detailed-Features-SP.pdf.

8. **Microsoft Corporation**. MSF for Agile Software Development Process Guidance. [En línea] 2006. [Citado el: 27 de Noviembre de 2010.]

9. **Hernández, Enriquez Orallo.** [En línea] 1999. [Citado el: 15 de Diciembre de 2010.] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

10. **Danysoft**. Haciendo visible lo invisible. [En línea] 2010. [Citado el: 14 de enero de 2011.] http://shop.danysoft.com/Altova-UModel.

11. **Danysoft**. Haciendo visible lo invisible. [En línea] 2010. [Citado el: 14 de enero de 2011.] http://shop.danysoft.com/embarcadero.

12. **Extremo, Unai Baigorri y Sotomayor, Borja Basilio.** [En línea] [Citado el: 16 de enero de 2011.] http://www.people.cs.uchicago.edu/~borja/pubs/revistaeside2002.pdf.

13. **Vilá, Fermí.** [En línea] [Citado el: 20 de Enero de 2011.] http://sgonzalez.debianchile.cl/files/biblioteca_nerd/Java_Script/Java_script.pdf.

14. **Kioskea .Net**. [En línea] [Citado el: 20 de Enero de 2011.] http://es.kioskea.net/contents/css/cssintro.php3.

15. [En línea] 15 de Diciembre de 2010. [Citado el: 15 de Enero de 2011.] http://lamonterapicona.es/2011/01/15/la-plataforma-net/.

16. [En línea] 10 de Enero de 2010 [Citado el 20 de Enero de 2011] http://www.intercambiosvirtuales.orgsoftwaremicrosoft-visual-studio-2010-ultimate-espanol

17. **Hernando, Roberto Velasco.** [En línea] [Citado el: 10 de Enero de 2011.] http://www.rhernando.net/modules/tutorials/doc/bd/oracle.html.

18. [En línea] [Citado el 20 de Enero de 2011] http://subversion.apache.org/

19. **MSDN. (2010)** [En línea] [Citado el: 20 de Enero de 2011.] http://msdn.microsoft.com/es-es/asp.net/aa336522.

20. [En línea] [Citado el: 5 de Enero de 2011.] https://ame.endesa.es/confluenceame/display/PPAME4NET/Entity+Framework.es-ES.

21. **Guthrie, Scott**. [En línea] 17 de Octubre de 2010. [Citado el: 12 de enero de 2011.] http://weblogs.asp.net/scottgu/archive/2007/10/14/asp-net-mvc-framework.aspx.

22. **Mestras Pavón, Juan.** [En línea] Abril de 2006. [Citado el: 10 de Abril de 2011.] http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

23. **Canales, Roberto.** [En línea] 22 de Diciembre de 2003. [Citado el: 25 de Marzo de 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp.

24. **Programación en castellano.** [En línea] [Citado el: 12 de Abril de 2011.] http://www.programacion.com/articulo/diseno_de_software_con_patrones_parte_4_145.

Bibliografía

Chandramouli, R., Kuhn, D. R., & Ferraiolo, D. (2007). *Role Based Access Control.* Norwood, MA: ARTECH HOUSE, INC.

Chapell, David. (2010). *Introducing Visual Studio 2010.* [En línea] 2010. http://www.davidchapell.com/introducing_visual_studio_2010_chapell.pdf

Madarro Capó, Evaristo., Rodriguez Griñán, Osniel. (2010). *Aplicación para la gestión de la configuración del RBAC en el Sistema de Administración de Identidades".* Ciudad de La Habana, Cuba: Universidad de las Ciencias Informáticas, 2010.

Mendoza Sánchez, María, A. (2004). *Metodologías de desarrollo de software.* http://www.informatizate.net

MSDN: Inicio de .NET Framework. (2010) [En línea] 2010. http://msdn.microsoft.com/es-es/netframework/default.aspx.

Pérez, Lamancha, Beatriz. (2007) *Gestión de las Pruebas Funcionales*. [En línea] 2010. http://www.calidadysoftware.comtestingpruebas_funcionales.php.

Sanderson, S. (2009). *Pro ASP .NET MVC Framework*. New York.

Villarroel, González, Luís R., Montalvo Cesar. A. (2008) *Aplicación de la Metodología MSF 4.0 a la definición e implementación de arquitecturas orientadas a objetos.* SANGOLQUI.

Vogelmann Martínez, Estela. (2008). *Políticas y modelos de seguridad.* FACENA-UNNE. Argentina.

Glosario de términos

***Active Directory*:** Es una base de datos que permite almacenar información relativa a los recursos de una red con el fin de facilitar su localización y administración.

***AJAX*:** Java Script asíncrono y XML.

Apache/BSD: Es una licencia de software libre creada por la Apache Software Foundation.

Biz Talk: 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.

***Clúster*:** Es un conjunto de servidores independientes pero interconectados.

CLR: Es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET.

CVS: Es una aplicación informática que implementa un sistema de control de versiones.

***DSD*:** Restricción dinámica de separación de responsabilidades.

***DAC*:** Control de Acceso Discrecional.

Framework: Es un marco de trabajo que expone un conjunto de bibliotecas que no forman parte de una aplicación específica y que han sido creadas para ser utilizados por cualquier aplicación. Sus componentes pueden ser usados para propósitos generales y están ampliamente probados para funcionar en gran variedad de escenarios.

***GRASP*:** Patrones de asignación de responsabilidad.

GOF: Patrones creacionales, estructurales y de comportamiento.

***HTML*:** Lenguaje de Marcado de Hipertexto.

JSP: Es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

Jquery: Es una librería basada en Java Script para acceder a los objetos del DOM de un modo simplificado.

Glosario de términos

LINQ: Es un proyecto de Microsoft que agrega consultas nativas semejantes a las de SQL a los lenguajes de la plataforma.

***LDAP*:** Protocolo Ligero de Acceso a Directorios.

MSF: Microsoft Solution Framework.

***MAC*:** Control de Acceso Obligatorio.

MVC: Patrón de arquitectura Modelo-Vista-Controlador.

***NIST*:** Instituto Nacional de estándares y tecnologías.

***RBAC*:** Control de acceso basado en roles.

SSO: Es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación

***SSD*:** Restricción estática de separación de responsabilidades.

SQL: Es un lenguaje declarativo de acceso a bases de datos.

