Universidad de las Ciencias Informáticas

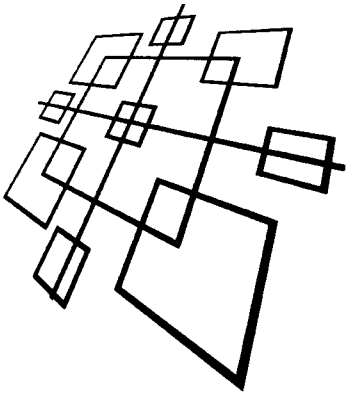
Dirección de Informatización

Akademos
Gestión Académica

SUBSISTEMA GESTIÓN DE SEGURIDAD

Trabajo de diploma para optar por el título de:

Ingeniero Informático



Autor: Yandy Valdés Cabrera
Tutor: Ing. Aracelys García Armenteros

Ciudad de la Habana, Diciembre 2005

Agradecimientos

A mis padres, por el cariño y el amor que me han brindado toda la vida, por ser mis principales educadores, por ser el motivo de inspiración de llevar todo este sueño adelante y poder siempre contar con ellos.

A toda mi familia, por brindarme su amor y apoyo incondicional.

A mis hermanos, que tanto han luchado porque este sueño sea realizado y sin su apoyo nada de esto fuera realidad, por ser mis ejemplos y guías ante la vida.

A Daimi, por siempre estar a mi lado y brindarme su ayuda en todo momento.

A mis compañeros de aula, por haber tenido la posibilidad de compartir estos años con ellos y haber conocido personas maravillosas que realmente se merecen todo mi apoyo.

A mis profesores, de toda la vida, por siempre educarme.

A Emil, por ser realmente una persona excepcional, amigo y poder contar con su apoyo.

A mi tutor, por su consagración y apoyo para que este trabajo resultara lo mejor posible.

A Pepe, Frank y Andry, por convertirse en mis hermanos y estar siempre dispuestos a ayudarme, por ser parte de esto.

A nuestra revolución, que nos educa y nos prepara ante la vida, con el simple propósito de convertirnos en mejores personas cada día.

Resumen

La Universidad de las Ciencias Informáticas (UCI) posee una infraestructura tecnológica, que permite la creación de una ciudad digital, donde su principal objetivo es la automatización de los procesos involucrados en el quehacer cotidiano. El correcto funcionamiento de la gestión académica es de vital importancia para un desempeño triunfante del centro; es por tal motivo que surge la necesidad de desarrollar un Sistema de Gestión Académica con el objetivo de automatizar todo el proceso docente y dentro de éste una herramienta que garantice la seguridad de la información manejada. El presente trabajo tiene como objetivo el diseño e implantación del Módulo de Seguridad del Sistema de Gestión Académica "Akademos", proporcionándoles mayor potencialidad al producto y confianza a los clientes.

La puesta en vigor de este sistema permitirá dar solución a problemas actuales existentes en el centro, lo que se traduce en mantenimiento de la confidencialidad, integridad y disponibilidad de la información manejada, reducción de costes asociados a pérdidas de información por incidentes de seguridad y evita la necesidad de implementar seguridad en cada uno del resto de los módulos del sistema al brindarle sus servicios.

Para su desarrollo se siguieron los pasos que proponen el Proceso Unificado del Software. Está implementado sobre la plataforma Microsoft .NET, específicamente en el lenguaje Visual C Sharp.NET, como gestor de base de datos se utilizó SQL Server. Actualmente se encuentra en explotación en la Universidad un conjunto de funcionalidades de este módulo.

Índice

Introducción

En la actualidad con el alto nivel de crecimiento que ha experimentado Internet y sus potencialidades, es difícil la creación de aplicaciones aisladas, de aplicaciones que no interactúen con otras. Por lo que estas aplicaciones en algún momento pueden revelar información confidencial, poniendo en gran peligro a sus clientes y por lo tanto a las instituciones a las que pertenecen, significando en el peor de los casos enormes pérdidas económicas. Es por esta y otras muchas causas más que desde el mismo inicio del surgimiento de la Informática y el desarrollo de aplicaciones, se ha visto la necesidad de mantener la seguridad de los sistemas desarrollados, siguiendo este objetivo se han implementado diferentes métodos y vías que garanticen la confidencialidad e integridad de la información por su vital importancia.

En la Universidad de las Ciencias Informáticas (UCI), como centro rector de la implementación de nuevas tecnologías se viene desarrollando un conjunto de proyectos con el objetivo de informatizar los procesos que a diario son parte de la vida universitaria. Entre ellos cuenta la automatización de la gestión académica con un alto nivel de importancia, debido a la gran cantidad de matrícula que forma parte del centro, con perspectivas de ampliación en cursos venideros y con ello también una inmensa cantidad de profesores y personal de secretaría, tornando engorroso la gestión académica de forma manual.

Situación Problémica y problema a resolver

En la UCI se ha venido utilizando un grupo de sistemas con el objetivo de automatizar los procesos que intervienen en la gestión académica debido a la creciente necesidad de resolver los problemas existentes en esta área, por ser una Universidad con perspectivas diferente a la tradicional, orientada al mundo de la informática. Los cuales presentan ineficiencia para garantizar seguridad en la información manejada, poca o ninguna integración con otros sistemas desarrollados por el centro, lejos de ayudar a aliviar la labor del personal de secretaria, de directivos, docentes y estudiantes hacen el proceso más engorroso, presentando como principal dificultad no poder adaptarse a flujos de gestión

específicos ni a cambios que ocurran, una rígida configuración de las políticas de seguridad implementadas, careciendo en conjunto de los principios que garantizan la seguridad de una aplicación ["Autenticidad, Confidencialidad, Integridad, No repudio"], comprometiendo en algún momento toda o parcialmente la información manejada por el sistema, significando esto un gran problema debido al carácter confidencial, legal y la necesidad de mantener la integridad de dicha información para el centro universitario. Por lo antes expuesto, se procede a desarrollar el subsistema "Gestión de Seguridad", que forma parte del Sistema de Gestión Académica "Akademos".

Para ello se propone realizar la investigación a partir del siguiente **problema:**

¿Cómo implementar los niveles de seguridad del Sistema Automatizado para la Gestión Académica "Akademos"?

Beneficios esperados:

Permitir la gestión automatizada de los elementos que intervienen en la labor académica de la Universidad, garantizado un alto nivel de seguridad, confidencialidad e integridad de los datos manejados por el sistema, integración con otros sistemas existentes en la UCI y un mayor control sobre las acciones de los usuarios en el sistema.

Objeto de Estudio

- Elementos que intervienen en la gestión académica.
- Tendencias actuales para el desarrollo de aplicaciones Web.
- Elementos de seguridad en aplicaciones Web.
- Tecnologías Web para el desarrollo de aplicaciones cliente-servidor.
- Sistemas Gestores de Bases de Datos.

Campo de acción

- La gestión académica en la UCI.
- Mecanismos de seguridad y protección.

- La plataforma Microsoft.Net, en específico, la tecnología ASP.Net para el desarrollo de aplicaciones Web.
- Microsoft SQL Server como sistema gestor de Base de Datos.

Objetivo General

Desarrollar un sistema automatizado que permita la gestión del proceso académico de forma confiable y segura.

Dada la problemática que se ha planteado tendrá como **objetivos específicos**, en su etapa de desarrollo:

- Implementar varios niveles de acceso para restringir las acciones que puedan realizar los usuarios.
- Garantizar integridad, autenticidad y confidencialidad de los datos.
- Controlar las acciones de los usuarios del sistema. Para de esta forma poder detectar comportamientos indebidos.
- Monitorear las incidencias generadas en tiempo real.
- Permitir el intercambio de información entre la aplicación y los servicios Web de forma segura.
- Modelar la solución que se propone, utilizando el Proceso Unificado de Desarrollo de Software.

Dentro de las **tareas** que se proponen para dar solución a los objetivos planteados están:

- Estudio del sistema actual implantado en nuestra universidad, de sus funcionalidades, deficiencias y de las nuevas características a implantar.
- Desarrollar fundamentación teórica del objeto de estudio.
- Analizar y discutir la arquitectura de la aplicación.
- Modelación e implementación del módulo, basado en el Proceso Unificado de Desarrollo de Software.
- Implantación oficial del módulo.

Estructuración del contenido

Capítulo 1: *Fundamentación teórica*: En este capítulo se explican las metodologías, los lenguajes usados, las herramientas utilizadas para el desarrollo de la aplicación y el estado del arte de la seguridad en las aplicaciones informáticas.

Capítulo 2: *Descripción de la solución propuesta*: Modelado del negocio del sistema de Gestión de Seguridad a través de un modelo de dominio. Descripción de la información que se maneja, de la aplicación que se propone así como de todos los requisitos funcionales y no funcionales que esta debe cumplir.

Capitulo 3: *Construcción de la solución propuesta*: Incluye la definición del modelo de análisis del sistema y de este el modelo de clases. Describe los diagramas de secuencia del modelo del diseño para cada realización de los casos de uso. Muestra el diagrama de clases del diseño y la descripción de cada una de las clases.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En la actualidad con el rápido incremento de las tecnologías informáticas, hasta las más pequeñas empresas están conectadas a la gran red de redes, tanto para brindar servicios como para consumirlos. Los sistemas operativos, hasta nuestros días, no cuentan con una seguridad lo suficiente fuerte como para no exponer a los usuarios a diversos peligros. Con ello la Web cambia día a día nuestras vidas y la seguridad no es una excepción a este postulado. El estar conectado permanentemente, además de brindar innumerables beneficios, reporta un gran peligro. Los ataques son un hecho constante que no dejamos de recibir a diario. A principios de la década de los noventa, afrontamos una nueva encrucijada en términos de seguridad informática. Hace algunos años atrás, muchas empresas tomaban el camino fácil (poca o ninguna protección de la seguridad) porque los riesgos eran menores y las consecuencias menos devastadoras. Esta situación ha cambiado totalmente, ya no existe. Según John Shwarz presidente y gerente de operaciones de Symantec, "Las amenazas cada vez son más frecuentes y complejas, una violación a la seguridad podría convertirse en algo devastador para cualquier institución, puesto que pone en peligro la reputación de la misma, la confianza de los clientes y toda la confidencialidad de la información." [12]. Así como el panorama de la seguridad ha evolucionado durante los últimos años, la seguridad de los sistemas informáticos ha dejado de ser una preocupación de la tecnología para convertirse en un asunto personal de los desarrolladores de software.

1.2 El proceso unificado de desarrollo de software

En la actualidad el software lleva a la construcción de sistemas mucho más complejos y grandes. Debido principalmente al auge de las computadoras y a su vez el aumento del rigor del usuario. Esto se ha visto también afectado por el rápido crecimiento en el uso de Internet para el intercambio de todo tipo de información. James Rumbaugh, Grady Booch e Ivar Jacobson, autores de "El proceso unificado

de desarrollo de software", opinan que "El problema del software se reduce a la dificultad que afrontan los desarrolladores para coordinar las múltiples cadenas de trabajo de un gran proyecto de software" [7]. Para desarrollar un software se necesita una forma coordinada de trabajo, un proceso que integre las múltiples facetas del desarrollo. Se necesita un método común, un proceso que:

- Proporcione una guía para ordenar las actividades de un equipo.
- Dirija las tareas de cada desarrollador por separado y del equipo como un todo.
- Especifique los artefactos que deben desarrollarse.
- Ofrezca criterios para el control y la medición de los productos y actividades de proyectos.

El proceso unificado de desarrollo, RUP, es el resultado de la evolución e integración de diferentes metodologías de desarrollo de software. RUP permite sacar el máximo provecho de los conceptos asociados a la orientación a objetos y al modelado visual. Esto permite a los grupos de desarrollo producir aplicaciones informáticas más robustas y flexibles que se adaptan a las necesidades de los usuarios. La correcta aplicación de RUP permite reducir los tiempos de desarrollo, aumentar la calidad de las aplicaciones y disminuir los costes de mantenimiento. "Está basado en componentes, lo cuál quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas" [7]. El RUP es un proceso de desarrollo de software que contiene un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software de forma eficiente (Figura1). Es el resultado de la experiencia de más de 30 años de trabajo y los autores [James Rumbaugh, Grady Booch e Ivar Jacobson] confirman que es la solución al problema del software.

Esta metodología utiliza el Lenguaje Unificado de Modelado (UML, Unified Modeling Language) para preparar todos los esquemas de un sistema de software. UML es una parte esencial del Proceso Unificado, fueron desarrollados paralelamente por las mismas personas, haciendo que su integración sea un éxito.

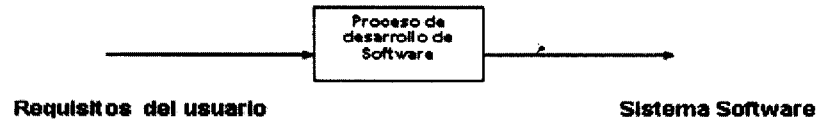


Figura 1: Proceso de Desarrollo de Software.

1.2.1 Características del Proceso Unificado de Software

Los aspectos definitorios y a la vez que lo convierten en único al Proceso Unificado, se resumen en tres fases: dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

- **Dirigido por casos de uso:** La razón de ser de un sistema es brindar servicios a los usuarios, RUP define caso de uso como el conjunto de acciones que debe realizar un sistema para dar un resultado de valor a un determinado usuario y los utiliza tanto para especificar los requisitos funcionales del sistema, como para guiar todos los demás pasos de su desarrollo, dígase diseño, implementación y prueba.

- **Estar centrado en la arquitectura:** La arquitectura es una vista del diseño completo con las características más importantes. Esta no solo incluye las necesidades de los usuarios e inversores, sino también otros aspectos técnicos como el hardware, sistema operativo, sistema de gestión de base de datos, protocolos de red, con los que debe coexistir el sistema. La arquitectura representa la forma del sistema, la cual va madurando en su interacción con los casos de uso hasta llegar a un equilibrio entre funcionalidad y características técnicas.

- **Ser iterativo e incremental:** El alto nivel de complejidad de los sistemas actuales, hace que sea factible dividir el proceso de desarrollo en varios mini-proyectos. Cada uno de estos se les denomina iteración y pueden o no representar un incremento en el grado de terminación del producto completo. En cada iteración los desarrolladores seleccionan un grupo de casos de uso, los cuales se diseñan, implementan y prueban. La

planificación de iteraciones hace que se reduzcan los riesgos de los costes de un solo incremento, no sacar al mercado un producto en el tiempo previsto, mantener la motivación del equipo pues puede ver avances claros a corto plazo y que el desarrollo pueda adaptarse a los cambios en los requisitos.

1.3 Rational Rose

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML [Booch, Rumbaugh y Jacobson] y que soporta de forma completa la especificación del UML. Esta herramienta propone la utilización de cuatro tipos de modelos para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. A continuación los cuatro tipos de modelos:

- **Desarrollo Iterativo:** Utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.
- **Generador de Código:** Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.
- **Ingeniería Inversa:** Proporciona mecanismos para realizar la denominada Ingeniería Inversa, a partir del código de un programa, se puede obtener su diseño.
- **Trabajo en Grupo:** Permite varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

1.4 Seguridad

1.4.1 Elementos de seguridad

El proceso por el cual se diseña la seguridad tiene carácter cíclico. La seguridad de una aplicación depende, en gran sentido, de la atención que presten los programadores y administradores. No solo durante la fase de diseño sino también durante el período que dure la aplicación.

Las medidas de seguridad implementadas sin tener en cuenta las amenazas reales que puedan afectar a la aplicación sólo dan una sensación falsa de seguridad. "Las tecnologías y los métodos que elija dependerán de las amenazas que determine que es posible que se produzcan. El grado de implementación de estos métodos y tecnologías dependerá del nivel de riesgo que pueda tolerar la aplicación, del nivel de riesgo que puedan tolerar los datos de la aplicación y del coste. Tanto el coste como el esfuerzo deberían guardar correlación con el valor de lo que se desea proteger." [9]

Dado que van surgiendo nuevas amenazas casi a diario, las aplicaciones deben examinarse constantemente con el fin de identificar posibles errores de seguridad. No obstante, el diseño inicial de la aplicación determina la frecuencia con la que se pueden producir dichos errores.

Se consideran una amenaza para la seguridad cualquier incidente potencial, malintencionado o de otra índole, que pueda tener un efecto no deseado en la aplicación. "Los puntos vulnerables de la aplicación o del sistema operativo hacen posible la existencia de estas amenazas. Un ataque a la aplicación se puede definir como una acción emprendida por un intruso malintencionado para aprovechar determinados puntos vulnerables con el fin de materializar la amenaza" [9]. El riesgo que esto comporta está constituido por los daños posibles que el ataque puede ocasionar en la aplicación o incluso en la empresa.

1.4.2 Seguridad en aplicaciones Web

La seguridad de los sitios Web es una cuestión de importancia fundamental, además de compleja, para los programadores de sitios Web. La protección de un sitio requiere la elaboración cuidadosa de un plan; por consiguiente, los programadores y administradores de sitios Web deben comprender perfectamente las opciones para proteger los sitios.

En algún momento, todas las aplicaciones Web sufren algún intento de pirateo. Siempre que usuarios desconocidos puedan alcanzar la aplicación Web, hay muchas probabilidades de que alguien intentará hacer algo no autorizado o incluso malintencionado con ella. En la mayoría de los servidores que permiten el acceso al público de Internet se intenta, al menos cuatro o cinco veces al día, encontrar puntos vulnerables. Por lo tanto, sin importar qué tipo de aplicación Web se desee crear, ésta deberá incorporar elementos de seguridad. Se debería equiparar la preocupación por la seguridad con la concienciación que existe sobre la necesidad de crear copias de seguridad. Aunque se piense que nunca se tendrá ningún problema, siempre resulta más seguro asumir que puede suceder y adoptar las precauciones necesarias.

La primera lección acerca de la seguridad de las aplicaciones Web, es que no existe el método de seguridad perfecto. Incluso en los sistemas seguros, los usuarios malintencionados pueden utilizar herramientas sofisticadas para intentar introducirse en el sistema. Afortunadamente, la mayoría de los ataques provienen de usuarios relativamente poco sofisticados, que sólo buscan sitios en los que resulte fácil penetrar. El objetivo inicial debería ser proteger el sitio de los ataques comunes y redundantes, que suponen la mayoría de los ataques.

Una fase importante en el proceso de generación de aplicaciones seguras consiste en ser capaz de anticipar las amenazas que se puedan sufrir. Microsoft ha desarrollado un modo de categorizar las amenazas que se conoce como STRIDE (Spoofing-Tampering-Repudiation-Information disclosure-Denial of service-Elevation of privilege, suplantación-manipulación-repudio-revelación de información-denegación de servicio-concesión de privilegio):

- **Burlar la identidad:** Esto sucede cuando el método de autenticación se ve afectado. Un atacante ha pirateado o puede reproducir la secuencia de autenticación de un usuario. Como medida se recomienda que se utilice métodos de autenticación más rigurosos, como la autenticación Kerberos y los certificados de autenticación de cliente, para evitar que se burle la identidad. El almacenamiento de la información de autenticación en cookies y los esquemas de autenticación de diseño propio son los métodos menos fiables y más fáciles de frustrar.

- **Manipular datos:** La destrucción o alteración deliberada de datos. Los datos se ven amenazados tanto si están en tránsito (física o electrónicamente) como si están almacenados. Los datos confidenciales deberían cifrarse mediante algoritmos hash y firmas digitales dándole una mayor fortaleza.

- **Repudiabilidad:** Es la noción de negar que se ha producido una acción. En este caso deben registrarse las acciones que se desea que no sean realizadas por usuarios no autorizados. Dicha no repudiabilidad también puede obtenerse mediante el uso de firmas digitales.

- **Revelación de información:** La gravedad de la revelación de información depende de la confidencialidad de la información revelada.

- **Denegación de servicio:** La disponibilidad y la confiabilidad de las aplicaciones se ven directamente afectadas por los ataques de denegación de servicio (DoS), que son una forma de sabotaje que hace que las aplicaciones no estén disponibles para los usuarios autorizados. Los ataques DoS se producen cuando un sistema está desbordado por el tráfico hasta el punto de no poder procesar las solicitudes de servicio legítimas. La mayoría de las medias que se pueden tomar para defenderse de los ataques DoS consisten en filtrar los paquetes mediante un servidor de seguridad para separar los paquetes legítimos de los malintencionados. Además, se

puede regular el límite de ancho de banda y de los recursos para evitar que un sitio Web sobrecargado bloquee todo un servidor.

- **Concesión de privilegio:** Se produce una concesión de privilegio cuando un usuario obtiene acceso privilegiado a partes de la aplicación o a datos a los que el usuario normalmente no tiene acceso. Lo más común es aprovecharse de un punto vulnerable como es un desbordamiento de búffer para ejecutar código malintencionado con el mismo contexto de seguridad que el proceso de la aplicación.

Implementar tecnología de seguridad es sólo parte de la solución. Otra parte consiste en la vigilancia. Aunque el sistema cuente con numerosos elementos de seguridad, es preciso vigilarlo de cerca, supervisando los registros de eventos del sistema, buscando los intentos repetidos de iniciar una sesión en el sistema o comprobando si existe un alto número de solicitudes en el servidor Web.

1.5 SQL Server

SQL Server es un completo Sistema de Gestión de Base de Datos Relacional con todas las posibilidades que tienen estas aplicaciones; un servidor disponible en varias ediciones en el que es posible utilizar un lenguaje, Transact-SQL, para programar procedimientos almacenados y desencadenadores, aparte de poder definirse tablas, índices, vistas, etc.

Microsoft SQL Server 2000

MS SQL Server 2000, potente motor de bases de datos de alto rendimiento, capaz de integrarse con herramientas de desarrollo como Visual Studio .NET, incorporando un modelo de objetos totalmente programable (SQL-DMO) con el cual se pueden desarrollar aplicaciones que usen componentes de SQL Server.

Microsoft SQL Server 2000 es la última versión del sistema de gestión de bases de datos relacionales (SGBDR), cliente servidor, basado en un lenguaje de consulta estructurada (SQL, Structured Query Language), que aprovecha la sólida base establecida por su predecesor SQL Server 6.5. y 7. Como la mejor base de datos

para Windows NT, MS SQL Server 2000 es el SGBDR ideal para un amplio campo de clientes corporativos y fabricantes independientes de software. Las necesidades y requisitos del cliente, han dado lugar a innovaciones significativas en SQL Server 2000, entre las que se incluyen la facilidad de uso, escalabilidad, fiabilidad, y almacenamiento de datos [10]. Proporcionando soporte para un conjunto de características que aportan las siguientes ventajas:

- **Facilidad de instalación, distribución y utilización:** SQL Server incluye un conjunto de herramientas administrativas y de desarrollo que mejoran la capacidad para instalar, distribuir, administrar y utilizar SQL Server entre varios sitios.

- **Escalabilidad:** Puede utilizarse el mismo motor de base de datos a través de plataformas que van desde equipos portátiles que ejecutan Microsoft Windows® 9X, 200X o hasta grandes servidores con varios procesadores que ejecutan Microsoft Windows NT®, Enterprise Edition.

