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 del RBAC en el Sistema de Administración de Identidades"

Autor(es): Evaristo José Madarro Capó

Osniel Griñán Rodríguez

Tutora: Ruth Yurina Vega Cutiño

Co-tutor: Maikel De La Torre

Ciudad de La Habana, junio 25 de 2010.
Año 52 de la Revolución

"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber…"

Albert Einstein…

A mis padres por el apoyo que me han dado durante todos estos años, de no ser por ustedes hoy no estaría aquí.
A mi hermana Celia por confiar en mí y apoyarme en los momentos buenos y malos.
A toda mi familia que ha sabido apoyarme y ha confiado siempre en mí.
A mis amigos todos, a los de la UCI por compartir conmigo todos esos momentos de alegría y tristeza de 5 años y a los otros que se han preocupado por mí y me han sabido dar apoyo.
A todos aquellos que han creído en mí.

Evaristo José Madarro Capó

A mis padres por guiarme en la vida, apoyarme, y ser ejemplo de consagración y sacrificio.
A mi hermano por aconsejarme y ser el mejor de todos los amigos.
A mi novia, que siempre me ha brindado su apoyo incondicional cuando lo he necesitado.
A mi cuñada, porque es una persona excelente.
A mi familia, por lo especial que es.
A todos los amigos que en el transcurso de los 5 años me facilitaron llevar a vías de hechos esta difícil empresa.

Osniel Griñán Rodríguez

Después de 5 años de grandes sacrificios y trabajo, hoy por fin se hace realidad nuestro sueño y se cumple nuestra gran primera meta en la vida, por lo que en este momento están más presentes que nunca las personas que nos han apoyado y por tanto también se hace suyo nuestro logro.

Agradecemos a nuestros padres, por apoyarnos en todo momento, ese apoyo moral, valió más que cualquier riqueza.

A nuestros hermanos y familiares en general.

A los amigos por todos esos momentos tan emocionantes que hemos vivido.

A todos los profesores que han contribuido en nuestra formación como profesionales.

Al colectivo del proyecto, Ruth, Maikel, Roberkis, Roberto, Felix, Diovis, Yoandy, Ernesto, Armando, Vivi, Carlos y Mario, ya que, todos pusieron su granito de arena en el desarrollo de la aplicación. A todos, Gracias.

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 firmo la presente a los ____ días del mes de _____ del año _____.

Evaristo José Madarro Capó	Ruth Yurina Vega Cutiño
Osniel Griñán Rodríguez	Maikel de la Torre Luis

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.

RESUMEN

Este trabajo propone el desarrollo de una aplicación web para la gestión de la configuración del control de acceso basado en roles (*Role Based Access Control*, RBAC) en la Universidad de las Ciencias Informáticas (UCI), específicamente en la línea de seguridad del Centro de Identificación y Seguridad Digital (CISED), donde se está desarrollando un Sistema de Administración de Identidades. Este sistema posee un componente de autorización, basado en el modelo RBAC, el cual se apoya para su funcionamiento en un repositorio, donde se encuentran almacenados los principales elementos del modelo. Actualmente para poder ofrecer este servicio de autorización se debe llevar un control estricto de cada cambio efectuado en el repositorio, labor que se hace engorrosa pues este proceso se desarrolla de forma manual.

Con el desarrollo de esta investigación se propone situar en la línea de seguridad del Centro un sistema que administre todo el proceso de configuración de los elementos del RBAC. Este documento recoge todo el trabajo realizado para la elaboración del sistema antes mencionado, incluyendo el estado del arte de aplicaciones con similares objetivos existentes en el mundo, el estudio y definición de las características del sistema, así como la planificación, desarrollo y pruebas del producto.

PALABRAS CLAVE

Control de acceso, autorización, rol, recurso, RBAC

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Controlar el acceso a los recursos es uno de los factores a tener en cuenta para garantizar la seguridad de un sistema. Un control por el que se determina qué usuarios pueden acceder a determinados recursos, además del dónde, cómo y cuándo. Proteger la información de posibles accesos no deseados ha motivado, a múltiples instituciones desarrolladoras de *software*, a generar soluciones y tecnologías que automaticen la gestión del control de acceso.

Desde los inicios del desarrollo de *software* ha sido un requisito la construcción de modelos de seguridad y acceso de diversas maneras, en los cuales los usuarios que intervienen en los procesos son inicialmente identificados y luego autorizados a su interacción con los mismos.

La tecnología ha evolucionado con los propios sistemas de información protegidos. Actualmente existen diversos modelos para el control de acceso que pretenden solucionar los problemas existentes en este contexto, uno de los más comunes es el control de acceso basado en roles (RBAC). Este modelo es un enfoque para implementar políticas de control de acceso basado en el concepto de rol y es considerado uno de los modelos más generales debido a su neutralidad respecto a la política de control de acceso y a su flexibilidad, lo que le permite poder simular enfoques alternativos como control de acceso obligatorio (MAC) y control de acceso discrecional (DAC). RBAC ha demostrado su efectividad en una gran variedad de ámbitos, desde organizaciones gubernamentales hasta complejas aplicaciones web.

El desarrollo informático en Cuba se ha incrementado gradualmente y con ello la necesidad de brindar una mayor seguridad a la información. En el país se ha estado al tanto de cómo en otros países desarrolladores de *software*, los temas de acceso y seguridad han sido diseñados e implementados. Sin embargo, existe una tendencia a establecer modelos de seguridad y acceso no basados en estándares ya bien definidos.

Actualmente en la Universidad de las Ciencias Informáticas (UCI), específicamente en la línea de seguridad del Centro de Identificación y Seguridad Digital (CISED), se está desarrollando un Sistema de Administración de Identidades que permitirá mantener un control adecuado sobre los recursos de las aplicaciones para evitar que personal no autorizado acceda a los mismos y realicen operaciones que

conlleven a la pérdida de información. Este sistema cuenta con un componente de autorización, basado en el modelo RBAC, a través del cual verifica los permisos de acceso de los usuarios sobre los recursos.

Para facilitar y agilizar el uso de este componente y con el objetivo de disminuir la ocurrencia de errores, tributando así al correcto funcionamiento del mismo, es necesario gestionar la configuración de los principales elementos de dicho modelo, tales como roles, recursos, operaciones, etc., en correspondencia con el estándar del modelo unificado que define el Instituto Nacional de Estándares y Tecnología (*National Institute of Standards and Technology,* NIST).

Teniendo en cuenta las **problemáticas** existentes dentro de la línea de seguridad del CISED en cuanto al control del acceso a información restringida**,** se ha determinado como **problema científico**: ¿Cómo gestionar la configuración del control de acceso basado en roles?

Para esto el **objeto de estudio** lo constituye: Modelos de control de acceso.

Derivándose que el **campo de acción** que abarca esta investigación sería: Gestión de la configuración del control de acceso basado en roles.

Para dar respuesta al problema planteado se definió el siguiente **objetivo**: Desarrollar una aplicación para la gestión de la configuración del RBAC en la línea de seguridad del CISED.

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

- ✓ Analizar soluciones existentes.
- ✓ Identificar las tecnologías y estándares relacionados con el tema propuesto.
- ✓ Realizar diseño e implementación del sistema a desarrollar.
- ✓ Realizar pruebas de calidad y buen funcionamiento a la solución propuesta.

Esta investigación tiene la siguiente **idea a defender**: El desarrollo de una aplicación para la gestión de la configuración del control de acceso basado en roles proveerá al Sistema de Administración de Identidades una interfaz para la administración de los principales componentes del RBAC.

Para garantizar el desarrollo de los objetivos se proponen las siguientes **tareas de investigación**:

- Análisis del modelo de control de acceso RBAC.
- Revisión de las soluciones más utilizadas para la administración de los elementos que componen el modelo RBAC para un mayor conocimiento del problema.
- Especificación de las tecnologías para la implementación del *software*.
- Especificación de los escenarios y los requisitos de calidad de servicio del *software* a desarrollar para definir los objetivos a seguir.
- Modelado del sistema para trazar una línea base que sirva como guía en todo el desarrollo del *software*.
- Realización del diseño del *software* a desarrollar para su posterior implementación.
- Implementación de la propuesta de solución con el objetivo de dar cumplimiento a los escenarios definidos.
- Ejecución de pruebas al funcionamiento del *software* concebido para comprobar su calidad.

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

Métodos Teóricos:

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

- Histórico – Lógico: Para constatar teóricamente cómo ha evolucionado el desarrollo de la administración del RBAC a través de los años.

Métodos Empíricos:

- Entrevista: Posibilitará obtener información referente a las diferentes vías para el diseño e implementación del *software*.

Este trabajo está conformado por 3 capítulos, a continuación se describe el contenido de los mismos:

Capítulo I: En este capítulo se hace referencia a los principales conceptos relacionados con la administración del modelo de control de acceso RBAC, se realiza un estudio del estado del arte y se analizan las tecnologías a utilizar para el desarrollo del *software*.

Capítulo II: En este capítulo se describe la propuesta de solución incluyendo sus principales características, partiendo del análisis previo de la problemática y el objeto de estudio en cuestión. Además del levantamiento y descripción de los escenarios y los requerimientos de calidad de servicio que debe cumplir la aplicación.

Capítulo III: En este capítulo se modela el diagrama de clases y se diseña el modelo de base de datos, se abordan conceptos de gran importancia para un mejor entendimiento del tema, se presenta la interacción entre los diferentes módulos de la aplicación y se modela el diagrama de centro de datos lógicos y el diagrama de aplicación. Además, se ejecuta el plan de pruebas y se realiza un resumen de evaluar las mismas.

Tabla 1: Cronograma de trabajo para el desarrollo de la propuesta de solución

No	Actividades	Fecha	Fecha	Responsables
1	Análisis del modelo de control de acceso RBAC	15/10/09	30/10/09	Osniel Griñán Evaristo J. Madarro
2	Revisión de las soluciones más utilizadas para la administración de los elementos que componen el modelo RBAC para un mayor conocimiento del problema.	1/11/09	30/11/09	Osniel Griñán
3	Especificación de las tecnologías para la implementación del *software*	1/12/09	15/12/09	Evaristo J. Madarro
4	Especificación de los escenarios y los requisitos de calidad de servicio del *software* a desarrollar para definir los objetivos a seguir.	5/01/10	30/01/10	Osniel Griñán Evaristo J. Madarro
5	Modelado del sistema para trazar una línea base que sirva como guía en todo el desarrollo del *software*.	1/02/10	2/03/10	Osniel Griñán Evaristo J. Madarro
6	Realización del diseño del *software* a desarrollar para su posterior implementación	5/03/10	25/03/10	Osniel Griñán Evaristo J. Madarro

7	Implementación de la propuesta de solución con el objetivo de dar cumplimiento a los escenarios definidos	30/03/10	1/05/10	Osniel Griñán Evaristo J. Madarro
8	Ejecución de pruebas al funcionamiento del *software* concebido para comprobar su calidad.	30/03/10	15/05/10	Osniel Griñán Evaristo J. Madarro

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

Las organizaciones y las empresas no solo están evolucionando hacia el desarrollo de sistemas óptimos y confiables, sino también a modelos donde la seguridad sea una característica fundamental para las tecnologías de la información (IT); por esta razón se hace necesario contar con herramientas que permitan la correcta ejecución de proyectos y planes de seguridad.

La gestión de identidades es apoyada por una amplia gama de difusión y tecnologías de protección que, al implementarse correctamente, permite que las organizaciones puedan operar de manera eficiente en el actual entorno competitivo. La gestión eficiente de la identidad y las cuestiones de control de acceso asociadas, deben ser motivo de especial preocupación para la seguridad de todos los organismos, con independencia de su tamaño o enfoque de negocios.

1.2. Gestión de identidades

1.2.1. Definición de gestión de identidades

La esencia de la gestión de identidades, como solución, es proporcionar una combinación de procesos y tecnologías para administrar y garantizar el acceso a la información y los recursos de una organización, al tiempo que protege los perfiles de los usuarios. La gestión de identidades garantiza que la identidad se derive de una fuente fidedigna y que la identidad es supervisada y controlada, es decir, impide que otros manipulen esa identidad y continuamente valida la autenticidad de la misma. Además, permite que la identidad sea compartida de manera eficaz, garantizando que la información se proporciona en forma oportuna y precisa, mientras que se protege la privacidad de la misma. Las soluciones de gestión de identidades deben proporcionar un mecanismo simple para entender el crecimiento y la complejidad, garantizar una configuración coherente de todos los sistemas cuando los usuarios se agregan, eliminan o modifican de alguna manera, así como la autenticación, autorización y control de acceso independiente. La gestión de identidades se refiere a la administración eficaz de una identidad definitiva para un usuario y

garantizar que los usuarios tengan acceso rápido y fiable a la información y a las aplicaciones de una organización. (1)

1.2.2. Principales sistemas de gestión de identidades existentes en el mundo

- ***Oracle Identity Management***

Oracle Identity Management ofrece métodos de autenticación muy sólidos, autorización basada en riesgos, otorgamiento de derechos específicos, abastecimiento de usuarios basado en roles y relaciones. Este sistema agrupa las tecnologías de administración en tres áreas funcionales:

- **Servicios de Directorio**

Oracle Internet Directory

Oracle Internet Directory es un directorio de identidades (*Lightweight Directory Access Protocol*, LDAP) que aprovecha las características de escalabilidad, alta disponibilidad y seguridad de la base de datos de Oracle. *Oracle Internet Directory* puede utilizarse como repositorio central de usuarios para las implementaciones de *Oracle Identity Management*, o puede servir como directorio basado en estándares, altamente escalable para cualquier organización. (2)

- **Gestión de identidad**

Oracle Identity Manager

Es un sistema de gestión de identidades empresariales altamente flexible y escalable que controla centralmente el ciclo de vida de las cuentas de los usuarios y los privilegios de acceso dentro de los recursos empresariales. *Oracle Identity Manager* ofrece una integración lista para usar con las tecnologías de infraestructura y aplicaciones empresariales comúnmente implementadas. (2)

Oracle Role Manager

Es una solución para la administración de roles construida sobre una arquitectura J2EE. *Oracle Role Manager* ofrece un grupo de características completas para la administración del ciclo de vida de los roles empresariales y la administración de sus relaciones. (2)

- **Gestión de Acceso**

Oracle Access Manager

Ofrece un control de acceso escalable para los entornos heterogéneos con una solución integrada, basada en estándares para la autenticación, el inicio de sesión web única y la creación y cumplimiento de las políticas. *Oracle Access Manager* soporta todos los servidores web, servidores de aplicaciones y servidores de directorios más importantes. (2)

Oracle Entitlements Server

Oracle Entitlements Server ofrece una administración centralizada de políticas basadas en estándares y cumplimiento de políticas distribuidas en todas las aplicaciones empresariales dando como resultado un entorno empresarial más seguro, mejor facilidad de administración y el cumplimiento de políticas consistentes. (2)

- ***Tivoli Identity Manager***

Tivoli Identity Manager proporciona una gestión de identidades basada en políticas en entornos heredados y de *e-business*. Las intuitivas interfaces web, administrativas y de autoservicio, se integran con los procesos empresariales ya existentes para ayudar a simplificar y automatizar la gestión y el suministro de usuarios. Incorpora un motor de flujo de trabajo y utiliza los datos de identidad para distintas actividades, como auditorías y generación de informes. (3)

- ***Tivoli Access Manager***

Tivoli Access Manager para *e-business* es una solución versátil para la gestión de problemas de autenticación y autorización. Centrado principalmente en las aplicaciones web, las implementaciones de *Access Manager* abarcan desde el inicio único de sesión (*Single Sign-On*, SSO) hasta despliegues de infraestructura de seguridad más compleja. Con esta herramienta es posible agrupar a los usuarios y asignar permisos a los grupos, con lo que se simplifica la administración del control de acceso cuando existen muchas aplicaciones y recursos. Se admiten reglas dinámicas y decisiones de autorización en función de datos externos para las aplicaciones que lo necesiten. (3)

1.2.3. Conceptos relacionados con los sistemas de gestión de identidades

A través de la investigación de los sistemas actuales para la gestión de identidades se pudo observar que aunque se implementan en dependencia de las características de su entorno, estos sistemas de gestión de identidades engloban cuatro conceptos fundamentales:

- **Autenticación (demostración de quién es el usuario)**

La autenticación es el proceso básico de la validación de que alguien, o alguna entidad, es quien dice ser. Este proceso se divide en varios métodos de desafío y respuesta:

- ✓ Algo que usted sabe: los nombres de cuenta, el número de ID del cliente, contraseña, PIN.
- ✓ Algo que tienes: tarjeta de crédito, licencia de conducir, pasaporte.
- ✓ Algo que eres: huellas dactilares, la retina, el ADN.
- ✓ Algo que puede hacer: la firma.

Ese proceso puede tomar muchas formas, e incluso puede utilizar combinaciones de estos métodos. La solución de autenticación más común en los sistemas informáticos de hoy es el nombre de cuenta y contraseña. La autenticación puede ser necesaria una o varias veces dependiendo de cómo está integrado el sistema o los sistemas. (5)

- **Autorización (determinación de los derechos de acceso y privilegios de usuario)**

La autorización es el proceso de determinar si una cuenta identificada y verificada tiene los permisos para el acceso a los recursos. La autorización es generalmente una comprobación básica de si la cuenta está activa y en buen estado, y se basa en puntos de datos específicos dentro de un sistema individual.

La autorización sucede después de la autenticación y usa atributos o derechos, asociados con la identidad digital para determinar a qué recursos puede acceder dicha identidad digital. Se basa en políticas de control de acceso (reglas para especificar quién puede acceder, a que recursos), modelos (formalismos para describir las políticas) y mecanismos (verificar la solicitud de acceso de un usuario para conceder o denegar el acceso). (5)

- **Control de acceso (gestión de los medios de acceso)**

El control de acceso es un conjunto más amplio de políticas dentro de un sistema de gestión de identidad que definen las normas en todo lo que un usuario se le permite hacer en el ámbito de aplicación de dicho sistema. El administrador de identidades permite que estas políticas se definan en un alto nivel independiente de los sistemas específicos, por lo que se pueden aplicar a cualquier sistema, ya que, no dependen del mismo. (5)

- **Auditoría (registro y control de eventos y acciones)**

La auditoría constituye un conjunto de procedimientos y técnicas para evaluar y controlar total o parcialmente un sistema informático con el fin de proteger sus activos y recursos, verificar si sus actividades se desarrollan eficientemente de acuerdo con las normas informáticas y generales existentes en cada empresa y para conseguir la eficacia exigida en el marco de la organización correspondiente. (5)

1.3. Modelos de control de acceso

Desde los inicios del desarrollo de *software* ha sido un requisito la construcción de modelos de seguridad y acceso de diversas maneras sobre los cuales los usuarios que intervienen en los procesos son inicialmente identificados y luego autorizados a su interacción con los mismos. Entre los modelos más comunes se encuentran:

❖ **Modelo de control de acceso discrecional**

El modelo de control de acceso discrecional (*Discretionary Access Control,* DAC), también llamado modelo de seguridad limitada, es un modelo no orientado al control del flujo de información. Todos los sujetos y objetos en el sistema son controlados y se especifican reglas de autorización de acceso para cada sujeto y objeto. Los sujetos pueden ser usuarios, grupos o procesos. Los modelos DAC están basados en la idea de que el propietario de un objeto, su autor, tiene el control sobre los permisos del objeto. Es decir, el propietario del objeto es autorizado a permitir u otorgar permisos para este objeto a otros usuarios.

La principal ventaja de DAC es que el usuario se beneficia de la flexibilidad del modelo. Sin embargo, es difícil para DAC garantizar las reglas de integridad como menor privilegio (*least privilege)* o separación de

responsabilidades (*separation of duty)* que son necesarias para garantizar una seguridad de alto nivel. DAC es apropiado en ambientes donde la compartición de información es más importante que su protección. (6)

- **Modelo de control de acceso obligatorio**

En el modelo de control de acceso obligatorio (*Mandatory Access Control,* MAC) todos los sujetos y objetos son clasificados basándose en niveles predefinidos de seguridad que son usados en el proceso de obtención de los permisos de acceso. Para describir estos niveles de seguridad todos los sujetos y objetos son marcados con etiquetas de seguridad que siguen el modelo de clasificación de la información militar (desde "desclasificado" hasta "alto secreto"), formando lo que se conoce como política de seguridad multinivel. A diferencia de DAC, los modelos MAC proporcionan mecanismos más sólidos para la protección de datos, y tratan con requerimientos de seguridad más específicos, así como, los requerimientos derivados de las políticas de control de los flujos de información. Además, en los modelos MAC es el sistema quien protege los recursos u objetos, el administrador es el que impone las reglas de forma segura, a diferencia del DAC en el cual el dueño es quién protege los recursos. Sin embargo, asegurar las políticas de MAC es a menudo una tarea difícil, ya que, no proporcionan soluciones factibles dado que les falta suficiente flexibilidad. (6)

- **Modelo de control de acceso basado en tareas**

El control de acceso basado en tareas (*Task Based Access Control,* TBAC) permite controlar el acceso en entornos representados por flujos de trabajo *(workflow)*. El modelo TBAC extiende los tradicionales modelos de control de acceso basados en sujetos/objetos incluyendo aspectos que aportan información contextual basada en las actividades o tareas.

El control de acceso en TBAC es garantizado por medio de "Etapas de autorización". Las "Etapas de autorización" son un concepto abstracto introducido por TBAC para modelar y manejar un sistema de permisos relacionados con el progreso de las tareas o actividades dentro del contexto de un flujo de trabajo (*workflow)*. (6)

❖ **Modelo de Control de Acceso Basado en Rol**

RBAC está basado en la definición de un conjunto de elementos y de relaciones entre ellos. 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. El modelo RBAC incluye un conjunto de sesiones donde cada sesión es 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. 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.

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. (6)

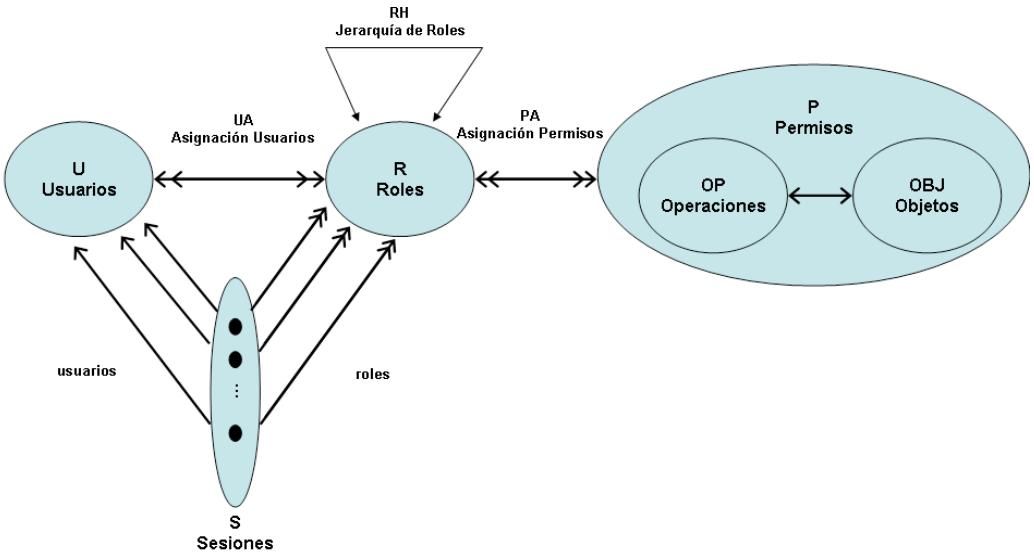


Figura 1: Modelo de control de acceso RBAC de la NIST

1.3.1. Modelo seleccionado: RBAC

Uno de los problemas más comunes en la gestión de grandes sistemas de información es la complejidad de la administración de seguridad. Los modelos DAC y MAC son inadecuados para cubrir las necesidades de la mayor parte de las organizaciones. El modelo DAC es demasiado débil para controlar el acceso a los recursos de información de forma efectiva, mientras el modelo MAC es demasiado rígido. El modelo TBAC presenta varias carencias cuando es aplicado, entre estas encontramos, que reconoce la necesidad de la inclusión de la información de contexto para realizar el control de acceso, pero centrándose solo en la información contextual referente a las tareas, así como al proceso del flujo de trabajo (*workflow)* y al uso o validez de los permisos, siendo necesario considerar una información de contexto más amplia. Además, la especificación de políticas de seguridad son complejas y la gestión, delegación y revocación de los privilegios son muy primitivas.

Desde su surgimiento, el modelo RBAC, se ha propuesto como un intento de unificar los modelos clásicos DAC y MAC, consiguiendo un sistema el cual impone el control de acceso, pero sin las restricciones rígidas impuestas por las etiquetas de seguridad. El control de acceso basado en roles permite expresar de forma sencilla y natural la política de acceso a los recursos de una organización compleja, 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). Permiten asimismo la construcción jerárquica de estas políticas de acceso, por herencia o especialización. A partir de lo anterior, se pudo definir que la tecnología RBAC es la más apta para el desarrollo de esta aplicación pues tiene el potencial de reducir la complejidad y el coste de la administración en un entorno de seguridad.

1.4. Gestión del RBAC

Un sistema de gestión del RBAC debe soportar y forzar el cumplimiento de las políticas y reglas que gobiernan el acceso a los recursos. Además, debe mantener las políticas de acceso de forma centralizada, evitando así la dispersión y la falta de control sobre las configuraciones. Las políticas de acceso deben mantener un conjunto de reglas sobre quién tiene acceso a qué recurso sobre la base de su rol.

1.4.1. Análisis de soluciones existentes para la gestión del RBAC

Como parte de la investigación se realiza un análisis de los sistemas informáticos existentes, tanto en el país como en el resto del mundo, que pudieran tributar a la solución del problema planteado.

1.4.1.1. Soluciones existentes en el mundo

Dentro de la comunidad del *software* libre 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, tal es el caso del entorno de escritorio multiplataforma GNOME, el cual a través de la herramienta *GNOME Display Manager*, incluida recientemente, controla el acceso a las características de apagado, reinicio y suspensión de las sesiones. Pero como productos dedicados a la gestión del modelo RBAC no se han obtenido resultados relevantes que puedan tributar a la investigación actual.

- ***Solaris Management Console***

La consola de administración de *Solaris* constituye una parte clave de la implementación de RBAC en el ambiente *Solaris*. Las herramientas proporcionadas por la consola de administración de *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. (7)

❖ ***InterChange IBM WebSphere Application Server***

Una de las características claves de esta herramienta es la capacidad de autorizar permisos a los usuarios para acceder al sistema mediante roles asignados a los mismos, a través de la implementación del conocido control de acceso basado en roles (RBAC). Los roles pueden ser fácilmente definidos por el administrador y asignados a un grupo de usuarios, restringiendo el acceso a componentes clave sólo a los usuarios no verificados. Se pueden asignar roles a lo largo de las asociaciones funcionales y reducir considerablemente la carga administrativa. La asignación de un rol a un usuario le permite el acceso sólo a los componentes del sistema incluidos en la definición del rol. (3)

❖ ***Oracle Entitlements Management Tool***

Esta herramienta de gestión puede ser utilizada por los usuarios de negocio para gestionar los roles y permisos. Esta interfaz de usuario permite gestionar los permisos de los usuarios a través del conocido modelo de control de acceso RBAC. Además, esta herramienta 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. (2)

❖ **Consola de administración de Visual Guard .NET**

Constituye una solución dirigida a cubrir los temas de seguridad de las aplicaciones .NET, no importando lo complejo de su entorno.
Visual Guard es un sistema de control de acceso basado en roles (RBAC), diseñado para el personal no técnico. Los administradores pueden añadir usuarios, eliminarlos, identificar permisos y gestionar roles desde una consola, usando la interfaz de administración de usuarios basada en la web.

Esta consola permite a los administradores gestionar el control de acceso sin necesidad de contar con una conexión directa con la base de datos o con la red local. Lo único que necesitan es tener una conexión a internet. Esto permitirá a *Visual Guard* soportar arquitecturas complejas (por ejemplo: diferentes distribuciones geográficas). Con esta consola los administradores podrán crear cuentas Usuario/Contraseña o cuentas declaradas desde el *Active Directory* sin necesidad de ayuda técnica. (8)

1.4.1.2. Soluciones existentes en Cuba

En Cuba el desarrollo informático se ha ido incrementado gradualmente, pues existe conciencia del valor que implica elevar la productividad, así como también de las necesidades de nuevos sistemas que cumplan requisitos específicos en procesos empresariales determinados, para un mejor desempeño de las entidades. Se han desarrollado múltiples intentos por garantizar la seguridad en los sistemas, pero existe una tendencia a establecer modelos de seguridad y acceso no basados en estándares ya bien definidos y en el peor de los casos ni siquiera se han implementado los mismos, por lo que se debe tener en cuenta la forma de integrar estos modelos, o un único modelo, a las aplicaciones que se desarrollan en un marco empresarial.

Uno de los compromisos que tiene la UCI con la Revolución es desarrollar sistemas para uso nacional e internacional. La nueva estrategia trazada por la universidad para alcanzar resultados relevantes en la producción ha traído consigo que los proyectos sean agrupados por centros o áreas de desarrollo. En algunos de estos centros se han intentado implementar soluciones para controlar el acceso a sus recursos a través del modelo RBAC, ejemplo de esto es el proyecto ERP-Cuba, el cual se encuentra enmarcado en el desarrollo de un componente de autorización basado en este modelo.

1.5. Principales tecnologías a utilizar

Con el objetivo de lograr un producto, no solo que solucione el problema existente, sino que además tenga la calidad requerida y cumpla con los requisitos establecidos por el cliente se realiza una investigación sobre la arquitectura, herramientas, metodologías y lenguajes a utilizar en la construcción de la solución.

Para el desarrollo de la solución se utilizarán las tecnologías definidas en la línea de la arquitectura del Sistema de Administración de Identidades, debido a que posteriormente esta solución será integrada al propio sistema, no obstante, se realiza una investigación para constatar las principales características de

estas herramientas. A continuación, se mencionan las tecnologías seleccionadas para la confección del *software*.

1.5.1. Metodología de desarrollo

Las metodologías se definen por pasos a seguir para lograr el cumplimiento de un objetivo, el cual lo constituye dentro del desarrollo del *software*, alcanzar un producto de alta calidad que cumpla con los requerimientos del cliente.

Las metodologías ágiles, proponen mejorar la calidad del *software* teniendo como premisa la comunicación inmediata y directa, mientras que las metodologías pesadas obtienen sus resultados a través del orden y la documentación.

1.5.1.1. Metodologías ágiles

Las metodologías ágiles están revolucionando la manera de producir *software*, y a la vez generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales. Aún son muchas las personas y organizaciones que desconocen los beneficios que traen consigo las prácticas ágiles de desarrollo de *software* en la actualidad. Sin embargo, esa realidad está cambiando a medida que aumentan los casos de éxito de proyectos que implementan dichos enfoques ágiles, demostrando con hechos lo que hace unos años era solo una moda.

- **MSF Agile**

MSF Agile (*Microsoft Solution Framework Agile*) es la nueva propuesta de Microsoft en el mundo de procesos y prácticas ágiles de desarrollo de *software*. *"MSF Agile*" renueva al ya exitoso MSF V 3.0, que es un marco de trabajo en cascada y espiral, implementando las mejores prácticas del mundo de desarrollo ágil de *software*. *MSF Agile* tiene como principales características el ser altamente insistente, de planificación adaptable a los cambios y enfocado a las personas.

El proceso definido por *MSF Agile* incorpora ideas clave del movimiento de *software* ágil, junto con los principios y prácticas de MSF. El proceso admite una estrategia ágil de ingeniería de *software* que utiliza múltiples iteraciones y un enfoque basado en escenarios para la construcción de aplicaciones. *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.

MSF Agile incorpora directamente prácticas para el manejo de calidad del servicio como el rendimiento y la seguridad. También está basado en contexto, es decir, utiliza un enfoque basado en contexto para determinar cómo operar el proyecto. Este enfoque ayuda a crear un proceso de adaptación que supera las condiciones de contorno de la mayoría de los procesos de desarrollo ágil de *software*. Cada pieza del trabajo de un proyecto es un elemento de trabajo y un objetivo importante de todo proyecto consiste en realizar el seguimiento de todos estos elementos de trabajo hasta que hayan finalizado. *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. (9)

1.5.2. Lenguaje de 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 introduce nuevos diagramas que representa una visión dinámica del sistema. Es decir, gracias al diseño de la parte dinámica del sistema podemos darnos cuenta de problemas en la estructura al propagar errores o en las partes que necesitan ser sincronizadas, así como del estado de cada una de las

instancias en cada momento. Permite la modificación de todos sus miembros mediante estereotipos y restricciones. Un estereotipo nos permite indicar especificaciones del lenguaje al que se refiere el diagrama de UML.

UML 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. (10)

1.5.3.Herramienta de modelado

- ***Altova UModel 2009***

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 2009 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. Las características de *UModel 2009* para el desarrollo de *software* basado en las capacidades de modelado avanzado son:

- Soporte para los 14 tipos de diagramas UML
- Modelado de esquemas XML en diagramas UML
- Diagramas de proceso de negocio
- Generación de código fuente en lenguajes Java, C#, y VB.NET
- Ingeniería inversa de código fuente y ficheros binarios Java, C# y VB.NET
- Sincronizado del modelo y el código a través de ingeniería de ida y vuelta
- Crea diagramas de secuencia desde el código fuente de la ingeniería inversa

- Generación de documentación personalizable de proyecto
- Compartir subproyectos para colaboración o reutilización
- Capas de diagramas con visibilidad selectiva
- *Vínculos (Hyperlinks)* entre diagramas, documentos, o páginas web
- Integración con sistemas de control de versiones
- Estrecha integración con *Visual Studio* y *Eclipse*. (11)

1.5.4. Lenguajes de 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# podremos acceder a una librería de clases muy completa y muy bien diseñada, que nos permitirá 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)

❖ ***JavaScript***

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 *JavaScript* podemos crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones *JavaScript* 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 *JavaScript* tenemos 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, *JavaScript* nos permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo. *JavaScript* 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, *JavaScript* pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente. Con *JavaScript* 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. (14)

- **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.

Las hojas de estilo pueden utilizarse para:

- Lograr una apariencia uniforme de todo el sitio al activar una sola definición de estilo en cada página.
- Cambiar un aspecto en todo el sitio web con tan sólo editar unas pocas líneas.

- Facilitar la lectura de los códigos HTML, ya que, los estilos se definen por separado.
- Permitir que las páginas se carguen más rápido, ya que, hay menos cantidad de código HTML en cada página.
- Posicionar los elementos de la página de una manera más uniforme. (15)

1.5.5. Plataforma de desarrollo

❖ **Plataforma .NET**

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

La plataforma .NET constituye un entorno para la construcción, desarrollo y ejecución de servicios web y otras aplicaciones 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, 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:

- 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.
- Interoperabilidad multilenguaje: El código puede ser escrito en cualquier lenguaje compatible con .NET, ya que, siempre se compila en código intermedio.
- 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.
- 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.

- 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.
- 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. (16)

1.5.6. Herramienta de desarrollo

❖ ***Visual Studio Team System 2008 (VSTS)***

A continuación se resumen las características más importantes que proporciona VSTS 2008:

- Provee una serie de herramientas para desarrollo, así como características de debugging, funcionalidad en base de datos y características innovadoras para la creación de aplicaciones en una variedad de plataformas.
- Incluye realces, como un diseñador visual, para desarrollo rápido con el *Framework* 3.5, esto ayuda mucho a los que desarrollan en web porque se incluyen las características de *Microsoft Expression Web*. *Visual Studio Team System 2008* provee a desarrolladores con todas las herramientas y el *framework*, el poder crear aplicaciones web con el soporte de AJAX.
- Incluye un nuevo lenguaje de consultas integrado para el manejo de la información, el cual se denomina *Language Integrated Query* (LINQ), el cual permite construir soluciones que analicen y actúen sobre la información.
- Provee a los desarrolladores la habilidad de escoger entre múltiples versiones del *framework* con el mismo entorno de desarrollo, así podemos desarrollar en la versión que queramos ya sea en .NET *Framework* 2.0, 3.0 o 3.5.
- Ofrece a los desarrolladores nuevas herramientas para la fácil creación de aplicaciones, conectadas a las últimas plataformas incluyendo la web, Windows Vista, Office 2007, SQL Server 2008 y Windows Server 2008. Para la web, tenemos ASP.NET, AJAX y otras tecnologías como *Silverlight*, WPF, *Windows Workflow*, *Windows Cardspace*, *Windows Communication Foundation*,

etc., que nos dará la posibilidad de crear aplicaciones con una rica interfaz de usuario, para dar una experiencia al usuario sin precedentes. (17)

1.5.7. Gestor de base de datos

❖ **Oracle Database 11g**

Oracle Database 11g Enterprise Edition ofrece confiabilidad, escalabilidad y desempeño de primer nivel para configuraciones en *cluster* y en un solo servidor. Ofrece las más completas características para soportar el procesamiento de transacciones más exigente, inteligencia de negocios, y aplicaciones para la administración de contenido. Protección ante las fallas del servidor, fallas del sitio, errores humanos, y 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. (18)

1.5.8. Herramienta para el control de versiones

Team Foundation Server 2008

Team Foundation Server (TFS) es una plataforma de colaboración en equipo que combina el portal del equipo, el control de versiones, el seguimiento de elementos de trabajo, la gestión de proyectos, los

procesos de orientación, y la inteligencia empresarial en un único servidor. Hay dos partes para *Team Foundation Server*. Por un lado, es una colección de características que son compartidas por los distintos miembros de un equipo de proyecto para que puedan trabajar juntos con mayor eficacia. Los miembros del equipo pueden compartir planes de proyectos, productos de trabajo, el progreso y las evaluaciones de manera fácil y natural.

Principales elementos que incluye el *Team Foundation Server*:

- **Control de versiones:** Gestión del código fuente y otras prestaciones que requieren de versiones.
- **Seguimiento de elementos de trabajo:** Permite mantener un seguimiento del estado actual de los elementos de trabajo como los defectos, riesgos, tareas y escenarios.
- **La recolección de datos y presentación de informes:** Ayuda en la evaluación del equipo de proyecto, basado en información obtenida de las herramientas de *Team Foundation Server*.
- ***Team Project Portal*:** Ofrece un punto central de comunicación para un equipo de proyecto como *Microsoft SharePoint Services*, sitio web de Windows.
- **Servicios compartidos de *Team Foundation*:** Proporciona una serie de servicios de infraestructura común e invisible a los usuarios finales.

Por otra parte, *Team Foundation Server* es una plataforma diseñada específicamente para la integración y extensibilidad. Los clientes y socios pueden personalizar los elementos de *Team Foundation Server* y complementarlo con nuevas funcionalidades. Las extensiones pueden ir desde lo más simple hasta la más compleja. Pueden ir desde cambiar el nombre de un campo de un elemento de trabajo, a la integración de una herramienta totalmente nueva.

Gran parte de la interfaz de usuario de *Team Foundation Server* se ofrece a través de *Microsoft Visual Studio*. Además, la funcionalidad se expone en el cliente a través de aplicaciones de Office, específicamente *Microsoft Word* y *Microsoft Project*. (19)

1.5.9. 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* (*JavaScript*) 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.

Las aplicaciones habilitadas para AJAX ofrecen:

- Mayor eficacia, porque las partes importantes del proceso de una página web se realizan en el explorador.
- Elementos de interfaz de usuario, como indicadores de progreso, información sobre herramientas y ventanas emergentes.
- Actualizaciones parciales de la página, que actualizan sólo las partes de la página web que han cambiado.
- Integración de clientes con los servicios de aplicación de ASP.NET para la autenticación de formularios, funciones y perfiles de usuario.
- Clases de *proxy* generadas automáticamente que simplifican las llamadas a los métodos del servicio web desde el *script* de cliente.
- Un marco que permite personalizar los controles de servidor para incluir funciones de cliente.
- Compatibilidad para los exploradores más populares y utilizados habitualmente, incluidos *Microsoft Internet Explorer* y *Mozilla Firefox*. (13)

❖ ***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* (EDM).
- El proveedor *EntityClient* 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 *EntityClient* 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. (21)

❖ ***MVC ASP .NET Framework***

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.

El *framework* MVC también nos ayuda a realizar un desarrollo basado en pruebas (*Test Driven Development*, TDD), donde implementamos pruebas unitarias automáticas, que definen y verifican los requerimientos del nuevo código, antes de que escribamos el código.

Principales características de este *framework*:

- Nos permite una separación clara de detalles y TDD por defecto. Todos los contratos del núcleo del *framework* MVC están basados en interfaces y son fácilmente intercambiables (incluye interfaces como *IHttpRequest/IHttpResponse*). Podemos ejecutar pruebas unitarias sin tener que ejecutar los controladores en un proceso ASP.NET (haciendo estas pruebas mucho más rápidas). Además, podemos usar el *framework* de pruebas unitarias que se desee (incluyendo *Nunit, MBUnit, MSTest*, etc.).
- Es altamente extensible. Todo en el *framework* MVC está diseñado para que pueda ser personalizado fácilmente.
- Incluye una potente herramienta para el mapeado de URL que nos permite crear aplicaciones con URL limpias. Las URL no necesitan tener extensiones, y están diseñadas para soportar un sistema de nombres amigable.
- El *framework* MVC soporta los archivos .ASPX, .ASCX y .MASTER como plantillas de vistas (es decir, podemos usar cualquier característica de ASP.NET como las plantillas anidadas, <%= %>, etc.). Sin embargo, no usa el modelo de interacción *post-back* para las comunicaciones con el servidor, sino que, se tiene que establecer una ruta para las interacciones del usuario a través de una clase controlador (lo que implica, entre otras cosas, que no existe el *viewstate* o el ciclo de vida de las páginas en las vistas de MVC).
- Soporta todas las características de ASP.NET como la autenticación a través de formularios Windows, roles, cacheo de datos, administración del estado de la sesión, monitoreo del estado y sistemas de configuración. (20)

1.6. 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. Además, se determinaron las diferentes herramientas a utilizar en el desarrollo del proyecto y se definió el modelo RBAC como la tecnología más apta para el desarrollo de la solución. También, se seleccionó la metodología a utilizar, así como los lenguajes que se utilizarán tanto para modelar la aplicación como para su posterior implementación.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción

En el presente capítulo se sentarán las bases por las cuales se propone el desarrollo del sistema así como la descripción de la propuesta de solución incluyendo sus principales características, partiendo del análisis previo de la problemática y el objeto de estudio en cuestión. Además del levantamiento y descripción de los escenarios y los requerimientos de calidad de servicio que debe cumplir la aplicación así como la planeación y descripción de las iteraciones que se realizarán en las diferentes fases dentro del ciclo de vida del proyecto.

2.2. Descripción del problema y objeto de informatización

En la UCI existen diferentes centros de desarrollo e investigación con el propósito de apoyar el desarrollo de la informatización del país, entre los que se encuentra el CISED, encaminado a desarrollar productos, servicios y soluciones integrales en el campo de la identificación y la seguridad digital.

Actualmente en el CISED específicamente en la línea de seguridad digital, se está desarrollando un Sistema de Administración de Identidades 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 Servidor de Seguridad es el encargado de orquestar todo el flujo en los procesos de autenticación y autorización de usuarios de las aplicaciones empresariales registradas en el Sistema de Administración de Identidades. Además, es capaz de compartir la carga con otros servidores de seguridad al interactuar con los demás módulos y agilizar los procesos antes mencionados a partir del almacenamiento en cache de resultados de peticiones repetidas.

El módulo Cliente de Seguridad es el punto de comunicación entre las aplicaciones empresariales y los servidores de seguridad disponibles. Las aplicaciones o proveedores de servicio delegan su seguridad en este módulo quien intercepta los pedidos de los recursos restringidos de las aplicaciones e inicia el

proceso de autenticación y autorización con el módulo Servidor de Seguridad; los procesos terminan igualmente en este módulo.

El módulo Gestor de Sesiones como su nombre lo indica se encarga de manejar las sesiones web del Sistema de Administración de Identidades, al exponer servicios que le permiten al Servidor de Seguridad persistir información de los usuarios autenticados en el sistema. Al centralizar el manejo de sesiones este componente provee de flexibilidad y escalabilidad al Sistema de Administración de Identidades.

El módulo Administrador de Acceso autentica y autoriza a los usuarios de las aplicaciones a partir de la interacción con el Servidor de Seguridad. Para garantizar la autenticación este módulo se vale del componente SSO que interactúa con un LDAP, validando las credenciales provistas por los usuarios de la organización; este componente permite la autenticación única para múltiples aplicaciones empresariales, es decir, que un usuario provee sus credenciales solo una vez y estará autenticado en todas las aplicaciones a las que tenga acceso dentro de la organización. Para el proceso de autorización el Administrador de Acceso utiliza el Servicio de Autorización, este componente controla el acceso basado en roles, siguiendo el modelo RBAC y evalúa políticas de acceso a los recursos de las aplicaciones registradas en el Sistema de Administración de Identidades interactuando con un repositorio donde se encuentran los principales elementos del modelo RBAC.

Actualmente el módulo no cuenta con un componente que administre la configuración, a través de una interfaz, de los principales elementos del modelo RBAC. Teniendo que gestionar los elementos de dicho modelo manualmente en el repositorio, lo cual impide que muchas de las características que plantea el estándar no sean manejadas y ejecutadas correctamente, lo que frena todo el potencial del proceso de autorización.

2.3. Descripción de la propuesta de solución

Se propone implementar una aplicación web para la gestión de la configuración del RBAC que brinde una interfaz sencilla e intuitiva que permita tanto a desarrolladores como a personal no técnico administrar el uso de los roles, el acceso a las aplicaciones y la asignación de permisos. El sistema contará con siete módulos en los que se agruparán todas las funcionalidades referentes a cada uno de los elementos del modelo RBAC así como otras funcionalidades que son necesarias por su vital importancia para la gestión del acceso.

Tabla 2: Módulos del Sistema

Módulos del Sistema	Descripción
Gestión de Roles	El módulo gestión de roles aborda la creación de roles, teniendo en cuenta las diferentes vías existentes siguiendo las políticas jerárquicas en el modelo RBAC, 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, y la asignación de determinados permisos a los roles existentes, siguiendo las restricciones que plantea el modelo RBAC.
Gestión de Recursos	El módulo gestión de recursos engloba la creación, eliminación y modificación de los recursos, la creación y eliminación de tipos de recursos, así como la asignación de recursos a las diferentes aplicaciones.
Gestión de Operaciones	El módulo gestión de operaciones administra las operaciones que se permiten realizar sobre los recursos existentes en cada una de las aplicaciones registradas.
Gestión de Permisos	El módulo gestión de permisos se centra en la administración de los permisos que posteriormente serán asignados a los roles para darle funcionalidad a los mismos, incluyendo funcionalidades como la creación y eliminación de los permisos. Además, en este módulo se permite la asignación de permisos a los roles existentes.
Gestión de Restricciones	El módulo gestión de restricciones, incluye todo lo referente a las diferentes restricciones que plantea el modelo RBAC, es el centro de la funcionalidad del sistema. Este módulo es el encargado de crear las restricciones, imponiendo diferentes reglas que se deben tener en cuenta para la realización de determinadas funcionalidades.
Gestión de Reportes	El módulo gestión de reportes se encarga de realizar los diferentes reportes, brindando información actual relacionada con los elementos del modelo RBAC. Este módulo efectúa los reportes a partir de diferentes criterios que se establecen en el sistema.
Gestión de Auditoría	El módulo gestión de auditoría realiza la administración de las trazas del sistema brindando información relacionada con los diferentes eventos que se han efectuado sobre los elementos del modelo RBAC a partir de diferentes criterios que se establecen en la aplicación.

2.4. Modelado del sistema

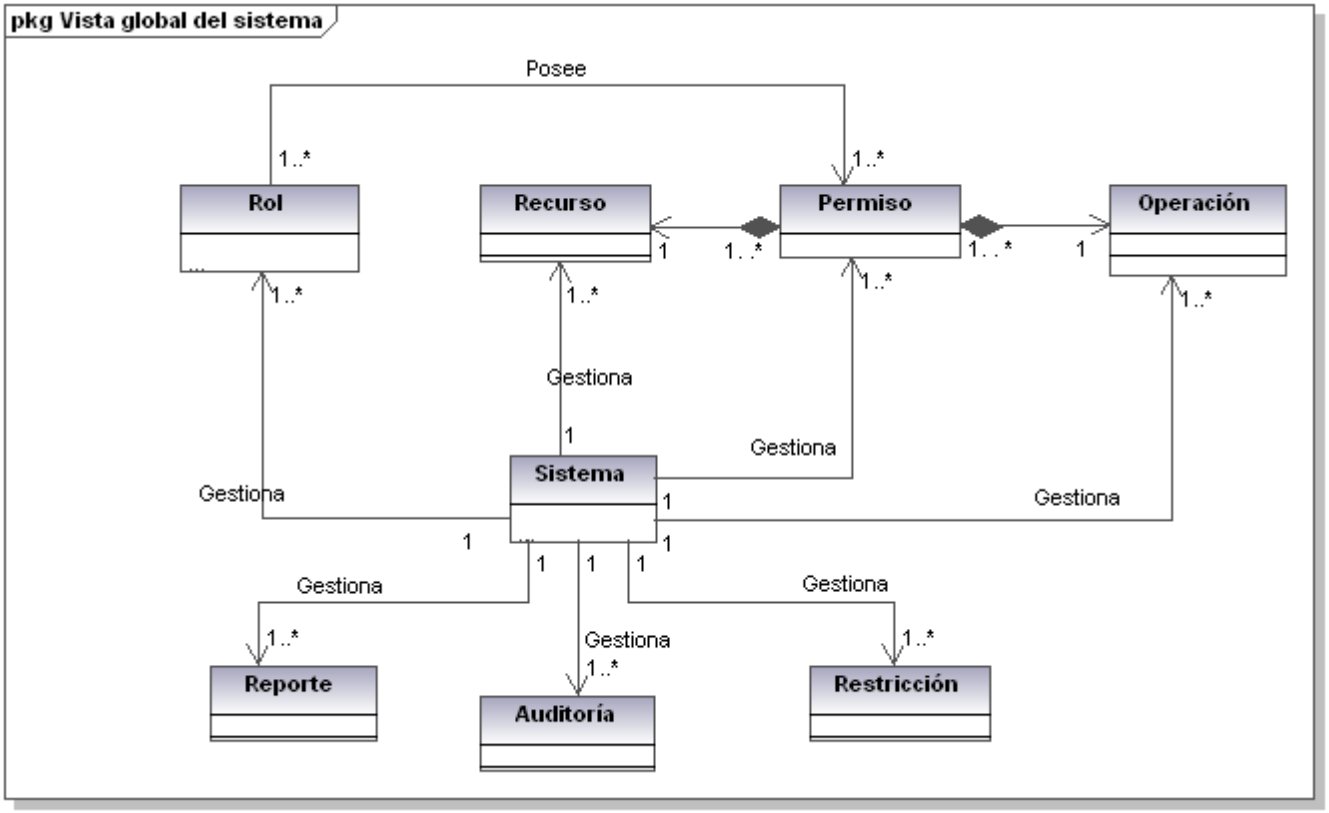


Figura 2: Vista global del sistema

2.5. Fase Visión y Alcance

El presente epígrafe tiene por objetivo documentar la fase de visión y alcance del proyecto. Esta etapa es una oportunidad para que el cliente y el equipo de trabajo discutan, acuerden y compartan la misma visión y alcance del proyecto. Al finalizar esta fase se entregará un documento que contiene la siguiente información: visión y alcance del proyecto, situación actual, conformación del equipo de trabajo, objetivos de diseño y la especificación y descripción de personas.

2.5.1. Visión y alcance de la propuesta solución

El objetivo que se persigue con esta aplicación, es dotar al componente de autorización del Sistema de Administración de Identidades de una solución final, para automatizar y agilizar las actividades de gestión de la configuración de los elementos del modelo RBAC, con una herramienta que provea todas las facilidades necesarias para que dicho componente pueda funcionar correctamente. Este proyecto

comprende las etapas de visión y alcance, planificación y desarrollo, según la metodología MSF para el desarrollo ágil de *software*.

2.5.2. Definición de personas

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

Tabla 3: Definición de personas

Persona	Descripción
Administrador del Sistema	Podrá realizar operaciones de administración y tendrá todos los privilegios que existan en el sistema

2.6. Fase Planificación

Es en esta fase en la que se termina la planificación del proyecto, el equipo prepara las especificaciones funcionales, los planes de trabajo y los cronogramas de los diferentes entregables del proyecto. Esta fase culmina con el hito “Proyecto Aprobado”. Los principales artefactos de esta fase son los escenarios y los requerimientos de calidad de servicios que sirven de guía para todo el proceso de desarrollo.

2.6.1. Escenarios del sistema

Para la implementación del sistema se realiza la especificación y descripción de los escenarios. Un escenario es un camino único de interacción del usuario a través del sistema. A medida que la persona trata de alcanzar una meta, el escenario registra los pasos concretos que se toman en el intento de alcanzar ese objetivo. Cuando se escriben los escenarios, se debe ser específico.

En esta investigación se lograron especificar 35 escenarios, que para un mejor entendimiento, fueron agrupados en los diferentes módulos existentes que a continuación se listan:

- **Módulo Gestión de Roles**

    1. Gestionar rol organizacional
    2. Gestionar rol administrativo
    3. Gestionar grupo de roles

4. Asignar usuarios a rol
5. Revocar asignación de usuarios a rol
6. Registrar trazas de la gestión de roles organizacionales
7. Registrar trazas de la gestión de roles administrativos
8. Registrar trazas de la gestión de grupos de roles

- **Módulo Gestión de Recursos**

9. Gestionar recurso
10. Gestionar tipo de recurso
11. Registrar trazas de la gestión de recursos
12. Registrar trazas de la gestión de tipos de recursos

- **Módulo Gestión de Operaciones**

13. Gestionar operación
14. Registrar trazas de la gestión de operaciones

- **Módulo Gestión de Permisos**

15. Gestionar permiso
16. Asignar permiso a un rol
17. Revocar asignación de permiso a un rol
18. Registrar trazas de la gestión de permisos

- **Módulo Gestión de Restricciones**

19. Gestionar restricción de cardinalidad de roles
20. Gestionar restricción de cardinalidad de permisos
21. Gestionar restricción de exclusión de roles
22. Gestionar restricción de exclusión de permisos de roles
23. Gestionar restricción de precondición de roles
24. Gestionar restricción temporal de roles
25. Registrar trazas de la gestión de restricción

- **Módulo de Gestión de Reportes**

26. Mostrar datos de los roles que cumplan con determinados criterios
27. Mostrar datos de los usuarios asignados a un determinado rol
28. Mostrar datos de los recursos que cumplan con determinados criterios
29. Mostrar datos de las operaciones que cumplan con determinados criterios
30. Mostrar los roles que posee un usuario determinado.
31. Exportar reportes en formato PDF.

- **Módulo Gestión de Auditoría**

32. Visualizar trazas almacenadas
33. Buscar trazas
34. Filtrar trazas por determinado criterio
35. Exportar trazas en formato PDF.

2.6.2. Requerimientos de calidad de servicio

- **Requerimientos de diseño e implementación**

El sistema debe contar con un conjunto de patrones que permita la reutilización del código así como facilidad en la actualización del mismo. El código debe cumplir con los estándares de codificación establecidos por el cliente.

- **Requerimientos de seguridad**

La seguridad del sistema será implementada a través de los diferentes módulos existentes en el proyecto tales como autenticación y autorización.

- **Requerimientos de usabilidad**

El sistema podrá ser usado por cualquier tipo de persona que posea conocimientos básicos en el manejo de la computadora, el sistema operativo Windows y el modelo RBAC.

- **Requerimientos de soporte**

Se debe ofrecer servicios de mantenimiento y actualización

- **Requerimientos de confiabilidad**

Se debe garantizar que el sistema esté disponible las 24 horas del día.
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.

2.6.3. Plan de iteraciones

Para planificar adecuadamente el desarrollo de la aplicación, es necesario estimar el tiempo que les tomará a los programadores la codificación de cada uno de los escenarios. En dependencia del tiempo necesario para la realización del escenario debe considerarse la división del mismo en partes más pequeñas. A partir de la prioridad de los escenarios se decide cuáles de ellos se implementarán en las primeras iteraciones, las funcionalidades críticas del sistema deben ser codificadas en iteraciones tempranas del ciclo.

Iteración #1: Se propone codificar los escenarios que proveen las funcionalidades pertenecientes a los módulos de “Gestión de Recursos”, “Gestión de Roles”, “Gestión de Operaciones” y “Gestión de Permisos” del sistema.

Iteración #2: Se codificarán los escenarios correspondientes al módulo “Gestión de Restricciones”.

Iteración #3: Última iteración, se implementarán los escenarios restantes, integrados en los módulos “Gestión de Reportes” y “Gestión de Auditoría”.

Tabla 4: Estimación de iteraciones

No	Módulo	Prioridad	Riesgo	Estimación (horas)	Iteración
1	Gestión de Roles	Alta	Medio	168	1
2	Gestión de Recursos	Alta	Medio	72	1
3	Gestión de Operaciones	Alta	Medio	48	1
4	Gestión de Permisos	Alta	Medio	120	1
5	Gestión de Restricciones	Media	Alto	240	2

6	Gestión de Reportes	Media	Medio	120	3
7	Gestión de Auditoría	Media	Alto	240	3

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. A continuación, mostramos las descripciones de los escenarios pertenecientes al módulo “Gestión de Operaciones”, todos de alta prioridad.

Tabla 5: Descripción del escenario “Gestionar operación”

Título	Gestionar operación
Área	Autorización
Iteración	1
Estado	Activo
Razón de estado	Proceso de construcción
Responsable	Evaristo José Madarro
Descripción	El administrador del sistema inicia el escenario al seleccionar la opción “Gestionar operación”. A continuación, el sistema muestra todas las operaciones existentes brindando opciones para adicionar, mostrar, editar o eliminar una operación determinada. Nota: Las opciones de adicionar, mostrar, editar y eliminar una operación han sido omitidas debido a que estas se describen como tareas del escenario en cuestión.
Historial	Inicio de descripción del escenario
Prioridad	Alta
Orden de Magnitud	1

Tabla 6: Descripción del escenario “Registrar trazas de la gestión de operaciones”

Título	Registrar trazas de la gestión operaciones
Área	Autorización
Iteración	1
Estado	Activo
Razón de estado	Proceso de construcción
Responsable	Evaristo José Madarro

Descripción	Este escenario es realizado de forma automática por el propio sistema. Consiste en la captura y persistencia de todos los eventos que se producen, relacionados con las funcionalidades que se realizan en el módulo “Gestión de Operaciones”. Nota: Las funcionalidades “Capturar evento”, y “Persistir datos del evento” han sido omitidas debido a que estas se describen como tareas del escenario en cuestión.
Historial	1. Inicio de descripción del escenario
Prioridad	Alta
Orden de Magnitud	1

2.6.4.1. Especificación de tareas por escenarios

El escenario “Gestionar operación” se dividió en 4 tareas fundamentales, que proveerán las funcionalidades de crear, editar, mostrar y eliminar una operación.

Tabla 7: Descripción de la tarea “Crear operación”

Título	Crear operación
Tipo de Tarea	Desarrollo
Iteración	1
Estado	Activo
Razón de estado	Proceso de construcción
Responsable	Evaristo José Madarro
Descripción	Se inicia al seleccionar la opción “Crear operación”. A continuación, el sistema muestra los campos necesarios que el usuario debe completar y luego de verificados los datos introducidos, el sistema introduce la información en la base de datos.
Historial	Inicio de descripción del escenario
Tiempo de desarrollo (horas)	30
Rango de prioridad	1

Tabla 8: Descripción de la tarea “Editar operación”

Título	Editar operación
Tipo de Tarea	Desarrollo
Iteración	1
Estado	Activo
Razón de estado	Proceso de construcción
Responsable	Evaristo José Madarro

Descripción	Se inicia al seleccionar la opción “Editar operación”. A continuación, el sistema muestra los campos con los datos que posee la operación en cuestión, brindando la posibilidad de modificar la información que el usuario desee, luego de verificados los datos modificados, el sistema introduce la información en la base de datos.
Historial	1. Inicio de descripción del escenario
Tiempo de desarrollo (horas)	48
Rango de prioridad	1

Tabla 9: Descripción de la tarea “Mostrar operación”

Título	Mostrar operación
Tipo de Tarea	Desarrollo
Iteración	1
Estado	Activo
Razón de estado	Proceso de construcción
Responsable	Evaristo José Madarro
Descripción	Se inicia al seleccionar la opción “Mostrar operación”. A continuación, el sistema muestra los datos que posee la operación en cuestión.
Historial	1. Inicio de descripción del escenario
Tiempo de desarrollo (horas)	48
Rango de prioridad	1

Tabla 10: Descripción de la tarea “Eliminar operación”

Título	Eliminar operación
Tipo de Tarea	Desarrollo
Iteración	1
Estado	Activo
Razón de estado	Proceso de construcción
Responsable	Evaristo José Madarro
Descripción	Se inicia al seleccionar la opción “Eliminar operación”. A continuación, el sistema elimina la operación seleccionada de la base de datos.
Historial	1. Inicio de descripción del escenario
Tiempo de desarrollo (horas)	48
Rango de prioridad	1

El escenario de “Registrar trazas de la gestión de operaciones” se dividió en 2 tareas fundamentales, que proveerán las funcionalidades de “Capturar evento” y “Persistir datos del evento”.

Tabla 11: Descripción de la tarea “Capturar evento”

Título	Capturar evento
Tipo de Tarea	Desarrollo
Iteración	1
Estado	Activo
Razón de estado	Proceso de construcción
Responsable	Evaristo José Madarro
Descripción	En este proceso el sistema captura los datos de los eventos que ocurran, relacionados con la gestión de operaciones.
Historial	1. Inicio de descripción del escenario
Tiempo de desarrollo (horas)	48
Rango de prioridad	1

Tabla 12: Descripción de la tarea “Persistir datos del evento”

Título	Persistir datos del evento
Tipo de Tarea	Desarrollo
Iteración	1
Estado	Activo
Razón de estado	Proceso de construcción
Responsable	Evaristo José Madarro
Descripción	El sistema almacena los datos de los eventos capturados, relacionados con la gestión de operaciones.
Historial	2. Inicio de descripción del escenario
Tiempo de desarrollo (horas)	48
Rango de prioridad	1

2.7. Conclusiones

En este capítulo se definieron un total de 35 escenarios y se describieron los escenarios correspondientes al módulo “Gestión de Operaciones” con sus respectivas tareas mostrando las funcionalidades que se desarrollarán en el ciclo de vida de la aplicación y se especificaron los requerimientos de calidad de servicio del sistema tales como seguridad, soporte, etc. Además, se realizó la descripción de la propuesta de solución enfatizando principalmente en su infraestructura, definida en 7 módulos.

CAPÍTULO 3: DESARROLLO Y PRUEBAS DE LA PROPUESTA DE SOLUCIÓN

3.1. Introducción

En el presente capítulo se describe el proceso de diseño de la solución propuesta a través de la especificación de los patrones de diseño, la arquitectura y los artefactos propuestos por la metodología utilizada, además se detallan las iteraciones realizadas durante la fase de desarrollo, así como las pruebas efectuadas sobre el sistema.

3.2. Fase Desarrollo

3.2.1. Especificación de la arquitectura

3.2.1.1. Estilo arquitectónico

El estilo arquitectónico Modelo-Vista-Controlador (MVC) 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.

- 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)

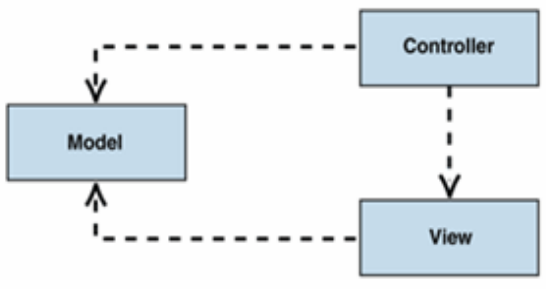


Figura 3: Estilo arquitectónico MVC

3.2.1.2. Vista lógica

El sistema en su vista más abstracta es una solución Cliente-Servidor, compuesta por diferentes módulos relacionados entre sí. La arquitectura se encuentra representada por 4 capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades, permitiendo reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas. Esta distribución de capas permite que se realicen grandes cambios sin tener que realizar cambios en las demás capas. Una vez que estas estén bien definidas la comunicación entre ellas se realizará solo a nivel de interfaces que permiten trabajar de manera transparente a las instancias reales.

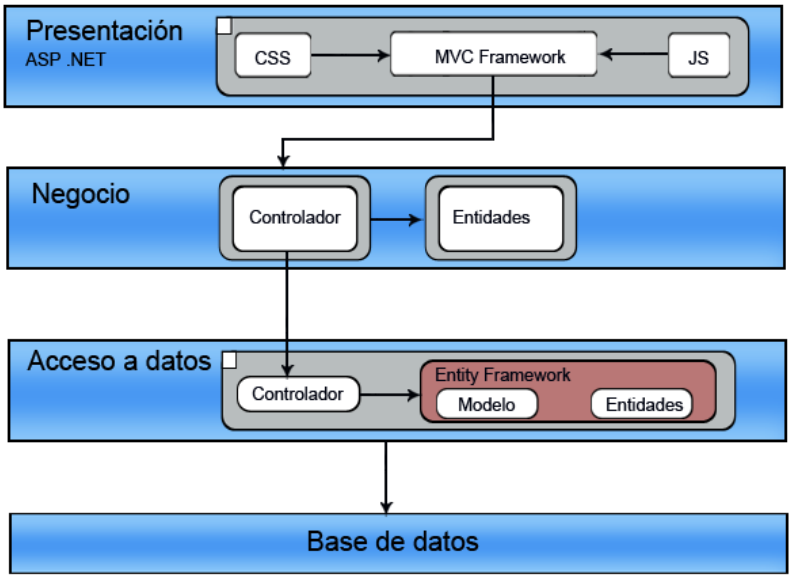


Figura 4: Vista lógica de la arquitectura del sistema

3.2.1.2.1. Descripción de capas

- **Capa de presentación**

Está compuesta por todas las interfaces de usuario y los componentes necesarios para su correcto funcionamiento. Estos elementos pueden ser ficheros *JavaScript* y CSS.

Esta capa se encuentra representada por un proyecto web y tiene interacción directa con la capa de negocio. Para el desarrollo de este proyecto web se hace uso del *framework* MCV ASP .NET el cual divide la implementación del proyecto en 3 componentes: modelos, vistas y controladores.

- **Capa de negocio**

En esta capa se recogen todas las funcionalidades necesarias para darle solución a los requerimientos de negocio. Las funcionalidades se encuentran definidas según el contexto en el que se desenvuelven. Tienen la responsabilidad de manejar todas las operaciones sobre una entidad de negocio en específico, así como todas las entidades que por conceptos de composición se encuentran relacionadas con esta. Por cada entidad de negocio se crea un controlador y una interfaz que debe ser implementada por el acceso a dato que le dará soporte.

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.

- **Capa de base de datos**

Está constituida por todo el conjunto de tablas y procedimientos que permiten el almacenamiento de la información recolectada y procesada.

3.2.2. Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Un patrón de diseño identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.

3.2.2.1. Patrones GRASP (*General Responsibility Assignment Software Patterns*).

Los patrones GRASP 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. A continuación, describiremos los patrones para asignar responsabilidades:

- **Patrón experto**

Un modelo de clase puede definir docenas y hasta cientos de clases de *software*, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Si estas se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se nos presenta la oportunidad de reutilizar los componentes en futuras aplicaciones. 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. (23)

- **Patrón creador**

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. (23)

- **Patrón 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. (23)

- **Patrón 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. (23)

- **Patrón 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. (23)

3.2.2.2. Patrones GoF (*Gang of Four*).

- **Patrón fachada (*facade*)**

Provee una interfaz unificada para un conjunto de interfaces de un subsistema. *Facade* define una interfaz de alto nivel que hace al sistema fácil de usar dividiendo el sistema en subsistemas, ayudando a reducir la complejidad del mismo, un diseño común disminuye las dependencias y comunicaciones entre subsistemas. Protege al cliente de los componentes del subsistema, así reduce el número de objetos con los que el cliente se relaciona. Promueve un débil acoplamiento entre el subsistema y los clientes. Esto permite modificar componentes en el subsistema sin modificar el cliente. (24)

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.

A continuación se muestra un diagrama donde se representan las clases pertenecientes al módulo “Gestión de Operaciones” con sus relaciones, debido a que el diagrama de clases general es demasiado extenso.

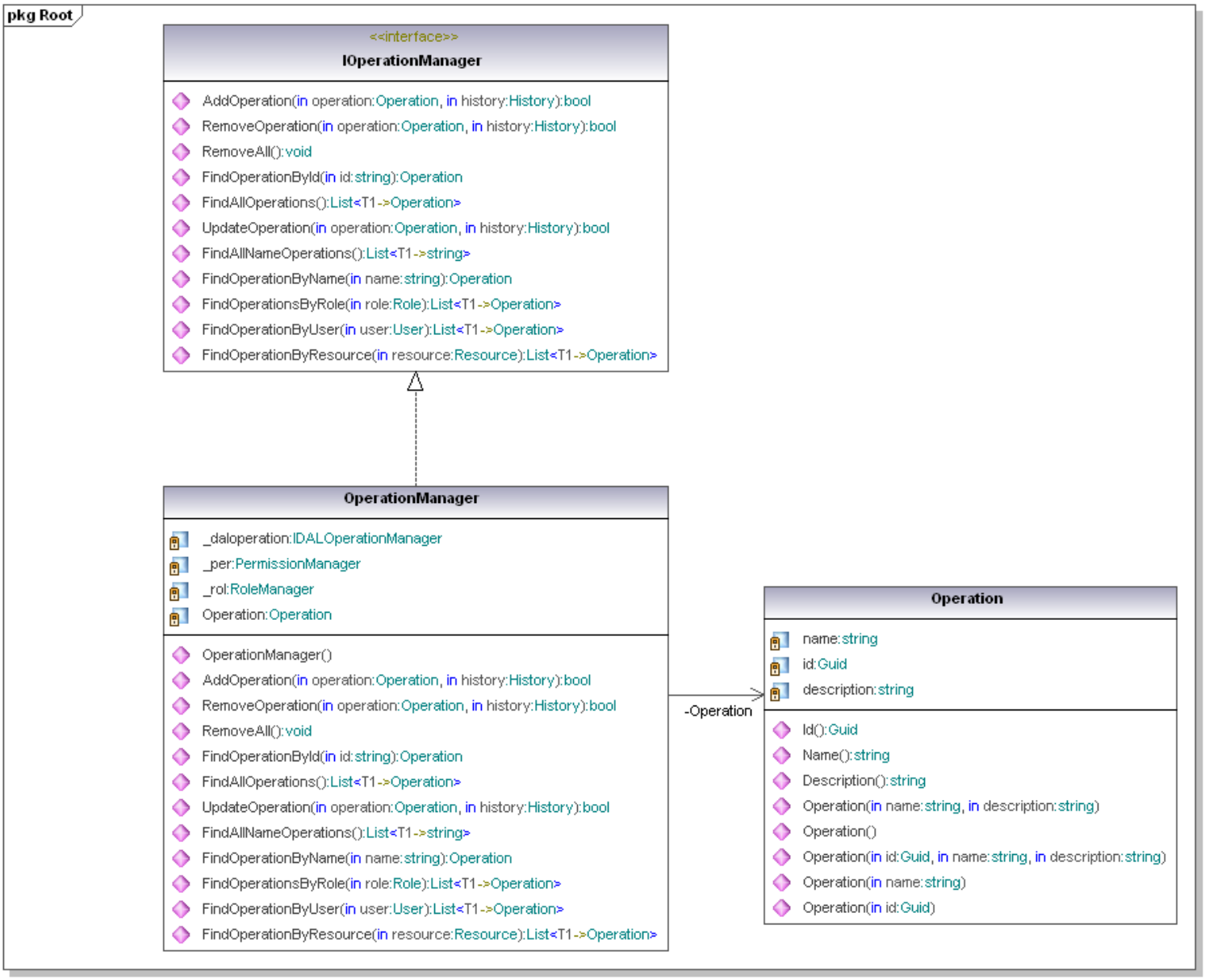
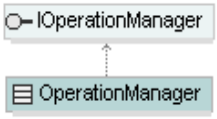


Figura 5: Diagrama de clases del módulo "Gestión de Operaciones"

3.2.4. Descripción de clases

Tabla 13: Descripción de la clase *"OperationManager"*

Nombre:	*OperationManager*
Tipo de Clase:	Controladora
Descripción:	Esta clase es la encargada de implementar todos los métodos de la interfaz *IOperationManager*. Maneja todas las acciones que se deseen realizar sobre las operaciones. 
Atributos	
Nombre	**Descripción**
_daloperation: IDAlOperationManager	Objeto de la clase interfaz de acceso a datos *IDALOperationManager*.
_per: PermissionManager	Objeto de la clase controladora *PermissionManager*
_rol: RoleManager	Objeto de la clase controladora *RoleManager*
Operation: Operation	Objeto de la clase entidad *Operation*
Métodos	
Nombre	**Descripción**
AddOperation (in operation: Operation, in history: History): bool	Permite adicionar una nueva operación.
RemoveAll (): void	Permite eliminar todas las operaciones existentes
FindOperationById (in id: string): Operation	Retorna la operación que posea el id especificado por parámetro.
FindAllNameOperation (): List<string>	Devuelve los nombres de cada una de las operaciones existentes.
FindAllOperation (): List<Operation>	Devuelve todas las operaciones existentes.
FindOperationByName (in name: string): Operation	Retorna la operación que posea el nombre especificado por parámetro.
FindOperationsByRole (in role: Role): List<Operation>	Devuelve todas las operaciones que el rol especificado por parámetro puede ejecutar en los permisos que posee.

FindOperationByUser (in role: User): List<Operation>	Devuelve todas las operaciones que el usuario especificado por parámetro puede realizar a través de los permisos asignados a los diferentes roles a los que está asignado.
UpdateOperation (in operation: Operation, in history: History): bool	Actualiza la operación especificada por parámetro.
FindOperationByResource(in resource: Resource): List<Operation>	Retorna todas las operaciones que se pueden ejecutar sobre el recurso especificado por parámetro.
RemoveOperation (in operation: Operation, in history: History): bool	Elimina la operación que se especifica por parámetro.
Asociaciones	RoleManager, PermissionManager, Operation

3.2.5. Clases persistentes

En este epígrafe se describirán las clases persistentes que presenta el módulo “Gestión de Operaciones”, mostrado anteriormente, para que el usuario pueda comprenderlas y conocer como manipularlas de una forma factible. Las clases persistentes se definen para conocer la información real representada en las tablas de la base de datos. De forma general, las clases persistentes y sus atributos son análogos a las entidades lógicas y sus atributos.

3.2.5.1. Descripción de clases persistentes

Tabla 14: Descripción de la clase *“Operation”*

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

3.2.6. Modelado de la base de datos

El modelo de datos es un conjunto de conceptos que pueden servir para describir la estructura de una base de datos, esto se refiere a tipos de datos, sus vínculos y las restricciones que deben cumplir estos datos. Además, éste se utiliza para describir la representación lógica y física de la información persistente manejada por el sistema.

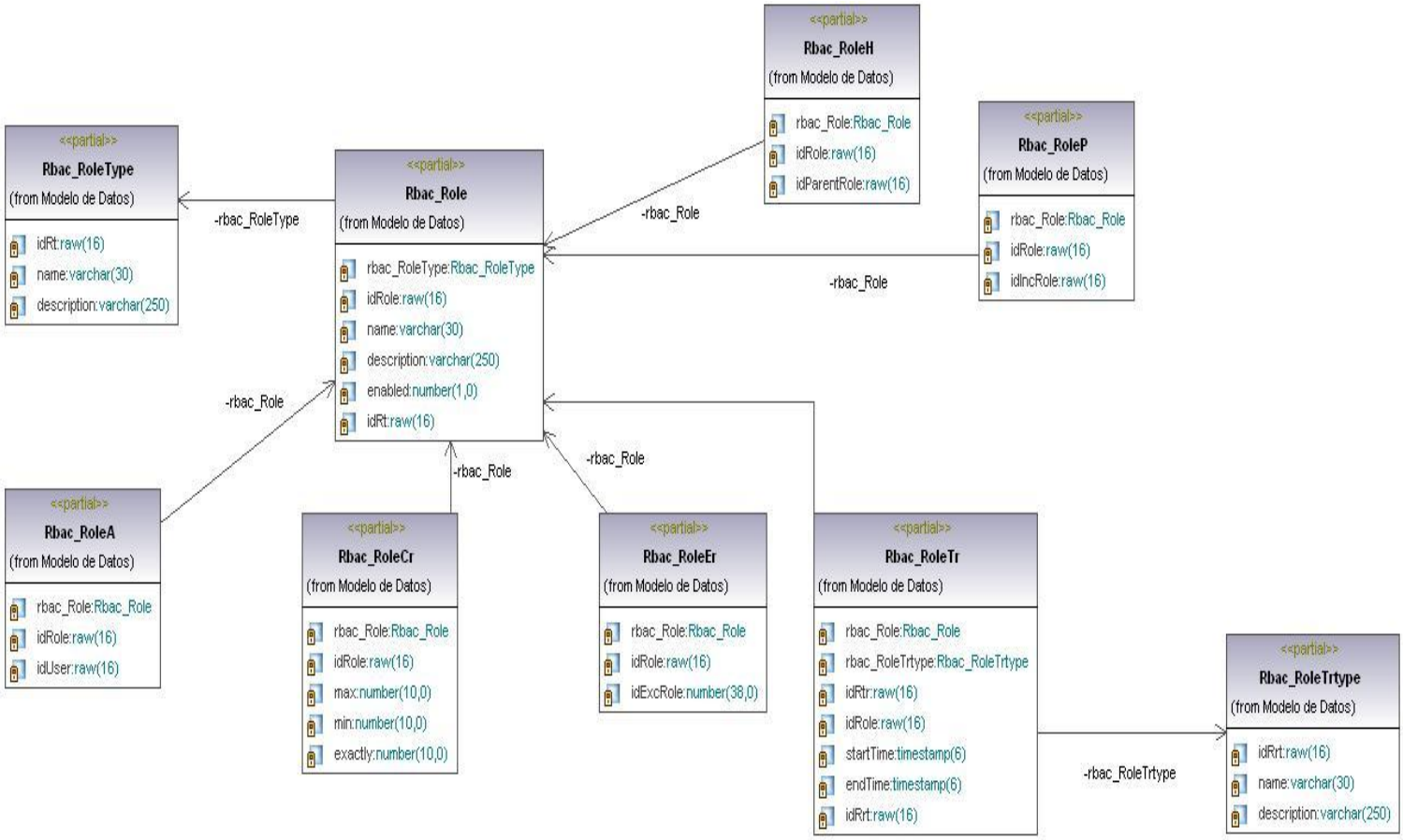


Figura 6: Modelo de datos del módulo "Gestión de Roles"

3.2.7. Diagrama de aplicación

Un diagrama de aplicación es una solución de ámbito, y 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*. El diagrama muestra las conexiones entre estas aplicaciones, que reflejan la configuración actual de la solución.

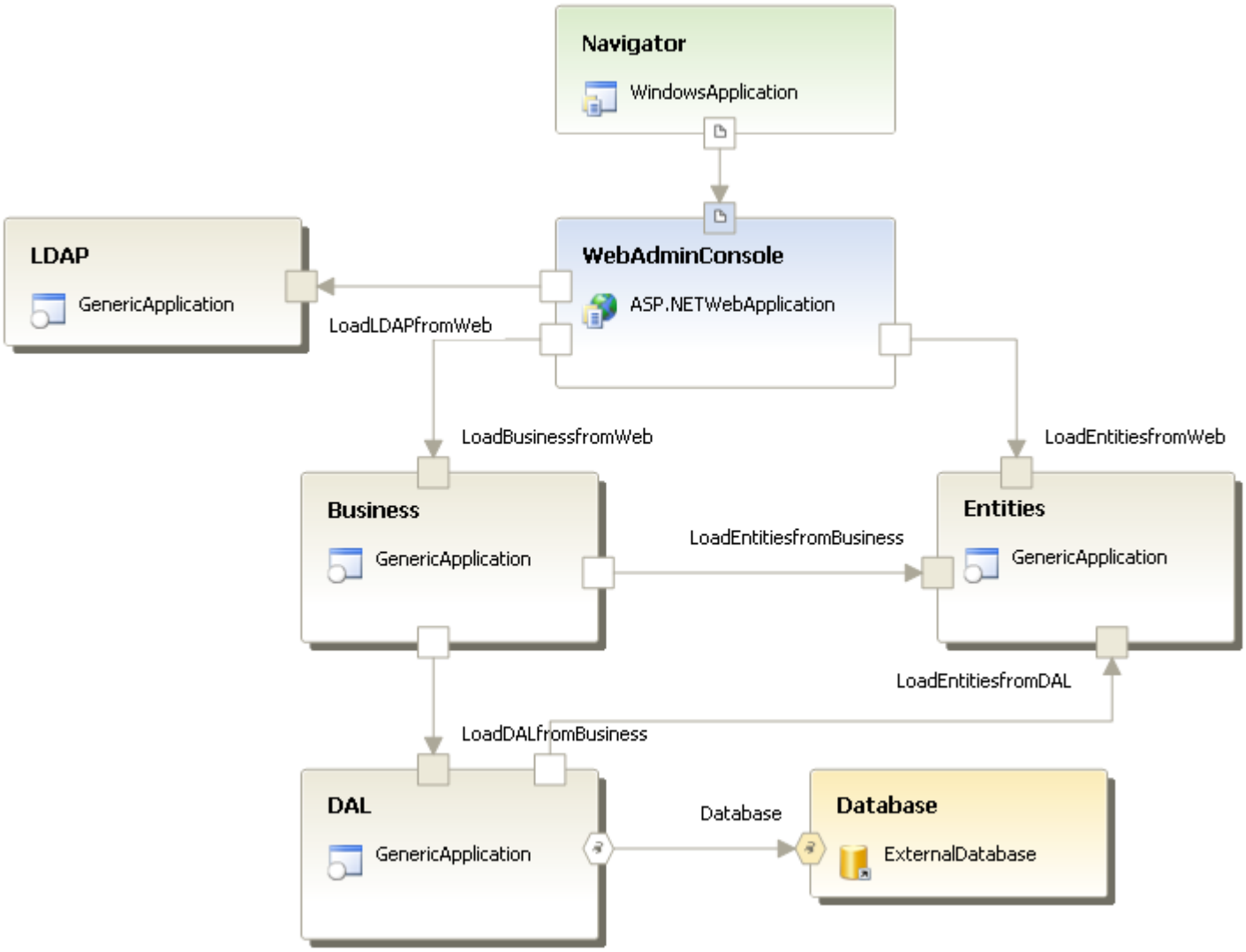


Figura 7: Diagrama de aplicación

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 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.

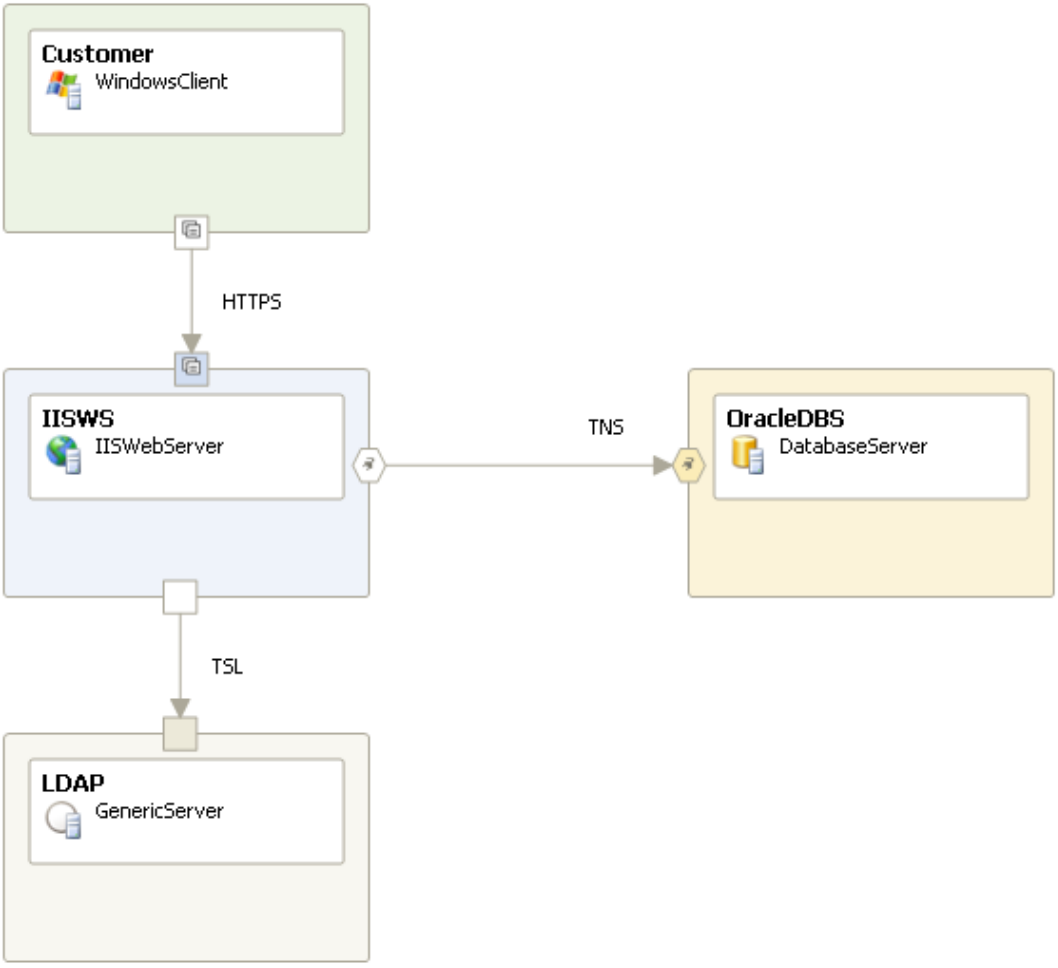


Figura 8: Diagrama de centro de datos lógicos

3.2.9. Implementación de escenarios

Durante el transcurso de las iteraciones se realiza la implementación de los escenarios que fueron seleccionados en cada una de ellas. Al principio de estas se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de ser necesario. Como parte del plan se descomponen estos escenarios en tareas. 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 con mayor prioridad para el usuario, 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 Roles	168	144
Gestión de Recursos	72	72
Gestión de Operaciones	48	48
Gestión de Permisos	120	96

3.2.9.2. Iteración 2

En esta iteración se implementaron los escenarios correspondientes al módulo “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 Restricciones	240	192

3.2.9.3. Iteración 3

En esta iteración se implementaron los escenarios de prioridad media. Esta iteración tiene la finalidad de proporcionar un ambiente expresivo y cómodo para el usuario. 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
Gestión de Reportes	120	96
Gestión de Auditoría	240	120

3.2.10. Interfaz de usuario

Gestión de Operaciones: Interfaz mediante la cual los administradores del sistema, podrán ver las operaciones existentes. Además, permitirá buscar operaciones por un criterio específico, así como crear, eliminar, editar y mostrar los detalles de una operación determinada.

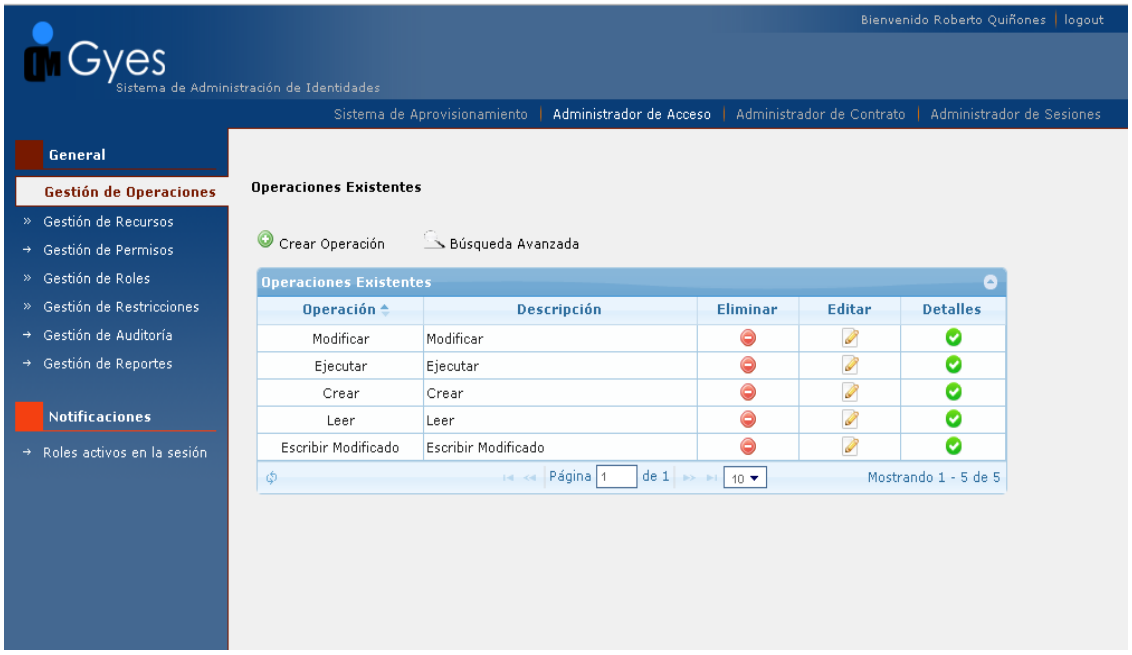


Figura 9: Interfaz del módulo "Gestión de Operaciones"

Gestión de Auditoría: Interfaz mediante la cual los administradores del sistema, podrán ver los eventos y acciones realizados en la aplicación. Además, permitirá buscar un evento determinado por diferentes criterios de búsqueda, tales como fecha, máquina cliente y entidad. También ofrece la funcionalidad de eliminar un evento seleccionado.

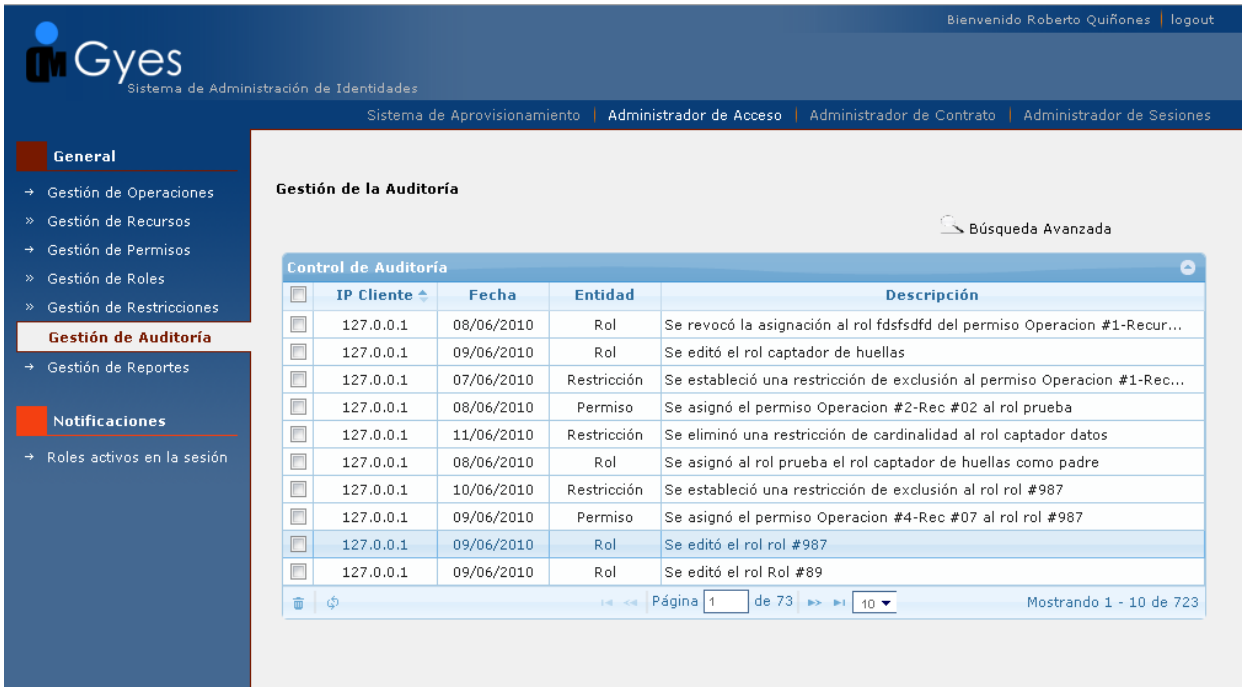


Figura 10: Interfaz del módulo "Gestión de Auditoría"

3.3. Pruebas

Las pruebas no pueden asegurar la ausencia de defectos; sólo pueden demostrar que existen defectos en el *software*. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados. Las condiciones controladas pueden ser normales o anormales. La prueba puede intencionalmente forzar al programa y producir errores en las respuestas para determinar si los sucesos ocurren cuando no tendrían que ocurrir o cuando los hechos no suceden cuando deberían suceder.

3.3.1. Pruebas unitarias

En programación, una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto se utiliza para asegurar que cada uno de los módulos funcione correctamente por separado. El objetivo de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores.

Las pruebas fomentan el cambio y la refactorización. Si se considera que el código es mejorable se puede cambiar sin ningún problema. Si el cambio no estuviese realizado correctamente las pruebas avisarán de ello. Se reducen drásticamente los problemas y tiempos dedicados a la integración y se simulan las dependencias, lo que permite que se pueda probar el código sin disponer del resto de módulos.

Las pruebas unitarias que se realizan a los servicios del sistema para validar la salida de los datos le aseguran al programador que su solución no presenta errores lógicos de programación y ante una entrada de datos determinada por el probador los valores obtenidos son los esperados.

3.3.1.1. Pruebas de caja blanca

El método de prueba utilizado para detectar defectos al *software* desarrollado fue el de caja blanca. Permitió asegurar que las operaciones internas se ajustan a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada. Con el uso intensivo de las técnicas de pruebas de caja blanca se lograron verificar los caminos de control importantes y permitieron descubrir errores dentro del ámbito del módulo.

Ventajas de utilizar *Visual Studio Team System* para realizar pruebas

- Proporcionan acciones preventivas sobre fallos.
- Crean primer nivel de la documentación
- 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.

- Configuración de los atributos, propiedades y métodos de la clase de prueba creando el contexto de la prueba que contiene la información relacionada con la prueba.
- Creación de las referencias para el proyecto objetivo de forma automática.

A continuación se muestran las pruebas realizadas a los métodos "*FindOperationByName"* y "*FindAllRoleByName":*

```
/// <summary>
///A test for FindOperationByName
///</summary>
[TestMethod()]
public void FindOperationByNameTest()
{
    DALOperationManager target = new DALOperationManager(); // TODO: Initialize to an appropriate value
    string name = "Leer"; // TODO: Initialize to an appropriate value
    Operation expected = new Operation() { Name = "Leer" }; // TODO: Initialize to an appropriate value
    Operation actual;
    actual = target.FindOperationByName(name);
    Assert.AreEqual(expected.Name, actual.Name);
    Assert.IsTrue(expected.Name.ToString() == actual.Name.ToString());
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 11: Prueba al método "*FindOperationByName"*

```
/// <summary>
///A test for FindAllRoleByName
///</summary>
[TestMethod()]
public void FindAllRoleByNameTest()
{
  DALRoleManager target = new DALRoleManager(); // TODO: Initialize to an appropriate value
  string name = "captador datos"; // TODO: Initialize to an appropriate value
  Role expected = new Role() { Name="captador datos"}; // TODO: Initialize to an appropriate
  Role actual;
  actual = target.FindAllRoleByName(name);
  Assert.AreEqual(expected.Name.ToString(), actual.Name.ToString());
  Assert.IsTrue(expected.Name.ToString() == actual.Name.ToString());
  //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figura 12: Prueba al método "*FindAllRoleByName"*

Arrojando como resultado de las pruebas realizadas que el código no presenta problemas. Mostrándose los resultados en las siguientes tablas:

Tabla 18: Prueba al método "*FindOperationByName*"

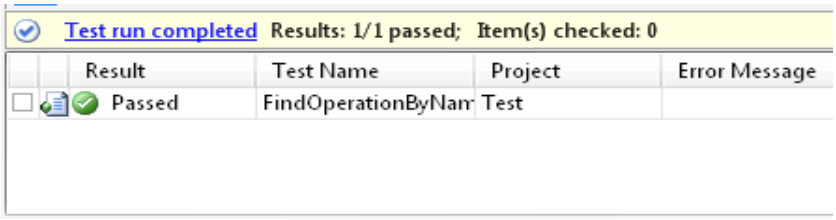
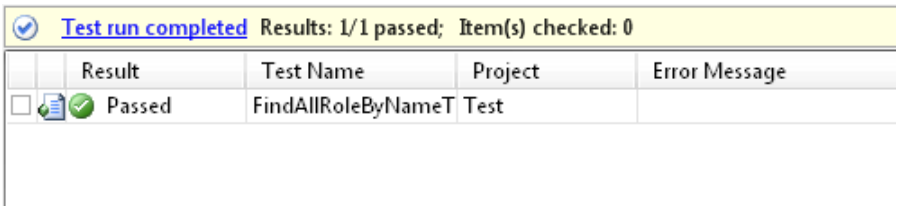
Prueba de Unidad		
Nombre Prueba: *FindOperationByNameTest*		
Estado: Satisfactoria	**Tipo:** Caja blanca	**Última ejecución:** 07/06/2010
Ejecutado por: Osniel Griñán Rodríguez	**Verificado por:** Maikel De La Torre Luis	
Descripción: Para poder ejecutar la prueba se debe pasar un tipo de dato *string* correspondiente al nombre de la operación, en caso de que exista una operación con este nombre, el método debe buscarla y retornarla, en caso contrario devuelve vacío.		
Entrada: *String name*		
Criterio de aceptación: Muestra un objeto del tipo operación, con el *id* de la operación, el nombre de la operación y la descripción.		
Resultado: 		

Tabla 19: Prueba al método "*FindAllRoleByName*"

Prueba de Unidad		
Nombre Prueba: *FindAllRoleByNameTest*		
Estado: Satisfactoria	**Tipo:** Caja blanca	**Última ejecución:** 07/06/2010
Ejecutado por: Osniel Griñán Rodríguez	**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 rol, en caso de que exista un rol con este nombre, el método debe buscarlo y retornarlo, en caso contrario devuelve vacío.		
Entrada: *String name*		
Criterio de aceptación: Muestra un objeto del tipo rol, con el *id* del rol, el nombre del rol, la descripción y el tipo de rol.		
Resultado: 		

3.4. Conclusiones

En el presente capítulo se definió la estructura básica del sistema, que consiste en una aplicación web basada en el estilo arquitectónico Modelo-Vista-Controlador (MVC) en la capa de presentación y una arquitectura multicapa para desarrollar la comunicación entre las distintas capas del sistema. En el desarrollo de la propuesta de solución se tuvieron en cuenta diferentes patrones de diseño, tales como, fachada, experto, creador, etc., permitiendo agilizar el proceso de implementación. Se especificó la estructura de los módulos, y una descripción de las clases. Se muestra el diagrama lógico de centro de datos en el cual se puntualiza la comunicación de la aplicación con el cliente, servicios, servidores y otras aplicaciones, así como el diagrama de aplicaciones.

CONCLUSIONES GENERALES

El presente trabajo posibilitó desarrollar una aplicación web que gestiona la configuración de RBAC en el Sistema de Administración de Identidades, eliminando de esta manera los problemas planteados en la situación problemática. Para la realización de la propuesta de solución, se determinó desarrollar una aplicación web sobre la plataforma .NET, por las ventajas que esta brinda. Se utilizó *Entity Framework* como tecnología para generar el acceso a datos, el MVC ASP .NET como *framework* para la arquitectura, además de bibliotecas con tecnologías AJAX como *AJAX Control Toolkit* y JQuery para el desarrollo de las interfaces de la solución y MSF como metodología de desarrollo.

Se realizó el diseño de la aplicación logrando un acercamiento a la misma para su posterior implementación. Con la aplicación se garantizó la gestión de la configuración del RBAC en el Sistema de Administración de Identidades de la línea de seguridad del CISED. Luego de realizadas las pruebas de calidad, se validó la solución y se determinó que esta cumple con las necesidades de la línea de seguridad del CISED.

Por todo lo anterior expuesto se concluye que los objetivos propuestos para el presente trabajo han sido cumplidos satisfactoriamente, ya que, la aplicación da solución a la situación problemática que le dio origen.

RECOMENDACIONES

Una vez concluido el presente trabajo de diploma se recomienda:

- Desarrollar un sistema de cacheo para la aplicación que proporcione un mejor funcionamiento en cuando a tiempo de respuesta de la aplicación.
- Ampliar el sistema hacia un entorno distribuido con el objetivo de administrar de forma centralizada las aplicaciones en más de una organización.
- Implementar un editor gráfico para facilitar la interacción del usuario con la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

1. **Microsoft Corporation.** [En línea] 2004. [Citado el: 5 de Diciembre de 2009.] http://download.microsoft.com/download/b/0/c/b0c210bd-6d80-4d40-9a43-60c7b643ab04/w86-microsoft-folleto.pdf.

2. **Oracle Corporation.** [En línea] 2009. [Citado el: 2 de Diciembre de 2009.] http://www.oracle.com/products/middleware/identity-management/docs/intro-oracle-idm-whitepaper.pdf.

3. **IBM Corporation.** [En línea] [Citado el: 3 de Diciembre de 2009.] http://www.infovision.info/clientes/appscan/IBM_TAM_ESP.pdf.

4. **Novell Corporation.** [En línea] 2010. [Citado el: 15 de Enero de 2010.] http://www.novell.com/products/identitymanager/.

5. **Reed, Archie.** [En línea] 2004. [Citado el: 7 de Diciembre de 2009.] http://searchcio.techtarget.com/searchCIO/downloads/DGIM_Ch1Excerpt.pdf.

6. **Sánchez, Miguel, Jiménez, Beatriz y Gutiérrez, Francisco L.** [En línea] [Citado el: 20 de Noviembre de 2009.] http://www.sistedes.es/sistedes/pdf/2007/SCHA-07-Sanchez-Acceso.pdf.

7. **Novalys Corporation.** [En línea] [Citado el: 25 de Noviembre de 2009.] http://www.visual-guard.com/download/VisualGuard-Detailed-Features-SP.pdf.

8. **Solaris Corporation.** [En línea] [Citado el: 19 de Noviembre de 2009.] http://www.sun.com/software/whitepapers/wp-rbac/wp-rbac.pdf.

9. **Microsoft Corporation.** MSF for Agile Software Development Process Guidance. [En línea] 2006. [Citado el: 27 de Noviembre de 2009.] http://www.microsoft.com/downloads/details.aspx?FamilyId=9F3EA426-C2B2-4264-BA0F-35A021D85234&displaylang=en.

10. **Hernáandez, Enriquez Orallo.** [En línea] 1999. [Citado el: 15 de Diciembre de 2009.] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

11. [En línea] [Citado el: 12 de Diciembre de 2009.] http://www.cepeu.edu.py/LIBROS_ELECTRONICOS_2/Lenguaje%20c.pdf.

12. **Extremo, Unai Baigorri y Sotomayor, Borja Basilio.** [En línea] [Citado el: 16 de Diciembre de 2009.] http://www.people.cs.uchicago.edu/~borja/pubs/revistaeside2002.pdf.

13. **Pérez, Damián Valdés.** Maestros del Web. [En línea] [Citado el: 25 de Enero de 2010.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

14. Danysoft. [En línea] 2009. [Citado el: 14 de Diciembre de 2009.] http://shop.danysoft.com/Altova-UModel.

15. Kioskea .Net. [En línea] [Citado el: 23 de Enero de 2008.] http://es.kioskea.net/contents/css/cssintro.php3.

16. [En línea] 25 de Enero de 2010. [Citado el: 5 de Febrero de 2010.] http://lamonterapicona.es/2010/01/25/la-plataforma-net/.

17. Tecnologías Microsoft. [En línea] 12 de Diciembre de 2007. [Citado el: 3 de Febrero de 2010.] http://mredison.wordpress.com/2007/12/02/caractersticas-de-visual-studio-2008/.

18. **Hernando, Roberto Velasco.** [En línea] [Citado el: 10 de Frebero de 2010.] http://www.rhernando.net/modules/tutorials/doc/bd/oracle.html.

19. **Dennis.** MSDN. [En línea] Enero de 2006. [Citado el: 25 de Enero de 2009.] http://translate.googleusercontent.com/translate_c?hl=es&langpair=en|es&u=http://msdn.microsoft.com/en-us/library/ms364062(VS.80).aspx&rurl=translate.google.com.cu&usg=ALkJrhgYxxNTJ3uVwg7oTIq-upHmzlyffg#team%20foundation%20fundamen_topic1.

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

21. [En línea] [Citado el: 5 de Marzo de 2010.] https://ame.endesa.es/confluenceame/display/PPAME4NET/Entity+Framework.es-ES.

22. **Ciudad Ricardo, Febe Ángel.** [En línea] Abril de 2006. [Citado el: 10 de Abril de 2010.] http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista2.shtml.

23. **Canales Mora, Roberto.** [En línea] 22 de Diciembre de 2003. [Citado el: 20 de Marzo de 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp.

24. WorldLingo. *WorldLingo.* [En línea] [Citado el: 12 de Abril de 2010.] http://www.worldlingo.com/ma/enwiki/es/Design_pattern_%28computer_science%29.

BIBLIOGRAFÍA

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

KATZ, Y., & BIBEAULT, B. (2008). *jQuery in Action.* Greenwich, CT: Manning Publications Co.

Palermo, J., Scheirman, B., & Bogard, J. (2010). *ASP .NET MVC In Action.*

Pressman, R. S. (2002). *Ingeniería del Software: Un Enfoque Práctico.* Madrid: McGraw-Hill/Interamericana.

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

Walther, S. (2010). *ASP .NET MVC Framework UNLEASHED.* Indianapolis.

GLOSARIO DE TÉRMINOS

A

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. · ***16

AJAX: JavaScript asíncrono y XML · ***23***, ***27***, ***64***

B

BizTalk*:** Herramienta utilizada para conectar varios sistemas diferentes a través de un sistema de mensajes y orquestación basado en XML. · ***55, ***56***

C

CISED: Centro de Identificación y Seguridad Digital · ***VI***, ***1***, ***2***, ***32***, ***64***

cluster: Es un conjunto de servidores independientes pero interconectados. · ***24***

D

DAC: Control de Acceso Discrecional · ***1***, ***10***, ***11***, ***13***

DSD: Restricción dinámica de separación de responsabilidades · ***12***

E

e-business*:** Se refiere a todas las transacciones, negocios y operaciones comerciales que se realizan usando las tecnologías de la información. · ***8

F

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. · ***22, ***23***, ***29***, ***30***, ***47***, ***64***, ***67***

G

GRASP: Patrones de asignación de responsabilidad · ***48***, ***63***

H

HTML: Lenguaje de Marcado de Hipertexto · ***21***, ***22***, ***27***

I

IT: Tecnologías de la información · ***6***

J

JQuery: Es una librería basada en JavaScript para acceder a los objetos del DOM de un modo simplificado. · ***64***

L

LDAP: Protocolo Ligero de Acceso a Directorios · ***7***, ***33***

M

MAC: Control de Acceso Obligatorio · ***1***, ***11***, ***13***

MSF: Microsoft Solution Framework · ***17***, ***18***, ***36***, ***64***

MVC: Arquitectura modelo-vista-controlador · ***45***, ***63***, ***64***

P

post-back*:** Un mecanismo introducido en el Microsoft ASP.NET para permitir la comunicación entre cliente y servidor. · ***30

R

RBAC: Control de acceso basado en roles · ***I***, ***VI***, ***1***, ***2***, ***3***, ***4***, ***12***, ***13***, ***14***, ***15***, ***33***, ***34***, ***35***, ***39***, ***64***

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. · ***VI***, ***21***, ***37***, ***53***, ***85***, ***86***, ***87***, ***89***, ***90***, ***91***

rol: Un rol es una función de trabajo en el contexto de una organización con una semántica asociada sobre la autoridad y la responsabilidad conferida en el usuario asignado a la función. · ***VI***, ***1***, ***12***, ***14***, ***15***, ***36***, ***37***, ***38***, ***52***, ***62***, ***74***, ***75***, ***76***, ***77***, ***78***, ***79***, ***81***, ***82***, ***83***, ***84***, ***85***, ***99***, ***100***, ***102***, ***103***, ***105***, ***107***

S

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. · ***22***

SSD: Restricción estática de separación de responsabilidades · ***12***

SSO: Inicio único de sesión · ***8***, ***33***

T

tablespaces*:** Es una unidad lógica de almacenamiento dentro de una base de datos ORACLE. · ***25

TBAC: Control de acceso basado en tareas · ***11***, ***13***

U

UCI: Universidad de las Ciencias Informáticas · ***VI***, ***1***, ***32***

UML: Lenguaje unificado de modelado · ***18***, ***19***

V

viewstate*:** Mantener el estado de los controles de una misma página en una petición al servidor. · ***30

X

XML: Lenguaje de marcado extensible · ***19***