***TBAC*:** Control de acceso basado en tareas.

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

Viewstate: Mantener el estado de los controles de una misma página en una petición al servidor.

***XML*:** Lenguaje de marcado extensible.

Anexos

Anexo I: Descripción de escenarios

Descripción del escenario "Asignar usuario al ámbito"

Nombre del Escenario: Asignar usuario al ámbito		**Identificador:** RBAC 1.5
Objetivo del Escenario: Permitir la asignación de usuarios al ámbito previamente seleccionado.		
Persona: Administrador de ámbito		
Iteración: 3ra	**Prioridad:** 2	**Complejidad:** 1
Descripción: El administrador de ámbito accede a la aplicación con la intención de asignar un usuario a un determinado ámbito. Después de autenticarse, el sistema muestra en pantalla los ámbitos en los cuales el administrador tiene permisos de acceso. El administrador de ámbito selecciona el ámbito en el cual desea trabajar y el sistema muestra la información referente al ámbito seleccionado y brinda la opción Asignar Usuario. Esta opción permite que el administrador pueda asignar un usuario existente en el directorio de identidades del ámbito o puede crear un nuevo usuario (Ver Escenario Crear Usuario) y asignarlo al ámbito. El Administrador oprime el botón Asignar Usuario y el sistema muestra un listado de todos los usuarios que se encuentran registrados en el directorio de identidades, el administrador de ámbito escoge el usuario que quiere asignar y oprime el botón Aceptar. El sistema asigna el usuario al ámbito y actualiza el listado de usuarios del ámbito mostrándolo finalmente.		
Validaciones: • El sistema debe verificar los permisos del administrador.		

Descripción del escenario "Revocar asignación de usuario al ámbito"

Nombre del Escenario: Revocar asignación de usuario al ámbito		**Identificador:** RBAC1.6
Objetivo del Escenario: Permitir la eliminación de un usuario de la lista de usuarios asociados a un ámbito.		
Persona: Administrador de ámbito		
Iteración: 3ra	**Prioridad:** 2	**Complejidad:** 1
Descripción: El administrador de ámbito desea revocar un usuario de un ámbito determinado. Después de		

autenticarse en el sistema, este le muestra los ámbitos a los cuales el administrador de ámbito tiene permisos de acceso.

Después de seleccionado el ámbito por el administrador, el sistema muestra la información referente al ámbito seleccionado, muestra el listado de usuarios existentes en el directorio de identidades y brinda la opción de Revocar Usuario.

Esta opción permite al administrador de ámbito un sistema de búsqueda el cual le permite encontrar el usuario fácilmente, de lo contrario el administrador puede buscarlo y seleccionarlo directamente del listado de usuarios.

Después de ser seleccionado el usuario, el administrador de ámbito oprime el botón Revocar y aparece en pantalla una ventana con la siguiente información: ´´ Está seguro que desea revocar el usuario del ámbito seleccionado´´, y muestra el botón Revocar y Cancelar.

El administrador de ámbito selecciona el botón Eliminar y el sistema actualiza el listado de usuarios del ámbito seleccionado.

El administrador de ámbito también tiene la opción de no revocar el usuario del ámbito seleccionando el botón Cancelar.

Finalmente el sistema realiza las operaciones correspondientes en dependencia de la opción seleccionada por el administrador.

Validaciones:

- El sistema debe verificar los permisos del administrador.

Descripción del escenario "Gestionar grupo"

Nombre del Escenario: Gestionar grupo		**Identificador:** RBAC2
Objetivo del Escenario: Brindar la posibilidad de crear, editar, mostrar y eliminar uno o varios grupos en el sistema.		
Persona: Administrador de grupo		
Iteración: 1ra	**Prioridad:** 1	**Complejidad:** 3

Descripción:

El administrador de ámbito accede a la aplicación con la intensión de Gestionar grupo. Una vez autenticado en la aplicación, se muestra en pantalla todos los ámbitos existentes dentro del nivel institucional en los cuales el administrador de grupo tiene los permisos de acceso asignados para realizar las operaciones.

Entre las operaciones que puede gestionar el administrador de grupo se encuentran la de Crear grupo cumpliendo con la estructura jerárquica establecida; la de Editar y Eliminar grupo, teniendo siempre como precondición que sea seleccionado previamente el ámbito para realizar dichas acciones sobre el grupo que pertenece a este ámbito; y por último Mostrar las características

generales del grupo seleccionado. Para consultar las descripciones se recomienda ver escenarios: • Crear grupo (RBAC2.1) • Editar grupo (RBAC2.2) • Eliminar grupo (RBAC2.3) • Mostrar grupo (RBAC2.4)
Validaciones: • El sistema debe verificar los permisos del administrador.

Descripción del escenario "Crear grupo"

Nombre del Escenario: Crear grupo.		**Identificador:** RBAC2.1
Objetivo del Escenario: Crear grupos en los ámbitos del sistema.		
Persona: Administrador de grupo		
Iteración: 1ra	**Prioridad:** 1	**Complejidad:** 1
Descripción: El administrador de grupo accede a la aplicación con la intención de crear un nuevo grupo en un determinado ámbito. Una vez autenticado en el sistema, éste muestra en pantalla una estructura jerárquica de los ámbitos existentes dentro del ámbito institucional y le permite seleccionar un ámbito dentro del cual desea poder crear un nuevo grupo. El administrador de grupo selecciona el ámbito y el sistema le permite crear un nuevo grupo dentro de este y le muestra además, varios campos a ser llenados obligatoriamente que corresponden al Nombre y la Descripción del nuevo grupo. El administrador de grupo introduce los datos necesarios del grupo que ha creado, y oprime el botón Crear Grupo. Finalmente el sistema crea el nuevo grupo dentro del ámbito seleccionado.		
Validaciones: • El sistema debe verificar los permisos del administrador.		

Descripción del escenario "Asignar recurso a grupo"

Nombre del Escenario: Asignar recurso a grupo		**Identificador:** RBAC2.5
Objetivo del Escenario: Permitir la asignación de recursos a determinados grupos.		
Persona: Administrador de grupo		
Iteración: 3ra	**Prioridad:** 2	**Complejidad:** 1
Descripción: El administrador de grupo accede a la aplicación con la intención de asignar recurso a grupo. Después de autenticarse en la aplicación, se muestra en pantalla la estructura jerárquica definida de todos los ámbitos existentes dentro del nivel organizacional. El sistema le permite al administrador seleccionar el ámbito y el grupo al cual le asignará el usuario, y muestra la opción Asignar Recurso. El administrador selecciona el ámbito, después el grupo y finalmente oprime el botón Asignar Recurso. Entonces el sistema muestra un listado de todos los recursos existentes, el administrador selecciona entonces los recursos que desea asignar y finalmente oprime el botón Aceptar. El sistema asigna el recurso al grupo y actualiza el listado de usuarios del grupo mostrándolo finalmente.		
Validaciones: • El sistema debe verificar los permisos del administrador.		