- **Almacenamiento de datos:** SQL Server incluye herramientas para extraer y analizar datos resumidos para el proceso analítico en línea (OLAP, Online Analytical Processing). SQL Server incluye también herramientas para diseñar gráficamente las bases de datos y analizar los datos mediante preguntas en lenguaje normal.

- **Integración del sistema con otro software de servidor:** SQL Server se integra con el correo electrónico, Internet y Windows.

1.6 Microsoft .NET

Microsoft .NET es un nueva plataforma sencilla y potente que simplifica el desarrollo de software. Proporciona un ambiente de alta productividad, basado en estándares y multilingüe para integrar versiones existentes con aplicaciones y servicios de la siguiente generación, así como la agilidad para resolver los retos de implementación

y operación de las aplicaciones a escala de Internet [2]. Está diseñada para cumplir con los siguientes objetivos:

- Proveer a los desarrolladores de un entorno consistente de programación orientado a objetos, en el cual los objetos se almacenan y se ejecutan localmente o de forma remota.
- Proveer un entorno de ejecución de código en el que se reduzcan los riesgos asociados al despliegue y al control de versiones.
- Brindar un entorno de ejecución seguro, incluyendo al código creado por partes desconocidas.
- Hacer que la experiencia de los desarrolladores sea aplicable en diversos tipos de aplicaciones como las basadas en Windows o las aplicaciones Web.[13]

.NET Framework es el nuevo modelo de programación de Microsoft para desarrollar, implementar y ejecutar servicios Web XML y todos los tipos de aplicaciones: de escritorio, para dispositivos móviles o basadas en Web (Figura 2). Microsoft perseguía dos objetivos principales en la creación de la plataforma .NET. El primero era mejorar el desarrollo de Microsoft Windows haciéndolo más orientado a objetos. El segundo era ofrecer un marco de trabajo moderno de tercera generación para crear e implementar servicios Web XML. Con .NET Framework, Microsoft mejora enormemente la productividad del programador, la facilidad de desarrollo y la ejecución de aplicaciones confiables. .NET Framework incorpora servicios Web XML. Estos integran aplicaciones y componentes poco complementados diseñados para el heterogéneo entorno informático actual mediante la comunicación con protocolos de Internet estándar como SOAP (Protocolo simple de acceso a objetos), WSDL (Lenguaje de descripción de servicios Web) y UDDI (Integración, descubrimiento y descripción universal).

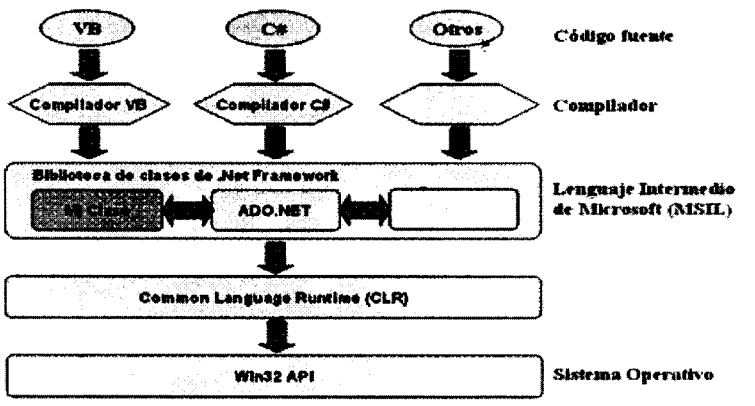


Figura 2: Arquitectura .NET Framework.

.NET Framework consta de tres áreas principales. La primera es Common Language Runtime (CRL), que asume la responsabilidad de ejecutar la aplicación. CRL garantiza que se cumplan todas las dependencias de la aplicación, administra la memoria y controla cuestiones como la seguridad y la integración de lenguajes. Common Language Runtime proporciona muchos servicios que simplifican el desarrollo del código y la implementación de la aplicación a la vez que aumenta la fiabilidad de la aplicación. La mayor parte de este trabajo se controla de manera transparente, simplificando las tareas de los desarrolladores y administradores.

La segunda área es la de las clases principales unificadas. Estas clases proporcionan todos los recursos que requiere un desarrollador para generar una aplicación moderna, incluida la compatibilidad con XML, las conexiones de red y el acceso a datos (Figura 3). Tener estas clases unificadas significa que un desarrollador que desarrolle cualquier tipo de aplicación, basada en Windows o en Web, utiliza las mismas clases. Esta coherencia aumenta la productividad del desarrollador y la reutilización de código.

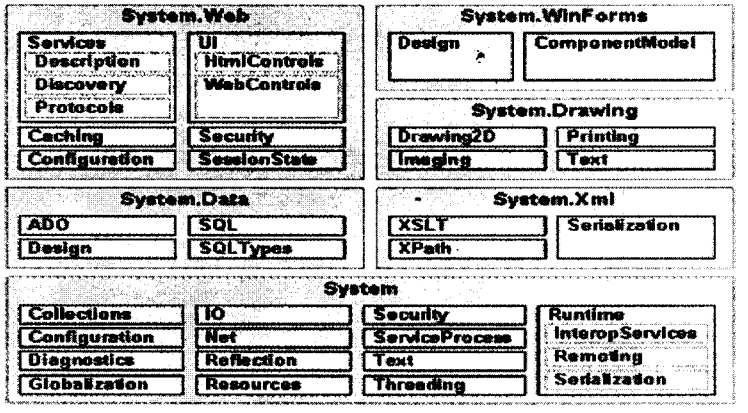


Figura 3: Biblioteca de Clases .NET Framework.

La tercera y última área es la de las clases de presentación, que incluyen ASP.NET para el desarrollo de aplicaciones Web, así como servicios Web XML y Windows Forms para el desarrollo de aplicaciones basadas en Windows o de "cliente inteligente".

1.6.1 Arquitectura propuesta por Microsoft para aplicaciones .Net

El diseño de aplicaciones distribuidas basadas en un modelo de componentes por capas ha sido de gran aceptación por los arquitectos de sistemas. Este modelo generalmente está compuesto por una capa de presentación, de negocio y de datos, en cada una de las cuales se agrupan componentes que realizan tipos de funciones similares. Un componente específico puede utilizar las funcionalidades proporcionadas por los componentes de su capa y de capas inferiores.

El análisis de la mayoría de las soluciones empresariales, basadas en modelos de componentes por capas, demuestra que existen varios tipos de componentes habituales, los cuales se muestran en la figura 4.

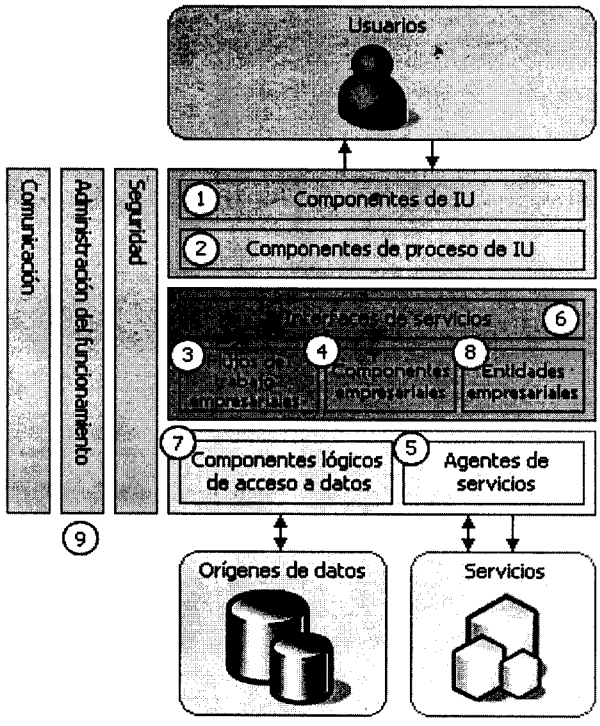


Figura 4: Tipos de Componentes.

En el Subsistema de Gestión de Seguridad solo estarán presentes algunos de estos tipos de componentes, los cuales son descritos a continuación:

Componentes de interfaz de usuario (IU)

La mayoría de las aplicaciones necesitan ofrecer al usuario un modo de interactuar con ella. Las interfaces de usuario se implementan utilizando formularios Windows Forms, páginas Microsoft ASP.Net, controles u otro tipo de tecnología, que permita procesar y dar formato a los datos de los usuarios.

Los componentes de la interfaz de usuario deben mostrar datos a este, obtener y validar los datos procedentes del mismo e interpretar las acciones que indican la operación que desea realizar con los datos. Asimismo, la interfaz debe filtrar las acciones disponibles, con el fin de permitir al usuario realizar sólo aquellas operaciones que le sean necesarias, en un momento determinado.

Interfaces de servicios

Para exponer lógica empresarial como un servicio, es necesario crear interfaces de servicios, que admitan los contratos de comunicación (comunicación basada en mensajes, formatos, protocolos, seguridad y excepciones, entre otros) que requieren los clientes. Éstas también se denominan fachadas empresariales.

Una interfaz de servicios es una entidad de software implementada normalmente como una fachada que controla los servicios de asignación y transformación, para permitir la comunicación con un servicio aplicando un proceso y una política de comunicación. Una interfaz de servicios expone métodos, a los que se puede llamar de forma individual o en una secuencia específica, para formar una conversación que implemente una tarea empresarial.

Componentes lógicos de acceso a datos

La mayoría de las aplicaciones y servicios necesitan obtener acceso a un almacén de datos, en un momento determinado del proceso empresarial. Por tanto, es razonable abstraer la lógica necesaria para obtener acceso a los datos, en una capa independiente de componentes lógicos de acceso a datos, ya que de este modo se centraliza esta funcionalidad de acceso a datos, facilitando la configuración y el mantenimiento de la misma.

Los componentes lógicos de acceso a datos, suelen implementar un patrón de diseño sin estado, que separa el procesamiento empresarial de la lógica de acceso a datos. [1]

1.6.2 Microsoft Visual C Sharp .NET

Microsoft Visual C Sharp .NET es el nuevo lenguaje de propósito general diseñado por Microsoft para la plataforma .NET. Entre sus principales creadores se encuentra Anders Hejlsberg, diseñador del lenguaje Turbo Pascal y la herramienta RAD (desarrollo de aplicaciones rápida) Delphi. Según José Antonio González Seco "C Sharp ha sido diseñado específicamente para ser utilizado sobre la plataforma .NET, aunque esta soporta código en varios lenguajes. Microsoft ha escrito la mayor

parte de la BCL (Librería de Clase Base) usándolo, por lo que su compilador es el más depurado y optimizado de los utilizados en el .NET Framework. Es por esta razón que C Sharp es el lenguaje nativo de .NET. Construido especialmente para adaptarse de manera natural al Framework y aprovechar al máximo todas sus características."[6]

Su sintaxis y estructura son muy similares a la de C++, ya que la intención de Microsoft es migrar los códigos escritos en estos lenguajes a C Sharp. Su sencillez y alto nivel de productividad suelen compararse con los de Visual Basic. En un comienzo la idea fue utilizar el lenguaje Java con estos fines, pero por problemas con la compañía creadora Sun, Microsoft tuvo que desarrollar su propio lenguaje añadiéndole las principales características que le daban fortaleza a Java y otras modificaciones que tenían pensado para hacerlo más orientado a componentes. Según un artículo encontrado en Internet [5] "C Sharp, al igual que C y C++, permite programar fácilmente a bajo nivel. Gracias a esto, acceder a las características avanzadas de la plataforma sobre la que trabajamos, crear código muy eficiente en aquellos puntos de la aplicación que son críticos y acceder a las interfaces de programación de aplicaciones (APIs) existentes es perfectamente posible."

En este sentido, Miguel de Icaza y su empresa Ximian (líder en tecnología "open source" para aplicaciones y servicios bajo Linux y Unix) ha lanzado el proyecto Mono como esfuerzo para crear una implementación "open source" del framework de desarrollo de .NET. Por su parte Fabián Seoane plantea "Su objetivo es permitir que los desarrolladores de GNU/Linux desarrollen aplicaciones multiplataforma basadas en .NET" [11]. El proyecto Mono implementará varias tecnologías desarrolladas por Microsoft que han sido enviadas al ECMA (Asociación Europea de Fabricantes de Informática) para su estandarización. Los frutos del mismo por ahora han sido los siguientes:

- Un compilador JIT compatible con la tecnología .NET capaz de correr en arquitecturas x86 bajo Linux y Windows.
- Un interprete compatible con .NET para x86, PPC, Sparc y Strong/Arm, para sistemas Linux, Windows, FreeBSD y Solaris.
- Un compilador de C# capaz de compilarse a sí mismo.

- Montones de clases .NET.
- Clases adicionales y Gtk# con el cual és ya posible crear aplicaciones simples.
- Un compilador de Logo para Mono.
- Las bases de ejecución de ASP.NET y ADO.NET.

1.6.3 ASP .NET

Con la aparición de la plataforma .NET se ha iniciado una nueva era en el campo de la programación de aplicaciones que conducirá la Internet de nueva generación. ASP.NET, una parte de la plataforma .NET de Microsoft, es una estructura de programación revolucionaria que permite el desarrollo de aplicaciones Web dirigidas a corporaciones. Es un marco de trabajo de programación generado por un Common Language Runtime que puede utilizarse en un servidor para generar eficaces aplicaciones Web (Figura 5).

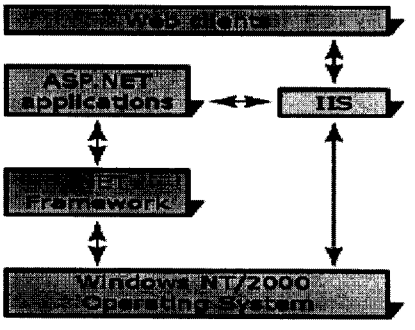


Figura 5: Arquitectura ASP.NET.

Constituye la forma más rápida y escalable de desarrollar, implementar y ejecutar aplicaciones Web en cualquier navegador o dispositivo. ASP.NET facilita el desarrollo de aplicaciones si lo comparamos con el modelo ASP clásico, por lo que la productividad de los programadores mejorará considerablemente. Esta plataforma permite dotar de funciones adicionales a una aplicación Web y escribir una menor cantidad de código, entre otras características. Además de las aplicaciones Web habituales, ASP.NET permite crear otros tipos con lo que puede

ampliar el alcance de las mismas a nuevos clientes y empresas. Por ejemplo, los servicios Web XML permiten compartir datos por Internet independientemente del sistema operativo y del lenguaje de programación, lo que amplía considerablemente el alcance de las aplicaciones. Al mismo tiempo, ASP.NET cuenta con controles móviles para que las aplicaciones puedan llegar a un mayor número de dispositivos Web móviles [3]. Entre sus principales ventajas frente a los modelos de programación Web anteriores se encuentran:

- **Mejor rendimiento:** Es un código de Common Language Runtime compilado que se ejecuta en el servidor. A diferencia de sus predecesores, ASP.NET puede aprovechar las ventajas del enlace anticipado, la compilación just-in-time, la optimización nativa y los servicios de caché desde el primer momento. Esto supone un incremento espectacular del rendimiento antes de siquiera escribir una línea de código.

- **Eficacia y flexibilidad:** Debido a que este se basa en Common Language Runtime, la eficacia y la flexibilidad de toda esta plataforma también esta disponible para la Web. Se comparte la misma biblioteca de clases, la mensajería y soluciones de acceso a datos de forma uniforme. ASP.NET es también independiente del lenguaje de programación utilizado.

- **Simplicidad:** Facilita la realización de tareas comunes. Su marco de trabajo permite generar interfaces de usuario, que separan claramente la lógica de aplicación del código de presentación.

- **Facilidad de uso:** Emplea un sistema de configuración jerárquico, basado en texto, que simplifica la aplicación de la configuración al entorno de servidor y las aplicaciones Web. Debido a que la información de configuración se almacena como texto sin formato, se puede aplicar la nueva configuración sin la ayuda de herramientas de administración local.

- **Escalabilidad y disponibilidad:** ASP.NET se ha diseñado teniendo en cuenta la escalabilidad, con características diseñadas específicamente a

medida, con el fin de mejorar el rendimiento en entornos agrupados y de múltiples procesadores. Además, el motor de tiempo de ejecución de ASP.NET controla y administra los procesos de cerca, por lo que si uno no se comporta adecuadamente (filtraciones, bloqueos), se puede crear un proceso nuevo en su lugar, lo que ayuda a mantener la aplicación disponible constantemente para controlar solicitudes.

- **Seguridad:** Con la autenticación de Windows integrada y la configuración por aplicación, se puede elevar el nivel de seguridad de las aplicaciones.

1.6.4 Servicios Web

Las aplicaciones Web actuales ya no son suficientes. El modelo actual de negocio electrónico no facilita la integración de las aplicaciones de Internet con el resto de software de las empresas. Si las compañías quieren extraer el máximo beneficio de Internet, los sitios Web deben evolucionar. Este es el contexto en el que surgen los Web services.

Los Web Services son componentes software que permiten a los usuarios usar aplicaciones de negocio que comparten datos con otros programas modulares, vía Internet. Son aplicaciones independientes de la plataforma que pueden ser fácilmente publicadas, localizadas e invocadas mediante protocolos Web estándar, como XML, SOAP, UDDI o WSDL. El objetivo final es la creación de un directorio online de Web Services, que pueda ser localizado de un modo sencillo y que tenga una alta fiabilidad (Figura 6).

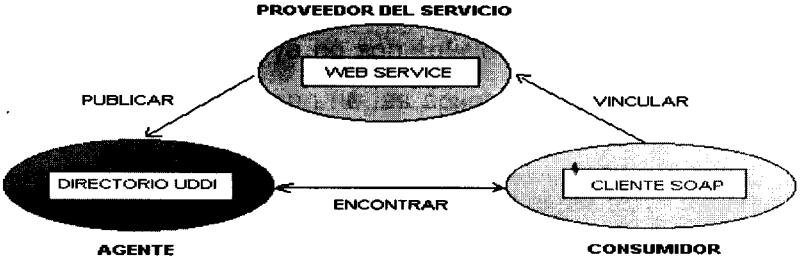


Figura 6: Arquitectura Web Services.

Aunque la idea de la programación modular no es nueva, el éxito de esta tecnología reside en que se basa en estándares conocidos en los que ya se tiene una gran confianza, como el XML. Además, el uso de los Web Services aporta ventajas significativas a las empresas. El principal objetivo que se logra, es la interoperabilidad y la integración. Mediante los Web Services, las empresas pueden compartir servicios software con sus clientes y sus socios de negocio. Esto ayudará a las compañías a escalar sus negocios, reduciendo el coste en desarrollo y mantenimiento de software, y sacando los productos al mercado con mayor rapidez. La integración de aplicaciones hará posible obtener la información demandada en tiempo real, acelerando el proceso de toma de decisiones. La evolución de Internet hacia los Web Services, mejorará los resultados globales de las empresas, reduciendo sus gastos y guiándolas hacia una mejora progresiva de la calidad. [8]

1.6.4.1 Protocolos que usan los Servicios Web

XML (Extensible Markup Language): Es un metalenguaje de marcas que permite definir cómo es la información que se transmite. Esto permite una comunicación de datos entre distintos sistemas. Es la base de los Servicios Web, y a pesar de su sencillez aparente, está transformando completamente la creación y el uso de software. Es la solución a un problema de comunicación entre programas de ordenador, pues la información generalmente queda fuertemente ligada al programa con el cual fue creada, y es así como se pierde mucho tiempo en pasar de un formato de definición a otro. El contenido almacenado en un documento XML se puede transferir fácilmente a través de la red. [4]

SOAP (Single Object Access Protocol): Es un protocolo de mensaje liviano basado en XML, usado para codificar los mensajes de Web Services antes de enviarlos por la red. Los mensajes SOAP son independientes de cualquier sistema operativo y protocolo, y pueden ser transportados usando una variedad de protocolos de Internet, incluyendo HTTP, SMTP y MIME. Permite la comunicación entre programas que corren en diferentes sistemas operativos. [14]

WSDL (Web Service Description Language): Es un lenguaje en formato XML que define las operaciones que proporciona un servicio, desarrollado conjuntamente por Microsoft e IBM.

UDDI (Universal Description Discovery and Integration): Es un directorio universal de Servicios Web basado en XML, que permite publicar, localizar y utilizar servicios Web.

Ventajas de los servicios Web

- Aportan interoperabilidad entre aplicaciones de software, independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- El software como un Servicio, en lugar de un producto empaquetado. Los Servicios Web pueden ser brindados y pagados en línea y accedidos desde cualquier plataforma. Esto contribuye a sistemas más flexibles y estables.
- Los Servicios pueden ser completamente descentralizados y distribuidos sobre Internet y accedidos a través de una gran variedad de dispositivos.
- Especificaciones Universalmente Aceptadas. Los Servicios Web se basan en especificaciones estándar para el intercambio de datos, mensajería, búsqueda, descripción de la interfaz y coordinación de los procesos.
- Integración con sistemas existentes. Mayor agilidad y flexibilidad debido a una mejor integración con los sistemas existentes.

Inconvenientes de los servicios Web

- Para realizar transacciones no pueden compararse en su grado de desarrollo, con los estándares abiertos de computación distribuida como CORBA.
- Su rendimiento es bajo, si se compara con otros modelos de computación distribuida, tales como RMI, CORBA, o DCOM. Es uno de los inconvenientes derivados de adoptar un formato basado en texto, y es que entre los objetivos de XML no se encuentra la concisión, ni la eficacia de procesamiento.
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall, cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

1.7 Conclusiones

En el presente capítulo se realizó un análisis de cómo el panorama de la seguridad ha evolucionado durante los últimos años, dejando de ser una preocupación de la tecnología para convertirse en un asunto personal de los desarrolladores de software, además de cómo se manifiesta en aplicaciones Web y un conjunto de amenazas que define Microsoft como las principales a las que se ven expuesta las aplicaciones Web en la actualidad.

Por otra parte se realizó un análisis de la metodología usada, de la herramienta Case para modelar, de la plataforma .Net, la arquitectura basada en componentes que propone Microsoft para el desarrollo de aplicaciones distribuidas, de la tecnología ASP.Net para el desarrollo de estas aplicaciones así como los Servicios Web, exponiendo de éstos últimos los principales protocolos que usan. También se realizó un estudio sobre los sistemas gestores de base de datos, específicamente Microsoft SQL Server 2000.

Capítulo 2: Descripción de la solución propuesta

2.1 Introducción

En este capítulo, se realiza un análisis del proceso de gestión de seguridad del Sistema de Gestión Académica que se emplea en la Universidad, dándonos una mayor compresión del mismo y sus características. El desarrollo de la aplicación se centra en el Proceso Unificado de Desarrollo de Software, haciendo uso del Lenguaje Unificado de Modelado (UML Unified Model Language) para la modelación de los artefactos. Ha sido de gran utilidad el empleo de la herramienta Case Rational Rose, que asiste al desarrollo de software para una mayor calidad de éste.

Se presenta el modelo de dominio como alternativa al modelo de negocio, en sistemas altamente centrados en tecnologías informáticas, debido a la poca estructuración de los procesos que describen el negocio y con el objetivo de entender el contexto en que se emplaza el sistema y por tanto contribuir a la comprensión de los requisitos del sistema que se desprenden de este contexto.

Posteriormente se describe la solución de software propuesta, realizándose un análisis de la relación costo-beneficio. Además se presenta la propuesta del sistema y se especifican los requerimientos funcionales y no funcionales, se determinan los casos de uso del sistema y los actores que interactuarán con éste, elaborándose una descripción en formato expandido de todos estos.

2.2 Estado actual del negocio

En la Universidad de las Ciencias Informáticas se han utilizado un conjunto de sistemas con el objetivo de automatizar los procesos involucrados en la gestión académica, sistemas los cuales limitan su seguridad a la pertenencia de usuarios a distintos roles definidos, donde todos los miembros tienen por igual los mismos permisos sobre el sistema, imposibilitando la restricción de acciones de un usuario

en particular o un conjunto de ellos, pudiendo esto representar un problema a la hora de tomar decisiones sobre cualquier sucêso ocurrido en el sistema que represente una violación o un incidente provocado sobre el mismo, comprometiendo la información manejada, que posee un carácter legal, confidencial y en su mayoría de carácter privado, representando su divulgación o modificación por un personal no autorizado un delito informático vigente, castigado por la ley.

También es imposible tener un control sobre dichas acciones, limitando de esta forma el conocimiento de lo ocurrido sobre el sistema por el personal encargado de la gestión de seguridad o mantenimiento del mismo, impidiendo en un futuro a los desarrolladores mejorar la usabilidad del sistema y poderle dar mantenimiento en base a fallos ocurridos.

Por otra parte resalta como una de las principales dificultades, la imposibilidad de estos sistemas de ajustarse a las características dinámicas que rigen el comportamiento en la Universidad, no logrando adaptarse a flujos de gestión específicos ni a cambios que ocurren con gran frecuencia por el mismo dinamismo al que está sujeto la gestión académica en la UCI. La integración con otros sistemas como uno de los principales objetivos a los que están sujetas las aplicaciones en la Universidad se hace imposible, debido a que no están diseñados para soportar tales funcionalidades.

2.3 Modelo del dominio

Debido al bajo nivel de estructuración que presenta el negocio que se está estudiando y que está altamente centrado en tecnologías informáticas, se propone un modelo del dominio, ya que nos permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Esto ayuda a los usuarios, clientes y desarrolladores e interesados, a utilizar un vocabulario común para poder entender el contexto en que se emplaza el sistema. Para capturar correctamente los requisitos y poder construir un sistema correcto se necesita tener un firme conocimiento del funcionamiento del objeto de estudio. Este modelo va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema.

2.3.1 Descripción del modelo del dominio

El modelo del dominio se describe mediante diagramas UML, específicamente con un diagrama de clases conceptuales significativas en el dominio del problema.

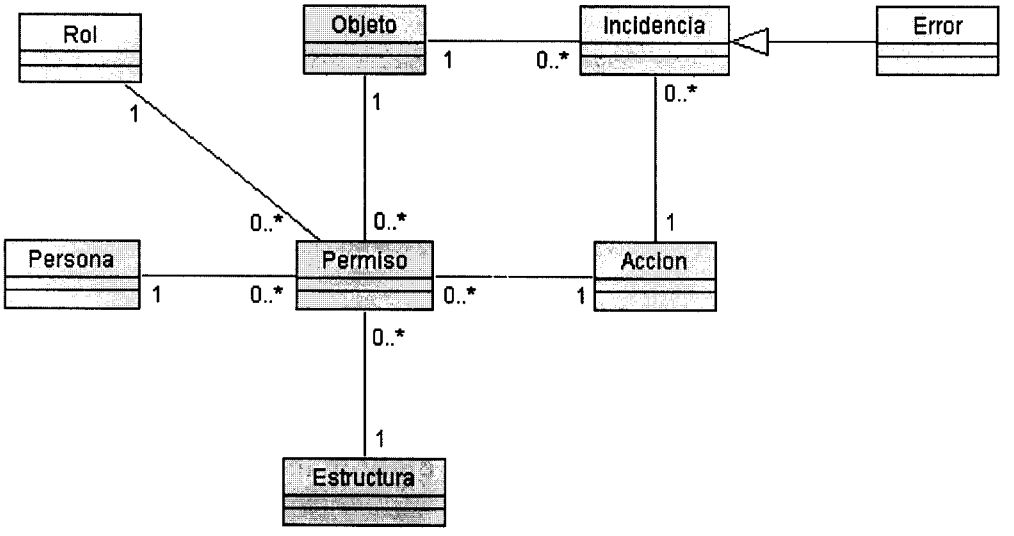


Figura 7: Diagrama del modelo del dominio.

Descripción del modelo del dominio

Un permiso está representado por las posibles acciones a realizar en el sistema por un grupo de personas o roles sobre un objeto o una estructura. Una persona en este caso pudiera ser cualquier estudiante, profesor, personal de secretaría u otra persona que interactúe con el sistema. En el caso de los roles, estos son los existentes en el directorio activo. Se describe a un objeto en fin, como cualquier cosa existente en el sistema que pueda representar a un objeto en la vida real, ejemplo: reportes, grupos, notas. Una estructura sería un grupo, una facultad o toda la universidad en general. Nos referimos a una incidencia como las trazas generadas por el sistema debido a la interacción de los usuarios con este. Un error es una especialización de una incidencia.

2.4 Solución propuesta

La solución que se propone es la elaboración de un Módulo de Gestión de Seguridad, que forme parte del Sistema de Gestión Académica "Akademos" e interactúe con el resto de los módulos, con el objetivo de brindarle servicios, presentando funcionalidades que ayudarán al trabajo de los involucrados en esta labor. Este mostrará una interfaz fácil y amigable donde tendrán las herramientas de trabajo necesarias para la gestión de los permisos sobre el sistema en los diferentes módulos, la visualización de incidencias tanto en tiempo real como las ocurridas en un período anteriormente. Esto sería parte de la seguridad de los elementos específicos del negocio que automatiza la aplicación.

La mayoría de las funcionalidades que brinda el Módulo de Gestión de Seguridad serán desarrolladas en un ambiente Web, haciendo uso de las facilidades que este brinda en términos de despliegue y con el objetivo de garantizar su interoperatividad con otras aplicaciones, exceptuando la visualización en tiempo real de incidencias, el que será elaborado sobre una aplicación dura y otras que estarán implícitas internamente en el propio sistema.

2.5 Requisitos funcionales

Conocidos los conceptos que rodean al objeto de estudio, se puede comenzar a analizar ¿Qué debe hacer el sistema para que se cumplan los objetivos planteados en este trabajo?, para ello se enumeran a través de requerimientos funcionales las funciones que el sistema deberá ser capaz de realizar. Dentro de ellos se incluyen las acciones que podrán ser ejecutadas por los usuarios y otro grupo que ejecutará el sistema de forma transparente al realizar alguna acción que implique esto. De acuerdo con los objetivos planteados el sistema debe ser capaz de:

Permisos

1. Comprobar permiso.
   - El subsistema especifica acción, objeto, tipo de objeto y estructura.

2. Asignar permiso.
   - El usuario especifica rol, idpersona, acción, objeto, tipo de objeto y estructura.

3. Eliminar permiso.
   - El usuario especifica rol, idpersona, acción, objeto, tipo de objeto y estructura.

4. Obtener estado permiso.
   - El subsistema especifica rol, idpersona, acción, objeto, tipo de objeto, estructura e inhabilitado.

5. Obtener los propietarios de los permisos.
   - El subsistema especifica objeto, tipo de objeto y estructura.

Incidencias

6. Registrar incidencia.
   - El subsistema especifica usuario, acción, tipo de objeto, ip y objeto.

7. Generar reporte de incidencias.
   - El usuario especifica el intervalo de tiempo.

8. Mostrar incidencias.
   - El usuario especifica tipo de objeto, acción, usuario, ip y fecha.

2.6 Requisitos adicionales

Requisitos de Seguridad

- Utilizar la seguridad integrada de Windows.
- La aplicación debe estar protegida de accesos no autorizados.
- La información manejada por el sistema será objeto de un alto nivel de protección.
- Realizar un control estricto de las acciones que lleven a cabo los usuarios en el sistema.
- Los usuarios autentificados solo tendrán acceso donde se les haya asignado.

Requisitos de Implementación

- Emplear la tecnología .Net como plataforma de desarrollo.
- Utilizar Microsoft SQL Server 2000 como Sistema de Gestión de Base de Datos.

Requisitos de Apariencia

- Debe poseer una interfaz amigable y en concordancia al estilo de las aplicaciones UCI.
- El sistema podrá configurar los permisos de los usuarios de una forma rápida y sencilla.

Requisitos de Software

Cliente:

- Sistema operativo con interfaz gráfica y soporte para red.
- Navegador Web.

Servidor:

- Windows 2000 Professional Service Pack 2.0 o superior.
- .NET Framework. 1.1.
- SQL Server 2000.

Requisitos de Hardware

Para el desarrollo

- Pentium 600 MHz o superior.
- 128 MB de memoria RAM.
- 20 GB de disco duro.

Para la explotación:

Clientes:

- Pentium de 133 MHz o superior.
- 128 MB de memoria RAM mínima, 256 MB de memoria RAM recomendada.

Servidor Web y de Correo:

- Dual PIII de 933MHz.
- 512 MB de memoria RAM.
- 60 GB de disco duro.
- Servidor de Base de Datos:
- Dual PIII de 933MHz.
- 512 MB de memoria RAM.
- 100 GB de disco duro.

Legales

- El empleo del sistema debe estar regido por un manual de normas y procedimientos que debe ser aprobado por la Dirección de la Universidad y estar basado en las disposiciones legales vigentes.

Análisis de costos y beneficios

Dada la importancia de la gestión académica dentro de una institución de estudios, en este caso la UCI, la utilización de una herramienta que automatice este proceso es indispensable, dado la importancia que representa para la institución la información manejada por el sistema. No obstante, es necesario tener en cuenta

para la puesta en marcha del proyecto los beneficios que reporta y los costos asociados a este.

Beneficios

El desarrollo de sistemas de este tipo en términos generales mejora rápida y efectivamente los niveles de calidad en la prestación de los servicios ofrecidos por la Universidad a nuestros clientes, asegurando la información, manteniendo su integridad y confiabilidad.

A continuación se listan los principales beneficios que reporta la implantación del sistema:

- Mantenimiento de la confidencialidad, integridad y disponibilidad de la información manejada por Akademos.
- Evita la necesidad de implementar seguridad en cada uno del resto de los módulos del sistema al brindarle sus servicios.
- Mejoras en la disponibilidad de la información.
- Prestación de servicios de mayor calidad.
- Minimiza los riegos a los que se pueda enfrentar el sistema.
- Disminución de problemas operacionales provocados por incidentes de seguridad.
- Reducción de costes asociados a pérdidas de información por incidentes de seguridad.
- Evita la manipulación de información confidencial por personas no autorizadas.

Teniendo en cuenta los beneficios reportados por el sistema propuesto y que la UCI posee la infraestructura tecnológica, así como el capital humano necesario para la elaboración y explotación del sistema, se considera factible el desarrollo del mismo.

2.7 Modelo del sistema

2.7.1 Modelo de Casos de Uso del Sistema

El modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones. Representa un esquema donde se recogen las funcionalidades del negocio que se automatizan y determina cómo será utilizado desde el punto de vista del usuario (Actor), pues se construye sobre la base de sus necesidades. [7]

Los actores representan los usuarios del sistema y otras aplicaciones que interactúan con él, es decir, representan terceros fuera del sistema, que interactúan con éste. Estos suelen corresponderse con trabajadores o actores del negocio. Los actores definidos en el sistema son los siguientes:

Actores	Justificación
Subsistema	Representa a cualquier componente software que necesite hacer uso de los servicios del subsistema de seguridad.
Usuario	Representa a cualquier usuario que utilice el sistema.
SGS	Representa la generalización de los actores Subsistema y Usuario.

2.7.2 Diagrama de Casos de Uso del sistema

El diagrama donde se representa la relación existente entre los actores y los casos de uso se representa a continuación:

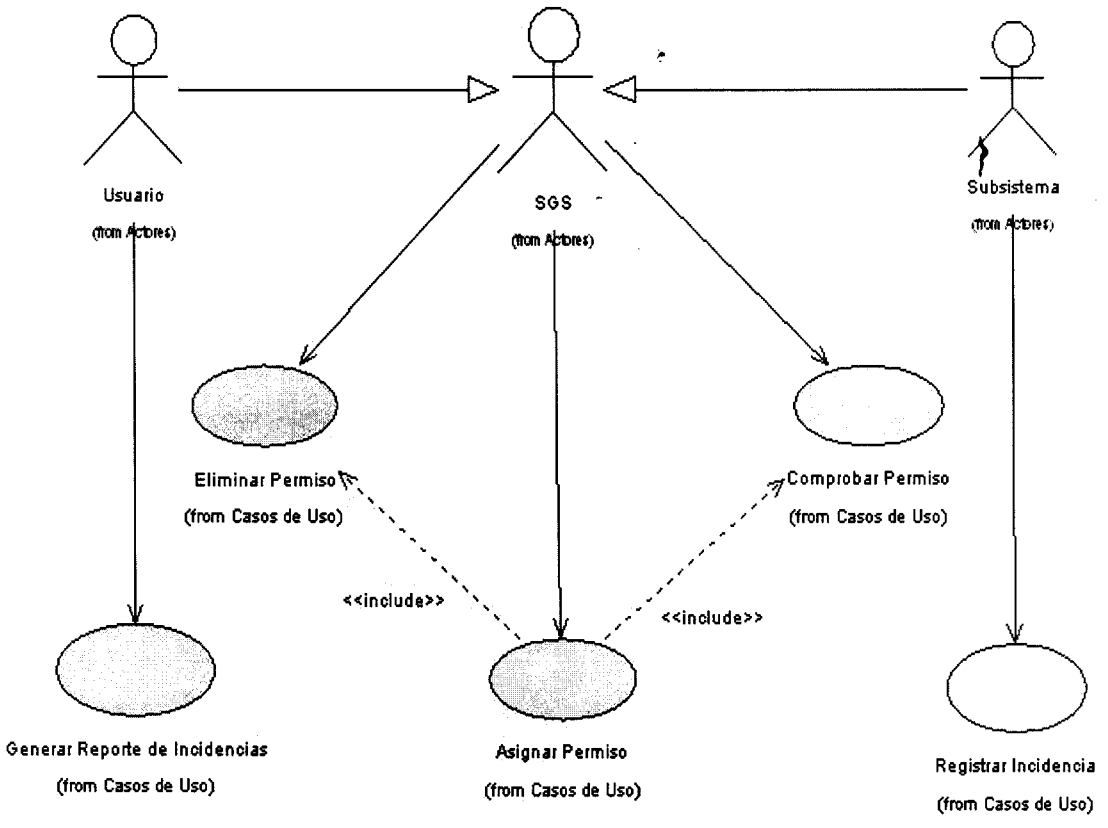


Figura 8: Diagrama global de casos de uso del sistema.

2.7.3 Descripción de los casos de uso del sistema

A continuación se presentan los casos de uso determinados para satisfacer los requerimientos funcionales del sistema:

CU1	COMPROBAR PERMISO.
Actor	SGS
Descripción	Un usuario o el subsistema piden la comprobación de un permiso sobre un determinado objeto del sistema, este verifica que el usuario posea el permiso suficiente para realizar dicha acción.
Referencia	RF 1, 5.

CU2	ASIGNAR PERMISO.
Actor	SGS
Descripción	El usuario o el subsistema asignan un nuevo permiso sobre una persona o un rol. Si el estado del permiso arroja que no existe se insertará uno nuevo, en caso que fuese negado se eliminará su negación y se insertará este nuevo y si existe se mantendrá el ya existente.
Referencia	RF 2, 3,4.

CU3	ELIMINAR PERMISO.
Actor	SGS
Descripción	El usuario o el subsistema elimina un permiso sobre un usuario o un rol negándole ejecutar tal acción sobre el sistema.
Referencia	RF 3.

CU4	REGISTRAR INCIDENCIA.
Actor	Subsistema
Descripción	El subsistema genera una nueva incidencia y solicita su registro en el sistema.
Referencia	RF 6.

CU5	GENERAR REPORTE DE INCIDENCIAS.
Actor	Usuario
Descripción	Se genera un reporte con todas las incidencias que cumplen con los parámetros seleccionados por el usuario.
Referencia	RF 7.

Mediante los casos de uso expandidos se describe paso a paso la secuencia de eventos que los actores utilizan para completar un proceso a través del sistema. A continuación se describen en formato expandido los casos de uso.

Caso de uso			
CU1	**COMPROBAR PERMISO**		
Propósito	Permitir que un usuario pueda realizar una acción sobre el sistema.		
Resumen: El caso de uso se inicia cuando un usuario o el subsistema (SGS) piden la comprobación de un permiso sobre un determinado objeto del sistema, este verifica que el usuario posea el permiso suficiente para realizar dicha acción.			
Referencias	RF 1, 5.		
Precondición	Se ha intentado realizar una acción en el sistema que necesita la verificación de un permiso.		
Acción del actor		**Respuesta del sistema**	
1	El usuario o el subsistema intentan realizar una determinada acción sobre el sistema.	2	El sistema verifica si dicho usuario o algún rol el cual este sea miembro posee un permiso igual al necesario para realizar dicha acción.
Flujo alternativo			
Acción del actor		**Respuesta del sistema**	
Paso 1	Si el usuario no posee el permiso requerido.	El sistema genera un mensaje de error.	
Poscondición	Se realiza la acción que el usuario estaba intentando ejecutar o se genera una incidencia en el sistema.		
Puntos de extensión.			
CU2			

Caso de uso	
CU2	**ASIGNAR PERMISO.**
Propósito	Permitir asignar un permiso de un rol o un usuario sobre un determinado objeto.

Resumen: El caso de uso se inicia cuando un usuario o el subsistema intentan asignar un determinado permiso de un rol o un usuario sobre un objeto, esto involucra poder insertar y eliminar usuarios o roles, el subsistema verifica el estado del permiso, en caso que sea no existe se insertará dicho permiso, en caso que sea negado se elimina el existente y se inserta un nuevo permiso y si existe este se mantiene.			
Referencias	RF 2, 3, 4, CU1, CU3.		
Precondición	El usuario ha accedido a la página designada para asignar permiso en los diferentes módulos del sistema o el subsistema intenta asignar un nuevo permiso.		
Acción del actor			**Respuesta del sistema**
1	El usuario (sección usuario) o el subsistema (sección subsistema) intentan asignar un determinado permiso sobre un usuario o un rol.		
Sección subsistema			
1	El subsistema intenta asignar un determinado permiso sobre un usuario (sección permiso usuario) o un rol (sección permiso rol).		
Sección permiso usuario			

1	El subsistema intenta asignar un permiso sobre otro usuario.	2	El sistema busca si existen permisos similares sobre este usuario o roles a los cuales sea miembro y verifica el estado de los permisos: I. No existe: Se inserta dicho permiso. II. Negado: Se elimina el existente y se inserta un nuevo permiso. III. Existe: Se mantiene el existente.
Sección permiso rol			
1	El subsistema intenta asignar un permiso sobre un rol.	2	El sistema busca si existen permisos similares sobre este rol y verifica el estado de los permisos: I. No existe: Se inserta dicho permiso. II. Negado: Se elimina el existente y se inserta un nuevo permiso. III. Existe: Se mantiene el existente.
Sección usuario			

		1	El sistema muestra las posibles opciones para interactuar con los permisos: I. Adicionar un nuevo usuario a la lista de miembro para posteriormente poder asignarle permisos. (sección adicionar usuario). II. Eliminar un usuario de la lista de miembro, eliminando todos sus permisos asignados sobre dicho objeto. (sección eliminar usuario). III. Asignar permisos a un usuario o rol (sección asignar permiso)
Sección adicionar usuario			
1	Un usuario elige adicionar un nuevo usuario o un rol a la lista de miembros con permisos.	2	El sistema muestra la posibilidad de realizar una búsqueda de personas y roles existente en el directorio activo.
3	El usuario elige el patrón de búsqueda: I. Por usuarios. II. Por roles.	4	El sistema muestra el resultado de la búsqueda según el patrón antes seleccionado.
5	El usuario selecciona los usuarios o roles que desee de la búsqueda arrojada por el sistema.	6	El sistema adiciona estos usuarios o roles a la lista de miembros.
Sección eliminar usuario			

1	Un usuario elige eliminar un usuario o un rol de la lista de miembros con permisos.	2	El sistema eliminará el usuario o rol de la lista de miembros con permiso y con ello todos los permisos que posea en este caso.
Sección asignar permiso			
1	Un usuario asigna permisos sobre la lista de miembros (usuarios o roles) con permisos, cambiando el estado inicial de los permisos mostrados.	2	El sistema actualiza el estado de los permisos según lo elegido por el usuario.
Poscondición	Quedan actualizados los permisos en el sistema.		
Prototipo de Interfaz	[Ver Anexo 1, Ver Anexo 2]		

Caso de uso			
CU3	**ELIMINAR PERMISO.**		
Propósito	Permitir eliminar un permiso de un rol o un usuario sobre un determinado objeto.		
Resumen: El caso de uso se inicia cuando un usuario o el subsistema intentan eliminar un determinado permiso de un rol o un usuario sobre un objeto, el sistema lo eliminará actualizando los permisos del sistema.			
Referencias	RF 3.		
Precondición	Se elimina un usuario con permisos en el sistema o el subsistema intenta promover un permiso antes asignado.		
Acción del actor		**Respuesta del sistema**	
1	El usuario o el subsistema intentan eliminar un determinado permiso.	2	El sistema eliminará el permiso actualizando el estado de los permisos en el sistema.

Poscondición	Han sido eliminados todos los permisos del usuario o rol que existían en el sistema sobre dicho objeto, quedando actualizado los permisos en el sistema.
Puntos de extensión.	
CU2	

Caso de uso			
CU4	**REGISTRAR INCIDENCIA.**		
Propósito	Permitir registrar una nueva incidencia sobre el sistema.		
Resumen: El caso de uso se inicia cuando el sistema intenta registrar una nueva incidencia, actualizando el estado de las incidencias en el sistema.			
Referencias	RF 6.		
Precondición	Se ha ejecutado una acción en el sistema que necesita ser registrada.		
Acción del actor		**Respuesta del sistema**	
1	El subsistema genera una incidencia.	2	El sistema registra la nueva incidencia.
Flujo alternativo			
Acción del actor		**Respuesta del sistema**	
Paso 1	La incidencia no se pudo registrar en la bases de datos.	El sistema registra la incidencia en un fichero local.	
Poscondición	Queda actualizado el estado de las incidencias en el sistema.		