Descripción del escenario "Asignar usuario a grupo"

Nombre del Escenario: Asignar usuario a un grupo.		**Identificador:** RBAC2.6
Objetivo del Escenario: Permitir la asignación de usuarios a el o los grupos pertenecientes a un ámbito específico.		
Persona: Administrador de grupo		
Iteración: 3ra	**Prioridad:** 2	**Complejidad:** 1
Descripción: El administrador de grupo accede a la aplicación con la intención de asignar usuario a grupo. Después de autenticarse en la aplicación, se muestra en pantalla la estructura jerárquica definida de todos los ámbitos existentes dentro del nivel organizacional a los cuales el tiene permisos de acceso. El sistema le permite al administrador seleccionar el ámbito y el grupo al cual le asignará el usuario, y muestra la opción Asignar usuario. El administrador selecciona primeramente el ámbito, después el grupo en el cual realizará las acciones y finalmente oprime el botón Asignar Usuario. Esta opción permite que el administrador pueda asignar un usuario existente en el directorio de identidades del ámbito o puede crear un nuevo usuario (Ver Escenario Crear Usuario) y asignarlo al		

grupo. El administrador oprime el botón Asignar Usuario y el sistema muestra un listado de todos los usuarios que se encuentran registrados en el directorio de identidades, el administrador de grupo escoge el usuario que quiere asignar y oprime el botón Aceptar. El sistema asigna el usuario al grupo y actualiza el listado de usuarios del grupo mostrándolo finalmente.
Validaciones: • El sistema debe verificar los permisos del administrador.

Descripción del escenario "Verificar permiso sobre recurso"

Nombre del Escenario: Verificar permiso sobre recurso		**Identificador:** RBAC3.5
Objetivo del Escenario: Verificar los permisos de los recursos		
Persona: Administrador de recursos		
Iteración: 3ra	**Prioridad:** 2	**Complejidad:** 3
Descripción: El administrador de recursos accede a la aplicación con la intención de verificar los permisos de un recurso. Después de autenticarse en la aplicación, se muestra en pantalla la estructura jerárquica definida de todos los recursos existentes a los cuales tiene permisos de acceso. El sistema le permite al administrador de recursos seleccionar el recurso. El administrador escoge un recurso de la estructura en forma jerárquica que muestra los recursos existentes. Por cada recurso que marque el administrador se muestran las operaciones establecidas para el recurso marcado y los roles asociados al recurso marcado. Además, el sistema muestra si el recurso seleccionado está asignado a algún rol o si el recurso esta heredando los permisos de algún rol. EL administrador de recursos una vez haya verificado los permisos sobre el recurso seleccionado puede reemplazar los permisos en los recursos hijos del recurso seleccionado (en caso de que este tenga recursos hijos) o puede copiar los permisos de su padre (en caso de que el recurso aún no tenga recursos hijos creados sobre él.)		
Validaciones: • El sistema debe verificar que se haya creado un recurso. • El sistema debe verificar que los permisos de los recursos padres que coincidan con los permisos de los recursos hijos no se repitan si los mismos se van a copiar.		

Descripción del escenario "Asignar permisos a rol"

<table>
<tr><td colspan="2">Nombre del Escenario: Asignar Permisos a rol</td><td>Identificador: RBAC4.4</td></tr>
<tr><td colspan="3">Objetivo del Escenario: Asignarle a los roles uno o varios permisos.</td></tr>
<tr><td colspan="3">Persona: Administrador de rol</td></tr>
<tr><td>Iteración: 3ra</td><td>Prioridad: 2</td><td>Complejidad: 2</td></tr>
<tr><td colspan="3">Descripción:

El administrador de rol accede a la aplicación con la intención de asignar permiso a un rol. Después de autenticarse en la aplicación, se muestra en pantalla la estructura jerárquica definida de todos los ámbitos existentes dentro del nivel organizacional a los cuales tiene permisos de acceso. El sistema le permite al administrador de rol seleccionar el ámbito, y una vez seleccionado, entonces le muestra un listado con todos los roles existentes para ese ámbito.