<table>
<tr><td colspan="4">Caso de uso</td></tr>
<tr><td>CU5</td><td colspan="3">GENERAR REPORTE DE INCIDENCIAS.</td></tr>
<tr><td>Propósito</td><td colspan="3">Permitir mostrar un reporte con todas las incidencias ocurridas que cumplan con los parámetros seleccionados.</td></tr>
<tr><td colspan="4">Resumen: El caso de uso se inicia cuando un usuario solicita al sistema un reporte con las incidencias que cumplan con los parámetros seleccionados. El sistema realiza la búsqueda y el caso de uso termina cuando el sistema devuelve el reporte solicitado.</td></tr>
<tr><td>Referencias</td><td colspan="3">RF 7.</td></tr>
<tr><td>Precondición</td><td colspan="3">El usuario ha accedido a la página destinada para generar reportes de incidencias.</td></tr>
<tr><td colspan="2">Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1</td><td>El usuario solicita un reporte con las incidencias, seleccionando los parámetros (tipoobjeto, acción, usuario, IP y fecha).</td><td>2</td><td>El sistema realiza la búsqueda.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema devuelve el reporte de incidencias solicitado.</td></tr>
<tr><td>Poscondición</td><td colspan="3">Han sido mostradas las incidencias que cumplieron con los parámetros seleccionados por el usuario.</td></tr>
<tr><td>Prototipo de Interfaz</td><td colspan="3">[Ver Anexo 3]</td></tr>
</table>

2.8 Conclusiones

El desarrollo de este capítulo ha permitido una mejor comprensión del contexto a automatizar y de las características y restricciones que deben existir en el sistema para cumplir con los requerimientos de los clientes. Se realizó una descripción de la solución propuesta y se definieron los requisitos funcionales y no funcionales que debe cumplir. Se elaboró el diagrama de casos de uso del sistema, donde se representa cada actor y su relación con cada uno de éstos. Fue elaborada una descripción de todos los casos de uso y posteriormente expandidos. Con el desarrollo de este flujo de trabajo y los artefactos obtenidos a partir de este, se puede pasar al flujo de diseño para comenzar la construcción de la solución de software propuesta, el cual será presentado en el próximo capítulo.

Capítulo 3: Construcción de la solución propuesta

3.1 Introducción

El presente capítulo expone la construcción de la solución propuesta, a través de los flujos de trabajo de diseño e implementación. Primeramente se presenta el modelo de diseño, donde son expuestas las realizaciones de los casos de uso definidos en el capítulo anterior, mediante diagramas de clases del diseño y diagramas de interacción. Aparecen además descritos los principios de diseño que se siguen, referentes a estándares de interfaz, tratamiento de excepciones y estándares de codificación. El diagrama de clases persistentes, así como el modelo de datos obtenido a partir de éste y el diagrama de despliegue donde se representan los nodos en los que se distribuye la aplicación, son otros artefactos del diseño que se presentan en este capítulo. Se muestra también el modelo de implementación con los diagramas de componentes definidos y una descripción de cada uno de estos.

3.2 Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar, constituyendo una entrada principal en la actividad de implementación. [7]

En este modelo, los casos de uso son realizados por las clases del diseño y sus objetos, lo cual se denota por la realización de casos de uso del diseño que describe cómo se realizan estos en particular. A continuación se muestran los diagramas de clases del diseño y diagramas de interacción para la realización de los casos de usos descritos en el capítulo anterior.

Diagrama de clases del caso de uso "Asignar permiso"

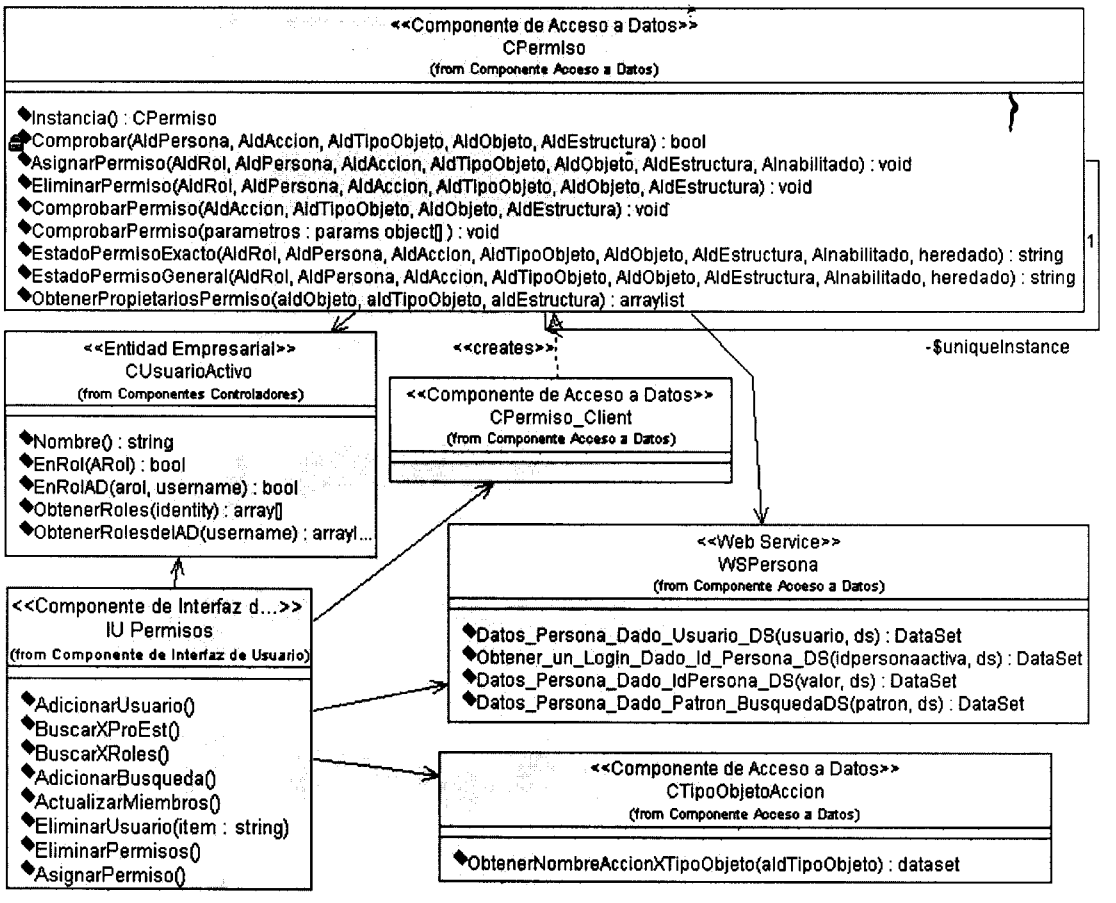


Figura 9: Diagrama de clases del CU "Asignar permiso".

Diagrama de secuencia del caso de uso "Asignar permiso". Sección "Subsistema"

[Ver Anexo 5]

Diagrama de secuencia del caso de uso " Asignar permiso ". Sección "Usuario ". Sección "Adicionar Usuario"

[Ver Anexo 6]

Diagrama de secuencia del caso de uso " Asignar permiso ". Sección "Usuario ". Sección "Asignar Permiso"

[Ver Anexo 7]

Diagrama de secuencia del caso de uso " Asignar permiso ". Sección "Usuario ". Sección "Eliminar Usuario"

[Ver Anexo 8]

Diagrama de clases del caso de uso "Comprobar permiso"

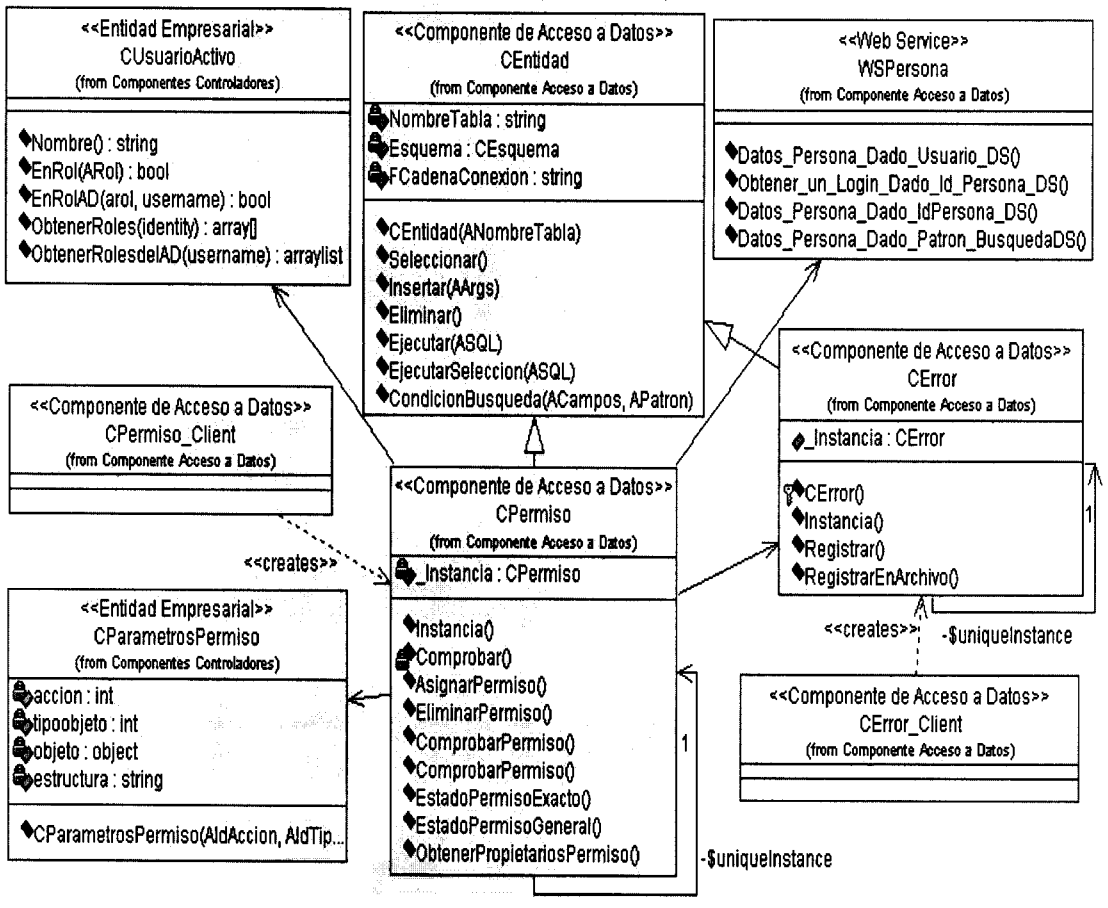


Figura 10: Diagrama de clases del CU "Comprobar permiso".

Diagrama de secuencia del caso de uso "Comprobar permiso"

[Ver Anexo 9]

Diagrama de clases del caso de uso "Eliminar permiso"

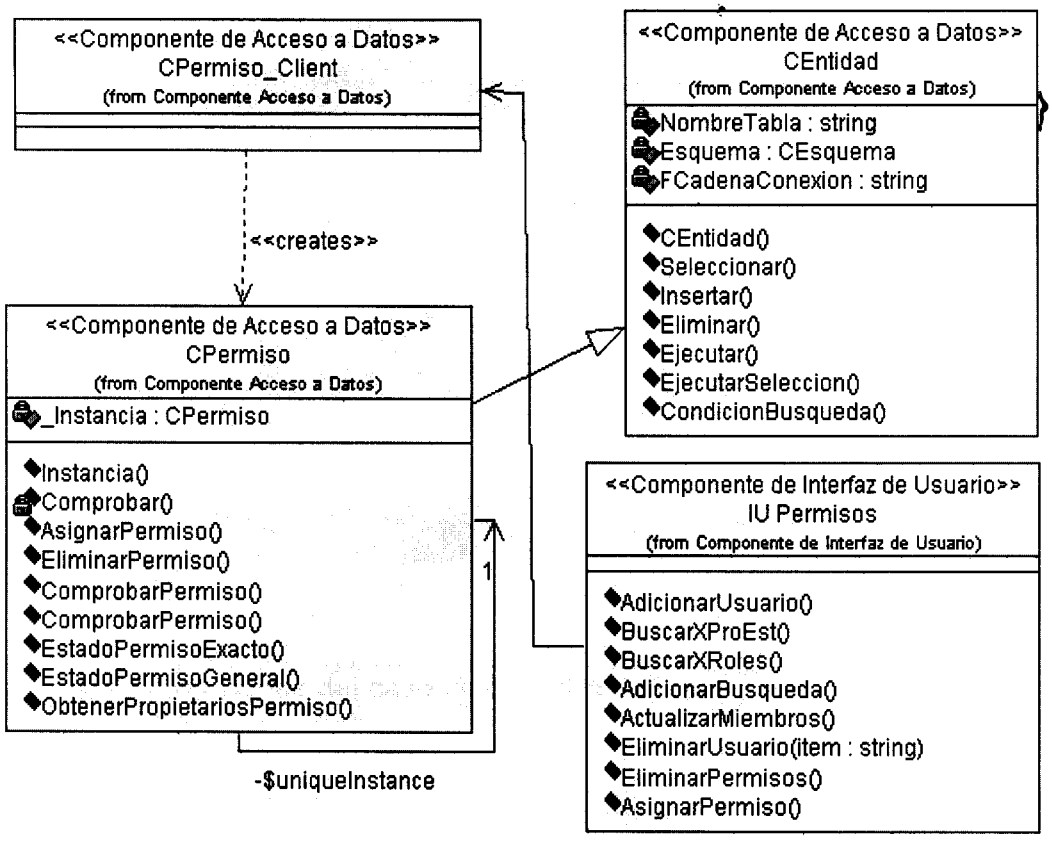


Figura 11: Diagrama de clases del CU "Eliminar Permiso".

Diagrama de secuencia del caso de uso "Eliminar permiso"

[Ver Anexo 10]

Diagrama de clases del caso de uso "Registrar incidencia"

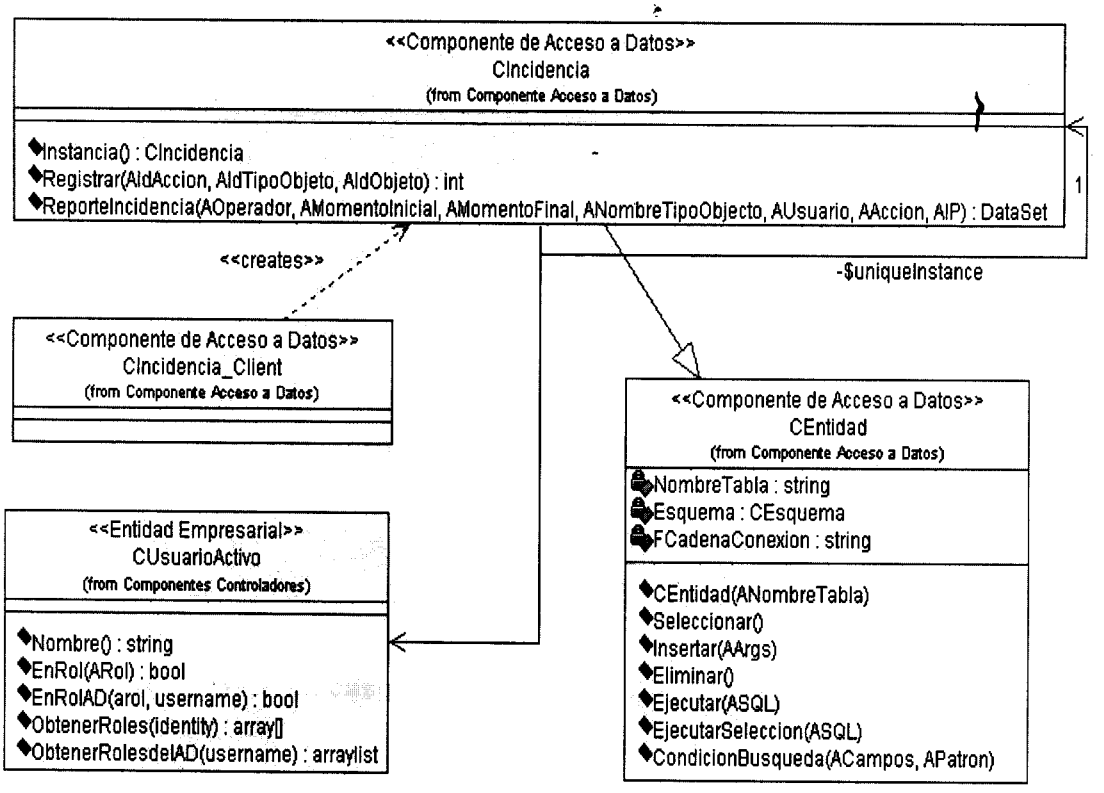


Figura 12: Diagrama de clases del CU "Registrar incidencia".

Diagrama de secuencia del caso de uso "Registrar incidencia"

[Ver Anexo 11]

Diagrama de clases del caso de uso "Reporte incidencias"

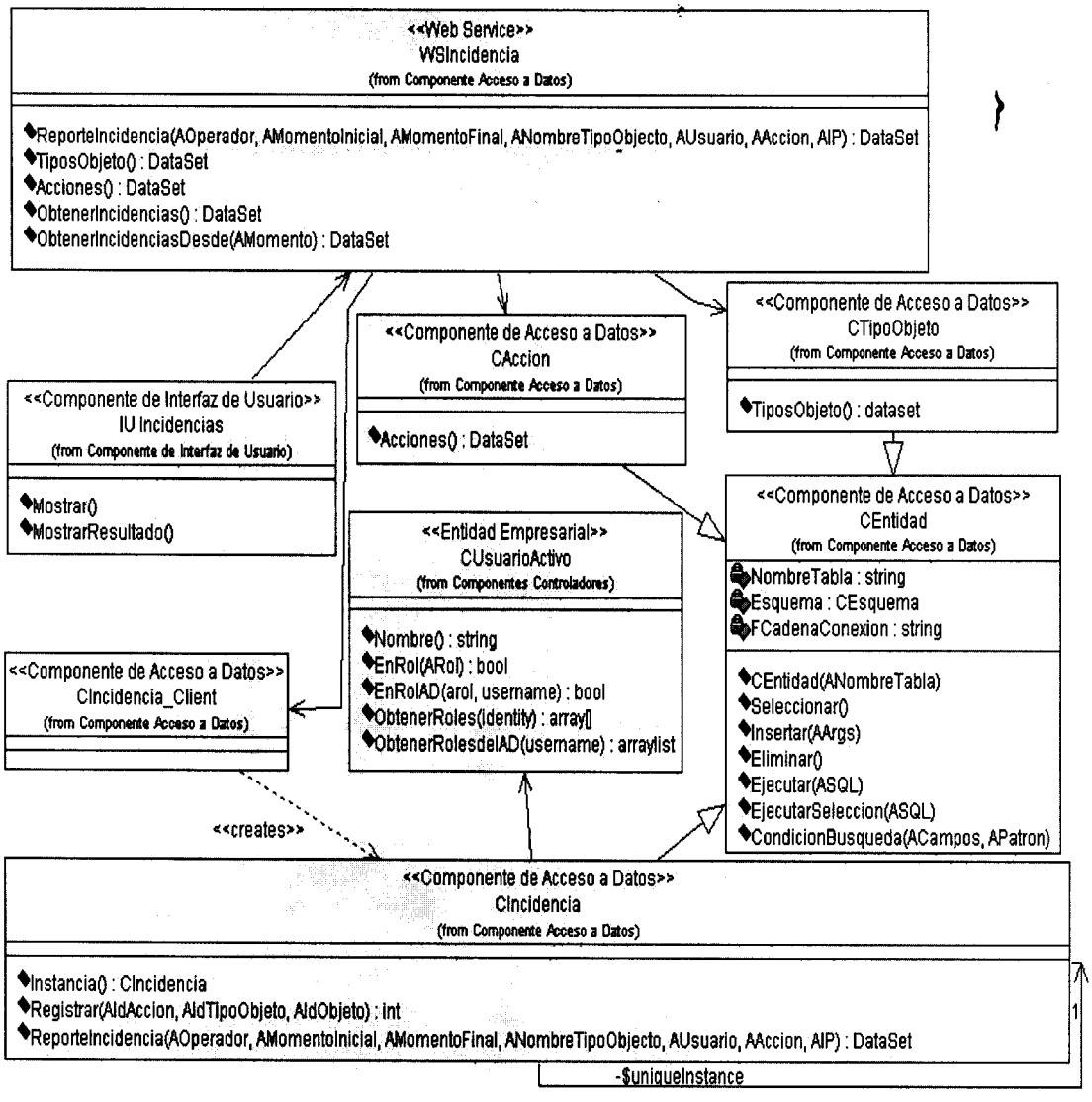


Figura 13: Diagrama de clases del CU "Reporte incidencias".

Diagrama de secuencia del caso de uso "Reporte incidencias"

[Ver Anexo 12]

3.2.1 Descripción de las clases del diseño

Nombre: CPermiso	
Tipo de clase: Componente de Acceso a Datos	
Atributo	**Tipo**
_Instancia	CPermiso
Para cada responsabilidad:	
Nombre:	*Instancia* ()
Descripción:	Retorna una única instancia de la clase.
Nombre:	*AsignarPermiso(AIdRol, AIdPersona, AIdAccion, AIdTipoObjeto, AIdObjeto, AIdEstructura, AInabilitado)*
Descripción:	Asigna permisos sobre una determinada persona o rol.
Nombre:	*GenerarConsultaInsert (AIdRol, AIdPersona, AIdAccion, AIdTipoObjeto, AIdObjeto, AIdEstructura, AInabilitado)*
Descripción:	Genera una consulta SQL para insertar un permiso.
Nombre:	*EliminarPermiso (AIdRol, AIdPersona, AIdAccion, AIdTipoObjeto, AIdObjeto, AIdEstructura)*
Descripción:	Elimina un permiso de un rol o de una persona en particular.
Nombre:	*ComprobarPermiso(AIdAccion, AIdTipoObjeto, AIdObjeto, AIdEstructura)*
Descripción:	Comprueba si el usuario activo posee este permiso. En caso negativo genera una excepción.
Nombre:	*ComprobarPermiso(parámetros)*
Descripción:	Sobrecarga de Comprobarpermisos con un arreglo de parámetros. En caso negativo genera una excepción.
Nombre:	*Comprobar(AIdPersona, AIdAccion, AIdTipoObjeto, AIdObjeto, AIdEstructura)*
Descripción:	Este método comprueba permisos sobre un determinado usuario y los roles a los que este pertenece. Retorna falso si no encuentra el permiso, verdadero en caso contrario.
Nombre:	*ObtenerPropietariosPermiso(aIdObjeto, aIdTipoObjeto,*

	aIdEstructura)
Descripción:	Retorna una lista con el idpersona o el rol de los propietarios de los permisos que cumplan con la condición de búsqueda.
Nombre:	EstadoPermisoGeneral(*AIdRol, AIdPersona, AIdAccion, AIdTipoObjeto, AIdObjeto, AIdEstructura, Ainabilitado, heredado*)
Descripción:	Retorna el estado de un permiso sin realizar una comprobación rigurosa y actualiza por referencia si es heredado.
Nombre:	*EstadoPermisoExacto(AIdRol, AIdPersona, AIdAccion, AIdTipoObjeto, AIdObjeto, AIdEstructura, Ainabilitado, heredado)*
Descripción:	Retorna el estado de un permiso mas rigurosamente, también comprobando en roles y actualiza por referencia si es heredado.