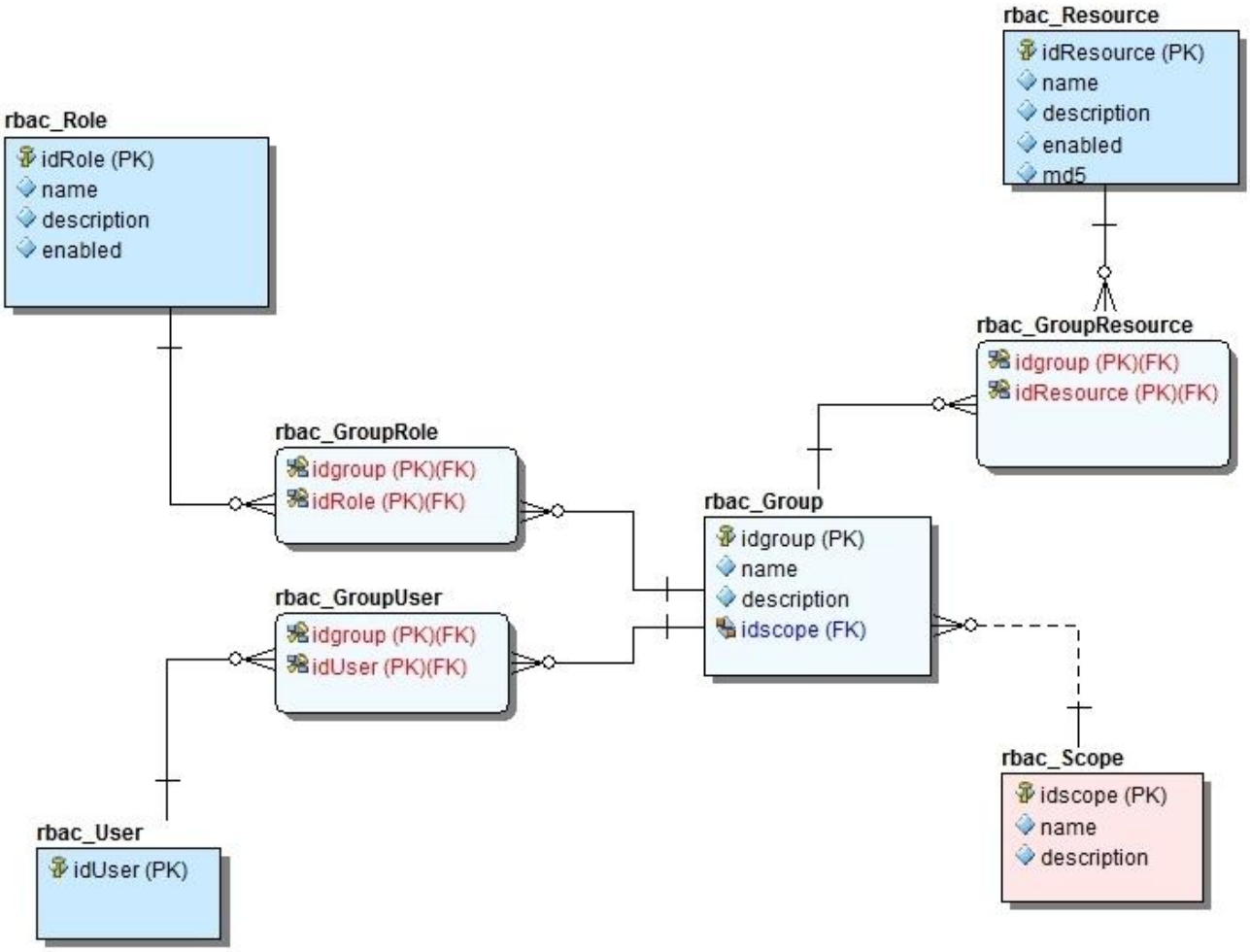
El administrador escoge un rol, y el sistema debe mostrarle en forma jerárquica todos los recursos asociados a ese rol. Por cada recurso que marque el administrador se muestran las operaciones establecidas para el recurso marcado. Además, el sistema muestra una tabla que contiene datos del recurso, como la ruta y el tipo de asignación del permiso.

El administrador selecciona esta opción, el sistema muestra el listado de los permisos existentes, escoge los roles a los cuales le quiere asignar el permiso, y oprime el botón Aceptar.

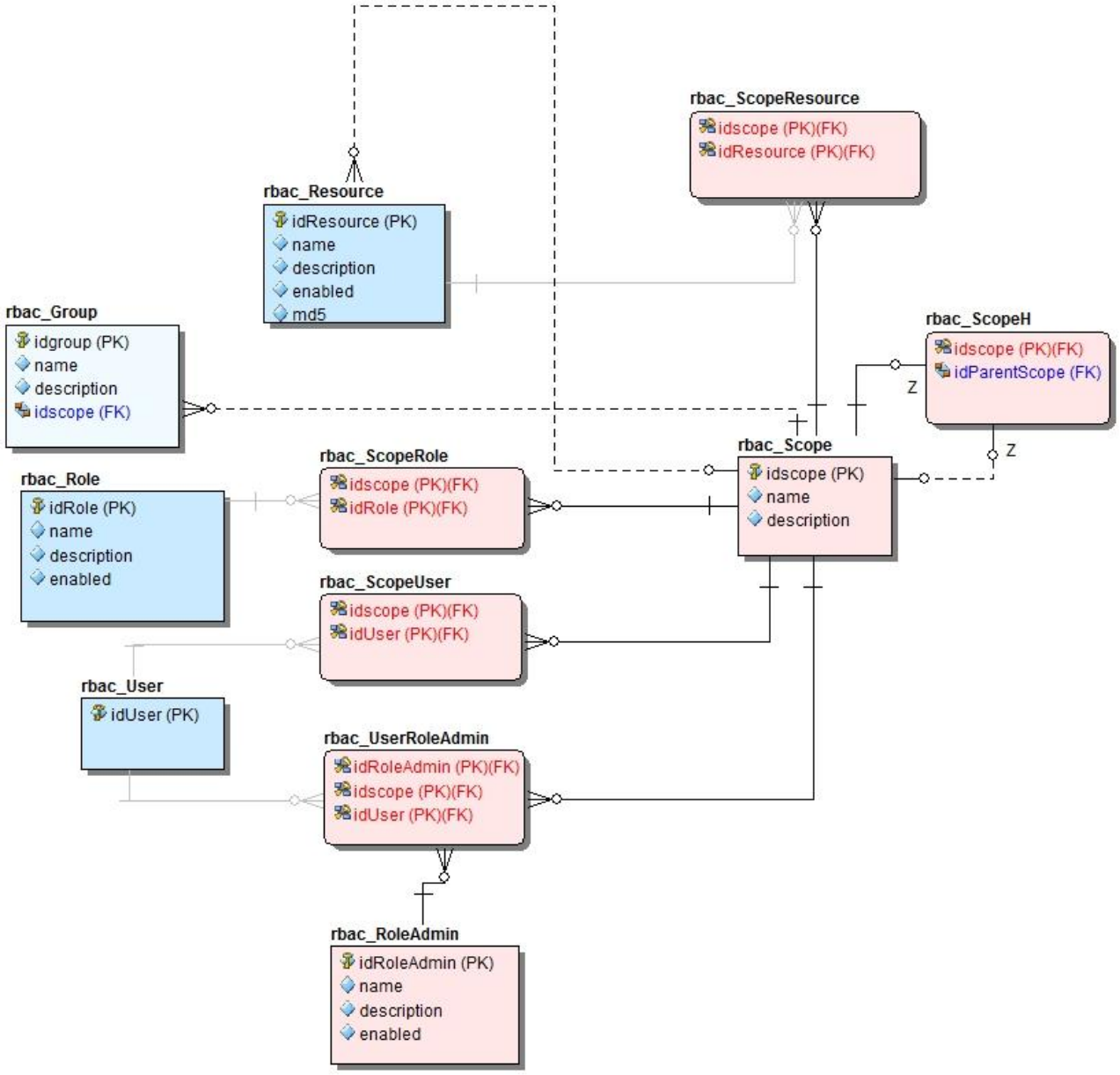
El Administrador de rol una vez que haya creado el rol puede asignarle varios permisos. Siempre se debe seleccionar antes el ámbito (al que pertenece el rol que se le van a asignar los permisos). Para ello se selecciona la opción Asignar Permisos a Rol. Se muestran los recursos existentes en una estructura jerárquica. Luego se debe seleccionar el recurso al que se le van a dar los permisos. Los permisos son de lectura, escritura o lectura y escritura. Al seleccionar el recurso se muestran operaciones predefinidas por tipo de recurso. Luego se crean los permisos correspondientes al recurso elegido.</td></tr>
<tr><td colspan="3">Validaciones:

• El sistema debe verificar que se haya creado un rol.</td></tr>
</table>

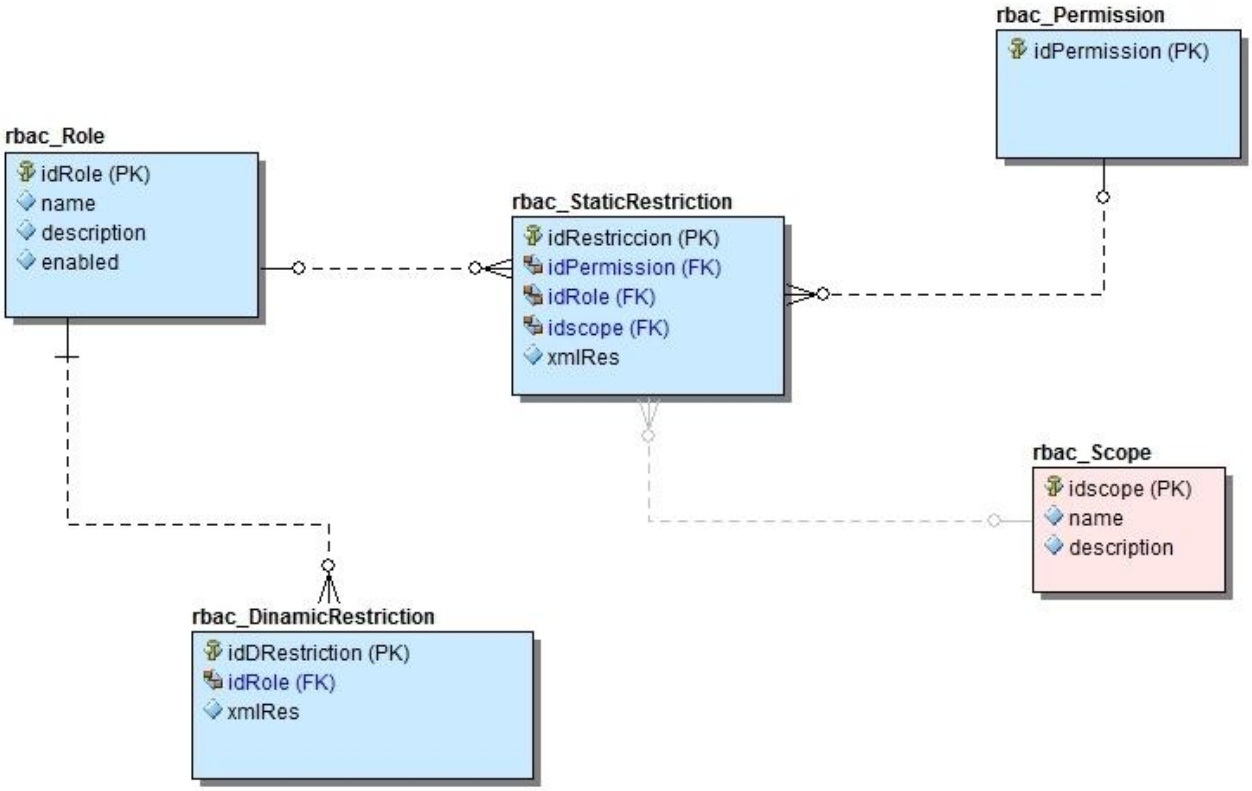
Anexo II: Modelo de datos



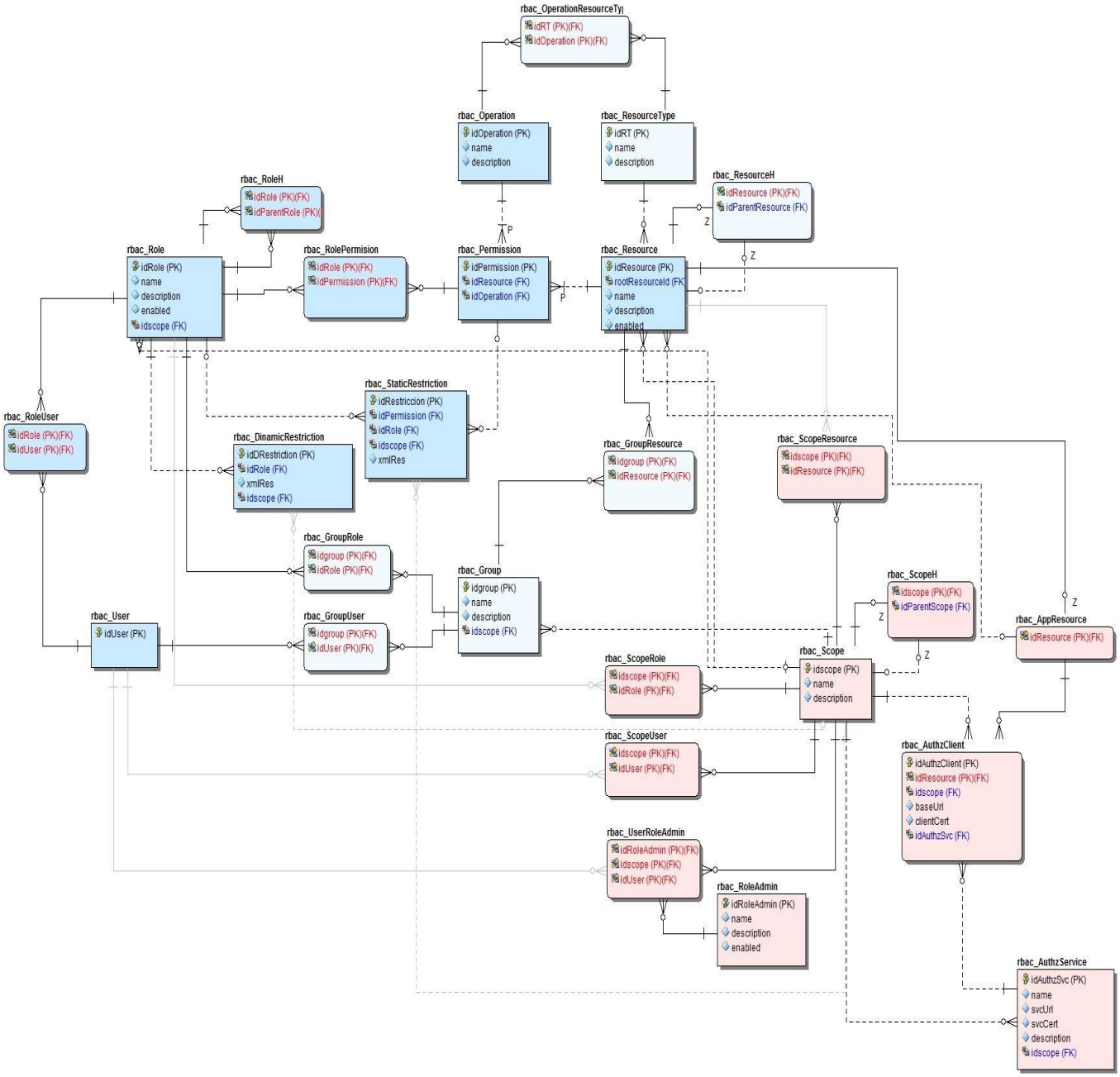
Modelo de datos del módulo "Gestión de Grupos"