Nombre: CTipoObjetoAccion	
Tipo de clase: Componente de Acceso a Datos	
Para cada responsabilidad:	
Nombre:	*ObtenerNombreAccionXTipoObjeto(aIdTipoObjeto)*
Descripción:	Retorna un dataset con los nombre de las acciones que tienen asociado un tipo de objeto.

Nombre: CError	
Tipo de clase: Componente de Acceso a Datos	
Atributo	**Tipo**
_Instancia	CError
Para cada responsabilidad:	
Nombre:	*Instancia()*
Descripción:	Retorna una única instancia de la clase.

Nombre:	*Registrar(AEx)*
Descripción:	Registra un error. Intenta registrarlo en la base de datos y si no puede lo registra en el fichero de logs.
Nombre:	RegistrarEnArchivo(AEx)
Descripción:	Registra un error en el fichero de logs.

Nombre: CUsuarioActivo	
Tipo de clase: Control	
Para cada responsabilidad:	
Nombre:	*Nombre()*
Descripción:	Retorna el nombre del usuario activo.
Nombre:	*PrepararNombre(ANombre)*
Descripción:	Retorna una cadena que pasa por un proceso de comprobación.
Nombre:	ObtenerRoles(identity)
Descripción:	Retorna una lista con los roles a los que pertenece el usuario tanto locales como en el Directorio Activo.
Nombre:	*EnRol(Arol)*
Descripción:	Determina si el usuario activo pertenece a dicho rol. Retornando verdadero o falso.
Nombre:	*EnRolAD(ARol, username)*
Descripción:	Determina si dicho usuario pertenece a este rol en el Directorio Activo. Retornando verdadero o falso.
Nombre:	*ObtenerRolesdelAD(strUser)*
Descripción:	Retorna una lista con los roles del directorio activo que pertenece un usuario.
Nombre:	*ObtenerTodosGruposADS()*
Descripción:	Retorna una lista de todos los grupos existentes en un directorio activo.

Nombre: CParametrosPermiso	
Tipo de clase: Control	
Atributo	**Tipo**
Acción	int
tipoobjeto	int
Objeto	object
estructura	string
Para cada responsabilidad:	
Nombre:	*CParametrosPermiso(AIdAccion, AIdTipoObjeto, AIdObjeto, AIdEstructura)*
Descripción:	Constructor de la clase que construye un objeto.

Nombre: CTipoObjeto	
Tipo de clase: Componente de Acceso a Datos	
Para cada responsabilidad:	
Nombre:	*TiposObjeto ()*
Descripción:	Retorna un dataset con todos los tipos de objetos.

Nombre: CAccion	
Tipo de clase: Componente de Acceso a Datos	
Para cada responsabilidad:	
Nombre:	*Acciones()*
Descripción:	Retorna un dataset con todos los tipos de acciones.

Nombre: CIncidencia	
Tipo de clase: Componente de Acceso a Datos	
Atributo	**Tipo**
_Instancia	CIncidencia
Para cada responsabilidad:	

Nombre:	*Instancia ()*
Descripción:	Retorna una única instancia de la clase.
Nombre:	*Registrar(AIdAccion, AIdTipoObjeto, AIdObjeto)*
Descripción:	Registrar una incidencia retornando el id de la incidencia.
Nombre:	*ReporteIncidencia(AOperador, AMomentoInicial, AMomentoFinal, ANombreTipoObjecto, AUsuario, AAccion, AIP)*
Descripción:	Retorna un dataset con el reporte.

Nombre: WSIncidencia	
Tipo de clase: Web Service	
Para cada responsabilidad:	
Nombre:	*ReporteIncidencia(AOperador, AMomentoInicial, AMomentoFinal, ANombreTipoObjecto, AUsuario, AAccion, AIP)*
Descripción:	Retorna un dataset con el reporte.
Nombre:	TiposObjeto()
Descripción:	Obtiene un dataset con la lista de objetos.
Nombre:	Acciones()
Descripción:	Obtiene un dataset con la lista de acciones.

Nombre: AKExcepcionN	
Tipo de clase: Componente de Acceso a Datos	
Atributo	**Tipo**
_IdError	int
_NumError	int
Para cada responsabilidad:	
Nombre:	*AKExcepcionN(ANumError, AExcepcionCausante)*
Descripción:	Constructor de la clase. Representa una excepción que ocurre en el negocio.

Nombre:	Registrar(ANumError, AExcepcionCausante)
Descripción:	Registra una excepción.

Nombre: IU Permisos	
Tipo de clase: Componente de Interfaz de Usuario	
Atributo	**Tipo**
dgPermiso	Datagrid
Item	string
Para cada responsabilidad:	
Nombre:	*LlenarListaMiembros()*
Descripción:	Se llena según las personas o roles que posean los respectivos permisos.
Nombre:	*AdicionarUsuario()*
Descripción:	Adiciona un usuario o un rol a la lista de miembros con permiso.
Nombre:	*BuscarXProfEst( )*
Descripción:	Busca estudiantes y profesores según un patrón de búsqueda.
Nombre:	*BuscarXRoles()*
Descripción:	Busca roles del directorio activo según un patrón de búsqueda.
Nombre:	*ElegirBusqueda(item)*
Descripción:	Adiciona de la búsqueda arrojada el elemento marcado.
Nombre:	AdicionarBusqueda()
Descripción:	Adiciona a la lista de miembros los elementos seleccionados.
Nombre:	*MostrarPermisos()*
Descripción:	Muestra los permisos asociados a un usuario o rol
Nombre:	*ActualizarEstadoPermisos(ds)*
Descripción:	Actualiza el estado del los permisos una vez que se halla efectuado algún cambio.

Nombre:	*ComprobarExistencia(item)*
Descripción:	Se muestran los permisos según la persona seleccionada y el ID tipo de objeto.
Nombre:	*ActualizarMiembros()*
Descripción:	Actualiza la lista de miembros con permiso.
Nombre:	*EliminarUsuario(item)*
Descripción:	Se elimina el usuario o el rol seleccionado.
Nombre:	*AsignarPermiso()*
Descripción:	En correspondencia con el estado de los permisos realiza una eliminación o actualización de los permisos.
Nombre:	*EliminarPermisos()*
Descripción:	Se eliminan los permisos al eliminar el usuario o rol.
Nombre:	*EstadoFinalMarcado(pos)*
Descripción:	Retorna un entero que significa el estado del permiso.

Nombre: IU Incidencias	
Tipo de clase: Componente de Interfaz de Usuario	
Atributo	**Tipo**
dgReporte	DataGrid
Para cada responsabilidad:	
Nombre:	*Mostrar()*
Descripción:	Muestra la pagina inicial donde el usuario puede elegir los parámetros para posteriormente mostrar las incidencias.
Nombre:	*MostrarResultado(ds)*
Descripción:	Muestra las incidencias según los parámetros seleccionados por el usuario.

3.3 Principios de diseño

3.3.1 Estándares en la interfaz de la aplicación

La interfaz gráfica del usuario es el medio por el cual este interactúa con el sistema, por lo que esta debe ser lo más amigable posible y lograr que se sienta identificado con la misma.

Para el diseño de la interfaz del sistema se tuvieron en cuenta aspectos necesarios, que garanticen la comodidad por parte del usuario, teniendo presente la organización de la información que se muestra y su distribución en la pantalla. Los elementos que se repiten en varias pantallas son ubicados en el mismo lugar permitiéndole al usuario acostumbrarse al ambiente y que éste no se vea desorientado. Éstas solo contienen la información necesaria para el usuario, evitando que estén sobrecargadas. Las pantallas son uniformes logrando un balance de los elementos que la componen.

3.3.2 Tratamiento de excepciones

El tratamiento de errores posibilita el buen funcionamiento de una aplicación dándole una mejor apariencia ante los clientes. En el sistema se controla la entrada de datos del usuario a nivel de interfaz. Cuando se produce un error por la entrada incorrecta de un valor suministrado por el usuario se le señaliza en la pantalla donde se encuentra para que sea rectificado.

Las excepciones que ocurren internamente en la aplicación son capturadas por el sistema y mostradas como mensajes de error al usuario (Figura 14). Estos mensajes aparecen en otra ventana, son específicos y a la vez concisos, además entendibles para el usuario; en algunos casos brindan información detallada de la excepción producida y en ocasiones se señala alguna solución de la misma.

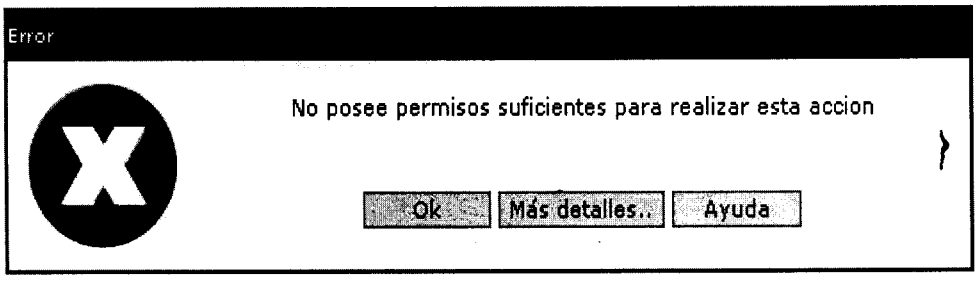


Figura 14: Errores personalizados.

3.4 Estándares de codificación

Para la implementación de la aplicación se siguieron estándares para la generación de código. Fue utilizada identación para una mejor organización y claridad del código fuente. Se describió cada operación de cada una de las clases haciendo uso de comentarios basados en XML que introdujo Microsoft, donde fue especificado su propósito y una breve descripción de cada uno de los argumentos que recibe. Las clases de acceso a datos fueron nombradas comenzando por la letra C y a continuación el nombre en mayúscula de la entidad que controla de la base de datos. Los componentes empresariales y los servicios Web se nombraron comenzando con la cadena CN y WS respectivamente.

3.5 Principios de protección y seguridad

3.5.1 Ataques SQL Injection

Los ataques por SQL Injection son realmente peligrosos, no exclusivamente por la capacidad de daño que conllevan, sino porque son vulnerabilidades que muchos programadores no corrigen a tiempo, el ataque es posible dadas ciertas características del dialecto o lenguaje SQL que lo dotan de flexibilidad, tales como:

- Poder embeber comentarios en una sentencia SQL.
- Poder escribir varias sentencias SQL juntas y ejecutarlas en bloque.
- Poder realizar consultas de metadatos por medio de tablas del sistema.

Por este motivo cualquier RDBM que entienda SQL, llámese Oracle, DB2, SQL Server, MySQL, es susceptible de recibir un ataque de este tipo a través de sus aplicaciones cliente, este ataque se produce a nivel de aplicación, ya sean de tipo Consola, Windows o Web e independiente de la plataforma de desarrollo (Java, Win32, .NET) con la que fue elaborada.

Existen ciertos principios a considerar para proteger nuestras aplicaciones de un SQL Injection:

1. No confiar en la entrada del usuario.
2. No utilizar cuentas con privilegios administrativos.
3. No proporcionar mayor información de la necesaria.

A continuación veremos algunas medidas implementas por el Subsistema de Gestión de Seguridad para combatir tales vulnerabilidades siguiendo los principios antes expuestos.

1. No confiar en la entrada del usuario:

- Se filtra la entrada del usuario de caracteres SQL para limitar los caracteres involucrados en un SQL Injection.
- Se protege las instrucciones de búsqueda de modelos coincidentes (LIKE).
- Se verifica cualquier tipo de entrada, no sólo lo introducido en los controles interfaz de usuario (IU) sino también aquellas que no son visibles, como parámetros de entrada y campos tipo hidden de las páginas Web.
- Se realiza la verificación en todos los niveles y capas de la aplicación, ya que si sólo protegemos la capa de presentación somos vulnerables a que un atacante salte a la siguiente capa y realice su ataque.

2. No se utilizan cuentas con privilegios administrativos:

- Ejecutar las sentencias SQL o invocar Procedimientos Almacenados con una cuenta con privilegios mínimos.

3. No se proporciona mayor información de la necesaria:

- No se expone al usuario final los mensajes de errores devueltos por el gestor de base de datos, para no brindar mayor información que sea útil al atacante.
- Se implementa una gestión de errores internamente que notifican al usuario solo errores personalizados y se establece en el archivo de configuración Web (Web.Config) el valor del atributo debug del elemento compilation en False y el atributo mode del elemento customErrors en On o en su defecto en RemoteOnly.

3.5.2 Políticas de respaldo y recuperación de información

Cuando utilizamos sistemas que son dependientes del buen funcionamiento de un sistema de bases de datos, nos volvemos dependientes en grado sumo de éste. En caso de ocurrir daños sobre cualquier porción de la base de datos, ya sea por errores humanos, una falla en el equipo, virus informáticos o de cualquier otro carácter, resulta esencial poder recuperar los datos implantados con un mínimo de tiempo y afectando lo menos posible el buen funcionamiento del sistema. Aunque no se puede prevenir cada una de éstas situaciones, podemos prepararnos para evitar las consecuencias que éstas tengan sobre nuestro negocio. Del tiempo que tardemos en reaccionar, dependerá la gravedad de las consecuencias.

Las empresas, ya sean grandes o pequeñas confían en la información computarizada para facilitar su operación, la pérdida de información puede representar cuantiosos daños económicos, pérdida de oportunidades, clientes decepcionados, reputación perdida, entre otros. De aquí que, es necesario la implementación de mecanismos básicos de seguridad para organizar las políticas de seguridad de la información, y así proteger toda la información. El establecimiento de procedimientos y garantías de seguridad frente a vulnerabilidades y amenazas, siempre actuará en beneficio de nuestra propia empresa y su funcionamiento.

Teniendo en cuenta la sensibilidad de la información manejada por el sistema de Gestión Académica y el daño que podría reportar su pérdida, ya sea en grado total o parcial, se decidió implementar una política de respaldo y recuperación de información, en este caso se propuso GFS (GrandFather – Father – Son). La cual sigue los siguientes principios:

- Respaldos completos semanales.
- Respaldos diferenciales diariamente.
- Mantener 4 semanas los respaldos completos realizados.
- Mantener 7 días los respaldos diferenciales realizados.

Se almacenan todos estos respaldos en servidores apartes y en dispositivos de almacenamiento externos, fuera del área de los servidores.

3.6 Diseño de la base de datos

3.6.1 Diagrama de clases persistentes

La arquitectura de aplicaciones de .Net basada en modelos de componentes por capas, que ha sido utilizada para la elaboración del subsistema, define un tipo de componente denominado entidad empresarial, que es utilizado para el paso de datos entre componentes. Estos suelen ser conjuntos de datos (DataSet) o clases orientadas a objetos personalizadas que representan entidades del mundo real. En el subsistema que se presenta, han sido utilizados los DataSet para el paso de datos entre componentes, pero partiendo de que estas clases que representan entidades del mundo real son válidas en la arquitectura utilizada, fue elaborado el siguiente diagrama de clases persistentes, para poder obtener el Modelo de Datos de la aplicación. A continuación se muestra el diagrama de clases persistentes del sistema, donde aparecen todas las entidades que se manejan en él.

Diagrama de clases persistentes

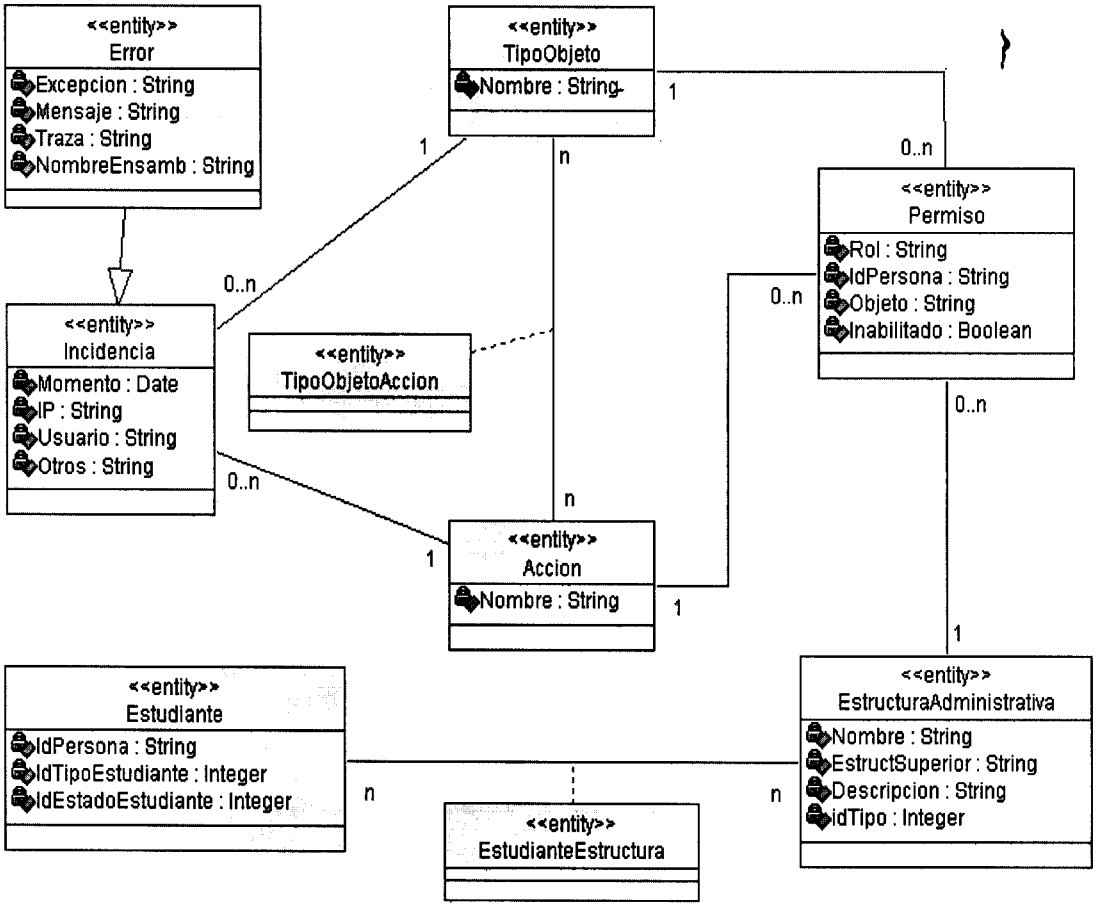


Figura 15: Diagrama de clases persistentes.

3.6.2 Modelo de datos

El modelo de datos que se obtuvo a partir del diagrama de clases persistentes se muestra a continuación.

Diagrama del modelo de datos

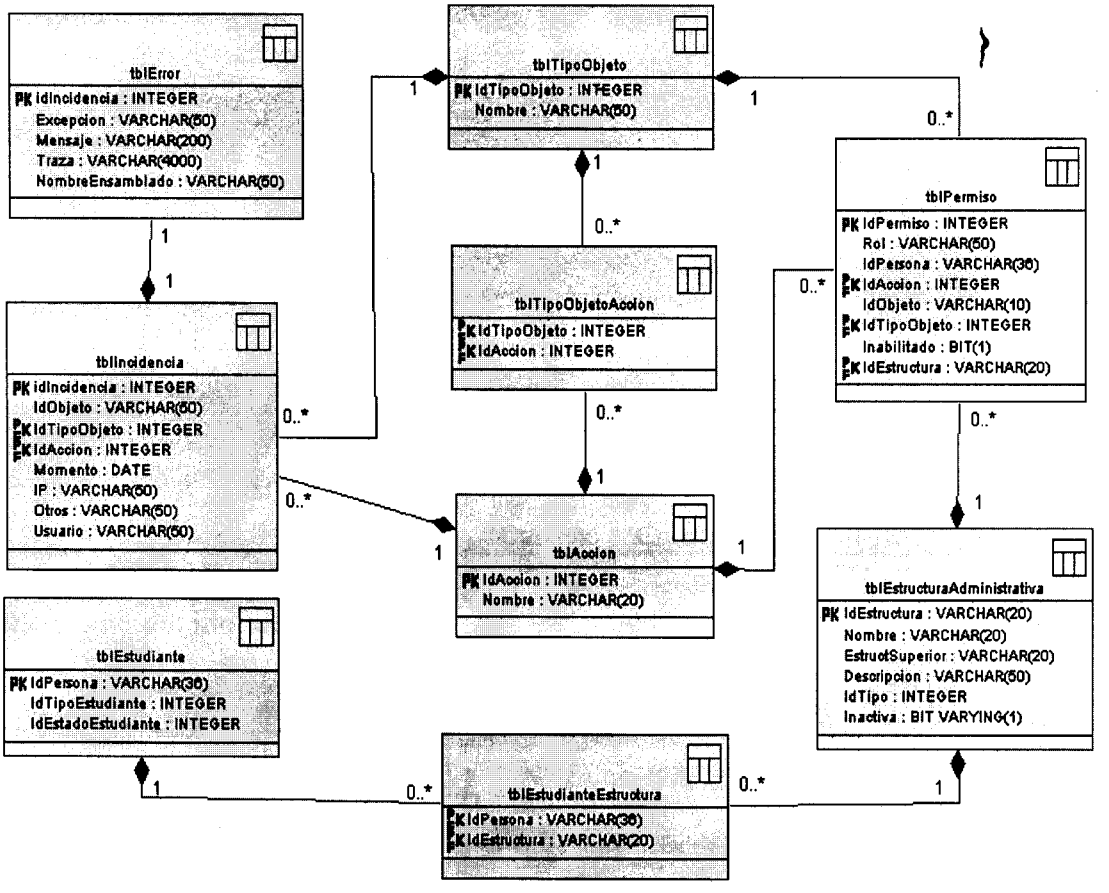


Figura 16: Diagrama del modelo de datos.

3.6.3 Descripción de las tablas de la bases de datos

Nombre: tblTipoObjeto		
Descripción: En esta tabla se almacenan los datos de los tipos de objetos existentes en el sistema.		
Atributo	**Tipo**	**Descripción**
IdTipoObjeto	int	Identificador de los objetos existentes en el sistema.
Nombre	varchar	Nombre de los objetos existentes en el sistema.

Nombre: tblTipoObjetoAccion		
Descripción: En esta tabla se almacenan las relaciones existentes entre objetos y acciones del sistema.		
Atributo	**Tipo**	**Descripción**
IdTipoObjeto	int	Identificador de un objeto.
IdAccion	int	Identificador de una acción.

Nombre: tblAccion		
Descripción: En esta tabla se almacenan los datos de las acciones existentes en el sistema.		
Atributo	**Tipo**	**Descripción**
IdAccion	int	Identificador de una acción.
Nombre	varchar	Nombre de una acción.

Nombre: tblPermiso		
Descripción: En esta tabla se almacenan los datos de los permisos existentes en el sistema.		
Atributo	**Tipo**	**Descripción**
IdPermiso	int	Identificador de un permiso.

Rol	varchar	Nombre de un rol.
IdPersona	varchar	Identificador de una persona.
IdAccion	int	Identificador de una acción.
IdObjeto	nvarchar	Identificador de un objeto.
IdTipoObjeto	int	Identificador de un tipo de objeto
IdEstructura	nvarchar	Identificador de una estructura.
Inabilitado	bit	Establece el estado del permiso (valido o negado).

Nombre: tblIncidencia		
Descripción: En esta tabla se almacenan las incidencias ocurridas en el sistema.		
Atributo	**Tipo**	**Descripción**
IdIncidencia	int	Identificador de una incidencia.
IdObjeto	nvarchar	Identificador de un objeto.
IdTipoObjeto	int	Identificador de un tipo de objeto.
IdAccion	int	Identificador de una acción.
Momento	datetime	Identifica en que tiempo ocurrió.
IP	varchar	Identifica IP donde ocurrió la incidencia.
Otros	varchar	Identifica otros datos.
Usuario	varchar	Identifica el nombre del usuario.

Nombre: tblError		
Descripción: En esta tabla se almacenan los errores ocurridos en el sistema.		
Atributo	**Tipo**	**Descripción**
IdIncidencia	int	Identificador de una incidencia.
Excepcion	varchar	Nombre de la excepción que generó el error.
Mensaje	varchar	Mensaje de error.
Traza	varchar	Traza del error.

NombreEnsamblado	varchar	Nombre del ensamblado.

Nombre: tblEstructuraAdministrativa		
Descripción: En esta tabla se almacenan los grupos a los que pertenecen los estudiantes administrativamente.		
Atributo	**Tipo**	**Descripción**
IdEstructura	nvarchar	Identificador de una estructura.
Nombre	varchar	Nombre de la estructura.
EstructSuperior	nvarchar	Identificador de la estructura superior.
Descripcion	varchar	Breve descripción.
IdTipo	int	Identificador del tipo de estructura.
Inactiva	tinyint	Identificador del estado.

Nombre: tblEstudiante		
Descripción: En esta tabla se almacenan los datos referentes a los estudiantes.		
Atributo	**Tipo**	**Descripción**
IdPersona	varchar	Identificador de una persona.
IdTipoEstudiante	int	Identificador del tipo de estudiante.
IdEstadoEstudiante	int	Identificador del estado del estudiante.

Nombre: tblEstudianteEstructura		
Descripción: En esta tabla se almacenan a las estructuras que pertenecen los estudiantes.		
Atributo	**Tipo**	**Descripción**
IdEstructura	nvarchar	Identificador de una estructura.
IdPersona	varchar	Identificador de una persona.

3.7 Modelo de despliegue

El modelo de despliegue es un modelo de objetos, que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. [7]

El diagrama de despliegue de la aplicación se muestra a continuación.

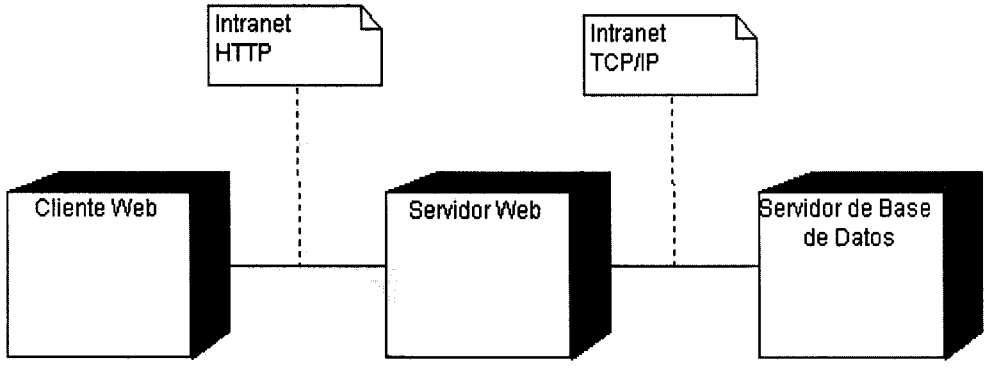


Figura 17: Diagrama de despliegue.

La solución propuesta se basa en la arquitectura de capas basada en componentes, propuesta por Microsoft para aplicaciones en .Net, que establece tres capas a través de las que se distribuye toda la aplicación. Dentro de ellas está presente la Capa de Presentación, la cual contiene los Componentes de Interfaz de Usuario que garantizan la interacción de éstos con la aplicación. La Capa Empresarial en la que se encuentran ubicados todos los componentes encargados de garantizar la lógica del negocio de la aplicación. En la solución propuesta fueron utilizados los Componentes Empresariales y las Interfaces de Servicios. En la Capa de Acceso a Datos, a la que pertenecen los componentes que acceden a los datos para realizar consultas y operaciones sobre éstos, de inserción, eliminación y modificación; fueron utilizados los componentes lógicos de acceso a datos que se proponen en la arquitectura utilizada y la Capa de Datos que contiene los datos del sistema, es decir, la base de datos en sí con toda la información almacenada.

Teniendo en cuenta lo anterior, en el diagrama de despliegue del sistema se representan tres nodos (Figura 17). Uno de estos és el Cliente Web, que representa los ordenadores de los usuarios, desde los cuales podrán acceder, utilizando el protocolo HTTP, a la aplicación que se encuentra publicada en el Servidor Web, donde se encuentran ubicados todos los componentes de las Capas de Presentación, Empresarial y de Acceso a Datos. Éste a su vez se comunica con el Servidor de Base de Datos (con SQL Server 2000 como sistema gestor de base de datos), a través del protocolo TCP/IP para realizar consultas y actualizaciones de la información que manipula el sistema.

3.8 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes. Describe también como se organizan y se relacionan unos con otros, definiendo un componente como el empaquetamiento físico de los elementos de un modelo, como es el caso de las clases del modelo de diseño.

A continuación se muestran los paquetes definidos y los componentes que pertenecen a cada uno de estos, así como la relación de dependencia que existe entre componentes de distintos paquetes, representando los paquetes según su definición por un color determinando. Además, se describe cada uno de estos componentes de acuerdo a su propósito y clases que contiene.

Los componentes representados en los siguientes diagramas que tienen extensión asmx, aspx, ascx están relacionados, cada uno de ellos, con un componente de extensión cs, los cuales no fueron representados en los diagramas para su mejor visualización.

Paquetes

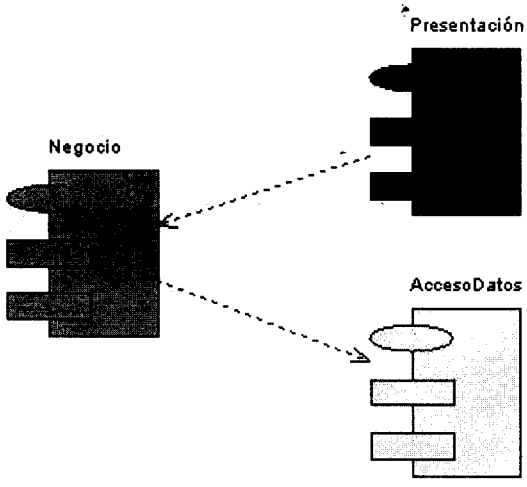


Figura 18: Diagrama de paquetes.

Diagrama de componentes

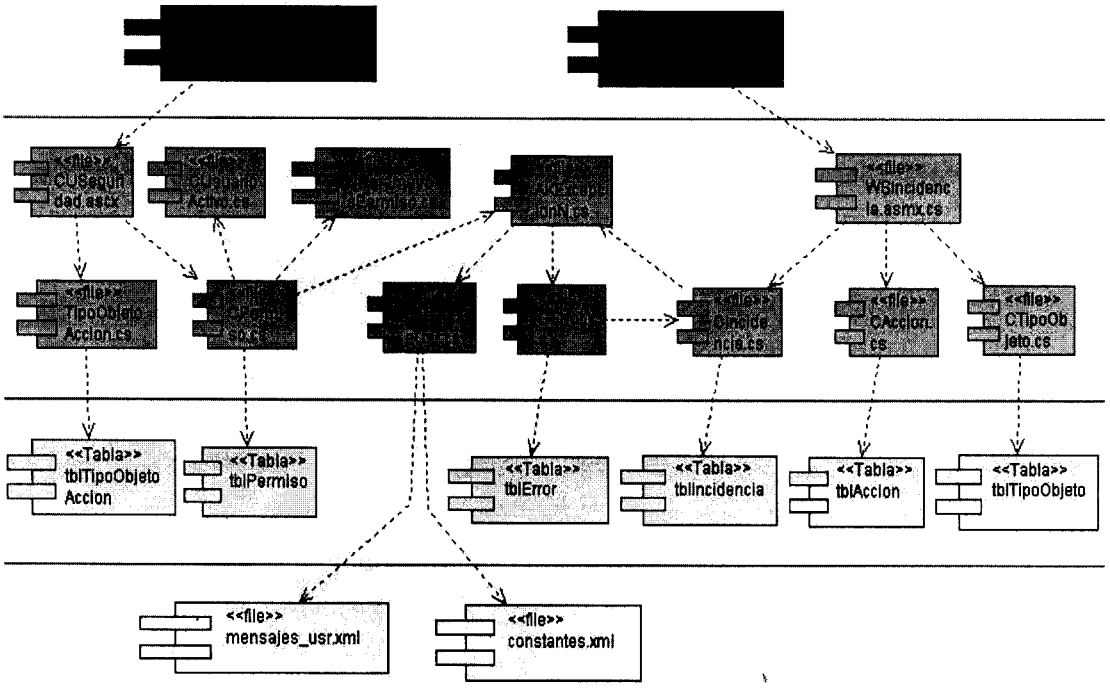


Figura 19: Diagrama de componentes.

Componente	Propósito	Contenido
Asignarpermiso.aspx	Página designada para la asignación de permisos en los diferentes módulos.	Asignarpermiso
Incidencias.aspx	Página designada para la visualización de las incidencias ocurridas en el sistema.	Incidencias
CUSeguridad.ascx	Control de seguridad designado para la interacción con los permisos, permitiendo asignar, eliminar y modificar tanto permisos como usuarios.	CUSeguridad
CUsuarioActivo.cs	Controla la gestión de usuarios y roles con el directorio activo del dominio uci.cu.	CUsuarioActivo
CParametrosPermiso.cs	Controla los parámetros recibidos en la gestión de los permisos.	CParametrosPermiso
AKExcepcionN.cs	Representa una excepción controlada por el sistema en la capa de negocios.	AKExcepcionN
WSIncidencia.asmx.cs	Brinda servicios para la gestión de incidencias ocurridas en el sistema.	WSIncidencia
TipoObjectoAccion.cs	Controla la relación existente entre los objetos y las acciones.	TipoObjectoAccion
CPermiso.cs	Controla la gestión de permisos en el sistema, tanto inserción como verificación, pertenencia, cambio de estado y	CPermiso

	eliminación, para usuarios y roles.	
CMensajeUsr.cs	Gestiona un grupo de mensajes que serán mostrados al usuario en su interacción con la aplicación.	CMensajeUsr
Cerror.cs	Gestiona los errores ocurridos en la aplicación dándole tratamiento personalizado.	CError
CIncidencia.cs	Gestiona las incidencias ocurridas en el sistema, registrándoles para posteriores auditorías sobre el mismo.	CIncidencia
CAccion.cs	Representa una posible acción a ejecutar sobre el sistema.	CAccion
CTipoObjecto.cs	Gestiona un conjunto de objetos tratados por la aplicación.	CTipoObjecto
tblTipoObjectoAccion	Representa una tabla en la base de datos donde se almacena la relación entre objetos y acciones.	
tblPermiso	Representa una tabla en la base de datos donde se almacenan los permisos establecidos en la aplicación.	
tblIncidencias	Representa una tabla en la base de datos donde se almacenan las incidencias ocurridas.	
tblAccion	Representa una tabla en la	

	base de datos donde se establecen las posibles acciones a realizar.	
tblTipoObjecto	Representa una tabla en la base de datos donde se almacenan los objetos tratados por la aplicación.	
tblError	Representa una tabla en la base de datos donde se almacenan los errores ocurridos en el sistema.	
mensajes_usr.xml	Representa un XML donde se almacenan los mensajes que se le mostraran al usuario en un determinando momento.	
constantes.xml	Representa un XML donde se almacenan las constantes tratadas en la aplicación.	

3.9 Conclusiones

En el presente capítulo se han desarrollado los flujos de trabajo de diseño e implementación que propone la metodología RUP. Se han elaborados los diagramas de clases del diseño, así como diagramas de secuencia para la realización de los caso de uso obtenidos en el capítulo anterior. Fueron descritos los principios de diseño en los que se basa la solución propuesta y elaborado el diagrama de clases persistentes, a partir del cual se obtuvo el modelo de datos. También fue realizada la modelación de los nodos en los que será distribuida la aplicación, especificando para cada uno de éstos el protocolo de comunicación y descrita la arquitectura en la que se basa la solución del software. Se obtuvo además el diagrama de componentes, elaborando una descripción que indica su propósito, contenido e interfaces.

Conclusiones

A partir de la investigación realizada para la elaboración de este subsistema utilizando el Proceso Unificado de Desarrollo de Software (RUP), para lograr una mejor comprensión de los requisitos de la aplicación y formalización de los mismos, se arriba a las siguientes conclusiones:

- La integridad, autenticidad y confidencialidad de los datos se garantizó, realizando un control estricto de las acciones que llevan a cabo los usuarios en el sistema y estableciendo niveles de acceso en función de las responsabilidades.

- El Subsistema de Gestión de Seguridad propuesto realiza un monitoreo de las incidencias ocurridas en tiempo real (Monitor de Incidencias), posibilitando a los administradores del sistema poder tomar medidas en un breve lapso de tiempo, evitando de esta forma consecuencias indeseables sobre el mismo y quedando preparado para posibles auditorías.

- La utilización de este subsistema contribuye al fortalecimiento del proceso docente en la gestión académica de la UCI, convirtiendo al sistema más sólido, y por tanto incrementa la confianza de los clientes.

- El Subsistema de Gestión de Seguridad se ajusta a las necesidades de la Universidad de las Ciencias Informáticas, y puede ser integrado con otros Sistemas de Gestión Académica del centro.

Recomendaciones

- Implementar el resto de las funcionalidades del subsistema propuesto para mejorar el proceso de gestión de seguridad en el proceso de control docente, como parte de la gestión académica que se lleva a cabo en la Universidad.

- Continuar la ejecución de las próximas iteraciones del proyecto, así como perfeccionar las funcionalidades existentes basada en los problemas detectados durante la fase de implantación.

- En vista a la política de migración hacia software libre llevada a cabo por nuestro país y a la que no se encuentra ajeno nuestra universidad, se recomienda en etapas venideras hacer un profundo estudio sobre la migración paulatina de los módulos que presenta este sistema a software libre.

- La implementación de una bitácora, para de esta forma tener un control más estricto y riguroso sobre los accesos a la aplicación almacenados en los logs, pudiendo determinar en un futuro, intentos maliciosos, con el objetivo de romper la seguridad del sistema o sencillamente dejarlo fuera de servicio.

- La implementación por parte de la Universidad de una infraestructura de clave pública (PKI) basada en los estándares actuales (X.509, SSL, S/MIME) para proporcionar certificados digitales a todos los miembros de la comunidad universitaria de forma fácil y eficaz, obteniendo así comunicaciones autenticadas, íntegras y confidenciales.

Referencias Bibliográficas

[1] Corporación Microsoft. "Arquitectura de aplicaciones de .NET: Diseño de aplicaciones y servicios". Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/default.asp

[2] Corporación Microsoft. "Introducción a Microsoft .Net, Que es .Net" Disponible en: http://www.microsoft.com/latam/net/introduccion/quees.asp

[3] Corporación Microsoft. "¿Qué es ASP.NET?". Disponible en: http://es.gotdotnet.com/quickstart/aspplus/doc/whatisaspx.aspx

[4] González C, Benjamín. "XML: el lenguaje de los Servicios Web" Disponible en: http://www.desarrolloweb.com/articulos/1574.php?manual=54

[5] González Seco, José Antonio "Descripción del nuevo lenguaje de Microsoft C#, vinculado a la plataforma .NET.". Disponible en: http://www.desarrolloweb.com/articulos/561.php.

[6] González Seco, José Antonio "El Nuevo lenguaje de Internet ", última modificación 10/5/2001.

[7] JACOBSON, Ivar, BOOCH, Grady, RUMBAUGH, James. El proceso unificado de desarrollo de software, Pearson Educación S.A., 2000.

[8] Merelo Cuervos, Juan Julián. "Microsoft .Net Servicios Web" Disponible en: http://geneura.ugr.es/~jmerelo/ws/

[9] Microsoft Corporation. "Diseñar aplicaciones distribuidas con Visual Studio .NET". Disponible en: http://www.microsoft.com/spanish/msdn.

[10] Riordan, Rebecca M. "Aprenda programación en Microsoft SQL Server 2000 ya". McGraw-Hill. Interamericana de España.

[11] Seoane, Fabián, García, Eduardo, Sánchez, Alejandro. "MONO Preguntas de Uso Frecuente". Disponible en: http://www.monohispano.org.

[12] Shwarz, John "La importancia de la seguridad en materia empresarial.". Disponible en: http://www.symantec.com

[13] Visual Studio .NET Convined Collection. Microsoft Corporation, 2001.

[14] W3C. "SOAP Specifications". Disponible en: http://www.w3.org/TR/soap/

Bibliografía

1. Ahmed, Meshbah. "ASP.NET Web Developer Guide". Syngress, 2002.

2. Archer, Tom. "A fondo C#". McGraw Hill / Interamericana de España, S.A. España, 2001.

3. Boggs, Wendy; Boggs, Michael. "Mastering UML with Rational Rose 2002". SYBEX Inc., 2002.

4. Cerami, Ethan. "Web Services Essentials". O´Reilly. 1ra Edición, 2002.

5. Daemen, Joan y Rijnmen, Vincent. "The Rijndael Block Cipher". En http://www.rsalabs.com, última modificación 09/04/2003.

6. Dellino, Domenick J. "The Evolution of Security on the Web: An Introduction to Cryptosystems of the Internet", En Visual Studio .NET Convined Collection. Microsoft Corporation, 1996.

7. Fowler Martin, Rice David, Foemmel Matthew, Hieatt Edward, Mee Robert y Stafford Randy. "Patterns of Enterprise Application Architecture". Addison Wesley, November 05, 2002.

8. Hansen, Gary W; Hansen, James V. "Diseño y Administración de Bases de Datos". Prentice Hall. 2da Edición.

9. Jacobson, Ivar; Booch, Grady; Rumbaugh, James. "El Proceso Unificado de Desarrollo de Software". Addison Wesley, 2000.

10. Parihar, Mridula. "La Biblia de ASP.NET". Ediciones Anaya Multimedia, S.A. España, 2002.

11. Ramió Aguirre, Jorge. "Aplicaciones Criptográficas". Universidad Politécnica de Madrid, España. 2da Edición, Junio, 1999.

12. Ramió Aguirre, Jorge. "Seguridad Informática y Criptografía". Universidad Politécnica de Madrid. España. 5ta Edición, Marzo, 2005.

13. Shwarz, John "La importancia de la seguridad en materia empresarial.". Disponible en: http://www.symantec.com

14. Trowbridge David, Mancini Dave, Quick Dave, Hohpe Gregor, Newkirk James, Lavigne David "Enterprise Solution Patterns Using Microsoft .NET". Microsoft Corporation, 1ra Edición, 2003.

15. Turstchi, Adrian. "C# .NET Web Developer Guide". Syngress, 2002.

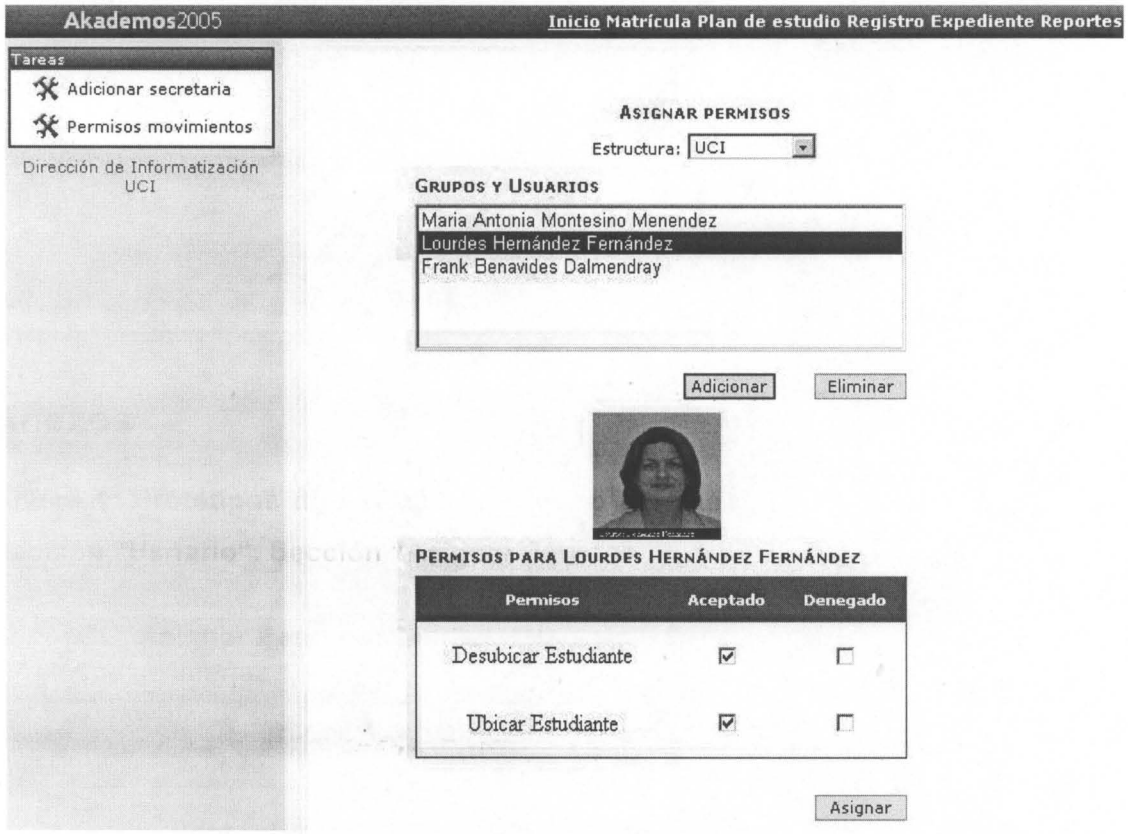
16. Visual Studio .NET Convined Collection. Microsoft Corporation, 2001.

Anexos

Anexo 1: Prototipos de interfaz de usuario del caso de uso "Asignar Permiso". Sección "Usuario". Sección "Asignar Permiso"

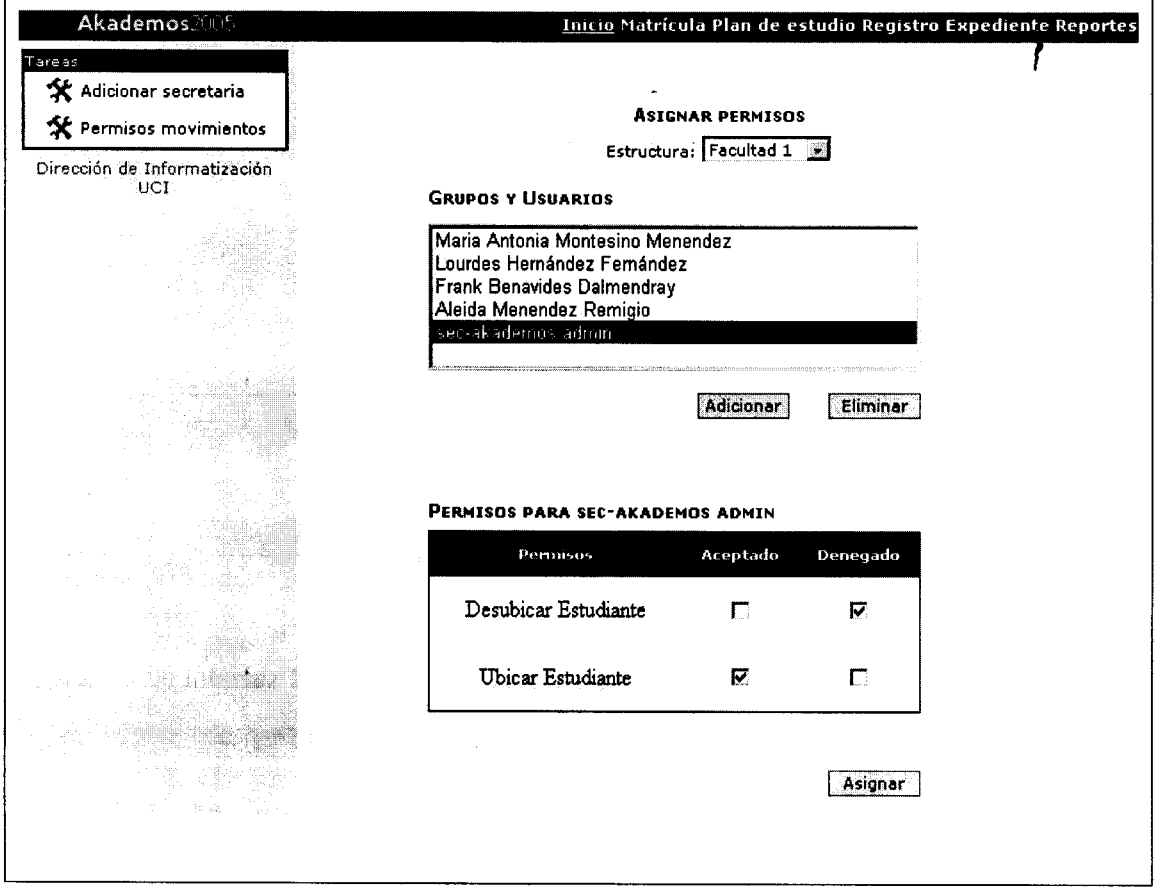
CU: Asignar Permiso. Sección "Usuario". Sección "Asignar Permiso".

Prototipo de interfaz 1



CU: Asignar Permiso. Sección "Usuario". Sección "Asignar Permiso".

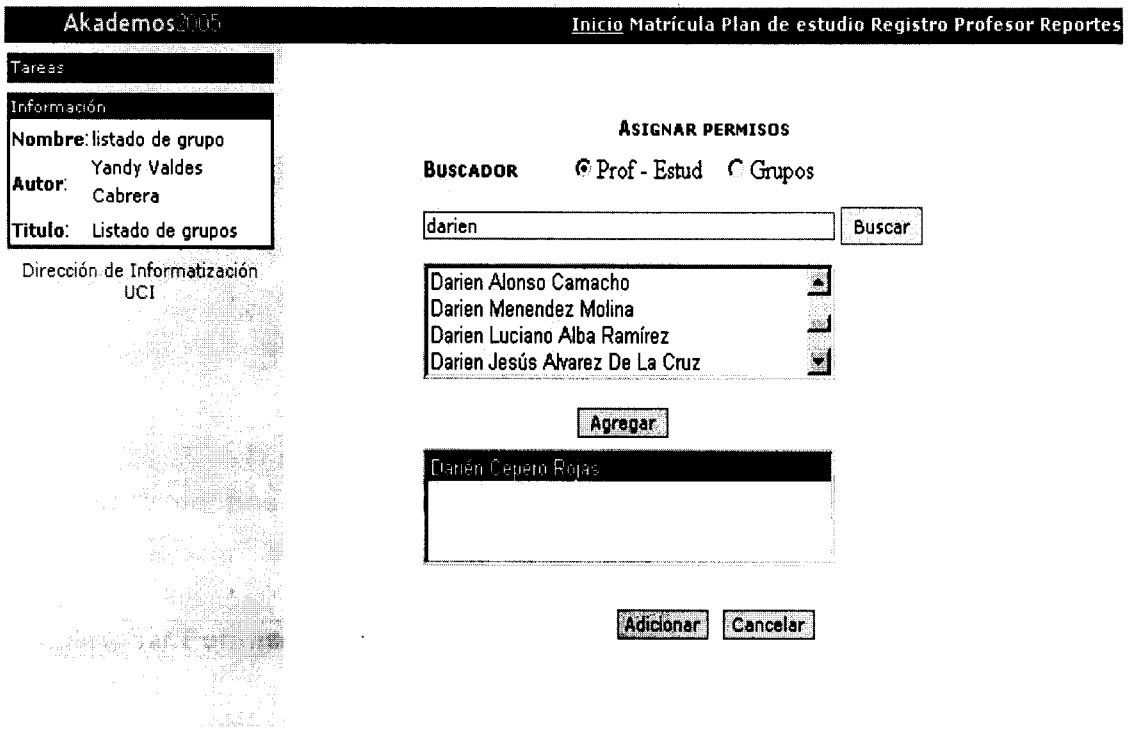
Prototipo de interfaz 2



Anexo 2: Prototipos de interfaz de usuario del caso de uso "Asignar permiso". Sección "Usuario". Sección "Adicionar Usuario"

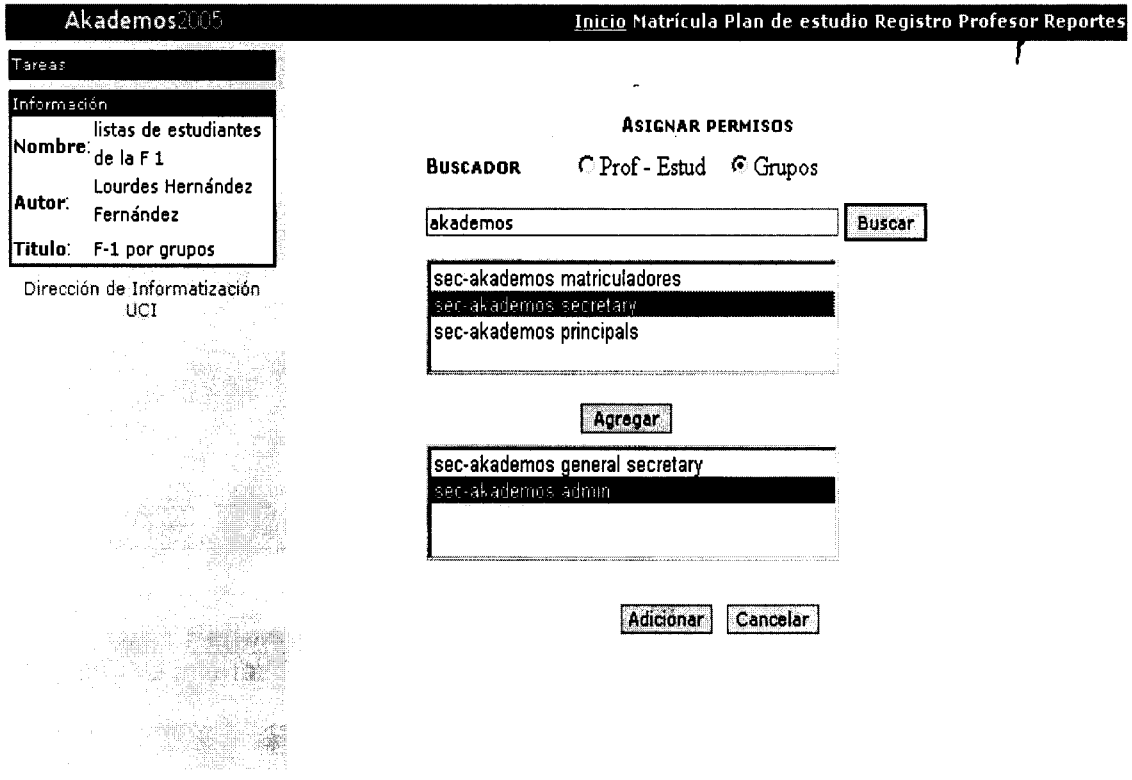
CU: Asignar Permiso. Sección "Usuario". Sección "Adicionar Usuario".

Prototipo de interfaz 1



CU: Asignar Permiso. Sección "Usuario". Sección "Adicionar Usuario".

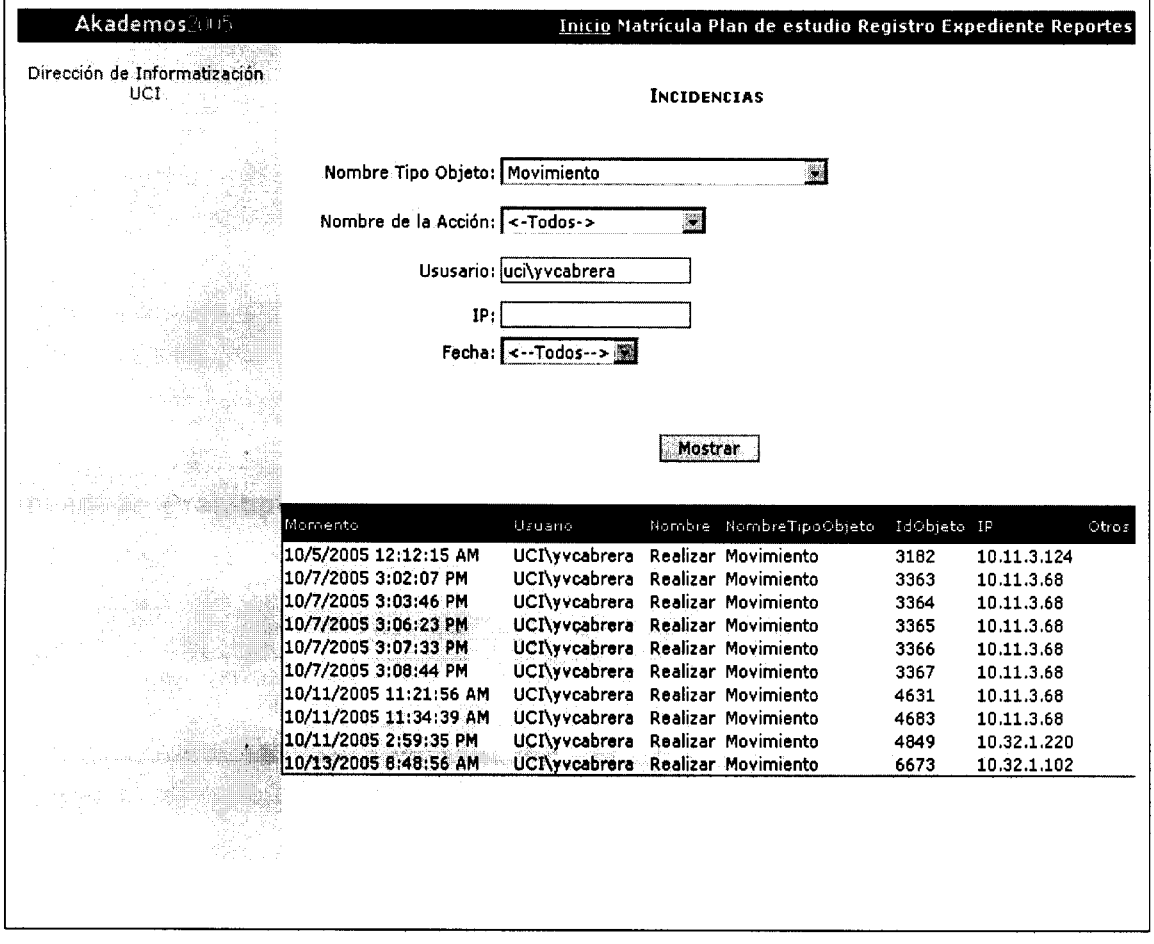
Prototipo de interfaz 2



Anexo 3: Prototipos de interfaz de usuario del caso de uso "Reporte de Incidencias"

CU: "Reporte de Incidencias".

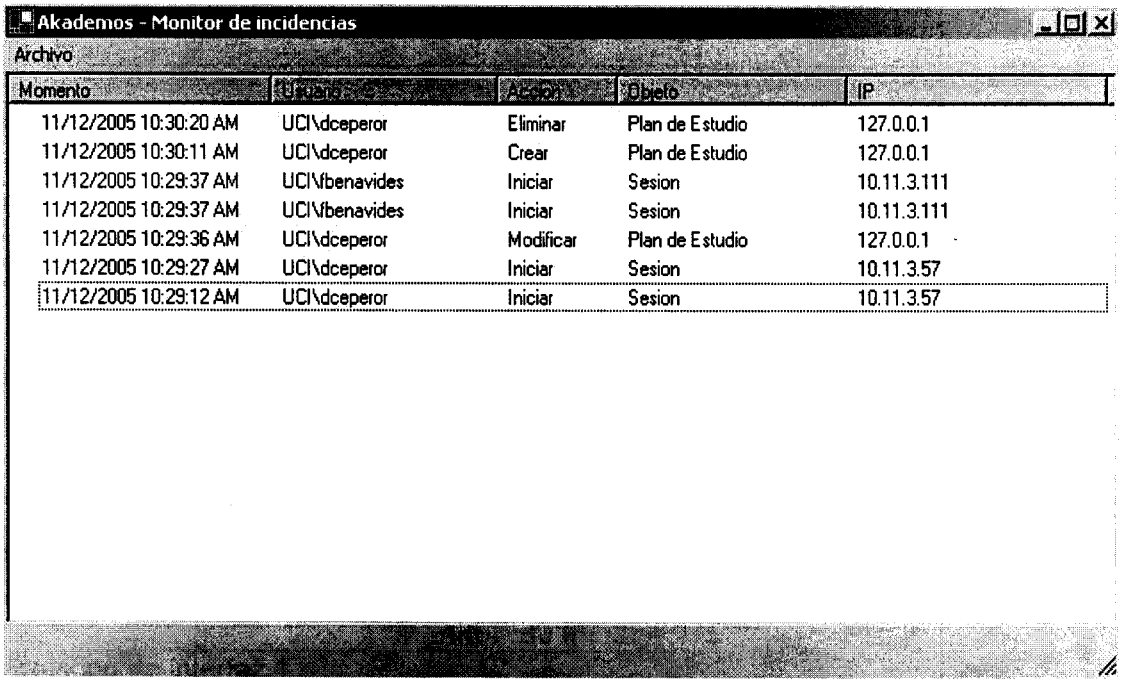
Prototipo de interfaz 1



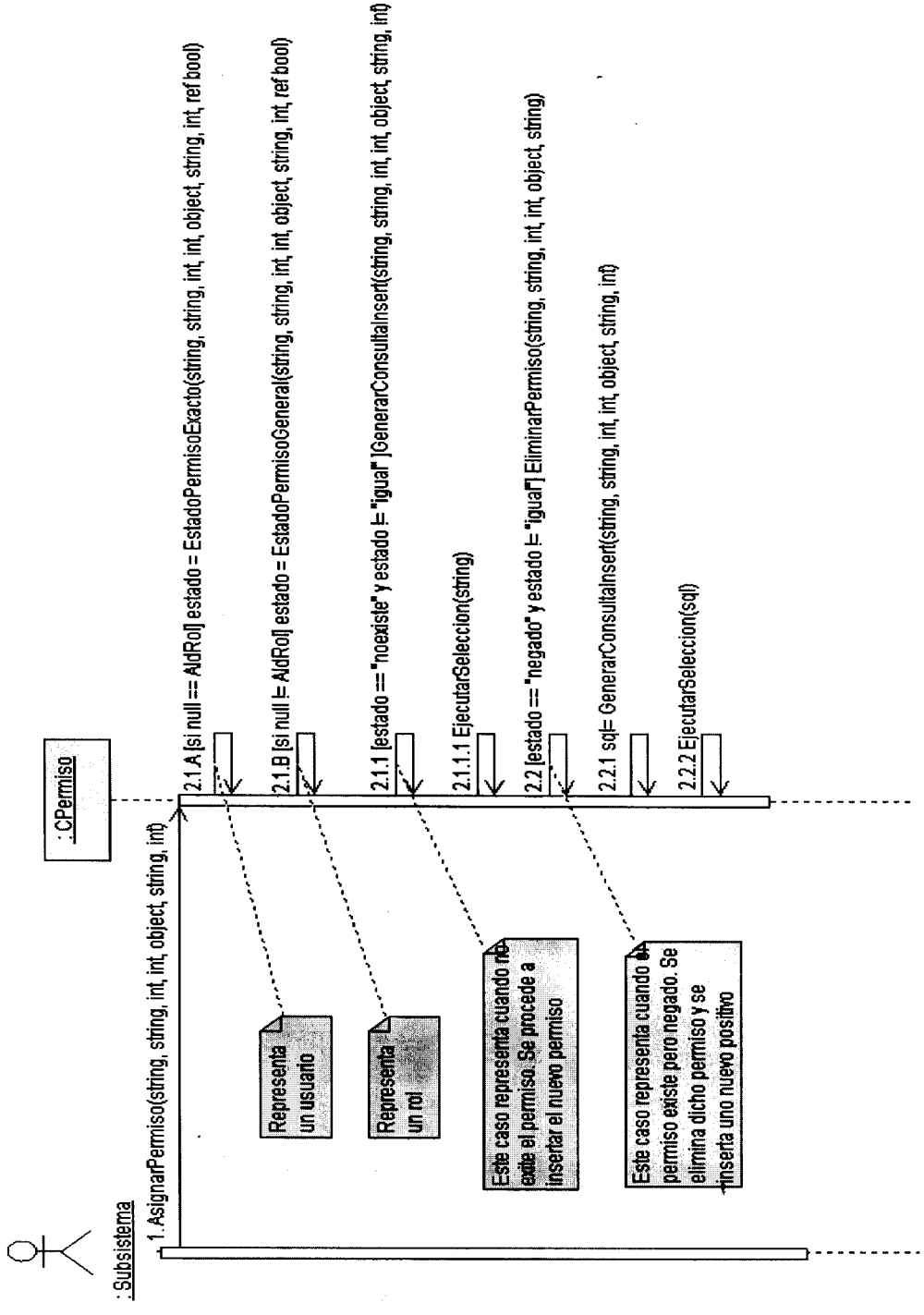
Anexo 4: Prototipos de interfaz de la aplicación de gestión de incidencias en tiempo real "Monitor de Incidencias"

"Monitor de Incidencias".

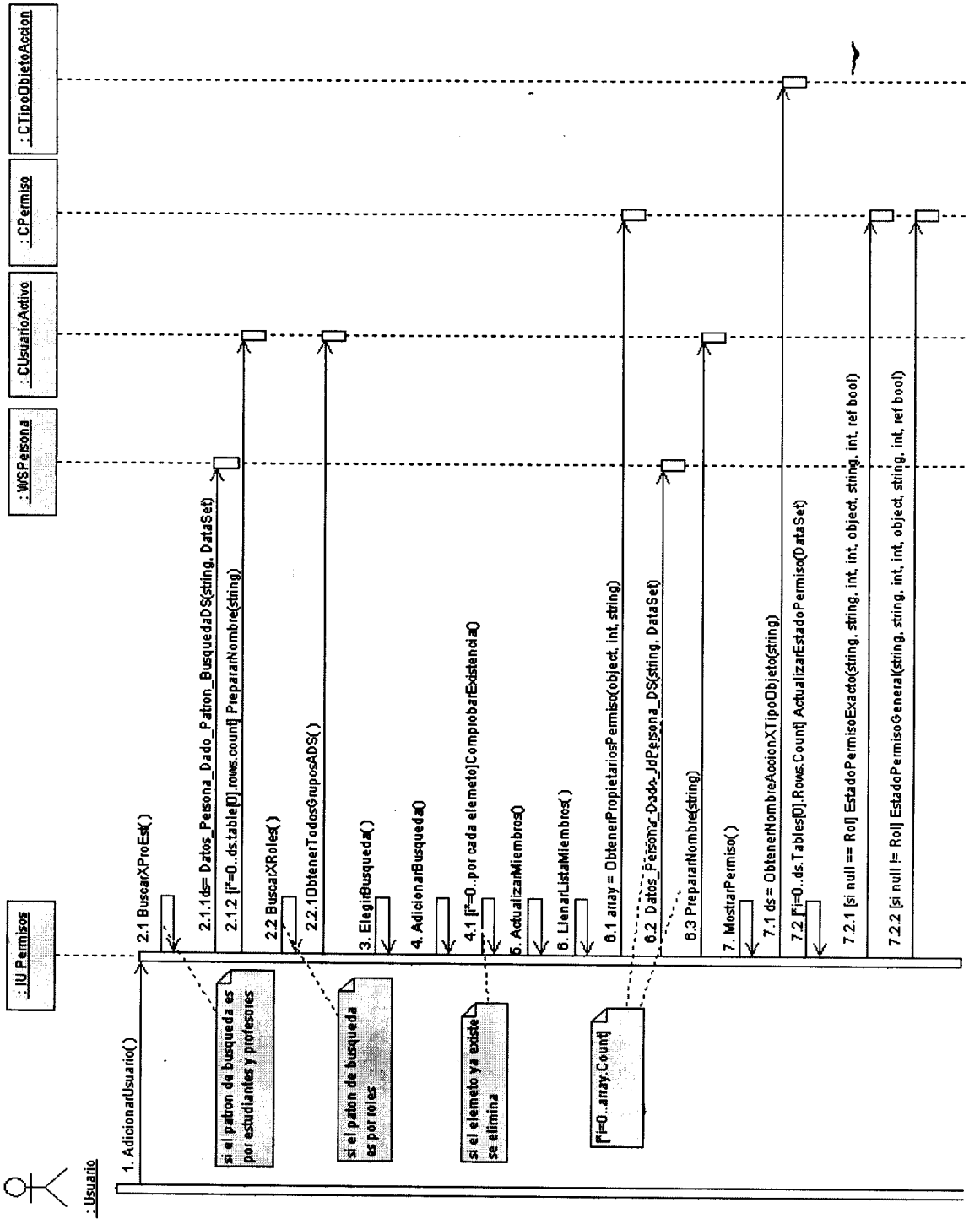
Prototipo de interfaz 1



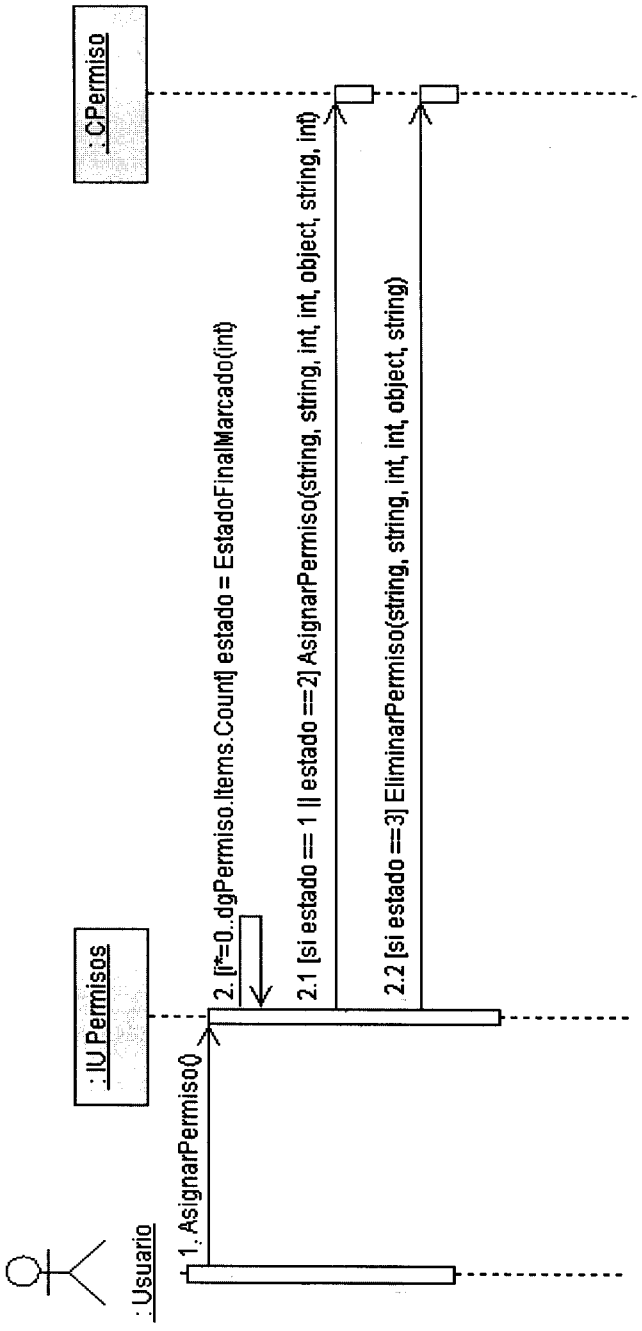
Anexo 5: Diagrama de secuencia del CU "Asignar permiso". Sección "Subsistema"



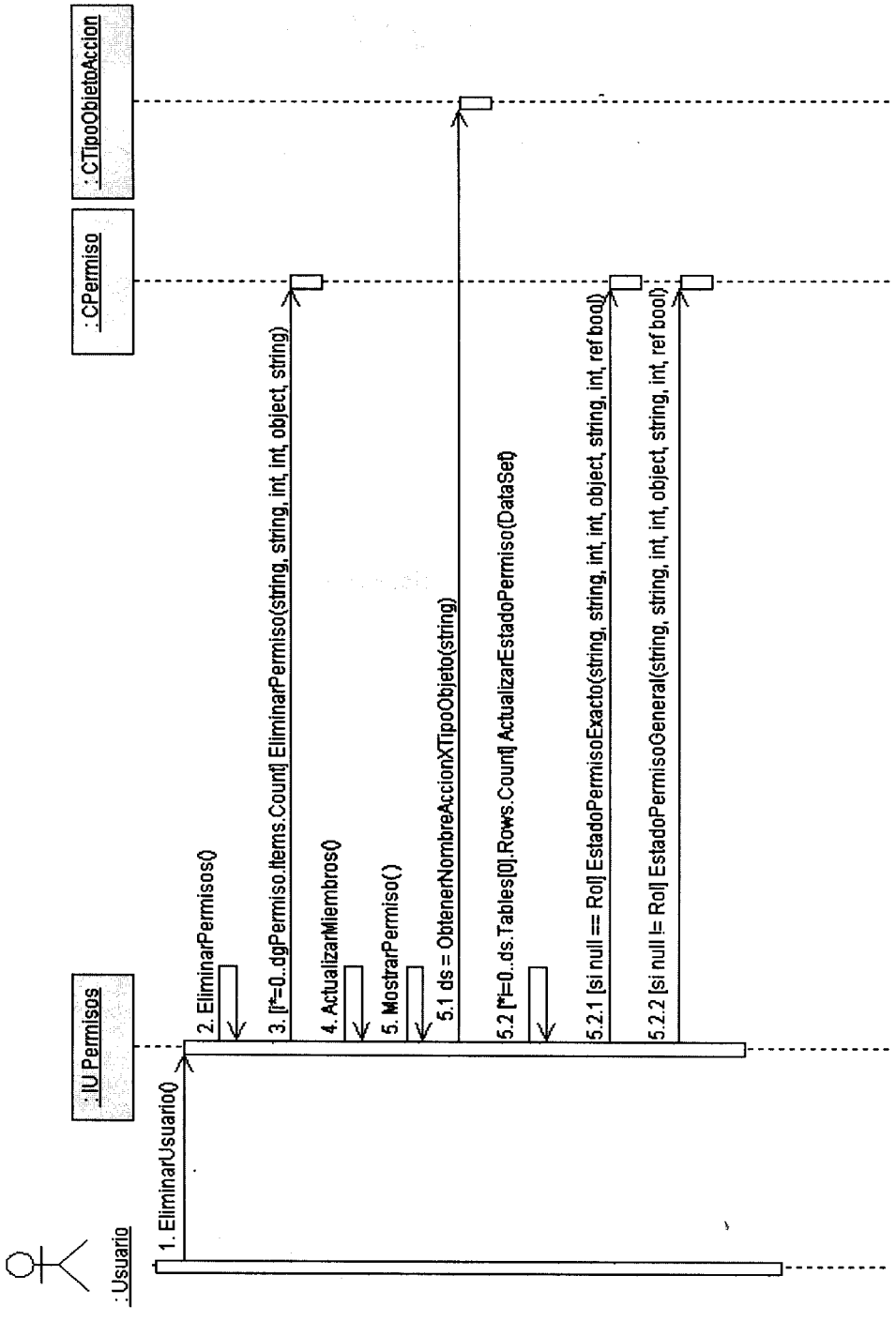
Anexo 6: Diagrama de secuencia del CU "Asignar permiso". Sección "Usuario". Sección "Adicionar Usuario"



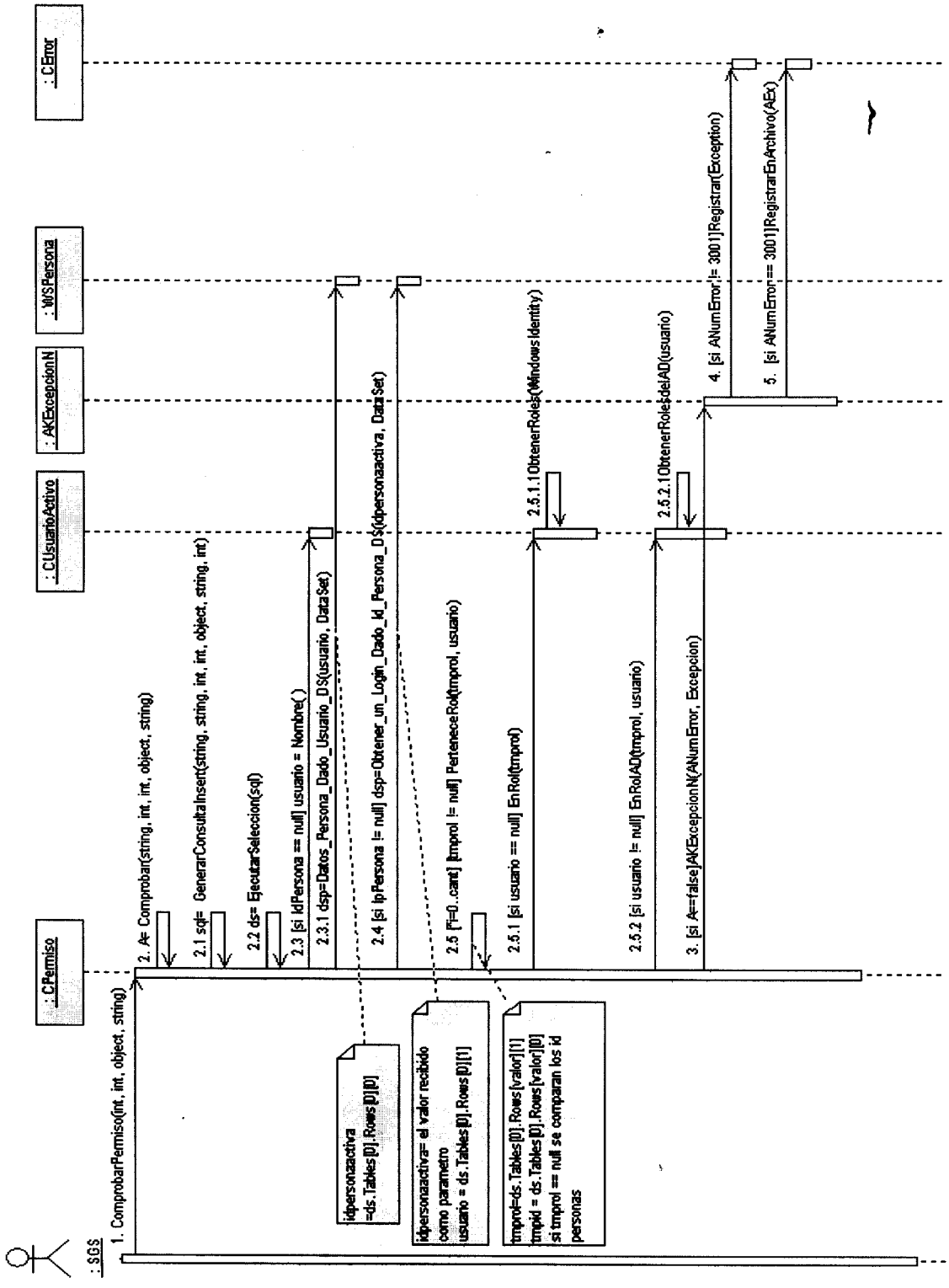
Anexo 7: Diagrama de secuencia del CU "Asignar permiso". Sección "Usuario". Sección "Asignar Permiso"



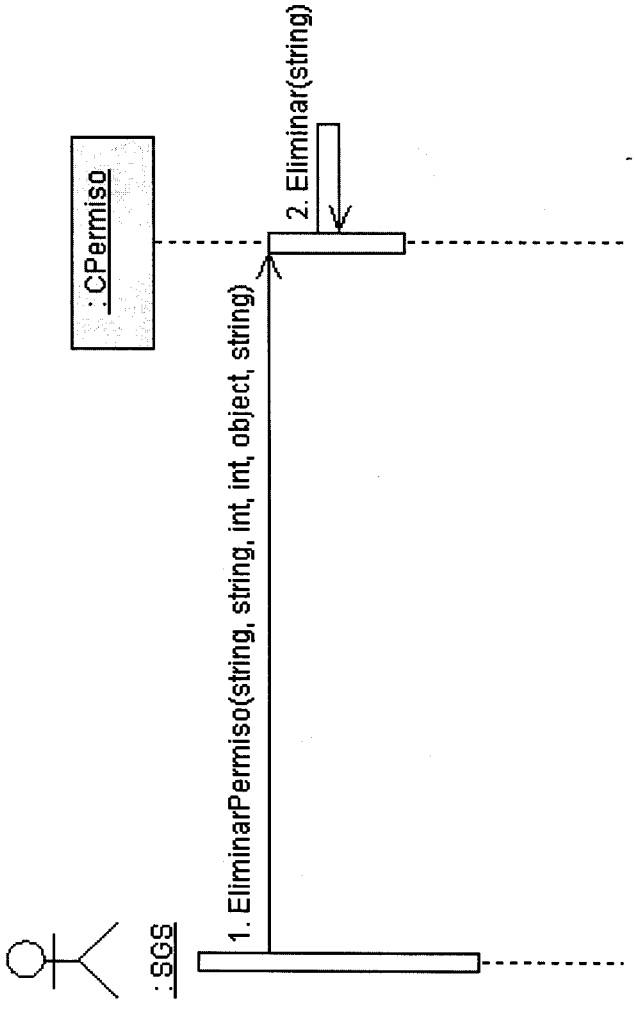
Anexo 8: Diagrama de secuencia del CU "Asignar permiso". Sección "Usuario". Sección "Eliminar Usuario"



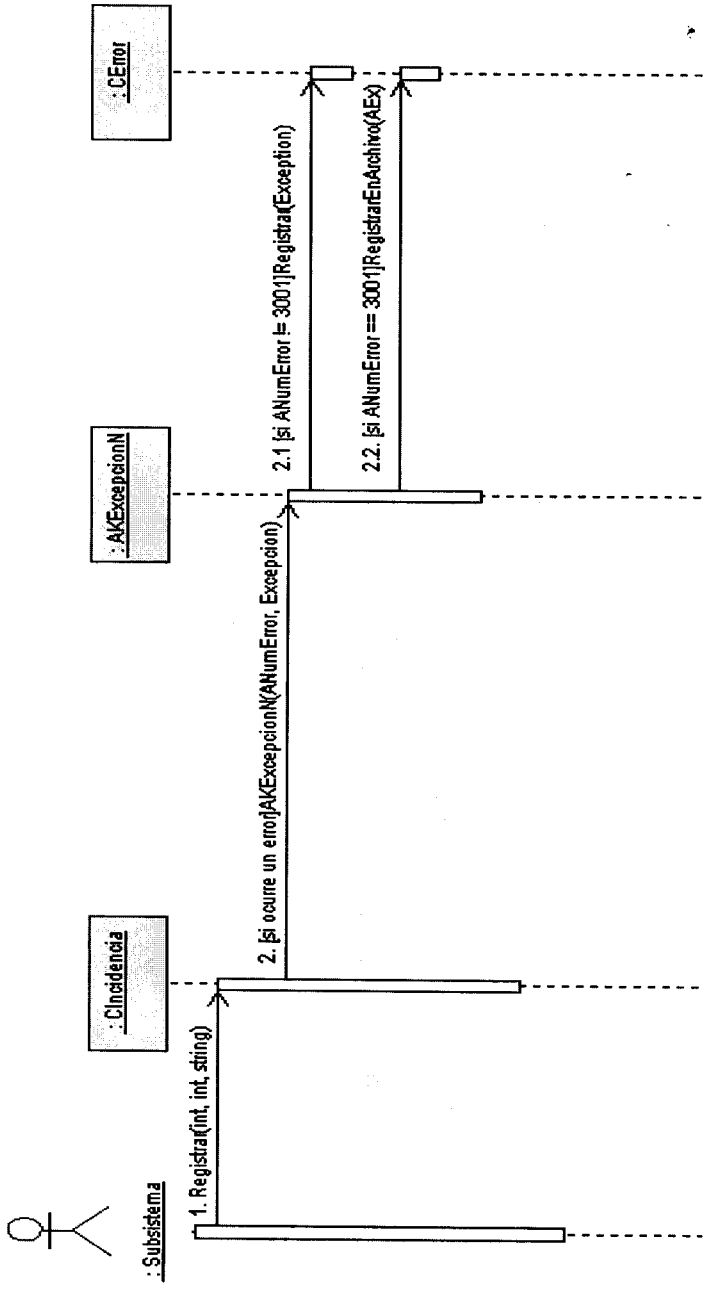
Anexo 9: Diagrama de secuencia del CU "Comprobar Permiso"



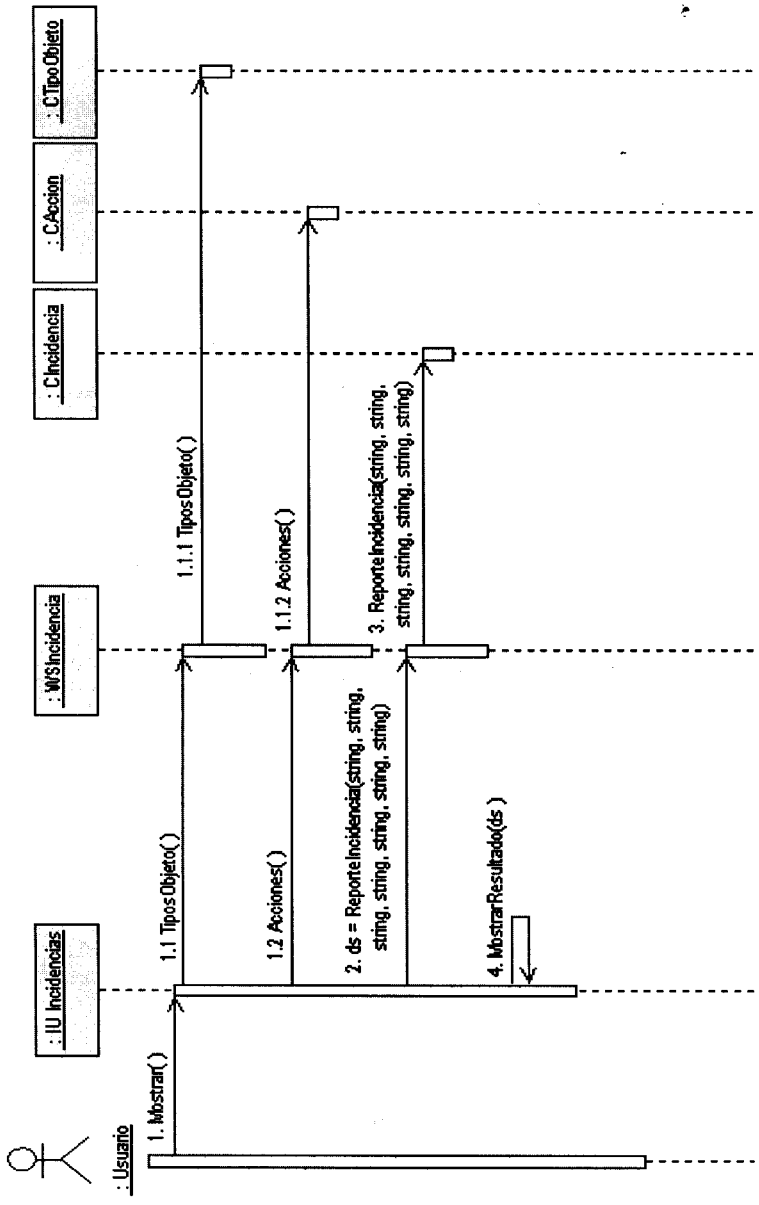
Anexo 10: Diagrama de secuencia del CU "Eliminar Permiso"



Anexo 11: Diagrama de secuencia del CU "Registrar Incidencia"



Anexo 12: Diagrama de secuencia del CU "Reporte de Incidencias"



Glosario de Términos

AKADEMOS: Sistema Automatizado para la Gestión Académica.

CORBA: Es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos. (Common Object Request Broker Architecture).

CU: Caso de Uso.

C++: Es un lenguaje híbrido, que se puede compilar y resulta más sencillo de aprender para los programadores que ya conocen C. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Las principales características son abstracción (encapsulación), el soporte para programación orientada a objetos (polimorfismo) y el soporte de plantillas o programación genérica (templates). Es un lenguaje que abarca tres paradigmas de la programación: La programación estructurada, la programación genérica y la programación orientada a objetos.

Data WareHouse: Se define como un conjunto de datos orientados por tema, integrados, variables en el tiempo y no volátiles que se emplea como apoyo a la toma de decisiones.

DCOM: Modelo de objetos componentes distribuido. Extensiones del Modelo de objetos componentes (COM) que facilita la distribución transparente de objetos a través de redes y de Internet.

Firewall: Es un equipo de hardware o software utilizado en las redes para prevenir algunos tipos de comunicaciones prohibidos por las políticas de red. (Cortafuegos).

GFS: (GrandFather – Father – Son), es una secuencia de respaldo de información de las más utilizadas y consiste en, respaldos completos cada semana y respaldos de incremento o diferenciales cada día de la semana.

HTTP: Protocolo para transferir archivos o documentos hipertexto a través de la red. (HyperText Transmission Protocol).

Identación: Técnica utilizada en programación con el objetivo de mantener claridad en el código escrito por el programador y que resulte fácil su compresión por otras personas.

IIS: Es el servidor Web de Microsoft que corre sobre plataformas Windows. (Internet Information Server).

Mono: Es el nombre de un proyecto de código abierto impulsado por Novell para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET según lo especificado por el ECMA.

MIME: (Multi-Purpose Internet Mail Extensions, Extensiones de correo Internet multipropósito), son una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo) de forma transparente para el usuario. En la actualidad ningún programa de correo electrónico o navegador de Internet puede considerarse completo si no acepta MIME en sus diferentes facetas (texto y formatos de archivo).

Open Source: (Código fuente abierto) Un programa que ofrece al usuario la posibilidad de entrar en su interior para poder estudiarlo o modificarlo, libre acceso al código fuente, que va unido a una serie de conceptos:

- Flexibilidad: Si el código fuente está disponible, los desarrolladores pueden modificar los programas a su antojo. Además, se produce un flujo constante de ideas que mejora la calidad de los programas.

- Fiabilidad y Seguridad: Con varios programadores a la vez escrutando el mismo trabajo, los errores se detectan y corrigen antes, por lo que el producto resultante es más fiable y eficaz que el comercial.
- Rapidez de desarrollo: Las actualizaciones y ajustes se realizan a través de una comunicación constante vía Internet.
- Relación con el usuario: El programador se acerca mucho más a la necesidad real de su cliente, y puede crear un producto específico para él.

RMI: Invocación de métodos remotos, estándar para manejo de objetos distribuidos que es parte de Java. (Remote Method Invocation).

SMTP: Protocolo simple de transferencia de correo electrónico. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras y/o distintos dispositivos. (Simple Mail Transfer Protocol).

SQL Injection: Es el nombre en inglés del ataque contra un Gestor de Bases de Datos Relacional que aprovecha la vulnerabilidad de una aplicación cliente del mismo. Dicha vulnerabilidad consiste en permitir mandar instrucciones SQL adicionales a partir de un campo o un parámetro de entrada, por lo que se dice han sido "inyectadas". La finalidad del ataque es realizar tareas sobre la base de datos, teniendo resultados indeseables e inesperados que van desde la alteración de un dato hasta apoderarse del servidor.

TCP/IP: Protocolo de comunicaciones estándar en Internet. (Transmission Control Protocol/Internet Protocol).

URI: Identificador unificado de recursos. (Uniform Resource Identifier).

URL: Localizador Universal de Recursos. Sistema de identificación en la red, es decir, la dirección en Internet de un sitio determinado. (Universal Resource Locutor).

UNIX: Sistema operativo portable, flexible, potente, con entorno programable, multiusuario y multitarea, muy difundido.

XML: Extensible Markup Language (Lenguaje extensible de etiquetas) Es un meta-lenguaje que permite definir lenguajes de marcado adecuado a usos determinados. Se propone como lenguaje de bajo nivel (a nivel de aplicación, no de programación) para intercambio de información estructurada entre diferentes plataformas.