Modelo de datos del módulo "Gestión de Ámbitos"



Modelo de datos del módulo "Gestión de Restricciones"



Modelo de datos general del sistema

Anexo III: Diagramas de clases

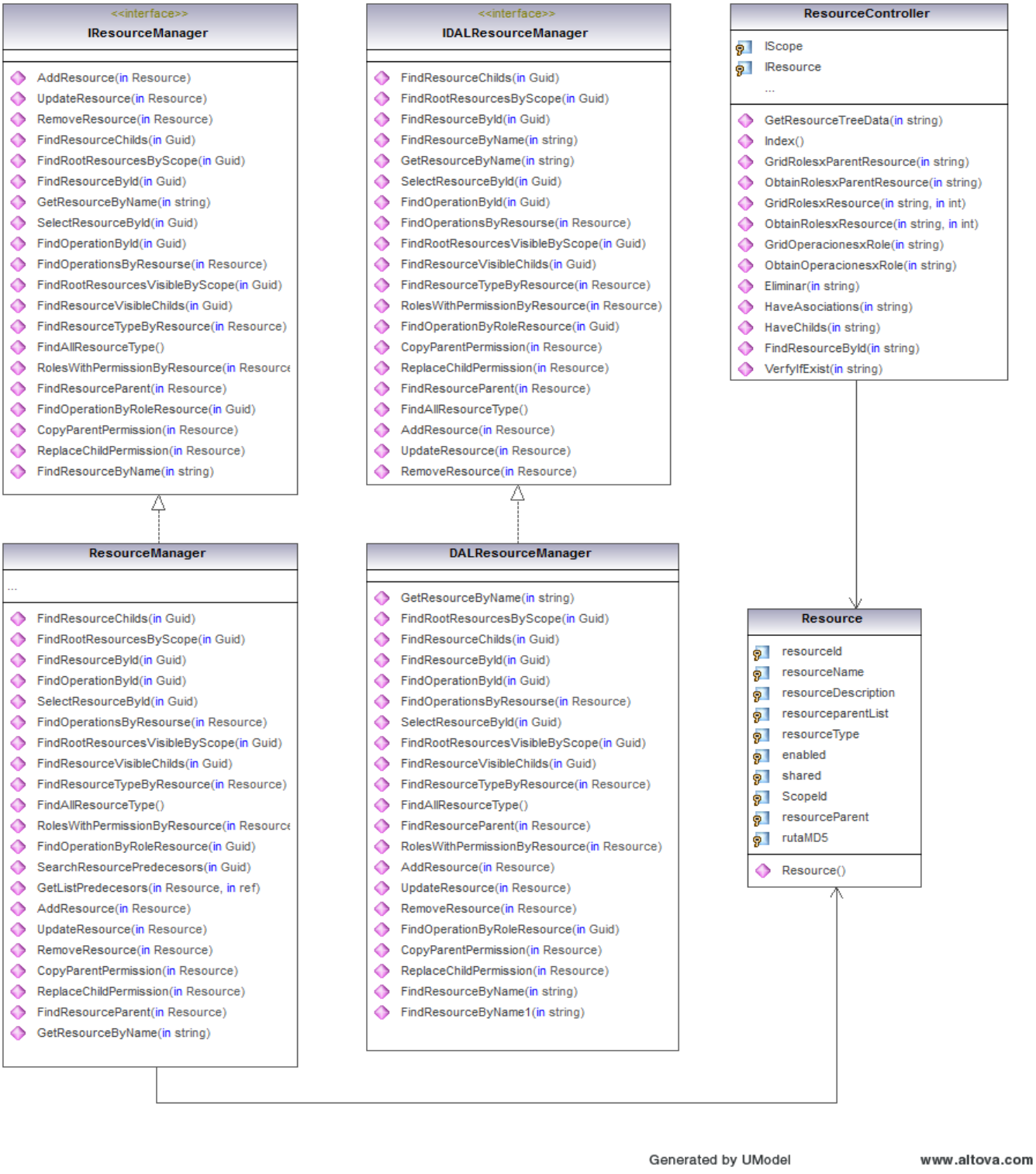


Diagrama de clases del módulo “Gestión de Recursos”

Anexo IV: Descripción de clases

Descripción de la clase "*ResourceManager*"

Nombre:	*ResourceManager*
Tipo de Clase:	Controladora
Descripción:	Esta clase es la encargada de implementar todos los métodos de la interfaz *IResourceManager*. Maneja todas las acciones que se deseen realizar sobre los recursos.
Atributos	
Nombre	**Descripción**
_dalresource: IDALResourceManager	Objeto de la clase interfaz de acceso a datos *IDALResourceManager*.
Métodos	
Nombre	**Descripción**
FindResourceChilds(Guid resourceId): List<Resource>	Devuelve una lista de los recursos hijos del recurso especificado por parámetro.
FindRootResourcesByScope(Guid scopeId): List<Resource>	Devuelve una lista de los recursos pertenecientes al ámbito especificado por parámetro.
FindResourceById(Guid resourceId):Resource	Devuelve el recurso con identificador igual al especificado por parámetro.
FindOperationById(Guid operationId):Operation	Devuelve la operación con identificador igual al especificado por parámetro.
SelectResourceById(Guid resourceId):Resource	Selecciona el recurso con identificador igual al especificado por parámetro.
FindOperationsByResource(Resource resource): List<Operation>	Devuelve una lista de las operaciones del re curso especificado por parámetro.
FindRootResourcesVisibleByScope(Guid scopeId): List<Resource>	Devuelve una lista de los recursos visibles para el ámbito especificado por parámetro.
FindResourceVisibleChilds(Guid resourceId, Guid scopeId): List<Resource>	Devuelve una lista de los recursos hijos visibles para el recurso especificado por parámetro.
FindResourceTypeByResource(Resource resource): ResourceType	Devuelve el tipo de recurso del recurso especificado por parámetro.

FindAllResourceType():List<Resource>	Devuelve una lista de todos los tipos de recursos.
RolesWithPermissionByResource(Resource resource):List<Role>	Devuelve una lista de roles con permisos sobre el recurso especificado por parámetro.
FindOperationByRoleResource(Guid roleId, Guid resourceId): List<Operation>	Devuelve una lista de operaciones con los recursos de los roles especificados por parámetro.
SearchResourcePredecesors(Guid idres): List<string>	Devuelve una lista de recursos predecesores
AddResource(Resource resource): bool	Devuelve verdadero si adiciona el recurso, falso en caso contrario.
UpdateResource(Resource resource):bool	Devuelve verdadero si actualiza el recurso, falso en caso contrario.
RemoveResource(Resource resource):bool	Devuelve verdadero si elimina el recurso, falso en caso contrario.
FindResourceByName(string name): Resource	Devuelve el recurso que tiene el nombre especificado por parámetro.
FindResourceParent(Resource child): Resource	Devuelve el recurso padre del recurso especificado por parámetro.
Asociaciones	*Resource, ResourceController*

Descripción de la clase "*RestrictionManager*"

Nombre:	*RestrictionManager*
Tipo de Clase:	Controladora
Descripción:	Esta clase es la encargada de implementar todos los métodos de la interfaz *IRestrictionManager*. Maneja todas las acciones que se deseen realizar sobre las restricciones.
Atributos	
Nombre	**Descripción**

document: XmlDocument	Objeto de tipo *XmlDocument.*
Métodos	
Nombre	**Descripción**
LoadXml(string FileName): XmlDocument	Devuleve el xml del archivo especificado por parámetro.
AddExclussionRole(List<Role> targets, List<Role> lists, string desde, string hasta):bool	Devuelve verdadero si se adiciona una restricción de exclusión de roles, falso en caso contrario.
FindAllExclussionRole():List<ExclusionRestriction>	Devuelve una lista de restricciones de exclusión de roles.
RemoveExclussionRole(Guid targetId, Guid excroleId):bool	Devuelve verdadero si se elimina una restricción de exclusión de roles, falso en caso contrario.
AddInclussionRole(List<Role> targets, List<Role> lists, string desde, string hasta): bool	Devuelve verdadero si se adiciona una restricción de inclusión de roles, falso en caso contrario.
FindAllInclussionRole():List<InclusionRestriction>	Devuelve una lista de restricciones de inclusión de roles.
RemoveInclussionRole(Guid targetId, Guid incroleId):bool	Devuelve verdadero si se elimina una restricción de inclusión de roles, falso en caso contrario
Asociaciones	*Restriction, RestrictionController*

Anexo V: Clases persistentes

Descripción de la clase "*Role*"

Nombre de la Clase:	*Role*
Tipo de Clase:	Entidad
Atributo	**Tipo**
id	*Guid*
name	*string*
description	*string*
enabled	*bool*
Responsabilidad	
Nombre	**Descripción**
Role ()	Constructor

Descripción de la clase "*Restrictions*"

Nombre de la Clase:	*Restrictions*
Tipo de Clase:	Entidad
Atributo	**Tipo**
id	*Guid*
name	*string*
description	*string*
xml	*string*
Responsabilidad	
Nombre	**Descripción**
Restrictions ()	Constructor

Descripción de la clase "*Resource*"

Nombre de la Clase:	*Resource*
Tipo de Clase:	Entidad
Atributo	**Tipo**
id	*Guid*
name	*string*
description	*string*
enabled	*bool*
resourceType	*ResourceType*
Responsabilidad	
Nombre	**Descripción**
Resource ()	Constructor

Anexo VI: Pruebas

Descripción de pruebas funcionales

Descripción de la funcionalidad Gestionar roles: En este escenario hay que comprobar la creación de un rol en la base de datos. Si los datos del rol son correctos se debe crear satisfactoriamente. Si existen errores mostrar mensajes indicando los mismos. Se debe además comprobar la modificación de un rol previamente seleccionado. Si los datos modificados son correctos, la modificación fue satisfactoria, en caso contrario mostrar mensajes indicando los errores que existen. También se comprueba la eliminación de un rol y los efectos que puede producir esta acción en el sistema.

Condiciones de ejecución:

- ✓ El administrador de roles debe estar previamente autenticado y autorizado.
- ✓ Debe estar disponible la base de datos que utiliza el sistema de gestión.

Caso de prueba escenario: Gestionar roles

Escenario	Descripción	Texto	Respuesta del sistema	Flujo central
Crear rol	Permite crear un rol en el sistema.	Nombre, descripción (Válido)	Se crea el rol correctamente.	Seleccionar el menú ”Gestión de roles” y oprimir la opción ¨Crear rol¨. Se introducen los datos del nuevo rol y luego se pulsa aceptar.
		Nombre, descripción (Inválido)	Muestra un mensaje de aviso informando que se entraron datos incorrectos.	
Editar rol	Permite modificar los datos de los roles del sistema.	Nombre, descripción (Válido)	Se modifican los datos de los roles correctamente.	Seleccionar el menú ”Gestión de roles”. Seguidamente se oprime la opción ¨Editar rol¨ y entonces se muestra un listado de los roles existentes en el sistema, Se selecciona el que se va a modificar y se introducen los nuevos datos. Finalmente se pulsa aceptar.
		Nombre, descripción (Inválido)	Muestra un mensaje de aviso informando que se entraron datos incorrectos.	
Eliminar rol	Permite eliminar un rol del sistema		Se elimina el rol correctamente.	Seleccionar el menú ”Gestión de roles”. Luego el sistema brinda la opción de ¨Eliminar¨ rol. Una vez que seleccione esta opción entonces el sistema muestra un listado de los roles existentes en el sistema. Se selecciona
			Muestra un mensaje de error informando que no se pudo eliminar el rol.	

				el rol, se oprime el botón “Eliminar” y el sistema muestra una ventana con un mensaje de aviso. Finalmente se elimina el rol del sistema.

Descripción de la funcionalidad Gestionar recursos: En este escenario hay que comprobar la creación de un recurso en la base de datos. Si los datos del recurso son correctos se debe crear satisfactoriamente. Si existen errores mostrar mensajes indicando los mismos. Se debe además comprobar la modificación de un recurso previamente seleccionado. Si los datos modificados son correctos, la modificación fue satisfactoria, en caso contrario mostrar mensajes indicando los errores que existen. También se comprueba la eliminación de un recurso y los efectos que puede producir esta acción en el sistema.

Condiciones de ejecución:

- ✓ El administrador de roles debe estar previamente autenticado y autorizado.
- ✓ Debe estar disponible la base de datos que utiliza el sistema de gestión.

Caso de prueba escenario: Gestionar recursos

Escenario	Descripción	Texto	Respuesta del Sistema	Flujo Central
Crear recurso	Permite crear un recurso en el sistema.	Nombre, descripción y tipo de recurso (Válido).	Se crea el recurso correctamente.	El sistema muestra una estructura jerárquica de los recursos existentes, y permite seleccionar un recurso dentro del cual se debe poder crear uno nuevo. Se oprime clic derecho sobre el recurso que se desea crear uno nuevo. Se muestra la opción “Crear recurso”. Se introducen los datos
		Nombre, descripción y tipo de recurso (Inválido).	Muestra un mensaje de aviso informando que se entraron datos incorrectos.	

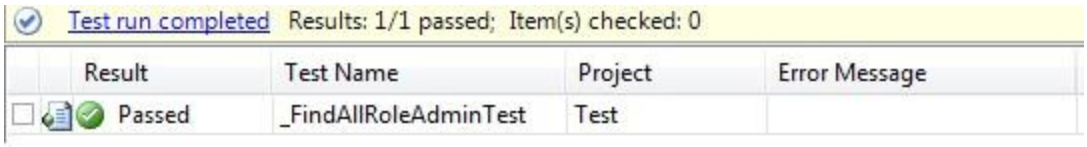
				del nuevo recurso y luego se pulsa aceptar.
Editar recurso	Permite modificar los datos de los recursos del sistema.	Nombre, descripción y tipo de recurso (Válido).	Se modifican los datos de los recursos correctamente.	Se selecciona el recurso que desea editar de la estructura jerárquica que muestra los recursos. Se oprime clic derecho y se selecciona la opción “Editar recurso”. Se selecciona el que se va a modificar y se introducen los nuevos datos. Finalmente se pulsa aceptar.
		Nombre, descripción y tipo de recurso (Inválido).	Muestra un mensaje de aviso informando que se entraron datos incorrectos.	
Eliminar recurso	Permite eliminar un recurso del sistema		Se elimina el recurso correctamente.	Se selecciona el recurso de la jerarquía que muestra los recursos. Se oprime clic derecho sobre el recurso a eliminar y el sistema brinda la opción de ¨Eliminar recurso”. Se oprime el botón “Eliminar” y el sistema muestra una ventana con un mensaje de aviso. Finalmente se elimina el recurso del sistema.
			Muestra un mensaje de error informando que no se pudo eliminar el recurso.	

Descripción de pruebas unitarias

```
/// <summary>
///A test for _FindAllRoleAdmin
///</summary>
[TestMethod()]
public void _FindAllRoleAdminTest()
{
    DALRoleManager target = new DALRoleManager(); // TODO: Initialize to an appropriate value
    List<RoleAdmin> expected = new List<RoleAdmin>(); // TODO: Initialize to an appropriate value
    List<RoleAdmin> actual;
    actual = target._FindAllRoleAdmin();
    Assert.AreEqual(7, actual.Count);
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Prueba al método *"FindAllRoleAdmin"*

Resultado de prueba al método *"FindAllRoleAdmin"*

Prueba de unidad		
Nombre de prueba: *FindAllRoleAdminTest*		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima ejecución:** 7/06/2011
Ejecutado por: Yaimi Manso Machado	**Verificado por:** Maikel de la Torre Luis	
Descripción: Devuelve una lista con todos los roles administrativos existentes, en caso de no haber ninguno devuelve una lista vacía.		
Entrada:		
Criterio de aceptación: Muestra una lista con todos los roles administrativos existentes.		
Resultado: 		

```
/// <summary>
///A test for FindScopeByName
///</summary>
[TestMethod()]
public void FindScopeByNameTest()
{
    DALScopeManager target = new DALScopeManager(); // TODO: Initialize to an appropriate value
    string name = "Guantanamo"; // TODO: Initialize to an appropriate value
    Scope expected = new Scope() { Name="Guantanamo" }; // TODO: Initialize to an appropriate value
    Scope actual;
    actual = target.FindScopeByName(name);
    Assert.AreEqual(expected.Name, actual.Name);
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Prueba al método *"FindScopeByName"*

Resultado de prueba al método *"FindScopeByName"*

Prueba de unidad		
Nombre de prueba: *FindScopeByNameTest*		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Ultima ejecución:** 7/06/2011
Ejecutado por: Yaimi Manso Machado	**Verificado por:** Maikel de la Torre Luis	
Descripción: Para poder ejecutar la prueba se debe pasar un tipo de dato *string* correspondiente al nombre del ámbito, en caso de que exista un ámbito con este nombre, el método debe buscarlo y retornarlo, en caso contrario devuelve vacío.		
Entrada: *string id*		
Criterio de aceptación: Muestra un objeto de tipo ámbito, con el *id* del ámbito, el *nombre* del ámbito y la *descripción*.		
Resultado: