Universidad de las Ciencias Informáticas

“Análisis y diseño de un sistema para la gestión de versiones en las etapas de despliegue y explotación de los sistemas informáticos”.

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

Autores:

Vitali Herrera Semenets

Angel Islán Leyva Vega

Tutor:

TC. Miriam Rosado Martínez

Tte. Ailema Pérez Medina

Ciudad de La Habana, Junio de 2009

DATOS DE CONTACTO

TC Miriam Rosado Martínez, Graduada de Ingeniera en Sistemas Automatizados de Dirección en la Ciudad Universitaria·"José Antonio Echevarría", CUJAE, en el año 1987. Es Investigadora Auxiliar y Especialista en Informática Operativa. Desde el momento de su graduación ha prestado servicio y transitado por varios cargos técnicos en la especialidad de informática en el Ministerio del Interior. Actualmente ocupa el cargo de Jefe de Sección del Departamento de Servicios Informáticos de la DIC, MININT, y cuenta con 21 años de graduada y de experiencia en la especialidad de informática.

Tte. Ailema Pérez Medina, Graduada con título de oro de Ingeniera en Ciencias Informáticas en la Universidad de las Ciencias Informáticas, UCI, en el año 2008. Actualmente ocupa el cargo de analista en la Sección de Sistemas Informáticos del Departamento de Servicios Informáticos de la DIC, MININT.

La gloria y el tiempo no son más que un estímulo al cumplimiento del deber.

José Martí.

AGRADECIMIENTOS

A Nuestro Comandante en Jefe Fidel Castro Ruz que tuvo la idea de crear este proyecto futuro para nosotros, y para el desarrollo de nuestro país.

A la Universidad de las Ciencias Informáticas, que nos brindó un nuevo camino para cursar nuestros estudios, que nos ha inculcado las ideas de lucha y esmero de ser cada día mejores.

Al profesor Pedro Martinto por sus consejos, su revisión y las conferencias ofrecidas a lo largo del desarrollo del trabajo de diploma.

A Javier Martinez, su ayuda fue fundamental para la terminación del diseño de nuestro sistema.

A nuestras tutoras Ailema y Miriam por su apoyo.

Angel y Vitali

Son muchas las personas que quisiera agradecerles, empezaré por mi familia, a mi mamá por estar siempre preocupada por mis resultados, gracias, se que en ocasiones te quité el sueño.

A mi papá, esa figura que quiero imitar.

A mi hermano, eres el mejor.

A mi tío Osvaldo, gracias por el apoyo y estar siempre pendiente de mí.

A mis amigos, en especial a Jose, Vitali, julio y Victor, son mis hermanos.

A mis compañeros de lucha en el módulo, me marcó la vida conocerlos a todos.

A todos aquellos que me han ayudado a lo largo de la carrera y que ahora sería imposible mencionarlos.

Angel.

Hay muchas personas a las cuales debo agradecer por su apoyo durante toda mi carrera, algunas que están hoy conmigo otras que no están, donde sobresale primero que todo mi familia, mi mamá quien desde pequeño me ha ayudado y apoyado para lograr que yo obtuviera siempre buenos resultados aunque siempre no es posible, pero si me ha inspirado para alcanzar los buenos resultados que he obtenido.

A mi papá quien me ha aconsejado y enseñado a ver las cosas del punto de vista mas correcto y quien también me a apoyado y ayudado en toda mi carrera.

A mi abuela la cual siempre ha querido verme graduado con el título y va a poder tener esa oportunidad.

A mis hermanos Karina y Luisito con los que tanto me divierto y paso gran parte del tiempo.

A mi novia Elaine que tanto apoyo me ha dado, tantos buenos y malos

Momento hemos pasado juntos.

A mi tío Leonel que me ha ayudado bastante para desarrollarme bien en mi carrera

y que siempre me tiene presente.

A mis amigos que han sido como hermanos Angel, Jose, Julio y Victor, y a todo quien de cierta manera me ha ayudado en la carrera que se que son unos cuantos gracias a todos.

Vitali.

DEDICATORIA

A toda mi familia, en especial a mi Mamá, mi Papá y mi hermano.

Angel.

A mi familia, aquella que tengo cerca y a la que tengo un poco más distante, en especial a mi mamá, mi papá, mis abuelas, mis abuelos, mis hermanos y mi novia.

Vitali.

RESUMEN.

Esta investigación consiste en realizar el diseño de un sistema de gestión de versiones en las etapas de despliegue y explotación de los *Sistemas Informáticos*[1] en el *MININT*[2], tal que nos permita una gestión centralizada de los sistemas informáticos que se generalizan y explotan en nuestro entorno garantizando la correcta utilización de las últimas versiones de los sistemas informáticos que se disponen y que cada puesto de trabajo tenga las condiciones necesarias para realizar un trabajo rápido y efectivo.

Con el desarrollo de esta investigación se tiene la intención de dar los primeros pasos para crear y explotar estas ventajas que son tan necesarias en nuestros días, pretendiendo que exista un control sobre los sistemas informáticos en explotación con que cuenta el Ministerio del Interior. El ministerio del interior no cuenta en estos momentos con un sistema que permita llevar a cabo este proceso, teniendo en cuenta lo antes expuesto y el nivel de importancia que tiene para el MININT este trabajo expone una propuesta de diseño para llevar a cabo lo anteriormente analizado.

PALABRAS CLAVES.

Sistema Informático.

[1] Sistemas informáticos: es la interconexión de hardware, software y de un soporte humano.
[2] MININT (**Ministerio del interior**): Órgano que vela por la seguridad del estado y el orden interior.

ÍNDICE DE CONTENIDOS.

ÍNDICE DE FIGURAS Y TABLAS.

INTRODUCCIÓN.

En las últimas décadas, la sociedad ha experimentado un desarrollo tecnológico acelerado. Con el surgimiento de las microcomputadoras es posible el almacenamiento de gran cantidad de información sin la utilización de papel. El desafío que se plantea en las instituciones en Cuba es la informatización de las mismas; para esto deben ser automatizados todos los procesos que en la actualidad se realizan manualmente.

El Ministerio del Interior está avanzando de manera rápida en el campo de la informática y esto lleva consigo el desarrollo de un gran número de software de los cuales se llegan a alcanzar más de una versión, esto implica que en algunos de los entornos donde se encuentran en explotación los sistemas informáticos no se cuente con la versión requerida y esto pueda proporcionar incompatibilidad entre productos que sean del mismo tipo pero de distintas versiones, además se puede llegar a desconocer cual es la versión que se encuentra en explotación en un determinado puesto de trabajo. Sobre los sistemas informáticos se han desarrollado y se explotan muchas versiones, lo que condujo a la idea de crear una aplicación capaz de controlar estas versiones y lograr de esta forma una mayor uniformidad en el MININT.

Un sistema que controle las versiones es fundamental para garantizar la mejor calidad de un producto determinado. Actualmente el ministerio no cuenta con un sistema que le garantice un control de las versiones que se encuentran en explotación lo que afecta directamente la informatización tanto del MININT como del país, por lo que este trabajo consiste en diseñar este sistema, y que con ello se garantice un trabajo efectivo, utilizando las tecnologías adecuadas, de lo contrario solo estaríamos añadiendo mas dificultades a las que diariamente enfrenta el MININT y su personal. Un adecuado control de versiones garantiza entre otras cosas que:

- Los productos que se exploten sean sobre la última versión desarrollada.
- Exista un control total por parte de los administradores de sistemas.
- Se puedan crear reportes con el 100% de veracidad en la información.
- Que productos de la misma versión se puedan desplegar, con una mayor facilidad en cualquier unidad del ministerio.

La *DIC*[3] tiene un Departamento de Servicios Informáticos (DSI) y dentro de este la Sección de Sistemas Informáticos, cuya responsabilidad radica en el despliegue de los sistemas informáticos, así

[3] DIC: Dirección de Informática Comunicaciones y Cifras

como brindar un seguimiento a las versiones instaladas en los distintos centros del MININT. Necesita de un conjunto de reportes para poder determinar y analizar el estado de los distintos sistemas que se explotan, así como el manejo de algunos indicadores, para poder tener al alcance las estadísticas útiles a la hora de analizar un sistema. Estos indicadores son controlados desde los distintos territorios de forma aislada.

En la actualidad, la información, relacionada a sistemas informáticos, que se manejan de los distintos territorios se hace de forma aislada y manual. Por otra parte, el monitoreo de las aplicaciones que se generalizan en la institución se realiza de forma empírica y no centralizada, con ausencia de criterios homogéneos, en algunos casos, ausencia de garantía en las condiciones del despliegue y reaccionando ante problemas, pero sin un seguimiento continuado a la calidad del desempeño.

Es por esta razón que para el desarrollo del buen funcionamiento del trabajo en todas las direcciones y unidades del Ministerio del Interior se hace indispensable la presencia de un Sistema de Control de Versiones (*COVER*[4]) que sea capaz de monitorear los sistemas informáticos que se generalizarán y explotarán en la red del MININT.

Las consecuencias de estos acontecimientos dan lugar a la **situación problémica** que ocupa a nuestro proyecto de investigación científica, cuyo contenido expresaremos a continuación: Dado que en la actualidad no se cuenta con una aplicación capaz de controlar de forma centralizada las versiones de los sistemas informáticos desplegados por todo el país, además existe ineficiencia en el control de la explotación de las versiones de dichos sistemas y el resultado de trabajo obtenido por algunos sistemas no se puede utilizar en todas las regiones del país, surge, en la DIC, la necesidad de crear una software para el control de las versiones de los sistemas informáticos.

Según la situación planteada el **problema científico** a resolver queda formulado de la siguiente manera: ¿Cómo resolver el control de las versiones de los sistemas informáticos, en las etapas de despliegue y explotación, en el entorno del MININT?

Para la solución de la problemática planteada el **objeto de estudio** lo constituye la informatización de los procesos de control y seguimiento de las versiones de los sistemas informáticos, cuyo **campo de acción** queda enmarcado en el análisis y diseño de los procesos de control y seguimiento de las versiones de los sistemas informáticos que se encuentran en las etapas de despliegue y explotación en el MININT.

[4] COVER: Sistema de Control de Versiones

INTRODUCCIÓN

Para llevar adelante la investigación se plantea como **idea a defender** que es posible el desarrollo de un sistema de gestión que logre una mayor efectividad en el control de las versiones en las etapas de despliegue y explotación de los sistemas informáticos.

Como **objetivo general** se propone modelar un sistema para la gestión de las versiones de los sistemas informáticos que se generalizan y explotan en el Ministerio del Interior.

Como métodos de investigación científica se utilizaron:

Métodos Teóricos:

- *Histórico lógico*: Posibilitó el análisis histórico del proceso de gestión de información. Además sirvió para analizar la trayectoria y evolución de la metodología de desarrollo de software y demás herramientas que se utilizan durante el trabajo.
- *Análisis y la Síntesis*: Permitió analizar la bibliografía y realizar síntesis de la misma.
- *Modelación*: Con la utilización de este método se modelaron los diagramas para la implementación de la aplicación Web.

Métodos Empíricos:

- *Entrevistas*: Permitió realizar entrevistas con el fin de precisar el problema a resolver, así como para la validación de la propuesta que se presenta.
- *Análisis de documentos*: Sirvió de guía en la revisión de los documentos utilizados para la investigación de nuestro trabajo de investigación.

Con el propósito de dar cumplimiento al objetivo general y resolver la situación problémica planteada, se proponen las siguientes **tareas investigativas**:

- Desarrollo de entrevistas con los especialistas de la DIC para obtener el modelo de negocio.
- Estudio de los sistemas de control de versiones existentes, que propicien información de interés para la toma de decisiones.
- Realización de un estudio del comportamiento de las tecnologías en el ámbito internacional para lograr la implementación de un sistema de alta calidad.
- Fundamentación de la metodología, herramientas y tecnologías a utilizar en el desarrollo del sistema.
- Construcción de los artefactos requeridos en la ingeniería de software del sistema comprendidos en las fases de análisis y diseño.

- Diseño de la base de datos que almacenará los datos sobre los sistemas informáticos que se generalicen y exploten en el MININT.
- Definición de la función del protocolo y el agente para la intercomunicación entre la información de los sistemas informáticos almacenados en la base de datos y las versiones correspondientes que se explotan en los puestos de trabajo, en todos los entornos involucrados.
- Diseño de un prototipo del sistema respetando la arquitectura definida.

Resultados Esperados:

Al concluir el presente trabajo se debe tener la modelación de una propuesta del Sistema de Control de Versiones (COVER) basado en una Aplicación Web, que constituya una herramienta de apoyo en el proceso de toma de decisiones para la generalización de un sistema informático, que posibilite un mayor control y brinde la información necesaria a los administradores sobre las versiones en explotación.

El trabajo de diploma consta de 3 capítulos más anexos, donde la información está distribuida de la siguiente manera:

El Capítulo 1 contiene la fundamentación teórica del trabajo de diploma. En él se realiza un estudio de las tecnologías actuales para desarrollar software y se hace una evaluación de cuál es la más factible a utilizar en función de las tendencias actuales.

El Capítulo 2 aborda como es el funcionamiento del negocio actualmente, los objetivos estratégicos de la organización y cuáles serán los principales procesos que serán objetos de automatización. Se estudia el entorno donde se implantará el futuro sistema mediante la modelación del negocio. Se hace una propuesta de cómo debe funcionar el sistema, obteniéndose como resultado final de todo este proceso, los requisitos funcionales y no funcionales que darán soporte al futuro sistema a automatizar.

En el Capítulo 3 se desarrollan los distintos diagramas de clases que se generan en el flujo de trabajo análisis y diseño, además se describe la arquitectura definida y los patrones utilizados, así como la seguridad y la interfaz en la aplicación. Se hace una breve descripción del diseño de la base de datos.

Los anexos muestran toda la documentación generada como resultado del flujo de trabajo de análisis y diseño de los procesos de control y seguimiento a la generalización de los sistemas informáticos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 **Introducción.**

En este primer capítulo se desarrollará la fundamentación teórica de la investigación donde se abordarán los principales conceptos y definiciones, los distintos tipos de sistemas de control de versiones que existen, la necesidad de su implementación en el MININT, así como los beneficios que trae consigo su utilización. Además se realiza un estudio de las tecnologías actuales para desarrollar software y se hace una evaluación de cuál es la más factible a utilizar en función de las tendencias actuales.

1.2 Análisis de otras soluciones existentes.

Los sistemas informáticos se construyen con un objetivo, facilitar el proceso masivo y rápido de datos previamente codificados y clasificados. Teniendo en cuenta las disciplinas de seguridad en los ámbitos del diseño, construcción y la explotación de los sistemas de información, existen retos importantes a los que es preciso hacer frente. Desde la identificación y autenticación de los clientes o usuarios de los sistemas, pasando por la arquitectura de las infraestructuras de los terminales, redes, servidores y centros de proceso, hasta el almacenamiento, confidencialidad y respaldo de los datos.

Una versión, revisión o edición de un producto, es el estado en que se encuentra en un momento dado de su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente específico).

El control de versiones se realiza principalmente en la industria informática para controlar las distintas versiones del código fuente. Sin embargo, los mismos conceptos son aplicables a otros ámbitos como documentos, imágenes, sitios web, etcétera. (1)

Actualmente existen en el mundo diferentes softwares que se dedican al seguimiento del control de versiones en los sistemas informáticos. Entre ellas se encuentran *CVS*[5], *SUBVERSION*, *GIT*, *MERCURIAL*, *BAZAAR.*

[5] CVS, del inglés Concurrent Version System (Sistema de Control de Versiones).

1.2.1 Sistema de Control de Versiones (CVS).

CVS es una herramienta que funciona en un esquema cliente y servidor. Existe un cliente o estación de trabajo en donde los desarrolladores hacen modificaciones al código y realizan las pruebas necesarias para satisfacer los requerimientos. En el servidor existe una versión consolidada del proyecto. Cada cierto tiempo los desarrolladores actualizan sus versiones de trabajo desde el servidor y por otra parte envían sus propios cambios hacia el servidor.

Existen servidores para *Linux* y *Unix*, así como clientes para una extensa gama de plataformas (Linux, *Windows*, *MacOSX*[6], etc.).

Uno de los beneficios de este esquema, es que se puede integrar la funcionalidad de cliente CVS en las herramientas de desarrollo integradas (*IDE*[7]). En la plataforma Linux es común que se realice esta integración, debido a lo fundamental que se ha convertido CVS en la mayoría de los proyectos que integran esta plataforma, incluyendo los mismos proyectos de desarrollo de estos IDE's. Es así como por ejemplo podemos encontrar *KDevelop*[8] de *KDE*[9], *Anjuta*[10] y *Eclipse*[11] en *GNOME*[12] integrados (y desarrollados) con CVS.

En la parte del servidor de CVS se maneja un repositorio. Un repositorio es simplemente un directorio en el servidor que contiene diversos módulos. Por ejemplo, un proyecto podría tener un repositorio, y en cada módulo estarían los subproyectos. A su vez, cada módulo contiene un conjunto de archivos, y representa al proyecto con el que se quiere trabajar. Por ejemplo en sitios grandes como *sourceforge*[13], cada uno de los proyectos tiene su propio repositorio CVS, y cada uno de ellos tiene uno o más módulos de trabajo. Cuando un desarrollador trabaja con CVS lo hace a nivel de módulo. Para conectarse al servidor se pueden usar distintos tipos de protocolos, sin embargo los más extendidos son "*pserver*[14]" para acceso anónimo o algún tipo de acceso donde la seguridad no es

[6] Sistemas Operativos: Conjunto de programas que se integran con el hardware para facilitar al usuario, el aprovechamiento de los recursos disponibles

[7] IDE, del inglés Integrated Development Environment (Entorno de Desarrollo Integrado): Es un programa compuesto por un conjunto de herramientas para un programador.

[8] KDevelop: IDE para sistemas GNU/Linux y otros sistemas Unix, orientado al uso bajo el entorno gráfico KDE, aunque también funciona con otros entornos, como Gnome.

[9] KDE, del inglés K Desktop Environment (Entorno de Escritorio K): es un entorno de escritorio e infraestructura de desarrollo para sistemas Unix/Linux.

[10] Anjuta: IDE para programar en los lenguajes C, C++, Java y Python, en sistemas GNU/Linux.

[11] Eclipse: IDE de código abierto multiplataforma para desarrollar Aplicaciones de Cliente Enriquecido.

[12] GNOME: Entorno de escritorio e infraestructura de desarrollo para sistemas operativos Unix/GNU/Linux, compuesto enteramente de software libre.

[13] Sourceforge: Es un software de colaboración para la administración de desarrollos.

[14] Pserver: Es un servidor de acreditación por clave.

importante. El otro es *ssh*[15], y se usa en casos más críticos, por ejemplo cuando se requiere acceso para poder realizar cambios al repositorio.

A pesar de que el uso de CVS permite automatizar la compleja tarea de administración de cambios sobre todo el proyecto, incluyendo la evolución en líneas paralela de un mismo proyecto, CVS como herramienta no resuelve todos los problemas del desarrollo en equipo. Simplemente es una ayuda para que las cosas más tediosas puedan ser automatizadas. Las áreas en donde CVS no está involucrado tienen relación con el tratamiento del proyecto a un nivel global. Actividades como la planificación, los *releases*[16], etc, quedan fuera de su ámbito de trabajo y deben ser abordadas por las personas y otras herramientas complementarias. (2)

1.2.2 Subversion.

Subversion es un sistema de control de versiones *Open Source*[17] diseñado para reemplazar a CVS. Subversion maneja tantos archivos como directorios, y los cambios introducidos en ellos, con el tiempo, le permite recuperar versiones antiguas de sus datos o examinar el historial de cómo cambió sus datos. Por esta característica mucha gente piensa en un sistema de control de versiones como una especie de "máquina del tiempo". Subversion puede operar a través de las redes, lo que le permite ser utilizado por personas en diferentes ordenadores.

Subversion es un sistema general que puede ser utilizado para manejar cualquier colección de archivos, estos ficheros pueden ser código fuente o información de los sistemas.

Subversion es un sistema centralizado para compartir información. En su núcleo está un repositorio, que es un almacén central de datos. El depósito almacena información en forma de un sistema de ficheros es decir un árbol típico de la jerarquía de archivos y directorios. Se pueden conectar al repositorio cualquier número de clientes, y estos a su vez pueden leer o escribir en estos archivos.

Subversion tiene un repositorio capaz de recordar cada cambio que se haga, cada cambio en los archivos, e incluso cambios en el propio árbol de directorios, tales como la adición, supresión, y reorganización de los archivos y directorios. (3)

[15] SSH, del inglés Secure Shell (Intérprete de Ordenes Seguro): Nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red.
[16] Releases (fases de desarrollo): Expresa cómo ha progresado el desarrollo de un software y cuánto desarrollo puede requerir.
[17] Open Source (Código abierto): Permite a los desarrolladores leer, modificar y redistribuir el código fuente.

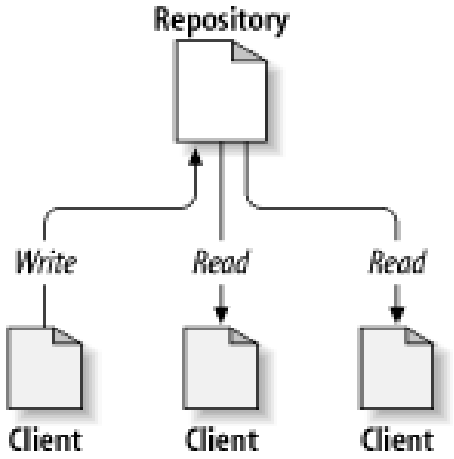


Figura 1 Repositorio Subversion usando arquitectura cliente / servidor.

1.2.3 GIT.

Git es la nueva estrella en ascenso rápido de sistemas de control de versiones. Inicialmente desarrollado por el creador del núcleo de Linux, Linus Torvalds. Git se ha adoptado recientemente en la comunidad de desarrollo web. Git es un tipo muy diferente de control de versiones ya que se trata de un sistema de control de versiones distribuidas, es decir no hay una base de código centralizada para extraer el código. Diferentes ramas ocupan diferentes partes del código. Esto lo hace diferente de otros sistemas de control de versiones, como CVS y SVN que usan un control centralizado de versiones, lo que significa que sólo muestran una copia del software que se utiliza.

Git se caracteriza por ser un sistema rápido y eficaz, y muchos de los principales proyectos de código abierto usan Git para trabajar con sus repositorios, proyectos como: *Kernel*[18] de Linux, *VINO*[19], *Fedora*[20], y muchos otros.

Otra de sus características se basa en que todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio (denominado autenticación criptográfica del historial).

A pesar de su rapidez presenta algunas desventajas como que no es tan fácil de levantar como CVS o SVN, por lo que es mucho más difícil de utilizar para un principiante. También Git expone los hashes como los números de versión para los usuarios, lo que proporciona garantías (un único hash se refiere siempre a un mismo contenido y así un atacante no puede modificar la historia sin ser detectado), pero

[18] Kernel: Se puede definir como el corazón del sistema operativo. Es el encargado de que el software y el hardware puedan trabajar juntos.
[19] VINO: Sistema de administración remota gráfica.
[20] Fedora: Es una distribución de GNU/Linux para propósitos generales.

esto puede resultar engorroso para el usuario. Git tiene un concepto único de seguimiento de los contenidos, pero no realiza un seguimiento a los directorios.

Otro problema con Git es que tiene muchas operaciones (de modificar), lo que hace más fácil de modificar el historial (el contenido de un hash no cambiará nunca, pero las referencias al mismo se pueden perder).

1.2.4 Mercurial.

Se trata de una aplicación multi-plataforma que está desarrollada en Python y cuyas características principales son el alto rendimiento, la escalabilidad, serverless, completamente distribuida, tratamiento robusto de texto plano y ficheros binarios, capacidades avanzadas para ramificaciones y fusiones, e incluye una interfaz web integrada.

Mercurial al igual que Git es otro sistema de control de versiones distribuidas de código abierto. Mercurial fue diseñado para grandes proyectos, muy probablemente fuera del alcance de los diseñadores y desarrolladores Web independientes, aunque esto no significa que los pequeños equipos de desarrollo no pueden o no deben utilizarlo.

Además de ser muy rápido y escalable, Mercurial es un sistema mucho más simple que Git, razón por la cual hace un llamamiento a algunos desarrolladores. No hay una gran variedad de funciones para aprender, y las funciones son similares a las de otros sistemas de control de versiones. También viene equipada con una única interfaz Web y una amplia documentación sobre la comprensión de Mercurial si el usuario ha estado utilizando otro sistema.

1.2.5 Bazaar.

Bazaar es un sistema de control de versiones distribuido que, al igual que CVS o Subversion, permite guardar progresivamente los cambios que se realizan sobre un conjunto de archivos de texto (habitualmente código fuente), recuperar versiones anteriores, mostrar diferencias, integrar el trabajo de diversos programadores, etc.… Bazaar ofrece una experiencia de usuario muy amigable. Se llama a sí mismo "El control de versiones para los seres humanos".

Sin embargo, a diferencia de CVS o Subversion, Bazaar nos permite trabajar de formas mucho más flexibles… desde el típico esquema cliente-servidor hasta la descentralización de los repositorios.

Una de sus principales características es el delicado control que tiene sobre la configuración. Como se indica con los flujos de trabajo, se puede usar para que se adapte a casi cualquier escenario de los usuarios y las configuraciones. Este es un gran sistema de control de versiones para cualquier tipo de proyecto porque es muy fácil de modificar. También es incrustable, lo cual quiere decir que se puede

añadir a los proyectos existentes. Bazaar también tiene una comunidad fuerte, que mantiene cosas como *plug-ins*[21] y muchas de las herramientas de terceros, tales como el software del *GUI*[22] para añadir una interfaz gráfica al sistema.

1.2.5 Sistemas de Control de Versiones en Cuba.

En Cuba se hace uso de sistemas de control de versiones que son usados a nivel mundial tales como Subversion, por ejemplo en la solución arquitectónica para proyectos de software[23] del polo productivo Bioinformática, perteneciente a la Universidad de Ciencias Informáticas (UCI), se tiene trazado dentro de los materiales y métodos para control de versiones el uso de Subversion.

Luego de un análisis y estudio de los sistemas de control de versiones que son utilizados a nivel mundial, se llegó a la conclusión que ninguno de ellos resuelve el problema que planteamos anteriormente, porque todos estos software realizan el control de las versiones en las etapas de implementación y desarrollo y la necesidad que existe actualmente en el MININT es realizar este control en las etapas de despliegue y explotación, por lo que el objetivo es diseñar una aplicación que sea capaz de controlar las versiones de los sistemas que son generalizados en el MININT.

1.3 Protocolos para la Intercomunicacion entre los diferentes modulos del Sistema.

La proliferación de redes de datos durante los últimos años, tanto *LAN*[24] como *WAN*[25], y el interfuncionamiento entre ellas hace que los aspectos relativos a su control y gestión cada vez sean más tenidos en cuenta, convirtiéndose en algo a lo que todos los responsables de redes han de prestar una gran atención. Es por esto que se hace necesario el uso de protocolos que permitan un adecuado control y gestión en las redes, dentro de estos protocolos se pueden destacar algunos como CMIP[26], SNMP, entre otros.

1.3.1 CMIP.

El protocolo CMIP es un modelo de sistema de interconexión abierta que define como crear sistemas comunes de administración de redes.

[21] Plug-in: es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.
[22] GUI, del inglés Graphic User Interface (Uso de Interfaces Graficas): Permite a los usuarios navegar e interactuar con las informaciones en la pantalla de su ordenador.
[23] Software: conjunto de programas elaborados por el hombre, que controlan la actuación del computador.
[24] LAN, del inglés Local Area Network (Red de área local).
[25] WAN, del inglés wide-area network **(**Red de area ancha).
[26] CMIP, del inglés Common Management Information Protocol (Protocolo de Informacion de Administracion Comun)

Algunas compañías telefónicas usan CMIP para la administración de redes públicas. Su estructura se basa en el modelo de administración OSI[27] que define los sistemas administrados. Cada nodo de comunicaciones debe tener una *MIB*[28]. Un *SMAP*[29] proporciona el interfaz con la información compartida MIB. Los SMAPs dialogan con otros SMAPs a través de la red. Una Entidad de SMAE[30] soporta la comunicación de los SMAP y las SMAEs usan CMIP para intercambiar datos entre nodos. CMIP define la metodología para diseñar el sistema de administración de red y las especificaciones del interfaz están indicadas en *CMIS*[31].

Sus funciones incluyen:

- ✓ *Administración Contable:* Proporciona una forma de monitorear el uso de la red para cobrar a los usuarios o para medir los costos y prevenir sobregiros de los presupuestos.
- ✓ *Administración de la Configuración*: Por medio de interfaces gráficos, el administrador puede seleccionar y reconfigurar puentes, enrutadores y otros dispositivos de comunicación.
- ✓ *Administración de Fallas:* Detecta y corrige fallas en la red. Analiza aspectos que ayudan a determinar las causas de falla. Dispone de alarmas para alertar a los administradores.
- ✓ *Administración del Comportamiento*: Proporciona servicios para monitorear la red y mejorar su comportamiento. Llevar estadísticas es uno de los elementos esenciales para administrar el comportamiento de las redes.
- ✓ *Seguridad*. Proporciona servicios de seguridad de alto nivel que puede autentificar a los usuarios, detectar intrusiones y asegurar la confidencialidad en la transmisión de datos. (4)

1.3.2 SNMP[32].

Es un protocolo que permite a los administradores de red administrar dispositivos de red y diagnosticar problemas en la red. El SNMP se basa en dos elementos principales: un supervisor y un agente. El supervisor es el terminal que le permite al administrador de red realizar solicitudes de administración. Los agentes son entidades que se encuentran al nivel de cada interfaz. Ellos conectan a la red los dispositivos administrados y permiten recopilar información sobre los diferentes objetos. Los

[27] OSI, del inglés Open Systems Interconnection (Interconexion de Sistemas Abiertos): Es un lineamiento funcional para tareas de comunicaciones.
[28] MIB, del inglés Management Information Base (Base de Información de Administración): Es un tipo de base de datos que contiene información jerárquica, de todos los dispositivos gestionados en una red de comunicaciones.
[29] SMAP (Proceso de Aplicación del Sistema de Administración).
[30] SMAE (Aplicación del Sistema de Administración)
[31] CMIS, del inglés Common Management Information Service (Servicio de Información de Administración Común): Ordenador en red estándar para la gestión común de servicios de información.
[32] SNMP, del inglés Simple Network Managemet Protocol (Protocolo Simple de Administracion de Red).

conmutadores[33], *concentradores*[34], *routers*[35] y *servidores*[36] son ejemplos de hardware que contienen objetos administrados. Estos objetos administrados pueden ser información de hardware, parámetros de configuración, estadísticas de rendimiento y demás elementos que estén directamente relacionados con el comportamiento en progreso del hardware en cuestión. Estos elementos se encuentran clasificados en algo similar a una base de datos denominada MIB. SNMP permite el diálogo entre el supervisor y los agentes para recolectar los objetos requeridos en la MIB.

La arquitectura de administración de la red propuesta por el protocolo SNMP se basa en tres elementos principales:

- ✓ *Los dispositivos administrados:* Son los elementos de red (puentes, concentradores, routers o servidores) que contienen "objetos administrados" que pueden ser información de hardware, elementos de configuración o información estadística;
- ✓ *Los agentes:* Una aplicación de administración de red que se encuentra en un periférico y que es responsable de la transmisión de datos de administración local desde el periférico en formato SNMP.
- ✓ *El sistema de administración de red: Es* un terminal a través del cual los administradores pueden llevar a cabo tareas de administración.

Para el correcto funcionamiento del COVER se hace imprescindible el uso de un protocolo de gestión de red y el que más cubre nuestras necesidades dadas sus características es SNMP a partir del cual pudiéramos hacer el diseño de un protocolo específicamente para nuestro sistema, ya que aporta ciertas funcionalidades que se manifiestan en tres áreas particulares que son de gran interés para el COVER, dichas áreas son: seguridad (autentificación, privacidad y control de accesos), transferencia de datos y comunicaciones Administrador a Administrador.

Los cinco tipos de mensajes SNMP intercambiados entre los Agentes y los Administradores, son:

- ✓ *Get Request: Es* una petición del Administrador al Agente para que envíe los valores contenidos en el MIB (base de datos).
- ✓ *Get Next Request*: Es una petición del Administrador al Agente para que envíe los valores contenidos en el MIB referente al objeto siguiente al especificado anteriormente.

[33] Conmutadores: Su función es interconectar dos o más segmentos de red.
[34] Concentradores, del inglés hubs: Es un dispositivo que permite centralizar el cableado de una red y poder ampliarla.
[35] Router: Permiten interconectar tanto redes de área local como redes de área extensa.
[36] Servidores: Aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes.

- ✓ *Get Response:* Es la respuesta del Agente a la petición de información lanzada por el Administrador.
- ✓ *Set Request:* Es una petición del Administrador al Agente para que cambie el valor contenido en el MIB referente a un determinado objeto.
- ✓ *Trap:* Es un mensaje espontáneo enviado por el Agente al Administrador, al detectar una condición predeterminada, como es la conexión/desconexión de una estación o una alarma. (5)

Este tipo de intercambio permitirá detectar cuando no se está explotando la versión correcta en cada uno de los entornos donde se encuentran en explotación los sistemas informáticos, y proveerlos a su vez, de la versión requerida, de igual manera, favorece con el uso de los agentes, a implementar alertas, anuncios y avisos por la red, tal que permitan la comunicación de los problemas relacionados con la explotación de las últimas versiones.

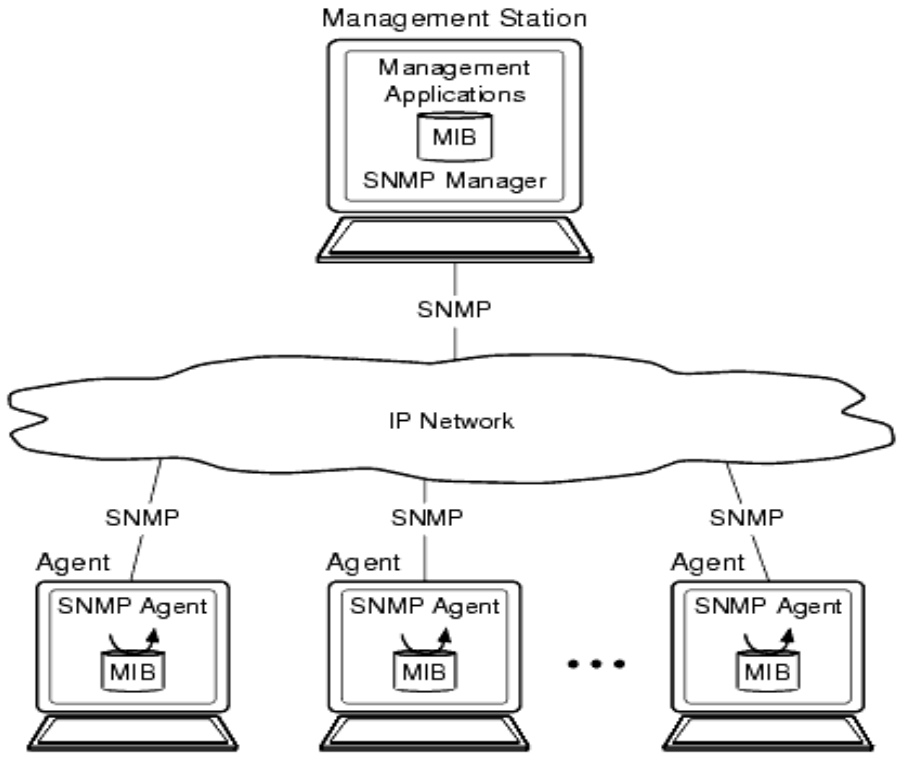


Figura 2 Estructura de comunicación del protocolo SNMP.

Tanto CMIP como SNMP definen estándares de administración de redes pero CMIP es más complejo y proporciona diversas características que deben observar los administradores de red.

1.3.3 Obtención de información.

El estudio del protocolo SNMP y CMIP sirvió de base para el conocimiento de la definición del protocolo que utilizaremos en la comunicación de nuestro sistema con el agente. Dicho protocolo será el encargado de transportar la información por la red y puede llegar a ser parecido en algunos aspectos al SNMP.

Por otra parte el agente instalado en las máquinas se encargará de encuestar los sistemas informáticos en explotación obteniendo de esta forma los datos sobre las versiones que se encuentran en explotación y brindará la información requerida al Sistema de Gestión de Versiones una vez realizada la consulta por este.

1.4 Tendencias Tecnológicas.

1.4.1 Rational Unified Process (*RUP*[37]).

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Estas surgen para mejorar el desarrollo de sistemas, según el tipo de proyecto y empresa, añadiendo y optimizando las prácticas de desarrollo de software. Actualmente existen y son usadas muchas metodologías de desarrollo de software a nivel mundial como *XP*[38], *MSF*[39], *OMT*[40], METRICA versión 2.1, *SSADM*[41] versión 4, HATLEY, MERISE, RUP entre muchas otras. Después de haber hecho un estudio de algunas de las metodologías existentes llegamos a la conclusión de que la más factible a utilizar es RUP.

RUP es un proceso de desarrollo de software y junto con el *UML*[42], constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado. Proporciona una guía en el orden de las actividades de un equipo, dirige las tareas individuales de los desarrolladores, especifica qué productos deberían ser desarrollados y ofrece criterios para monitorear y medir los productos y actividades del proyecto.

[37] RUP, del inglés Rational Unified Process (Proceso Unificado de Desarrollo).
[38] XP, del inglés Extreme Programing (Programación Extrema).
[39] MSF, del inglés Microsoft Solution Framework (Plataforma de Solución de Microsoft).
[40] OMT, del inglés Object Modeling Technique (Técnica de Modelado en Objeto).
[41] SSADM, del inglés Structured Systems Analysis and Design Method (Los Métodos de Análisis y Diseño de Sistemas Estructurados).
[42] UML, del inglés Unified Modeling Language (Lenguaje Unificado de Modelado).

Su objetivo es asegurar la construcción de sistemas de software de alta calidad que satisfagan las necesidades de los usuarios finales y clientes cumpliendo con los cronogramas y presupuestos previstos. Es adecuada para sistemas con extensos cronogramas y equipos de desarrollo numerosos.

RUP es un proceso que se caracteriza por ser:

- ✓ *Dirigido por casos de uso.* Los casos de uso describen los requisitos funcionales del sistema desde la perspectiva del usuario y se usan para determinar el alcance de cada iteración y el contenido de trabajo de cada persona del equipo de desarrollo.
- ✓ *Centrado en la arquitectura.* La arquitectura permite ganar control sobre el proyecto para manejar su complejidad y controlar su integridad. Hace posible la reutilización a gran escala y provee una base para la gestión del proyecto.
- ✓ *Iterativo e incremental.* Se divide en 4 fases: Inicio, Elaboración, Construcción y Transición, y cada una de ellas se divide en iteraciones. En cada iteración se trabaja en un número de disciplinas haciendo énfasis en algunas de ellas. Las disciplinas propuestas por RUP son: *Modelado del negocio*, *Requisitos*, *Análisis y Diseño*, Implementación, Pruebas, entre otras. Cada iteración añade funcionalidades al producto de software o mejora las existentes. (6)

En la Figura 1 se muestran los flujos de trabajo y las iteraciones que define RUP por las cuales debe transitar el desarrollo de un producto software. Además RUP identifica una serie de roles para los trabajadores que realizan las actividades de cada flujo, dentro de los principales roles se encuentran: ingeniero de requisitos, diseñador, arquitecto, implementador, jefe de proyecto y probador.

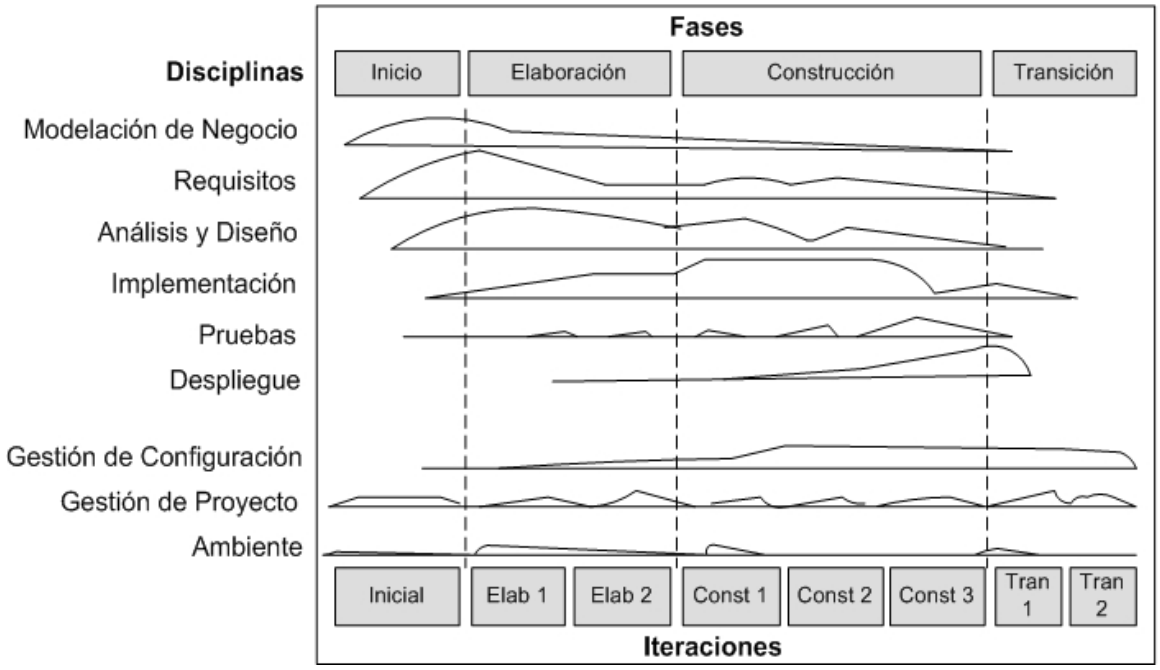


Figura 3 Fases y flujos de trabajo de RUP.

Los flujos de trabajo representados en la Figura 1 proponen la realización de un gran número de actividades y la elaboración de un amplio conjunto de artefactos, que usualmente los proyectos de desarrollo de software se comprometen a desarrollar pero que no realizan en su totalidad debido a la carencia de tiempo o a que descubren que no eran necesarios. RUP indica que al inicio del proyecto se realice una adecuación de cada flujo de trabajo de manera que se produzcan solo los artefactos y se realicen las actividades que tienen un propósito dentro del proyecto.

A continuación se describen las disciplinas de mayor interés de acuerdo a los roles desempeñados por los autores para la realización del trabajo de diploma.

Disciplina Análisis y Diseño.

El objetivo de la disciplina Análisis y Diseño es transformar los requisitos de software en un diseño del sistema, desarrollar una arquitectura robusta y adaptar el diseño al ambiente de implementación. Los principales artefactos que propone RUP para esta disciplina son: Modelo de Análisis, Modelo de Diseño, Modelo de Datos y Modelo de Despliegue. Las entradas fundamentales de esta disciplina son las especificaciones de los requisitos de software obtenidas en la disciplina Requisitos. Los trabajadores más relevantes involucrados en este flujo son el diseñador y el arquitecto.

El Modelo de Diseño es detallado en cuanto a la arquitectura porque debe ser implementado sin ambigüedad. Este modelo es esencial para las actividades de implementación y pruebas. Contiene subsistemas, paquetes, clases de diseño y diagramas de clases e interacción.

El Modelo de Datos describe la representación lógica y física de los datos persistentes utilizados por la aplicación. En casos de aplicaciones que usen sistemas gestores de bases de datos relacionales, este modelo debe incluir representaciones para procedimientos almacenados, disparadores, etcétera.

El modelado del portal ejecutivo se regirá por esta metodología ya que es la más recomendada para proyectos de gestión de este tipo, con gran número de procesos a desarrollar, por las ventajas y características expuestas anteriormente. Además es de la cual se posee mayor conocimiento debido a que es impartida y estudiada a lo largo de toda la carrera.

1.4.2 UML.

UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por el *OMG*[43]. Es un lenguaje gráfico para visualizar, especificar, construir y documentar cada una de las partes que conforman un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. (7)

EL lenguaje UML estandariza los artefactos y la notación, pero no define un proceso oficial de desarrollo, debido a:

- ✓ Aumenta las probabilidades de una aceptación generalizada de la notación estándar del modelado, sin la obligación de adoptar un proceso final.
- ✓ La esencia de un proceso apropiado admite mucha variación y depende de las habilidades del personal, de la razón investigación-desarrollo, de la naturaleza del problema, de las herramientas y de muchos otros factores. (8)

1.4.3 Visual Paradigm.

Visual Paradigm es una herramienta que facilita el diseño visual de diagramas. Esta herramienta le permite al equipo de desarrollo, realizar el modelo del ciclo de vida completo del proceso de desarrollo de software.

[43] OMG, del inglés Object Management Group (Grupo de Gestion de Objetos): Se encarga de la definición y el mantenimiento de estándares relacionados con la programación orientada a objetos.

Visual Paradigm soporta un conjunto de lenguajes, tanto en la generación de código, como en la ingeniería inversa en Java, C + +, CORBA IDL, PHP, XML Schema, Ada y Python, además soporta la generación de código en C #, VB. NET, *ODL*[44], Flash ActionScript, Delphi, Perl, Objective-C, y Ruby.

A diferencia de las herramientas de modelado en el mercado, Visual Paradigm emplea una respuesta rápida y bajos requisitos de memoria del motor de persistencia, lo que le permite manejar grandes y complicadas estructuras de un proyecto en una forma altamente eficiente y sólo requiere de una configuración de escritorio.

Visual Paradigm soporta la importación y exportación de las versiones 1.0, 1.2 y 2.1 de XMI. Los archivos de proyecto (.MDL / .CAT) de *Rational Rose*[45] también pueden ser importados en Visual Paradigm. Otra de las grandes ventajas del Visual Parading es que permite la importación y exportación de los proyectos en formato XML abierto.

1.4.4 Microsoft Share Point 2007.

Office SharePoint Server 2007 no es solamente una herramienta de colaboración de Microsoft, sino toda una plataforma de desarrollo debido a su flexible arquitectura y poderoso Modelo de Objetos. Este utiliza la organización para facilitar la colaboración, proporcionar características de administración del contenido, implementar procesos empresariales y dar acceso a la información imprescindible para los objetivos y procesos de la organización.

Mediante el uso de las plantillas del sitio y de otras características de Office SharePoint Server 2007, se pueden crear, fácil y eficazmente, sitios que admitan la publicación de contenido específico, administración del contenido, administración de registros o satisfacer las necesidades de la inteligencia empresarial que pueda tener la organización. Por ejemplo, se pueden crear sitios en el nivel de empresa, como portales de la organización, sitios en Internet o sitios especializados, como repositorios de contenido o áreas de reuniones. Estos sitios le permiten colaborar y compartir información con otros usuarios que estén tanto dentro como fuera de la organización.

[44] ODL, del ingles Object Definition Language (Lenguaje de Definicion de Objetos)
[45] Rational Rose: Poderosa herramientas de modelado visual para el análisis y diseño de sistemas basados en objetos. Cubre todo el ciclo de vida de un proyecto.

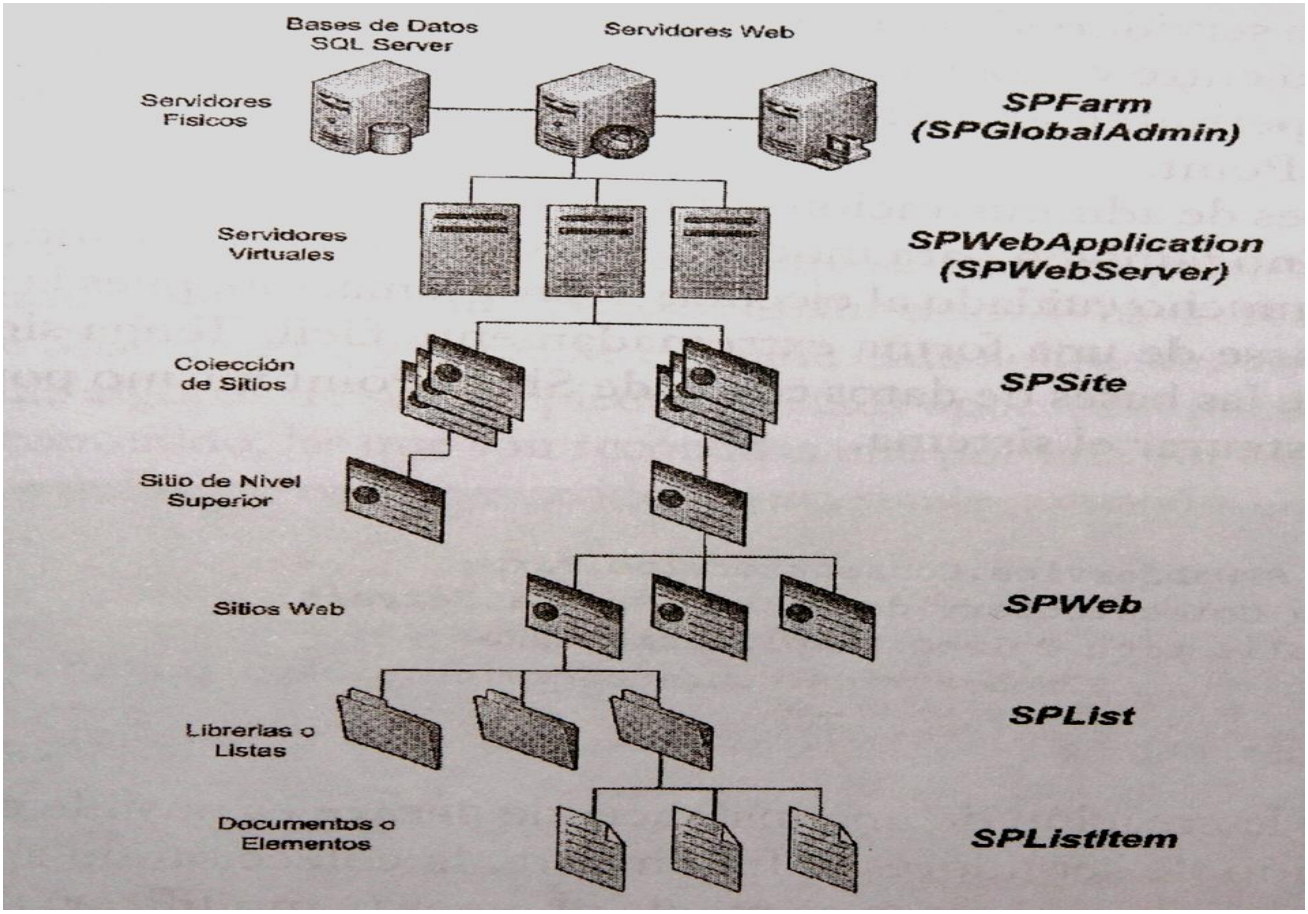


Figura 4 Arquitectura de SharePoint.

Utilizando Office SharePoint Designer 2007 se pueden crear e implementar soluciones interactivas en la plataforma SharePoint, sin necesidad de escribir código.

- ✓ Automatizar procesos de control empresariales como la aprobación de documentos.
- ✓ Crear aplicaciones de seguimiento y elaboración de informes mediante vistas de datos y formularios para recopilar y agregar fácilmente datos tanto de otros sitios como de listas de SharePoint y bibliotecas de documentos de su sitio Web.
- ✓ Tiene plantillas de aplicación integrada de Microsoft Windows SharePoint Services, totalmente personalizables y ampliable para permitir el trabajo rápido y eficaz.
- ✓ Permite ampliar las soluciones creando páginas avanzadas e interactivas de Microsoft ASP.NET.

- ✓ Inserta y edita controles con la eficacia de los mismos menús de actividad y cuadrículas de propiedades de control que anteriormente sólo se encontraban en herramientas de desarrollo como Microsoft Visual Studio 2005. (9)

1.4.5 Oracle.

Oracle es un sistema de administración de base de datos líder en la industria, utilizable para almacenar todo tipo de datos de negocio, incluyendo datos relacionales, documentos, multimedia, XML y datos de localización. Es fácil de desarrollar y administrar

Oracle es considerado uno de los Sistemas Gestores de Bases de Datos más completos, el cual representa una opción factible y potente para la información que se manejará en el sitio, para sistemas que generan un gran volumen de información al presentar un alto rendimiento. Es altamente escalable permitiendo grandes demandas de usuarios y manteniendo una alta capacidad de adaptarse a cambios bruscos de demanda.

Oracle permite la utilización de los *RAC*[46], con la tecnología de disco compartido. Está diseñado para satisfacer las demandas actuales de la propuesta de arquitectura. Los recursos, servidores, y almacenamiento pueden ser administrados como una entidad única dentro del ambiente del clúster. A medida que se agregan recursos, el RAC puede utilizarlos, esto asegura un costo total de propiedad más bajo, de esta forma no se necesita comprar nuevo hardware con los requerimientos necesarios.

Otras de sus principales características es su gran capacidad de almacenamiento y de réplica, máxima seguridad, administración simplificada, soporte de transacciones y facilidades en las tareas de recuperación y respaldo. Entre otras como:

- ✓ Posee una herramienta de administración gráfica que es mucho más intuitiva y cómoda de utilizar.
- ✓ Ayuda a analizar datos y efectuar recomendaciones concernientes a mejorar el rendimiento y la eficiencia en el manejo de aquellos datos que se encuentran almacenados.
- ✓ Apoya en el diseño y optimización de modelos de datos.
- ✓ Asiste a los desarrolladores con sus conocimientos de SQL y de construcción de procedimientos almacenados y triggers, entre otros.
- ✓ Apoya en la definición de estándares de diseño y nomenclatura de objetos.

[46]RAC, del inglés Real Aplication Cluster (Clúster de Aplicaciones Reales)

- ✓ Documenta y mantiene un registro periódico de las mantenciones, actualizaciones de hardware y software, cambios en las aplicaciones y, en general, todos aquellos eventos relacionados con cambios en el entorno de utilización de una base de datos.

Oracle está definido como el gestor de base de datos a utilizar para el desarrollo y despliegue del Sistema de Control de Versiones debido a que ofrece una excepcional disponibilidad, escalabilidad, fiabilidad y seguridad. Además proporciona un adecuado soporte para la réplica de los datos, necesarios para la comunicación entre el COVER y los sistemas ubicados en cada una de las provincias, para así tener un mayor control de las versiones instaladas. Además, también se tomó esta herramienta con el propósito de seguir la línea de desarrollo que se ha trazado el MININT en usar Oracle como gestor de Base de Datos.

La infraestructura del COVER en tiempo de despliegue identifica que en el DSI se utilice Oracle Enterprise Edition versión 10 y en cada provincia Oracle Standard Edition versión 10 que es más ligero. (10) (11)

1.4.6 Visual C#.

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos que permite a los desarrolladores crear una amplia gama de aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Este lenguaje se puede utilizar para crear aplicaciones cliente para Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y muchas tareas más. Microsoft Visual C# 2005 proporciona un editor de código avanzado, diseñadores de interfaz de usuario prácticos, un depurador integrado y muchas otras herramientas para facilitar un rápido desarrollo de la aplicación basado en la versión 2.0 del lenguaje C# y en .NET Framework.

El proceso de generación de C# es simple en comparación con el de C y C++, y es más flexible que en Java. No hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en un orden determinado. Un archivo de código fuente de C# puede definir cualquier número de clases, estructuras, interfaces y eventos. (12)

1.5 Conclusiones.

En este capítulo se describieron algunos de los sistemas de control de versiones que son usados a nivel mundial incluyendo Cuba. Se hizo un estudio del funcionamiento de dichos sistemas y de su estado del arte, así como de los protocolos para la intercomunicación entre los diferentes módulos del

sistema, este estudio proporcionó un mayor entendimiento de la línea que se quiere seguir para el desarrollo del COVER y en general permitió conocer como se comporta la tecnología actual.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

2.1 Introducción:

En este capítulo se describe el negocio a modelar, se muestran los diagramas del proceso del negocio, se exponen los requisitos funcionales y los no funcionales que deberá satisfacer el sistema y se definen cuáles son los actores y los casos de uso que intervienen en el mismo, además de llegar a conocer mejor el sistema que será desarrollado.

2.2 Propuesta del sistema.

Se propone un sistema capaz de controlar efectivamente las versiones de los sistemas informáticos que se explotarán y generalizarán en el ministerio del interior. Será capaz de detectar un uso incorrecto de sistemas informáticos en cualquier unidad del MININT correspondiente a cualquier provincia del país y reportarlo, de igual manera el sistema brindará la posibilidad de interactuar con los usuarios, enviar avisos a los usuarios de una nueva versión de un sistema informático. La utilización de este nuevo sistema será fundamental para el desarrollo del ministerio.

2.3 Característica del sistema COVER.

El sistema de control de versiones de los productos informáticos posee entre sus características que:

- Permitirá un control sistemático por parte de los responsables para detectar posibles fallas o violaciones.
- Será un sistema fácil de controlar.
- Se caracterizará por ser un sistema seguro.
- Tendrá una interfaz amigable.
- Controlará la utilización de las versiones informáticas desplegadas en el ministerio del interior.
- Reportará sistemáticamente la utilización debida del sistema al personal encargado de operarla.

2.3.1 Modos de operación del COVER.

El sistema a crear operará de la forma siguiente: tendrá una Base de Datos donde se almacenarán las últimas versiones que estén en explotación además de incorporarse las nuevas que lo estarán, el sistema será el encargado de detectar la utilización de versiones incorrectas en las distintas unidades desplegadas por todo el país y reportarlo al administrador que será el encargado operarlo a nivel central, el administrador una vez reportada la versión incorrecta enviará aviso por la red al operador

informándole que la versión que se está explotando en sus puestos de trabajo es antigua y debe actualizarla.

2.3.2 Situación en el Ministerio del interior.

En el ministerio del interior actualmente no existe un sistema parecido capaz de controlar las versiones que son utilizadas en sus distintas unidades, esto conlleva una cierta falta de control e ineficiencia en los procesos de gestión de las versiones de los sistemas informáticos

2.4 Modelo del Negocio.

Un modelo representa una forma de contemplar el sistema que se modela. El modelado del negocio es una técnica para comprender los procesos de negocio de la organización. (JACOBSON et al., 2000)

El modelado del negocio es la técnica por excelencia para alinear los desarrollos con las metas y objetivos de las empresas e instituciones. Si se realiza de tal forma que el modelo quede aprobado entre los grupos interesados, las posibilidades de éxito del proyecto aumentarán en forma muy importante.

El modelado de negocios, y más específicamente el modelado de procesos de negocio, es la forma idónea para comunicarnos con los usuarios de todos los niveles. (7)

Cualquier metodología de Análisis y Diseño para el desarrollo de sistemas tiene como punto de partida la captura de requisitos, obtenidos por los analistas en interacción con los usuarios, que más tarde serán analizados y plasmados en herramientas propias de cada metodología de manera que cubran las expectativas de los usuarios y que se ajusten a las tendencias actuales del desarrollo de aplicaciones.

La obtención de requerimientos es un paso muy importante para el desarrollo del sistema, pues un error en estas fases iniciales puede dar al traste con un sistema que no cumpla las expectativas de los usuarios. Este es un proceso complejo que depende de la comunicación entre clientes y especialistas y de la relación dentro de los grupos de trabajo, entre otros factores. Un sistema que no responda a las necesidades de los clientes no cumple los requerimientos mínimos de calidad. (13)

2.4.1 Objetivos del modelado del negocio.

El modelado del negocio nos permite entender mejor las características fundamentales del sistema, los principales motivos para ejecutar esta disciplina son los siguientes: asegurarse de que el producto será algo útil y no un obstáculo; conseguir que se ajuste de la mejor forma posible en la organización donde

se va a implantar; y tener un marco común para el equipo de proyecto, los clientes y los usuarios finales, algunos de sus objetivos son:

- Asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de la organización objetivo.
- Derivar los requerimientos del sistema necesarios para apoyar a la organización objetivo en su mejora.
- Entender el problema actual en la organización objetivo e identificar potenciales mejoras.
- Entender la estructura y la dinámica de la organización para la cual el sistema va a ser desarrollado (organización objetivo).

Para lograr estos objetivos, el modelado de negocio describe como desarrollar una visión de la nueva organización, basado en esta visión se definen procesos, roles y responsabilidades de la organización por medio de un Modelo de Casos de Uso del Negocio. Los artefactos del modelo de negocio sirven como entrada y referencia para la definición de los requerimientos del sistema. (14)

2.4.2 Alcance.

La modelación del negocio se realiza con el fin de de obtener una mejor comunicación con el cliente, que exista un intercambio de información adecuado que posibilite que el proyecto pueda realizarse con la máxima seguridad de alcanzar el éxito, además de conocer las necesidades del cliente que sería el objetivo principal y prioridad.

2.4.3 Actores y trabajadores que intervienen en el negocio.

Un actor del negocio es cualquier individuo, grupo, organización o máquina que interactúa con el negocio.

Un trabajador del negocio se utiliza en el modelado de negocio como punto de partida para derivar un primer conjunto de actores y casos de uso del sistema de información en construcción.

Tabla 1 Actores y Trabajadores del Negocio.

Actor del negocio	Justificación
Operador:	Es el administrador del sistema informático del cual se va a controlar la versión.

Trabajador del negocio	Justificación
Controlador:	Es el encargado de controlar que se estén explotando las versiones correctas de los sistemas informáticos.

2.4.4 Diagrama de casos de uso del negocio.

El modelo del negocio describe el negocio en términos de casos de usos del negocio, que corresponde a lo que generalmente se le llama procesos.

Figura 5 Diagrama de casos de uso del negocio.

2.4.5 Diagrama de clases del modelo de objetos.

El diagrama de objetos describe cómo colaboran los trabajadores y las entidades del negocio dentro del flujo de trabajo del proceso del negocio.

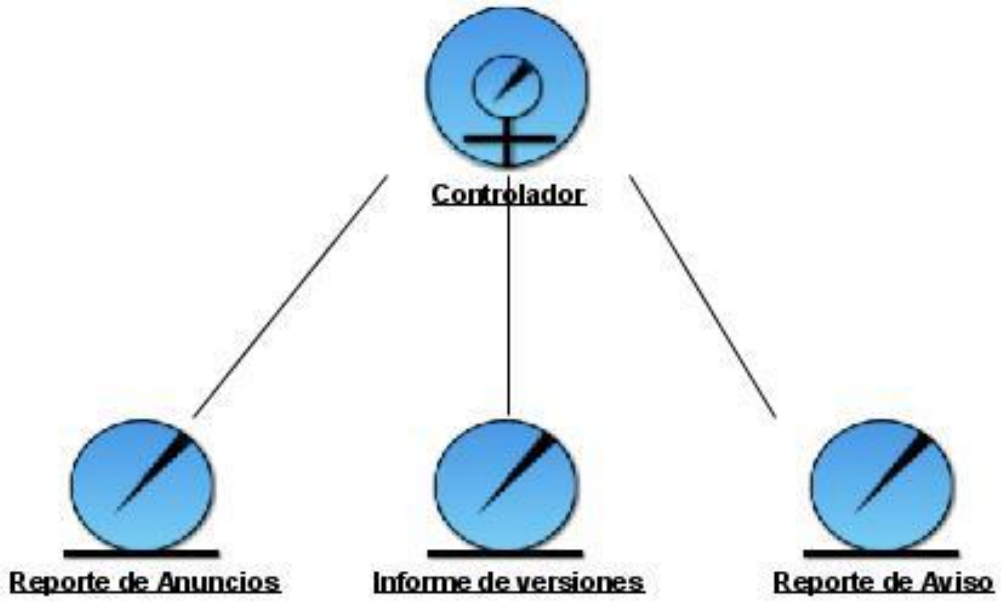


Figura 6 Diagrama de clases del modelo de objetos.

Tabla 2 Descripción del caso de uso del negocio "Recibir Control".

Nombre de Caso de Uso: "Recibir Control"	
Actores:	Operador (Inicia)
Propósito:	Verificar que se estén explotando las versiones correctas de los sistemas informáticos
Resumen:	El caso de uso se inicia cuando el Operador recibe al Controlador para realizar el control de las versiones
Curso Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1- El Operador recibe al Controlador	2- El Controlador verifica que las versiones que están en explotación sean las mismas del reporte de versiones 3- El Controlador crea un reporte de anuncios con las próximas versiones de los sistemas que tiene el Operador que se pondrán en explotación 4- El Controlador hace la entrega de reporte al Operador
5- El Operador recibe el o los reportes	
Flujo Alterno: "No están correctas las versiones"	
Acción del Actor	**Respuesta del Negocio**
	2.1- El Controlador crea un reporte de aviso donde le señala cuales son los sistemas informáticos que no cuentan con las versiones correctas y cuáles son las que

	se deben usar, pasa a la acción 3 del flujo normal de los eventos

2.4.6 Diagrama de actividades <Recibir Control>.

El diagrama de actividades describe gráficamente la estructura del flujo de actividades.

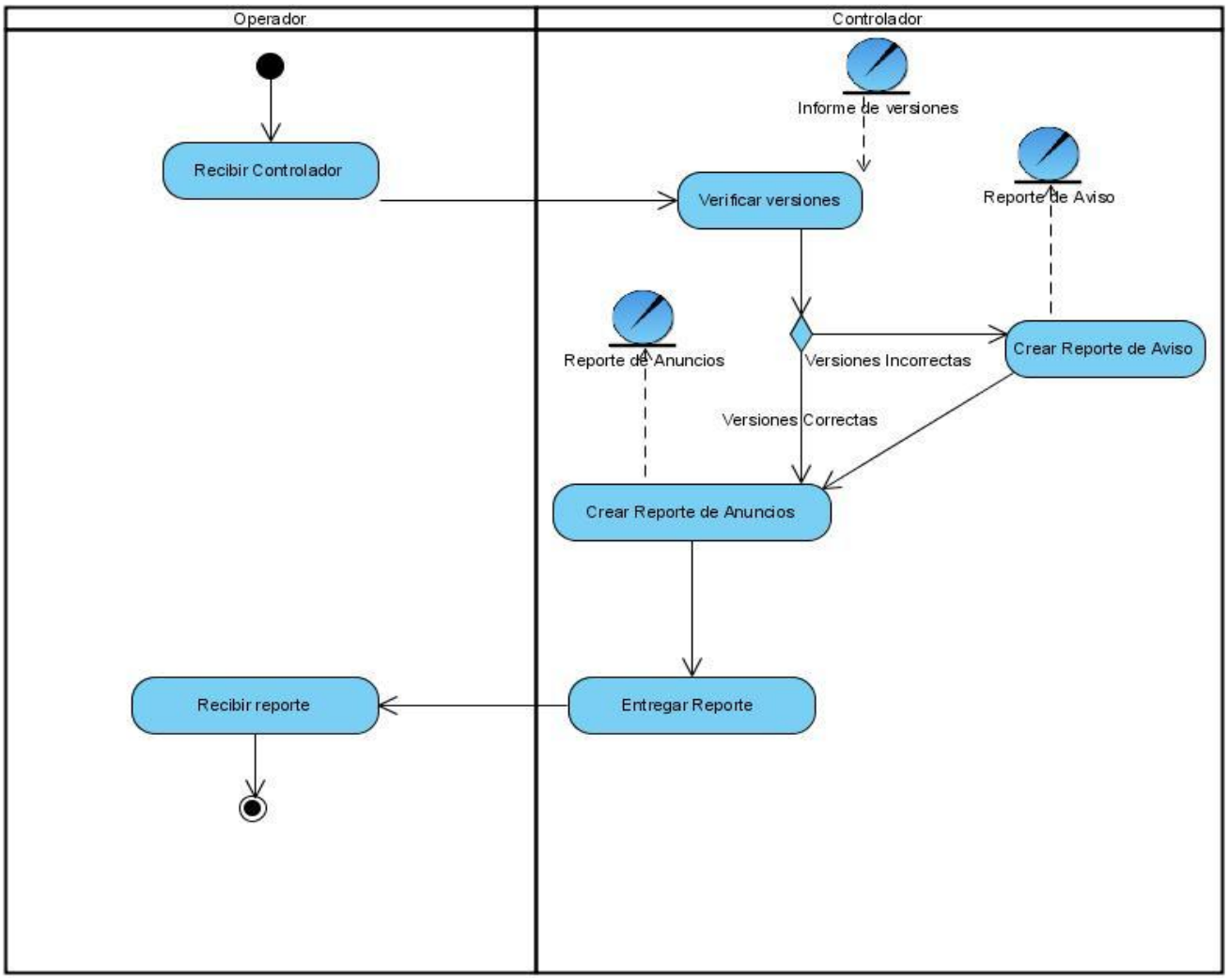


Figura 7 Diagrama de actividades "Recibir Control".

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.5 Especificación de los requisitos.

La captura de requisitos es la etapa principal de cualquier proyecto, si los desarrolladores no entienden en que consiste el proyecto su trabajo será deficiente y nunca será efectivo, no resolverá los problemas básicos que se requieren, y lógicamente lo que aportará para la institución en la que se encuentre será pérdida de tiempo y recursos, para una correcta captura de requisitos se empleó la técnica de la entrevista.

2.5.1 Requerimientos Funcionales.

Los Requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir, en nuestro trabajo de diploma fueron identificado los siguientes:

RF1. Brindar Aviso.

RF2. Realizar Control.

2.1 Consultar Versión.

2.2 Visualizar Reporte.

2.3 Eliminar Reporte.

RF3. Gestionar Anuncios.

3.1 Insertar Anuncio.

3.2 Modificar Anuncio.

3.3 Eliminar Anuncio.

RF4. Gestionar Usuarios.

3.4 Registrar Usuario.

3.5 Modificar Usuario.

3.6 Eliminar Usuario.

RF5. Gestionar Datos de Versiones.

3.7 Introducir Datos.

3.8 Actualizar Datos.

3.9 Eliminar Datos.

RF6. Consultar Datos.

RF7. Autenticarse.

2.5.2 Requerimientos no Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener.

Requisitos de Seguridad.

- Utilizar la seguridad integrada de Windows.
- Realizar un control sistemático de las acciones que lleven a cabo los usuarios en el sistema buscando posibles fallas de seguridad.
- Verificación sobre acciones irreversibles (Por ejemplo eliminaciones).

Requisitos de Implementación.

- Emplear la tecnología .Net como plataforma de desarrollo.
- Utilizar Oracle DB como Sistema de Gestión de Base de Datos.

Requisitos de Apariencia.

- Debe poseer una interfaz agradable.
- Mostrar las opciones que puede efectuar dentro del sistema según el rol del usuario.

Requisitos de Software.

- El cliente debe tener Microsoft Internet Explorer (Versión 5.0 o superior)
- Para el servidor Web: Internet Information Services con Windows Server 2003 como sistema operativo y SharePoint como gestor de contenido.

Requisitos de Hardware.

- Tarjeta de red.
- Para el servidor Web: mínimo 2GB RAM.
- Capacidad de disco duro, preferiblemente 80 GB para Servidor Web, y mínimo160 GB para servidor de base de datos.
- Microprocesador de 2.4 GHz o superior.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.6 Descripción del sistema propuesto.

2.6.1 Actores que intervienen en el sistema.

Actores: Terceros fuera del sistema que interactúan con él.

Tabla 3 Descripción de los actores del sistema.

Actores del sistema	Justificación
Administrador del Sistema:	Responde por el seguimiento en el control de versiones de un sistema informático. Participa y coordina el proceso de despliegue de una nueva versión, de conjunto con los desarrolladores, ingenieros de sistemas y administradores de los entornos involucrados. Le informa a los operadores de cualquier error o necesidad de nuevas funcionalidades del sistema que determinen la creación de una nueva versión, publican anuncios en el sitio relacionados con las versiones que se explotan o comenzaran a explotar. Responde a determinados sistemas informáticos que se le asignan para controlarlos y darles seguimiento.
Administrador Central	Es el administrador del sistema que más privilegios tiene. Es quien asigna los privilegios o roles a los usuarios en el sistema, además de realizar las funciones normales de un administrador del sistema. Puede controlar todos sistemas informáticos que estén guardados.
Operador:	Son quienes trabajan los sistemas informáticos en explotación. Pueden acceder al sistema y consultar datos de los sistemas con que trabajan u otros, así como ver los anuncios que se publican en el sitio relacionados con las versiones de los sistemas informáticos que se que se explotan.

Usuario del Sistema	Persona que va a interactuar con el sistema ya sea Administrador del Sistema, Administrador Central o el Operador.

2.6.2 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 como será utilizado desde el punto de vista del usuario (Actor), pues se construye sobre la base de sus necesidades. (15)

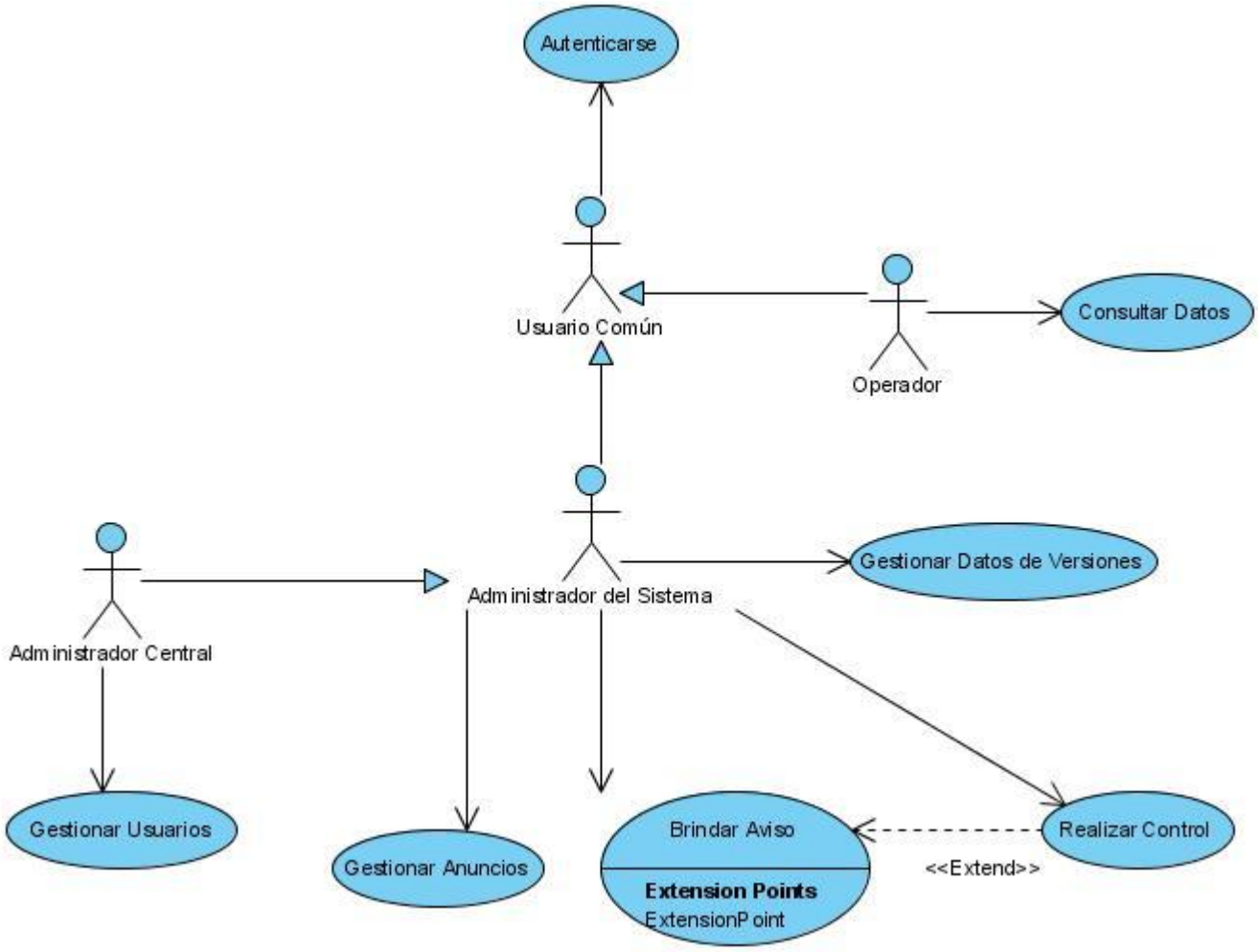


Figura 8 Diagrama de casos de uso del sistema.

2.6.3 Descripción de los Casos de Uso del Sistema.

Tabla 4 Descripción del caso de uso "Brindar Aviso".

Nombre de Caso de Uso: "Brindar Aviso"	
Actores:	Administrador del Sistema.
Propósito:	Que los usuarios sean informados de la utilización incorrecta de las versiones en explotación.
Resumen:	El caso de uso comienza después de que el sistema lanza una alerta al administrador y este crea un aviso y lo envía al operador.
Precondiciones:	1- Que el Administrador del Sistema se haya autenticado. 2- Que se haya realizado la consulta de versiones y se hayan detectado versiones incorrectas.
Referencias:	R1
Prioridad:	Crítica.

Curso normal de eventos	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema solicita enviar aviso.	2- El sistema pide los datos correspondientes.
3- El Administrador del Sistema llena los datos y confirma el envío del aviso	4- El Sistema valida los datos. 5- El sistema guarda el aviso y se encarga de enviarlo al operador correspondiente. 6- El Sistema muestra el mensaje "Envío realizado exitosamente".
7- El Administrador del Sistema es notificado del envío.	

Flujo Alterno	
	4.1- El Sistema muestra el mensaje “Los datos introducidos no son validos”, y pasa a la acción 2 del flujo normal de los eventos.
Prototipo de Interfaz 	

Tabla 5 Descripción del caso de uso “Realizar Control”

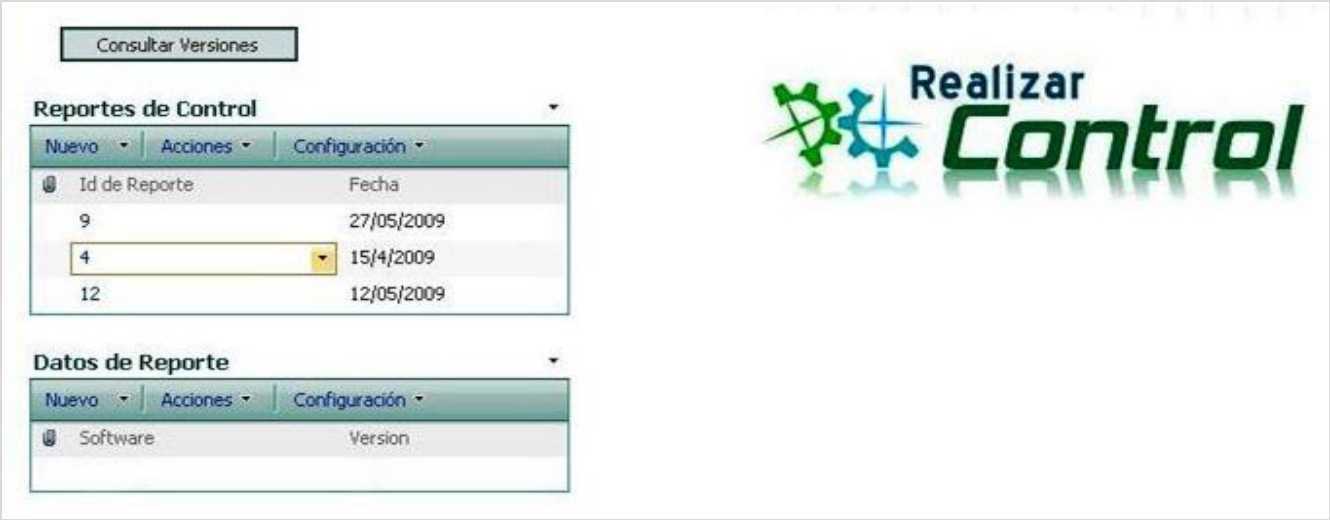
Nombre de Caso de Uso: “Realizar Control”	
Actores:	Administrador del Sistema.
Propósito:	Obtener información sobre las versiones de sistemas informáticos que se encuentran en explotación, así ver reportes anteriores o eliminarlos.
Resumen:	El caso de uso comienza cuando el Administrador del Sistema selecciona una opción ya sea realizar la consulta de versiones, ver reportes de consultas anteriores o eliminar dichos reportes.
Precondiciones:	Que el Administrador del Sistema se haya autenticado.

Referencias:	R2
Prioridad:	Crítica.

Curso normal de eventos	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema selecciona la opción que desea ejecutar.	2- a) Si desea consultar las versiones en explotación, ir a la sección "Consultar Versión". b) Si desea ver el reporte de una consulta determinada, ir a la sección "Visualizar Reporte" c) Si desea eliminar un reporte, ir a la sección "Eliminar Reporte".

Escenario Consultar Versión	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema solicita consultar las versiones en explotación.	2- El Sistema solicita a los Agentes en las máquinas la información de la versión de los sistemas en explotación. 3- El Sistema compara la información de los Agentes con los datos de las versiones que se deben estar explotando y lo almacena como reporte en la base de datos. 4- El Sistema muestra las versiones que se explotan, si hay alguna versión que no es la adecuada ver flujo alterno Brindar Alerta.
5- El Administrador del Sistema es informado.	

Prototipo de Interfaz



Flujo Alterno: "Brindar Alerta"

	1- El sistema detecta una versión no correcta. 2- El sistema muestra inmediatamente al Administrador del Sistema la versión no correcta.
3- El Administrador del Sistema en informado sobre la versión no correcta.	

Escenario Visualizar Reporte

Acción del actor	**Respuesta del Sistema**
1- El Administrador del Sistema solicita ver un reporte.	2- El Sistema muestra una lista con los reportes existentes.
3- El Administrador del Sistema selecciona el reporte deseado.	4- El Sistema muestra los datos del reporte.

Prototipo de Interfaz

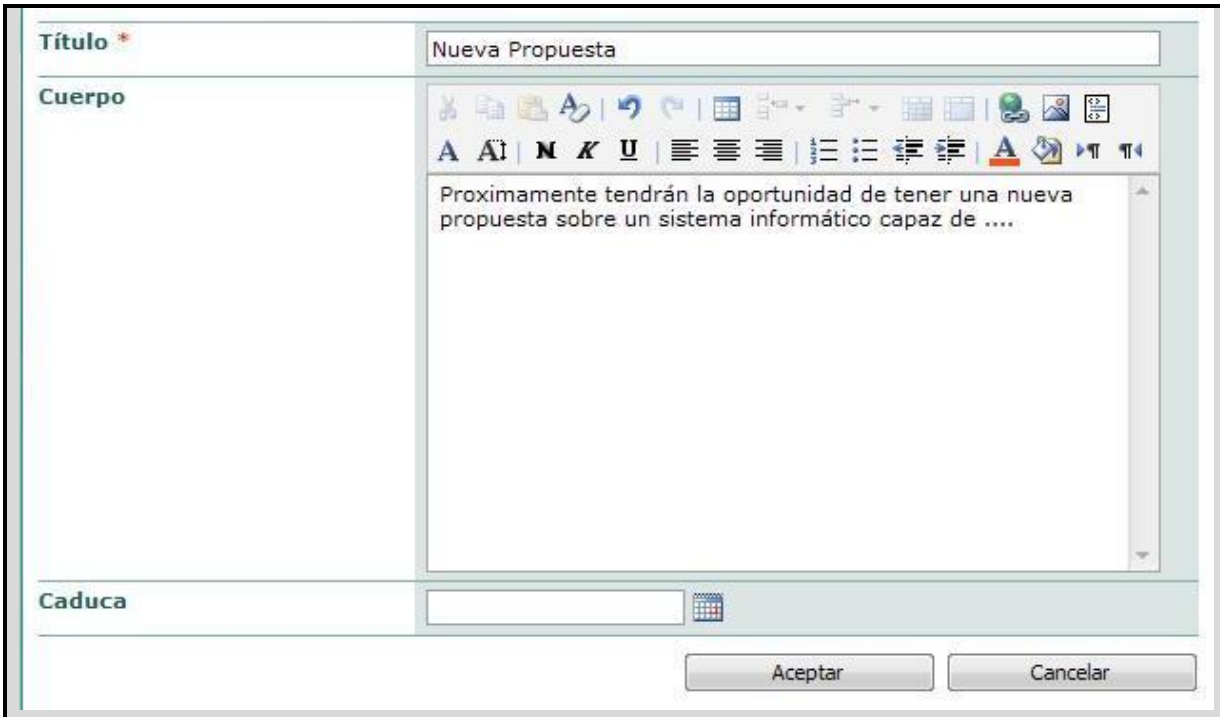
Escenario Eliminar Reporte	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema selecciona el reporte que desea eliminar.	2- El Sistema busca el reporte en la base de datos y lo elimina. 3- El Sistema muestra la lista de reportes actualizada.
Prototipo de Interfaz	

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Tabla 6 Descripción del caso de uso "Gestionar Anuncios".

Nombre de Caso de Uso: "Gestionar Anuncios	
Actores:	Administrador del Sistema
Propósito:	Proporcionar información sobre las versiones que se deben estar explota para que se actualice el usuario al conectarse al sitio.
Resumen:	El caso de uso comienza cuando el Administrador del Sistema desea publicar, modificar o eliminar los anuncios. De esta manera los usuarios al conectarse ven las informaciones importantes del día.
Precondiciones:	Que el Administrador del Sistema se haya autenticado.
Referencias:	R3
Prioridad:	Crítica.
Curso Normal de Eventos	

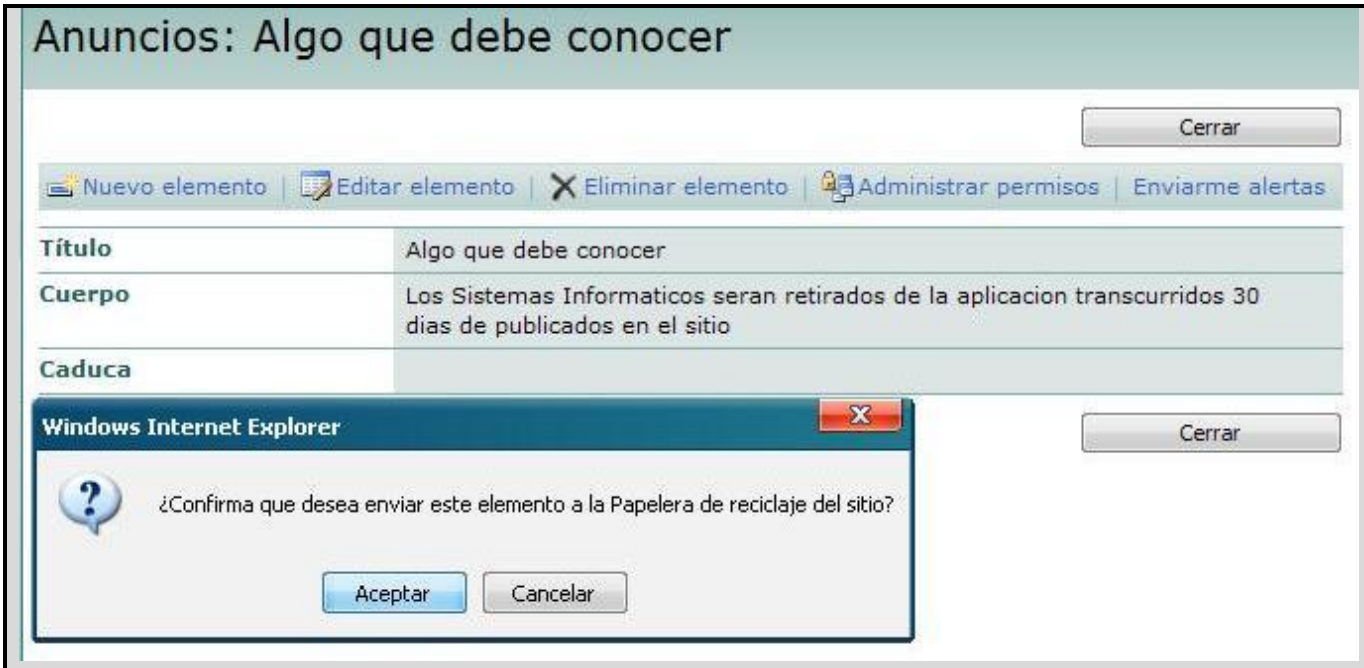
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema selecciona la opción que desea ejecutar.	2- d) Si desea insertar un anuncio, ir a la sección "Insertar Anuncios". e) Si desea eliminar un anuncio, ir a la sección "Eliminar Anuncios". f) Si desea modificar un anuncio, ir a la sección "Modificar Anuncios".
Escenario "Insertar Anuncios"	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema solicita al sistema la publicación de anuncios en el sitio.	2- El sistema muestra un formulario para que el Administrador del Sistema inserte el anuncio.
3- El Administrador del Sistema procede a introducir los datos del anuncio.	4- El sistema introduce el anuncio. 5- El sistema muestra los anuncios previamente insertados más el nuevo.
Prototipo de Interfaz	



Escenario "Eliminar Anuncios"

Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema solicita al sistema eliminar un anuncio en el sitio.	2- El sistema muestra una ventana con los anuncios que hay publicados para la eliminación del mismo.
3- El Administrador del Sistema selecciona el anuncio a eliminar.	4- El sistema elimina el anuncio seleccionado. 5- El sistema muestra el listado actualizado de anuncios.

Prototipo de Interfaz



Escenario "Modificar Anuncios"

Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema solicita al sistema modificar determinado aviso en el sitio.	2- El sistema muestra una ventana con los anuncios publicados.
3- El Administrador del Sistema selecciona el anuncio.	4- El sistema muestra el contenido del anuncio para modificar.
5- El Administrador modifica los datos y acepta.	6- El sistema guarda los datos del anuncio modificado. 7- El sistema muestra el listado de anuncios actualizado.

Prototipo de Interfaz

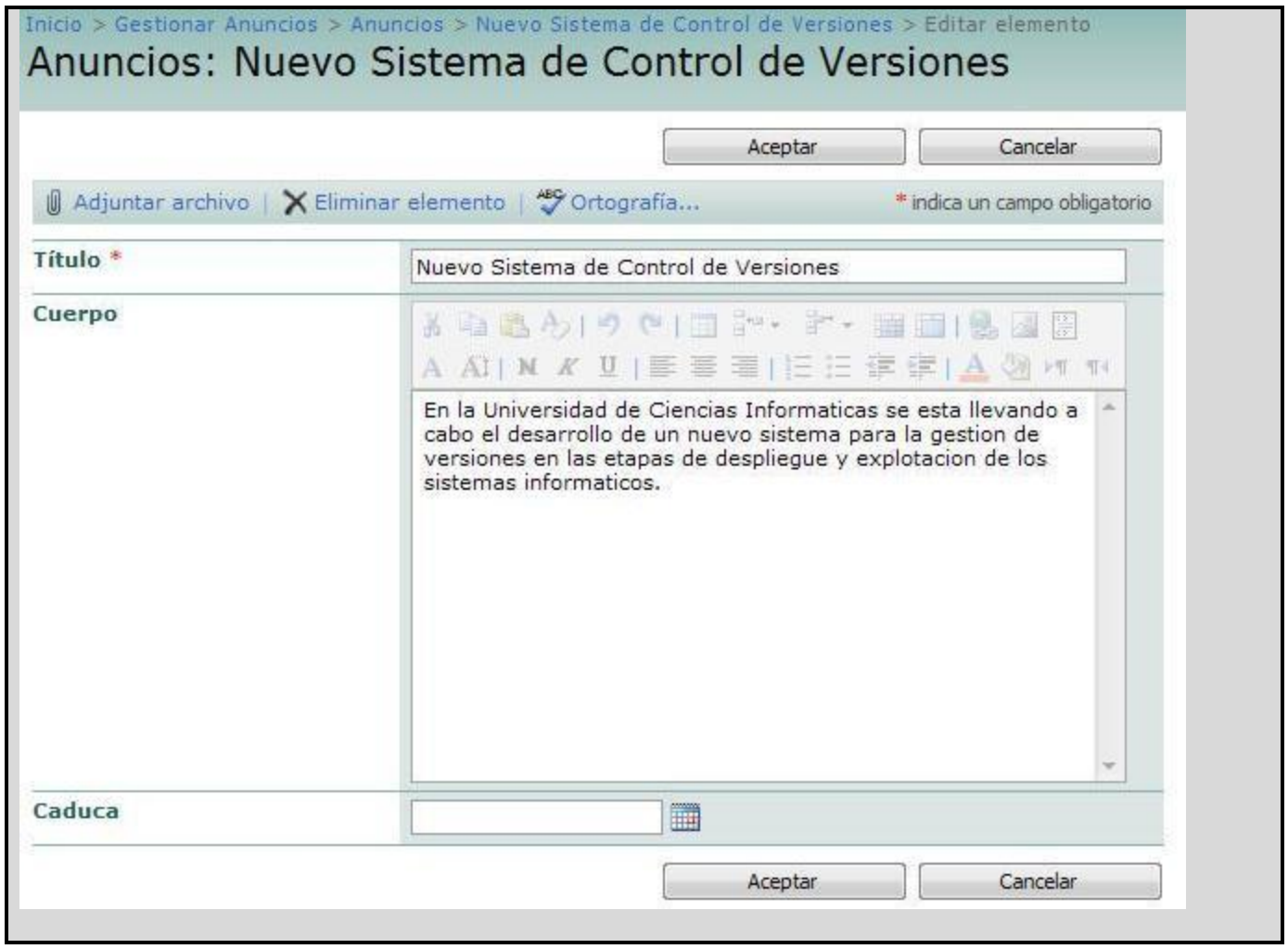


Tabla 7 Descripción del caso de uso "Gestionar Usuarios".

Nombre de Caso de Uso: "Gestionar Usuarios"	
Actores:	Administrador Central (Inicia).
Propósito:	Designar roles a los usuarios en el sistema, así como modificarlos o eliminarlos.
Resumen:	El caso de uso comienza cuando el Administrador Central solicita introducir, modificar o eliminar un usuario al sistema.
Precondiciones:	Que el Administrador Central se haya autenticado.

Referencias:	R4
Prioridad:	Crítica.
Curso Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador Central selecciona la opción que desea ejecutar.	2- a) Si desea registrar un usuario, ir a la sección "Registrar Usuario". b) Si desea modificar un usuario, ir a la sección "Modificar Usuario". c) Si desea eliminar un usuario, ir a la sección "Eliminar Usuario".
Escenario "Registrar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador Central solicita registrar un usuario.	2- El sistema muestra el formulario correspondiente.
3- El Administrador Central llena el formulario y acepta.	4- El sistema valida los datos. 5- El sistema verifica que el usuario no esté registrado. 6- El sistema guarda los datos del usuario en la base de datos. 7- El sistema informa que se ha introducido un nuevo usuario al sistema.

8- El Administrador Central es notificado.	

Prototipo de Interfaz

Flujos Alternos	
	4.1- El sistema muestra el mensaje “Datos de entrada no válidos” y pasa a la acción 2 del flujo normal de los eventos 5.1 El sistema muestra el mensaje “El Usuario ya existe en el sistema” y pasa a la acción 2 del flujo normal de los eventos

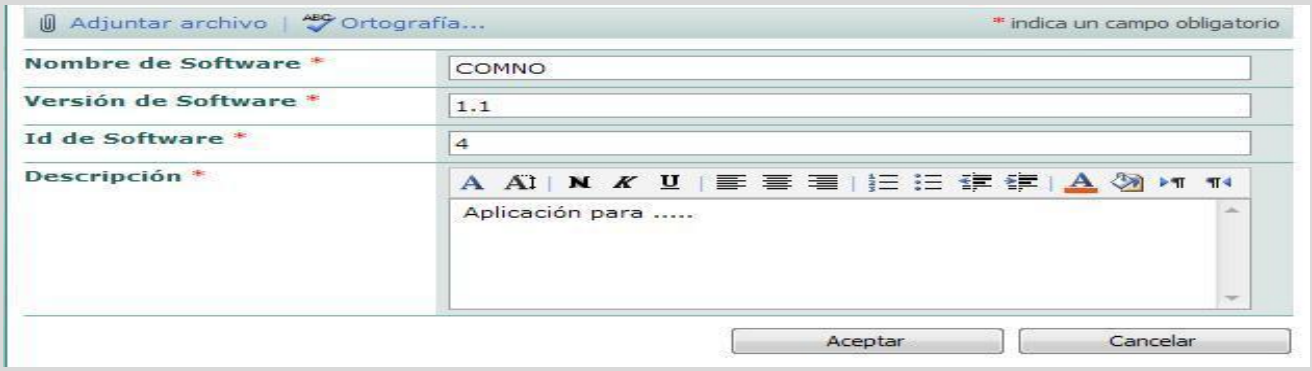
Escenario “Modificar Usuario”	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador Central solicita modificar los datos de un usuario.	2- El sistema muestra una lista con los usuarios que se encuentran registrados.
3- El Administrador Central selecciona el usuario que desea.	4- El sistema busca los datos del usuario seleccionado y los muestra en un formulario.
5- El Administrador Central modifica los datos y acepta.	6- El sistema valida los datos. 7- El sistema guarda los datos. 8- El sistema informa que se han guardado correctamente los datos.

9- El Administrador Central es notificado.	
Prototipo de Interfaz	
Flujos Alternos	
	5.1- El sistema muestra el mensaje “Datos de entrada no válidos” y pasa a la acción 4 del flujo normal de los eventos.
Escenario “Eliminar Usuario”	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador Central solicita eliminar un usuario.	2- El sistema muestra una lista con los usuarios registrados.
3- El Administrador Central selecciona el usuario a eliminar y acepta.	4- El sistema borra el usuario seleccionado de la base de datos. 5- El sistema informa que el usuario ha sido eliminado satisfactoriamente.
6- El Administrador Central es notificado.	
Prototipo de Interfaz	

Tabla 8 Descripción del caso de uso "Gestionar Datos de Versiones".

Nombre de Caso de Uso: "Gestionar Datos de Versiones"	
Actores:	Administrador del Sistema (Inicia).
Propósito:	Brindar la posibilidad de introducir, eliminar y modificar los datos de las versiones de sistemas informáticos que se van a explotar.
Resumen:	El caso de uso se inicia cuando el Administrador del Sistema solicita una de las opciones de introducir, eliminar o modificar.
Precondiciones:	Que el Administrador del Sistema se haya autenticado.
Referencias:	R5
Prioridad:	Crítica.
Curso Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema selecciona	2-

la opción que desea ejecutar.	a) Si desea introducir datos de un sistema, ir a la sección "Introducir Datos". b) Si desea actualizar datos de un sistema, ir a la sección "Actualizar Datos". c) Si desea eliminar datos de un sistema, ir a la sección "Eliminar Datos".
Escenario "Introducir Datos"	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema solicita introducir los datos de un sistema.	2- El sistema muestra el formulario correspondiente.
3- El Administrador del Sistema introduce los datos y acepta.	4- El sistema valida los datos. 5- El sistema lo guarda en datos de versiones. 6- El sistema informa que se han guardado correctamente los datos.
7- El Administrador del Sistema es notificado.	
Prototipo de Interfaz	



Flujos Alternos	
	4.1- El sistema muestra el mensaje “Datos de entrada no válidos” y pasa a la acción 2 del flujo normal de los eventos
Escenario “Actualizar Datos”	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema solicita actualizar los datos de una versión.	2- El sistema muestra una lista con las versiones de sistemas que se encuentran en la base de datos.
3- El Administrador del Sistema selecciona la versión que desea.	4- El sistema busca los datos de la versión seleccionada y los muestra en un formulario.
5- El Administrador del Sistema actualiza los datos y acepta.	6- El sistema valida los datos. 7- El sistema actualiza los datos en la base de datos. 8- El sistema informa que se han actualizado correctamente los datos.
9- El Administrador del Sistema es notificado.	
Prototipo de Interfaz	

Flujos Alternos

	5.1- El sistema muestra el mensaje “Datos de entrada no válidos” y pasa a la acción 4 del flujo normal de los eventos.
Escenario “Eliminar Datos”	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del Sistema solicita eliminar los datos de una versión.	2- El sistema muestra una lista con las versiones de sistemas que se encuentran en la base de datos.
3- El Administrador del Sistema selecciona la versión que desea eliminar y acepta.	4- El sistema borra la versión seleccionada de la base de datos. 5- El sistema informa que la versión ha sido eliminada satisfactoriamente.
6- El Administrador del Sistema es notificado.	
Prototipo de Interfaz	

Tabla 9 Descripción del caso de uso "Consultar Datos".

Nombre de Caso de Uso: "Consultar Datos"	
Actores:	Operador (Inicia).
Propósito:	Consultar los datos de los sistemas informáticos con los cuales van a operar.
Resumen:	El caso de uso se inicia cuando el operador solicita consultar los datos de un sistema informático.
Precondiciones:	Que el Operador se haya autenticado.
Referencias:	R6
Prioridad:	Crítica.

Curso Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El Operador solicita consultar los datos de un sistema informático.	2- El sistema muestra una lista con los sistemas informáticos que se encuentran en el repositorio.

3- El Operador selecciona el sistema informático que desea consultar.	4- El sistema carga los datos del sistema informático y los muestra.

Prototipo de Interfaz

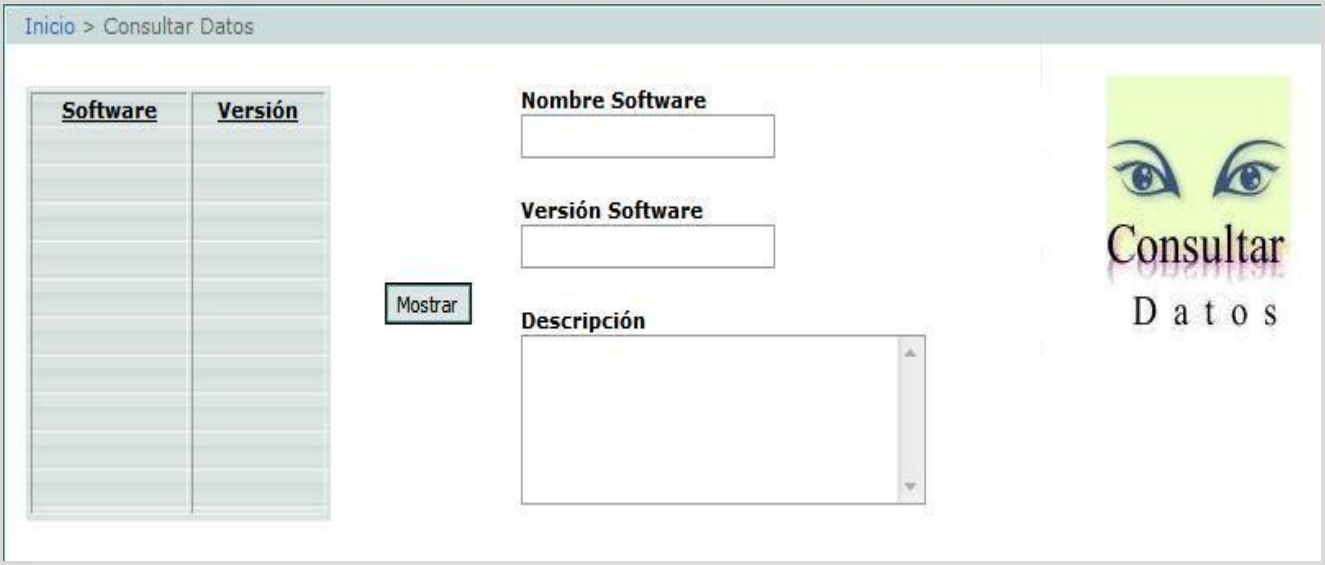
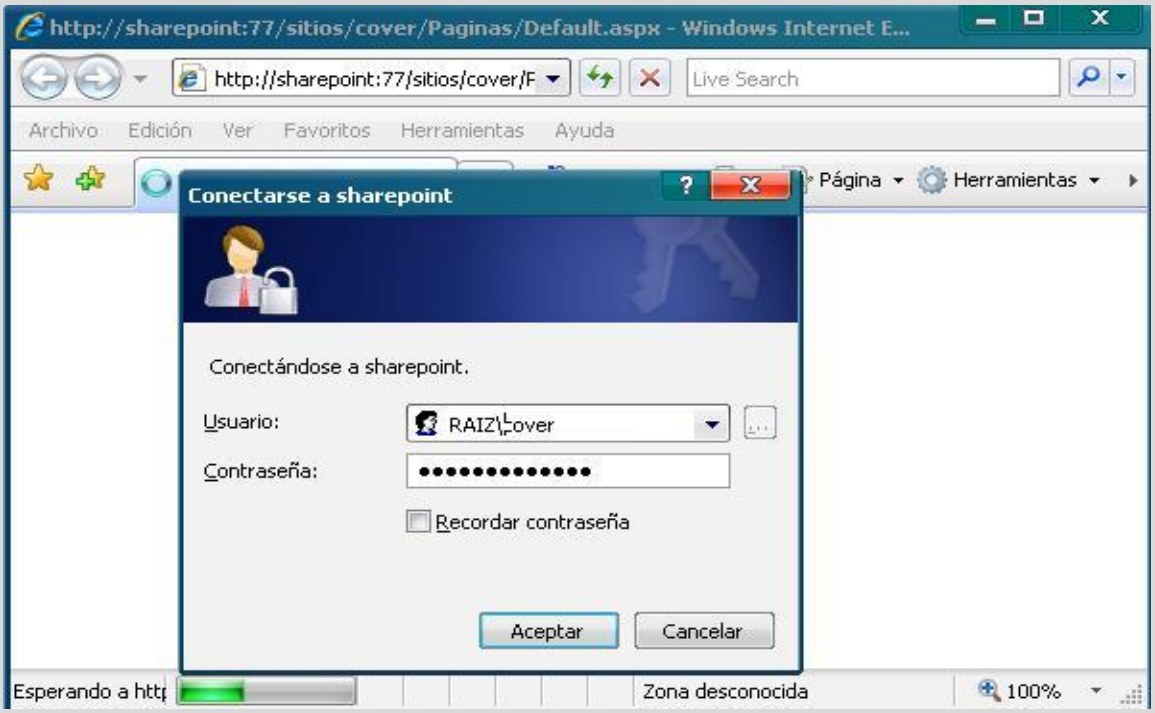


Tabla 10 Descripción del caso de uso "Autenticarse".

Nombre de Caso de Uso: "Autenticarse"	
Actores:	Usuario del sistema.
Propósito:	Adquirir privilegios según el rol que se tenga asignado.
Resumen:	El usuario introduce su nombre de usuario y contraseña. El sistema verifica si está registrado, si es así le da el acceso a las funcionalidades especificas, configuradas para el rol.
Precondiciones:	
Referencias:	R7
Prioridad:	Crítica.

Curso Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario introduce Usuario y Contraseña.	2- El sistema comprueba que el usuario exista en la base de datos. 3- El Sistema verifica si su contraseña es correcta. 4- El Sistema asigna privilegios al usuario y accede al sistema.

Prototipo de Interfaz



2.7 Conclusiones

En este capítulo se abordó que es un modelo de negocio, sus objetivos y alcance, se entendió de una mejor forma el sistema a desarrollar con el apoyo de los diagramas y la definición de los actores y casos de usos que intervendrán.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA.

3.1 Introducción.

En este Capítulo se presenta el modelo del análisis y el modelo del diseño, donde son expuestas las realizaciones de los casos de uso definidos en el capítulo anterior. Se describen las clases que se obtienen en el diseño y las tablas de la BD[47].

3.2 Análisis.

El flujo de trabajo de análisis y diseño juega un papel fundamental en la fase de elaboración. El objetivo principal del análisis es comprender los requisitos funcionales con que cuenta el software para así estructurar el proyecto de forma clara y precisa, se profundiza en el dominio de la aplicación lo que permite una mayor comprensión del problema para modelar la solución.

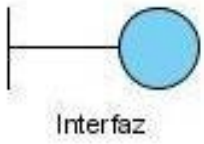
3.3 Diagramas de clases del análisis.

Los Diagramas de clases del análisis, se centran en los requisitos funcionales y son muy evidentes en el dominio del problema ya que representan conceptos y relaciones del dominio. Tienen atributos y entre ellos se establecen relaciones de asociación, agregación/composición, generalización/especialización y tipos asociativos. Existen tres estereotipos de clases estandarizados en UML y se utilizan para ayudar a los desarrolladores a distinguir el ámbito de las diferentes clases. Estas clases son:

Tabla 11 Clases del Análisis.

Nombre	Características	Figura
Entidad	Modelan información que posee larga vida y que es a menudo persistente.	

[47] BD: Base de Dato

Interfaz	Modelan la interacción entre el sistema y sus actores.	
Control	Coordinan la realización de uno o unos pocos casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.	

A continuación se presenta los diagramas de clases del análisis por cada caso de uso.

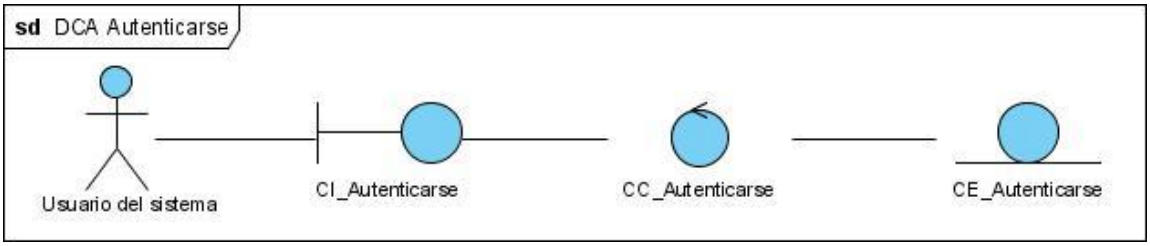


Figura 9 DCA: Autenticarse.

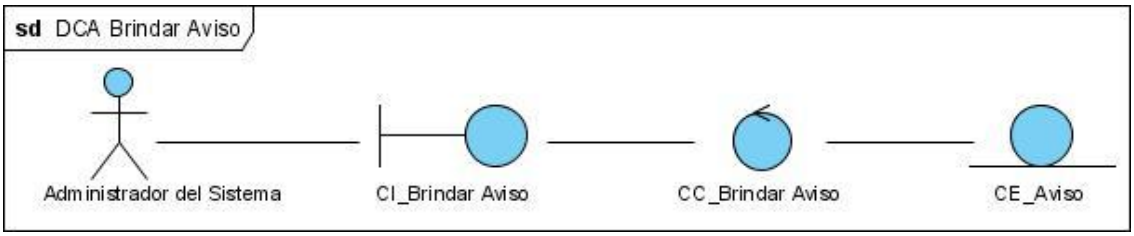


Figura 10 DCA: Brindar Aviso.

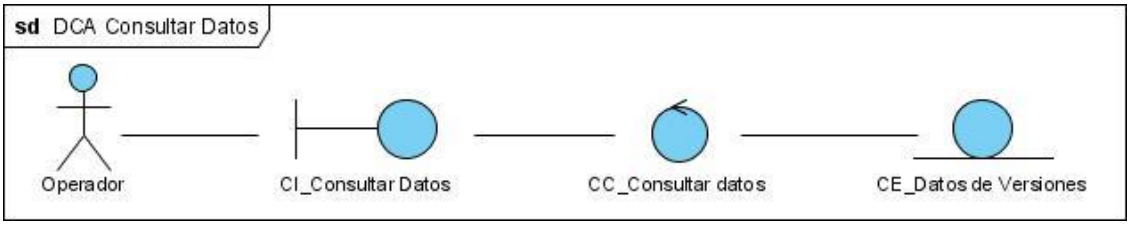


Figura 11 DCA: Consultar Datos.

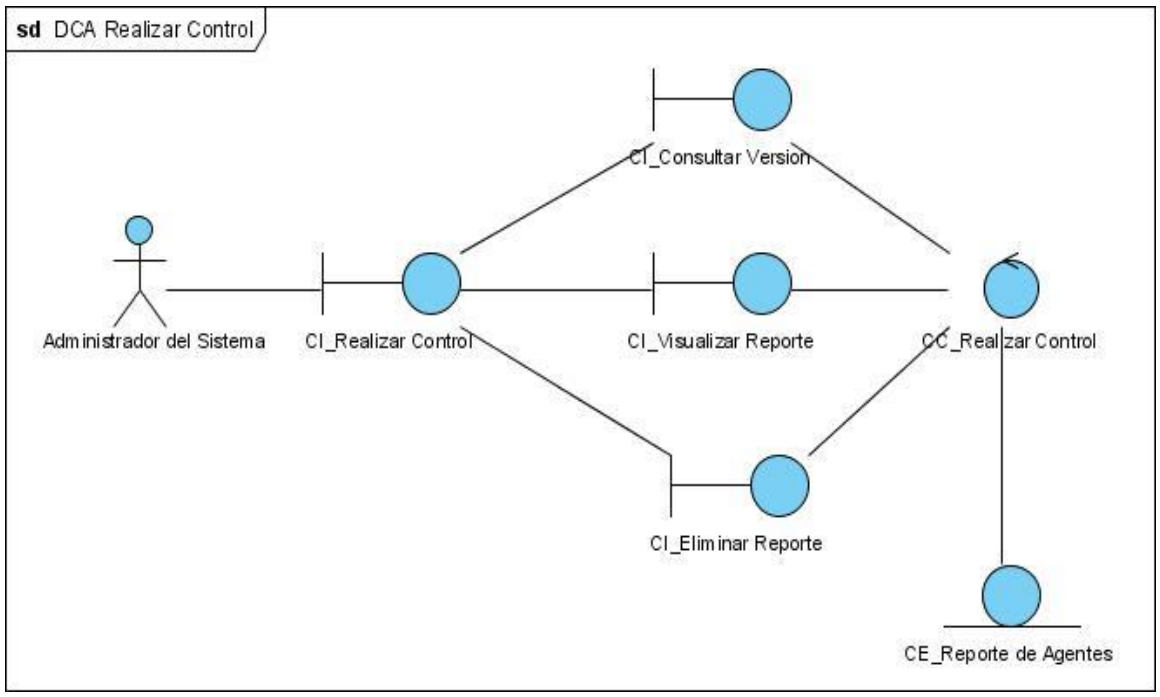


Figura 12 DCA: Realizar Control.

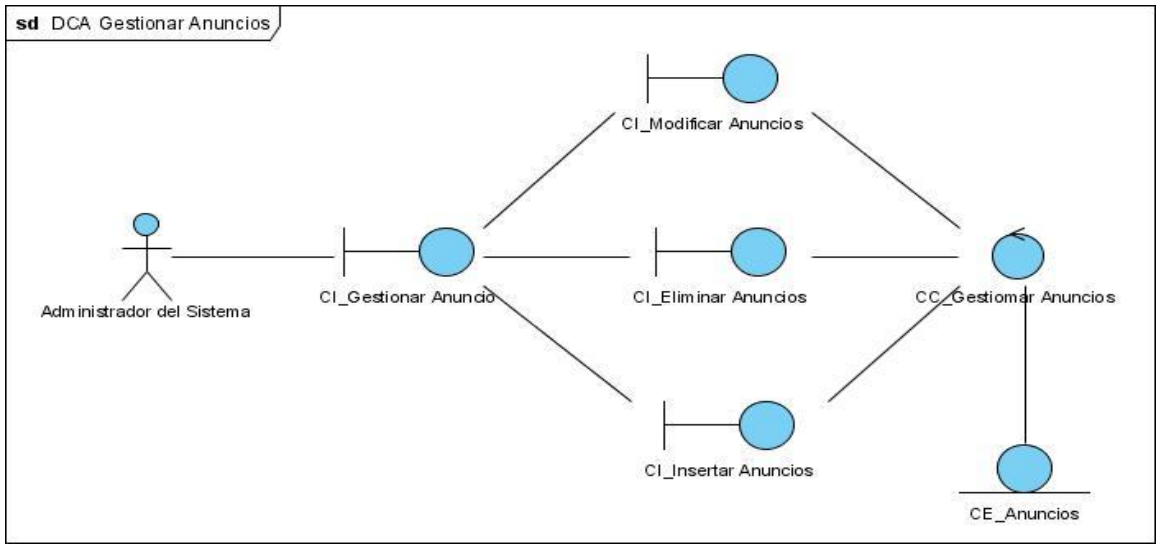


Figura 13 DCA: Gestionar Anuncios.

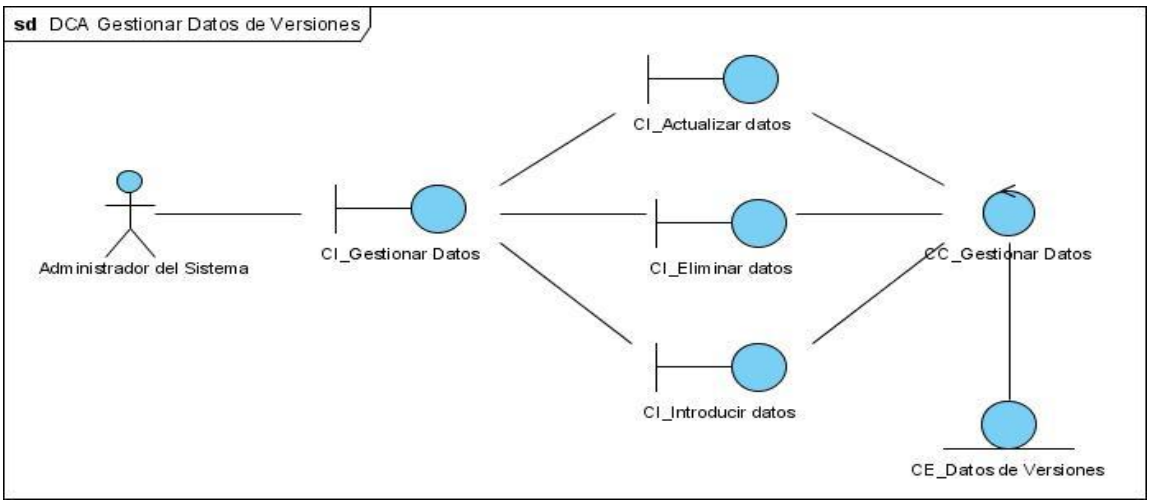


Figura 14 DCA: Gestionar Datos de Versiones.

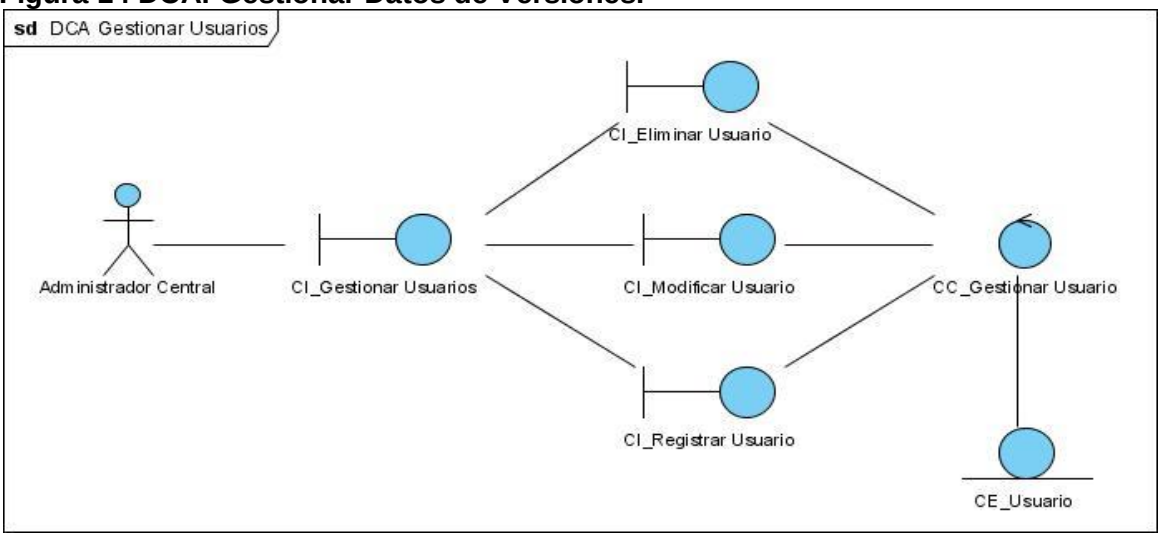


Figura 15 DCA: Gestionar Usuarios.

3.3.1 Diagramas de Colaboración del Análisis.

Los diagramas de interacción representan una vista dinámica del sistema y se pueden clasificar en dos tipos, diagramas de colaboración o diagramas de secuencia. Un diagrama de interacción representa la secuencia de acciones que ocurren desde que el actor comienza el caso de uso, así como los mensajes que se envían entre cada una de las clases. En el análisis se usan los diagramas de colaboración, ya que el objetivo principal es identificar las funcionalidades de cada objeto y las responsabilidades sobre ellos.

Desde el Anexo 1 hasta el Anexo 11 se representan los diagramas de colaboración del análisis del sistema COVER.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.4 Diseño.

En el diseño se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen.

Una entrada esencial en el diseño es el resultado del análisis, que proporciona una comprensión detallada de los requisitos, darle frente al diseño y lograr lo mejor en él, se puede alcanzar mediante los resultados que se obtengan en el análisis ya que este proporciona una comprensión bien definida de los requisitos.

3.4.1 Propósitos del Diseño.

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.
- Capturar las interfaces entre los subsistemas antes en el ciclo de vida del software, lo cuál es muy útil cuando se utiliza interfaces como elementos de sincronización entre diferentes equipos de desarrollo. (16)

3.5 Arquitecturas y Patrones del diseño utilizados.

Los patrones de diseño son soluciones a problemas ya conocidos que ayudan a un mejor rendimiento, desarrollo y mantenimiento del software donde se apliquen, contribuyen a la realización de una arquitectura más factible, simple y entendible. Ayudan a diseñar correctamente en menos tiempo, a construir problemas reutilizables y facilitan la documentación.

3.5.1 Arquitectura n-capas.

El modelo n-tiers (n-capas) de informática ha emergido como la arquitectura predominante para la construcción de aplicaciones multiplataforma en la mayor parte de las empresas debido a sus grandes ventajas.

Esta arquitectura permite hacer que tanto la interfaz de usuario, las reglas de negocios y el motor de datos se conviertan en entidades separadas unas de otras, lo importante es mantener bien definidas las interfaces que cada una de estas expongan para comunicarse con la otra. La que generalmente se evidencia es la de cuatro capas, la capa que se agrega es la que surge de separar definitivamente las reglas de negocio de la de "Acceso a Datos". Esta arquitectura brinda la ventaja de aislar definitivamente nuestra lógica de negocios de todo lo que tenga que ver con el origen de datos, ya que desde el manejo de la conexión, hasta la ejecución de una consulta, la manejará la capa de Acceso a Datos. De este modo, ante cualquier eventual cambio, sólo se deberá tocar un módulo específico, así como al momento de plantear la escalabilidad del sistema, si se respetan las reglas básicas de diseño no se deberían afrontar grandes modificaciones. (17) (18)

Principales ventajas:

- *Abstracción total acerca del origen de datos:* las distintas capas se especializan absolutamente en la funcionalidad que deben brindar, sin importar cuál es el origen de los datos procesados.
- *Bajo costo de desarrollo y mantenimiento de las aplicaciones:* es más sencillo cambiar un componente que modificar una aplicación monolítica, además de que brinda un control más cercano de cada componente, así como también la posibilidad de una verdadera reutilización del código.
- *Aplicaciones más robustas:* debido al encapsulamiento. [48]
- *Mayor flexibilidad:* se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad.
- *Alta escalabilidad:* la principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.
- *Mejor calidad en las aplicaciones:* como las aplicaciones son construidas en unidades separadas, estas pueden ser testeadas independientemente y con mucho más detalle, esto conduce a obtener un producto mucho más sólido. (19)

[48] Encapsulamiento: Propiedad de los objetos de permitir el acceso a su estado únicamente a través de su interfaz o de relaciones preestablecidas con otros objetos

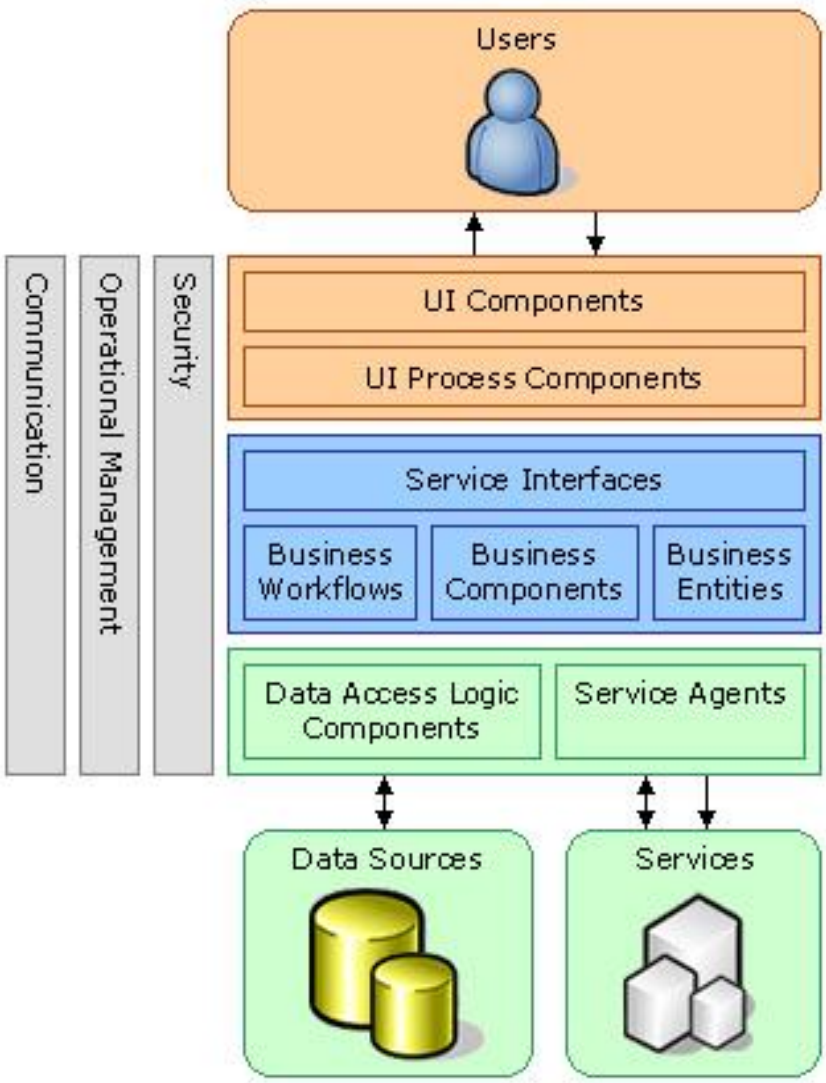


Figura 16 Arquitectura propuesta por Microsoft para aplicaciones .NET.

En la Figura se muestra un ejemplo de arquitectura en n-capas propuesta por Microsoft para el desarrollo de las aplicaciones en su plataforma .net, donde se encuentran bien definidas la capa de presentación o de interfaz de usuario en el nivel superior, donde se encuentran todas las interfaces y componentes de Interfaz de usuario, luego la capa de servicios o lógica de negocio en un nivel más abajo donde están presentes todos los servicios o componentes empresariales del sistema así como las entidades y en el nivel más bajo la capa de acceso a datos que será la que se encargará de interactuar con uno o varios almacenes de datos que utilice la aplicación.

3.5.2 Patrón Facade.

El patrón Facade[49] o Fachada proporciona una interfaz unificada de alto nivel para un subsistema, que oculta las interfaces de bajo nivel de las clases que lo implementan. Con esto se consiguen dos objetivos fundamentales: hacer el subsistema más fácil de usar y desacoplar a los clientes de las clases del subsistema.

Su objetivo es aislar a los clientes de las interfaces de bajo nivel del subsistema colocando entre ambos una clase denominada genéricamente "fachada" del subsistema, cuya interfaz pública recoja precisamente la semántica de los servicios ofrecidos por el subsistema que interesan a los clientes más habituales.

El valor que añade esta clase es el ofrecer a los clientes una forma única y simplificada de acceder a los servicios más generales del subsistema. Para ello, los clientes enviarán mensajes sólo a la fachada, y ésta se encargará de poner en funcionamiento la maquinaria del subsistema para conseguir el objetivo pretendido y devolver al cliente los resultados. (17)

Aplicación:

- Cuando se desee dotar de una interfaz sencilla y usable a un subsistema complejo. Una fachada proporciona una vista por defecto de la funcionalidad del subsistema suficiente para la mayoría de los programadores.
- Cuando se detecten demasiadas dependencias entre las clases clientes de una abstracción y las clases que implementan esta abstracción en un subsistema. En este caso debe introducirse una fachada que permita diseñar a los clientes y otros subsistemas para que dependan de una interfaz y no de una implementación.
- Cuando se quiera estructurar un sistema en subsistemas siguiendo un patrón de capas.
- Será de gran ayuda dotar de una fachada a cada nivel de subsistemas y utilizarla como punto de acceso al mismo. De este modo se simplificará al máximo el mantenimiento de las dependencias entre niveles.
- Cuando se tenga un subsistema que ofrezca una funcionalidad muy rica y compleja y un conjunto significativo de clientes que sólo necesitan usar una parte reducida de la misma **(20)**

[49] Facade (Fachada): simplifica el acceso a un conjunto de clases proporcionando una única clase que todos utilizan para comunicarse con dicho conjunto de clases.

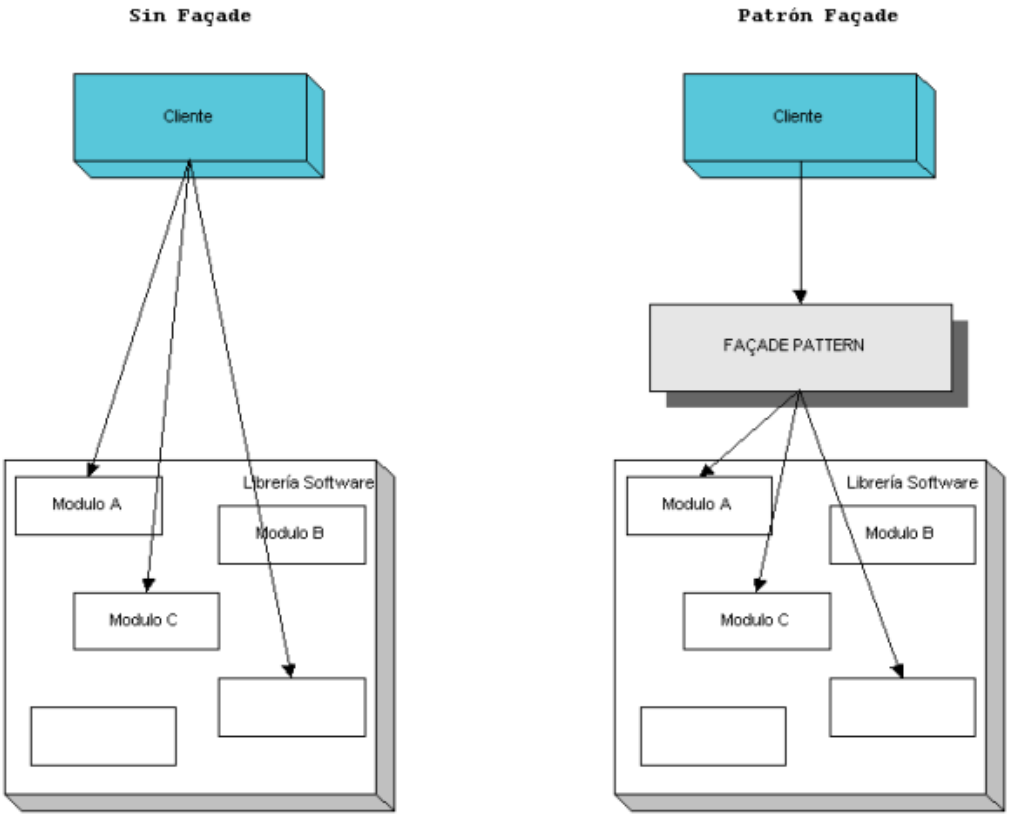


Figura 17 Patrón de diseño Fachada.

3.5.3 Patrón DAO.

El patrón DAO[50] se encarga de separar el acceso a datos del resto de funciones, o sea, tiene que haber independencia del almacén de datos. El DAO implementa el mecanismo de acceso requerido para trabajar con la fuente de datos. Esta fuente de datos puede ser un almacenamiento persistente o un servicio externo. Los componentes de negocio que tratan con el DAO utilizan un interface simple expuesto por el DAO para sus clientes. El DAO oculta completamente los detalles de implementación de la fuente de datos a sus clientes. Como el interface expuesto por el DAO no cambia cuando cambia la implementación de la fuente de datos subyacente, este patrón permite al DAO adaptarse a diferentes esquemas de almacenamiento sin que esto afecte a sus clientes o componentes de negocio. Esencialmente, el DAO actúa como un adaptador entre el componente y la fuente de datos.

El patrón DAO es una solución al problema del diferencial de impedancia entre un programa de aplicación orientado a objetos y una base de datos relacional, empleando únicamente la interfaz de

[50] DAO: Objeto de acceso a datos

programación (API) nativa del manejador de base de datos, o algún otro sustituto como el ODBC, DBI, etc.

El patrón DAO permite el encapsulamiento de componentes. Las clases DAO acceden a la fuente de datos y la encapsula para los objetos clientes. Entendiendo que oculta tanto la fuente como el modo de acceder a ella, logrando así desacoplar la lógica de negocios de la lógica de acceso a datos. Esto permite que la fuente de datos pueda cambiar y no es necesario cambiar la lógica del negocio, solo las API que utiliza la clase DAO para acceder a la fuente. (21)

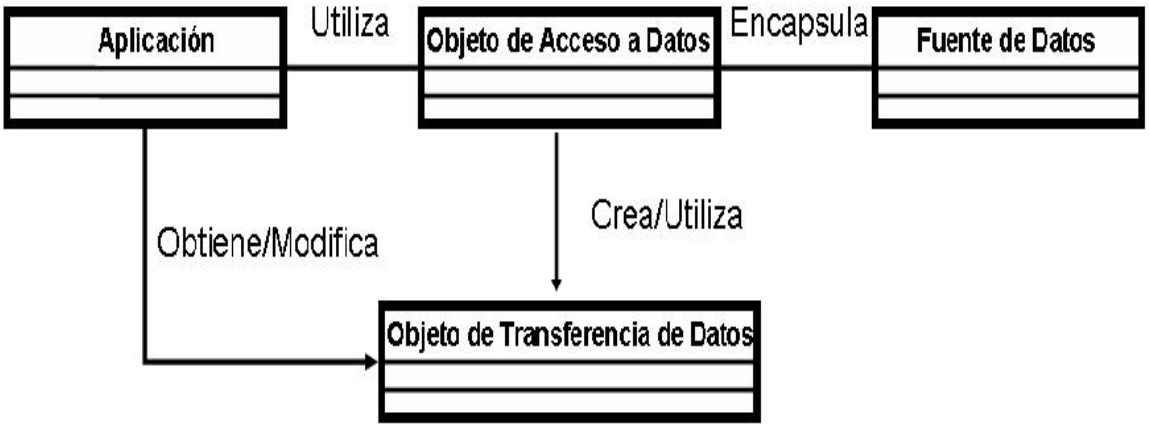


Figura 18 Patrón de diseño DAO.

3.5.4 Arquitectura definida.

La arquitectura definida para el sistema se basa en tener instalado en las máquinas donde se exploten los sistemas informáticos el agente, quien se encargará de recopilar la información solicitada del ordenador y enviarla haciendo uso de una conexión por socket[51] al servidor que tendrá el sistema de administración central, mediante el cual se almacenará la información en una base de datos central a través de una conexión ADO.

En la siguiente figura se representa la explicación anterior.

[51] Socket: Método para la comunicación entre un programa del cliente y un programa del servidor en una red.

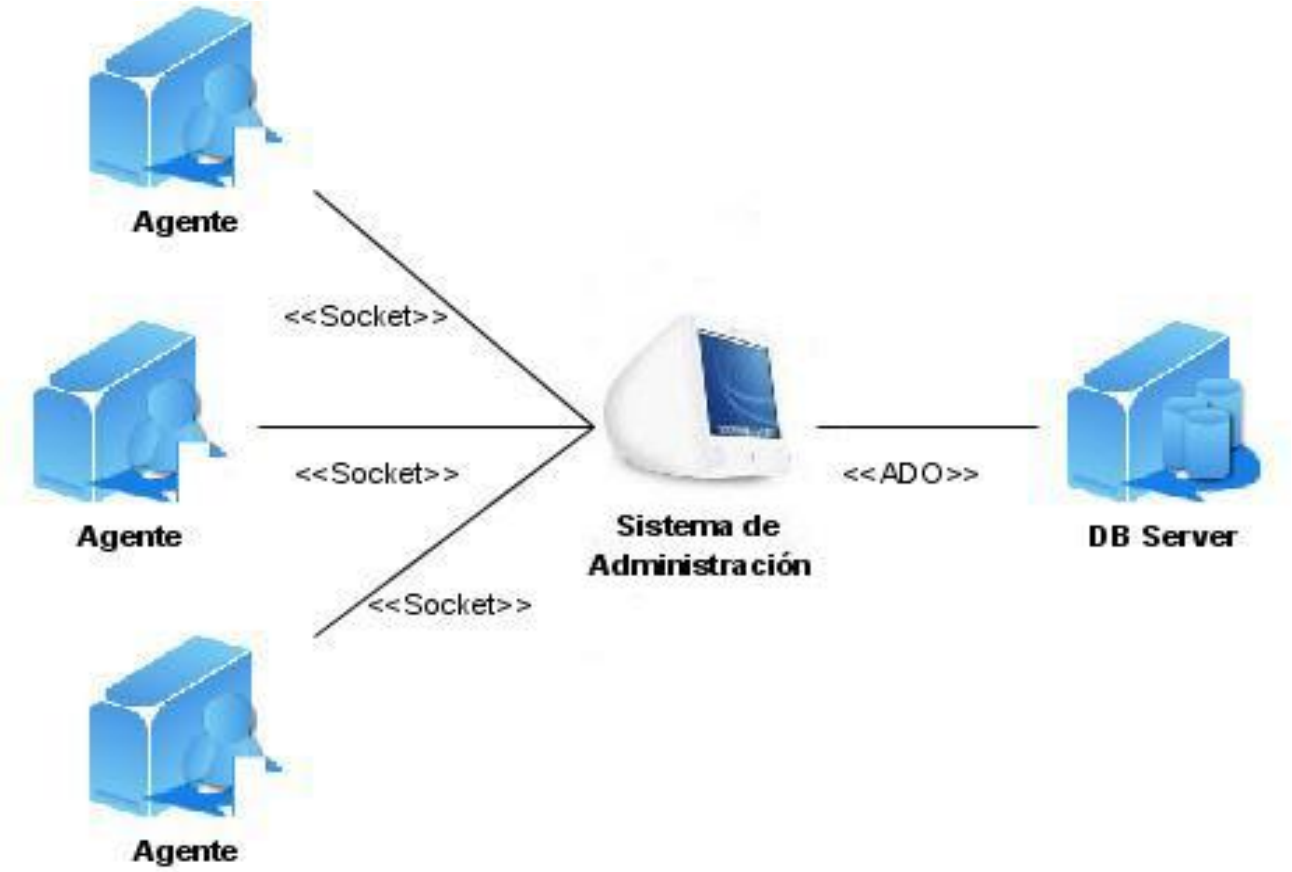


Figura 19 Arquitectura del Sistema.

3.6 Modelo del diseño.

El propósito del diseño es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y fácilmente extensible.

"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. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación". (15) (22)

3.6.1 Realización de los diagramas de clases del diseño.

Una realización de caso de uso – diseño es una colaboración en el modelo de diseño que describe cómo se realiza un caso de uso específico, y cómo se ejecuta, en términos de las clases de diseño y sus objetos. (15)

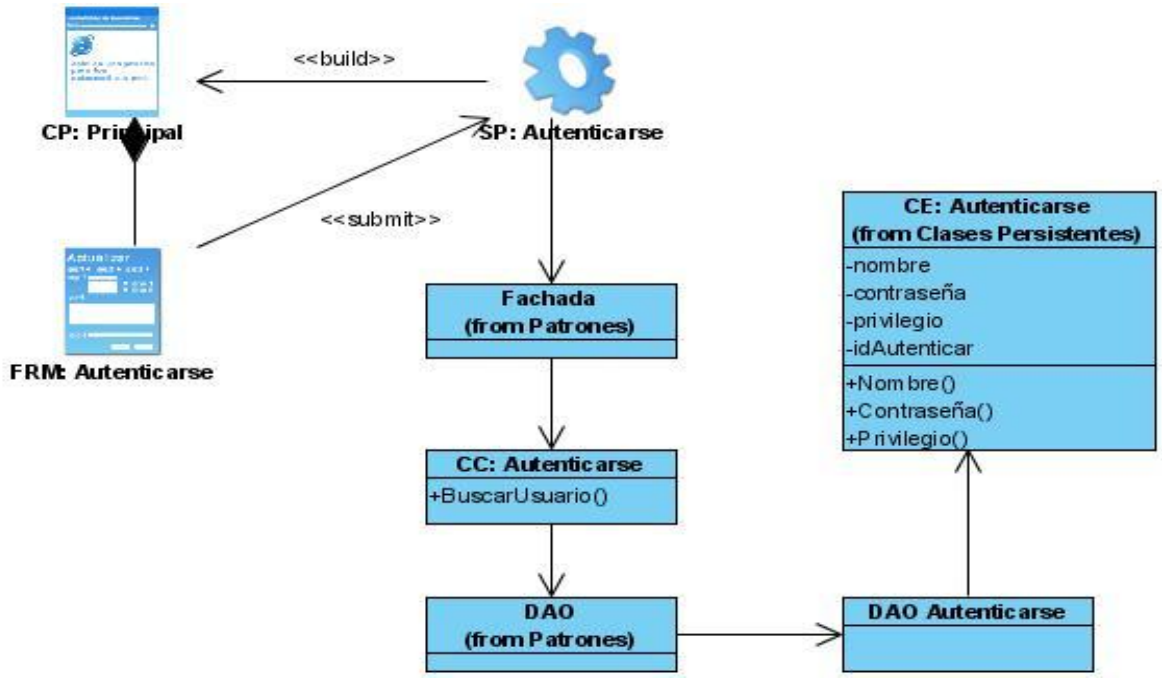


Figura 20 DCD: Autenticarse.

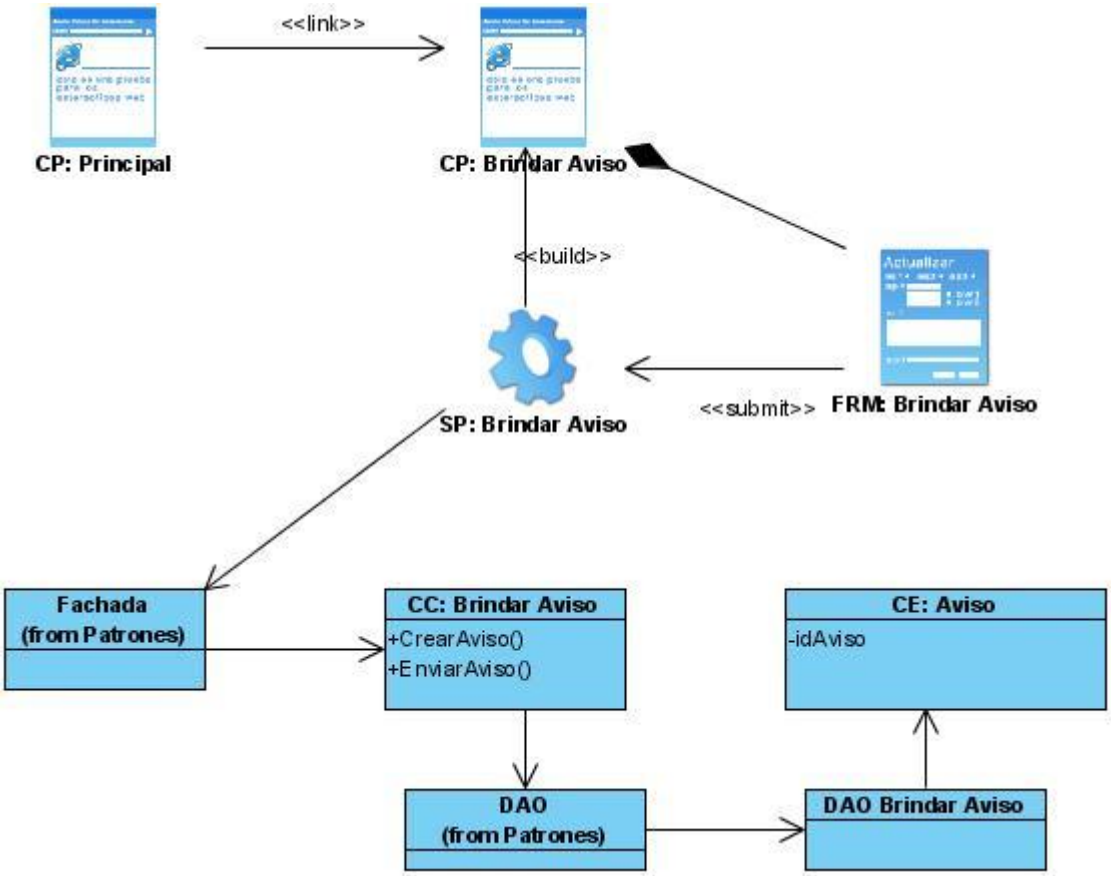


Figura 21 DCD: Brindar Aviso.

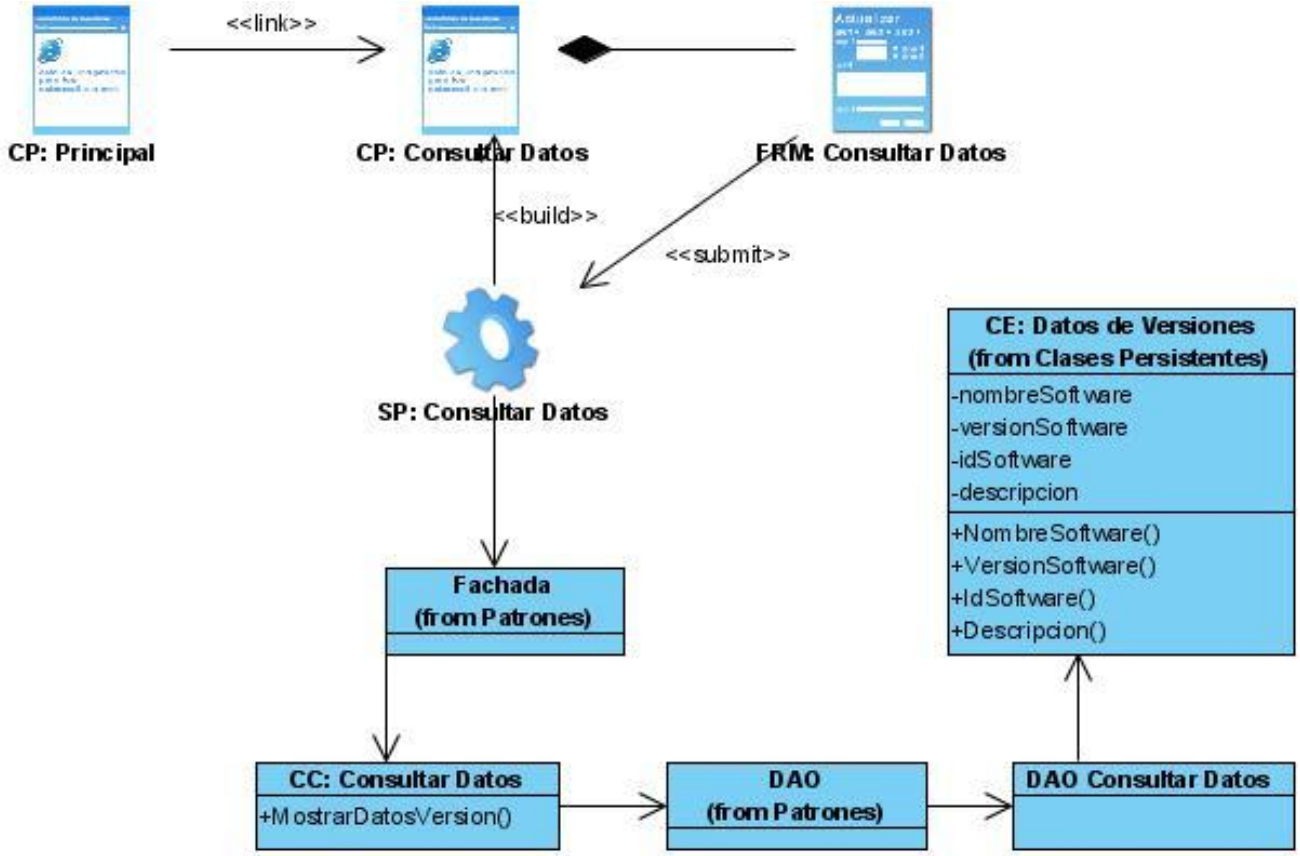


Figura 22 DCD: Consultar Datos.

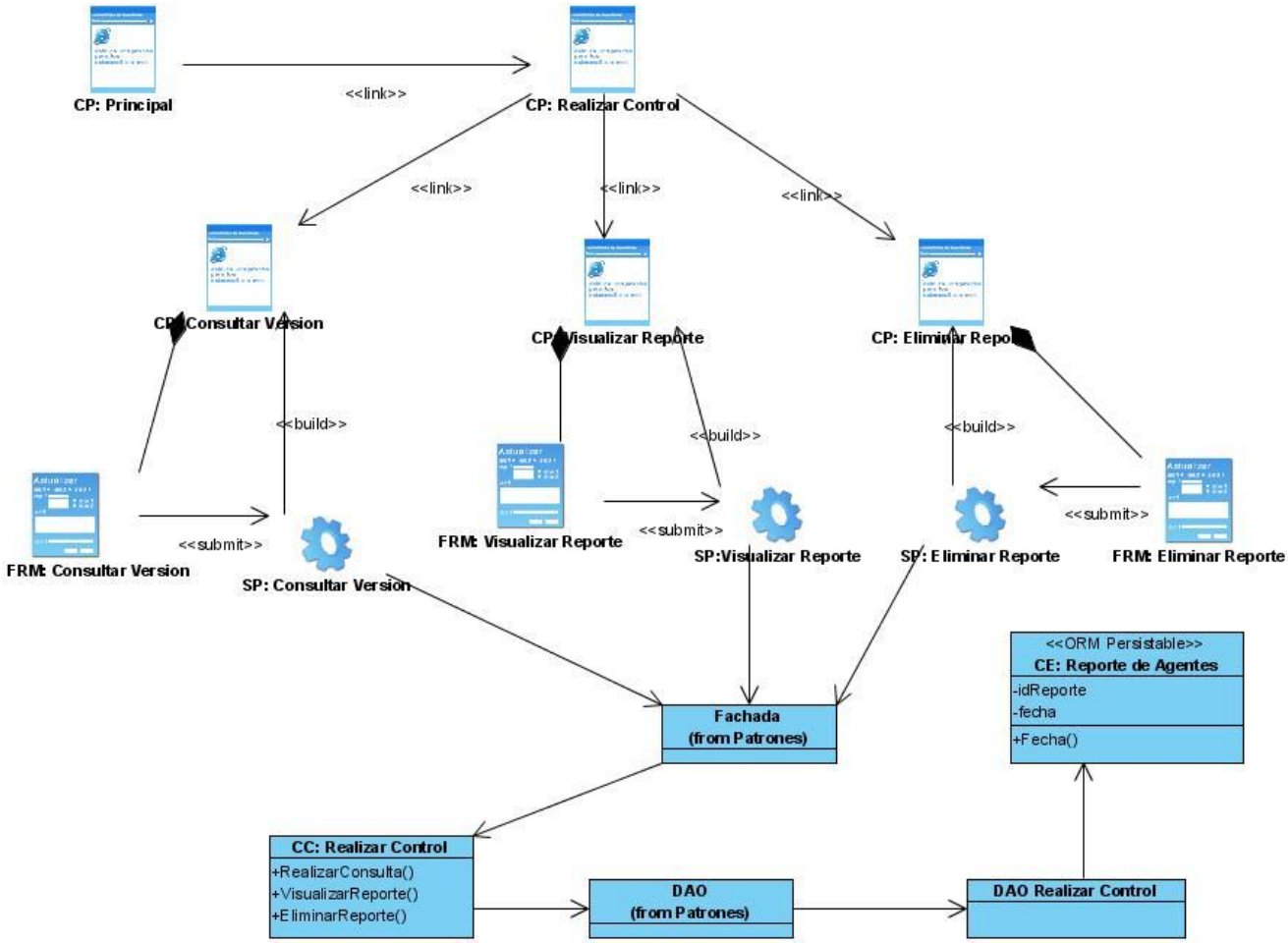


Figura 23 DCD: Realizar Control.

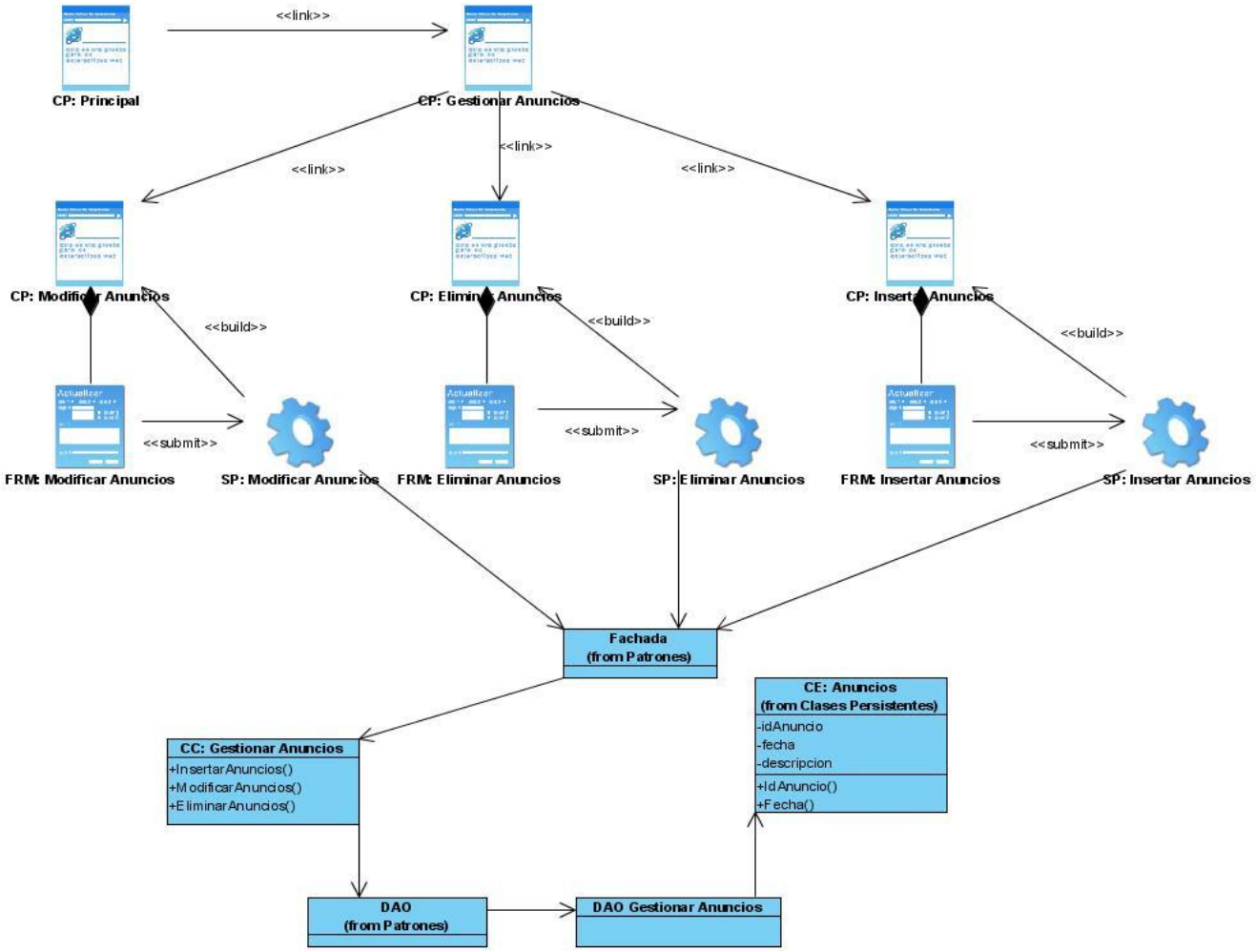


Figura 24 DCD: Gestionar Anuncios.

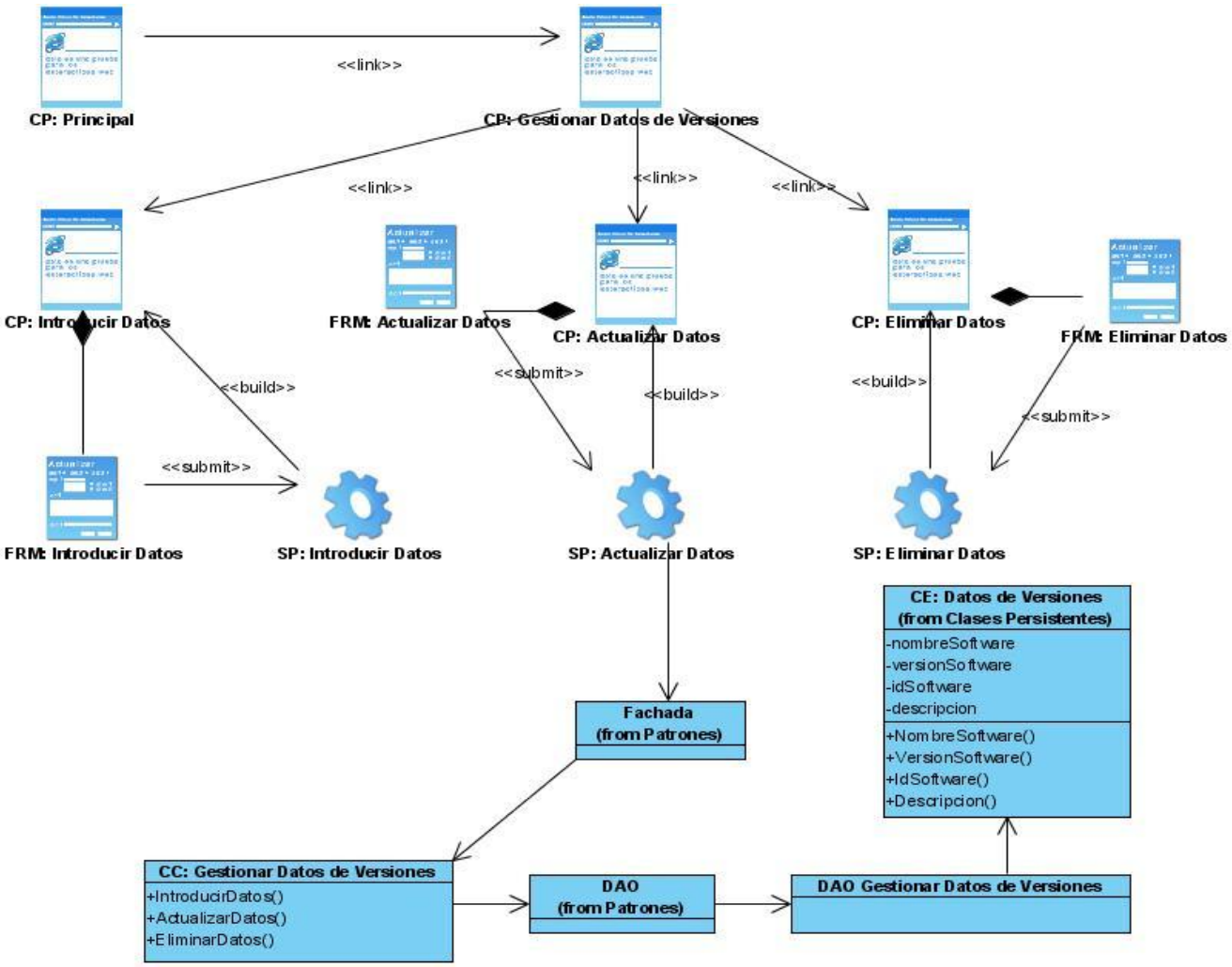


Figura 25 DCD: Gestionar Datos de Versiones.

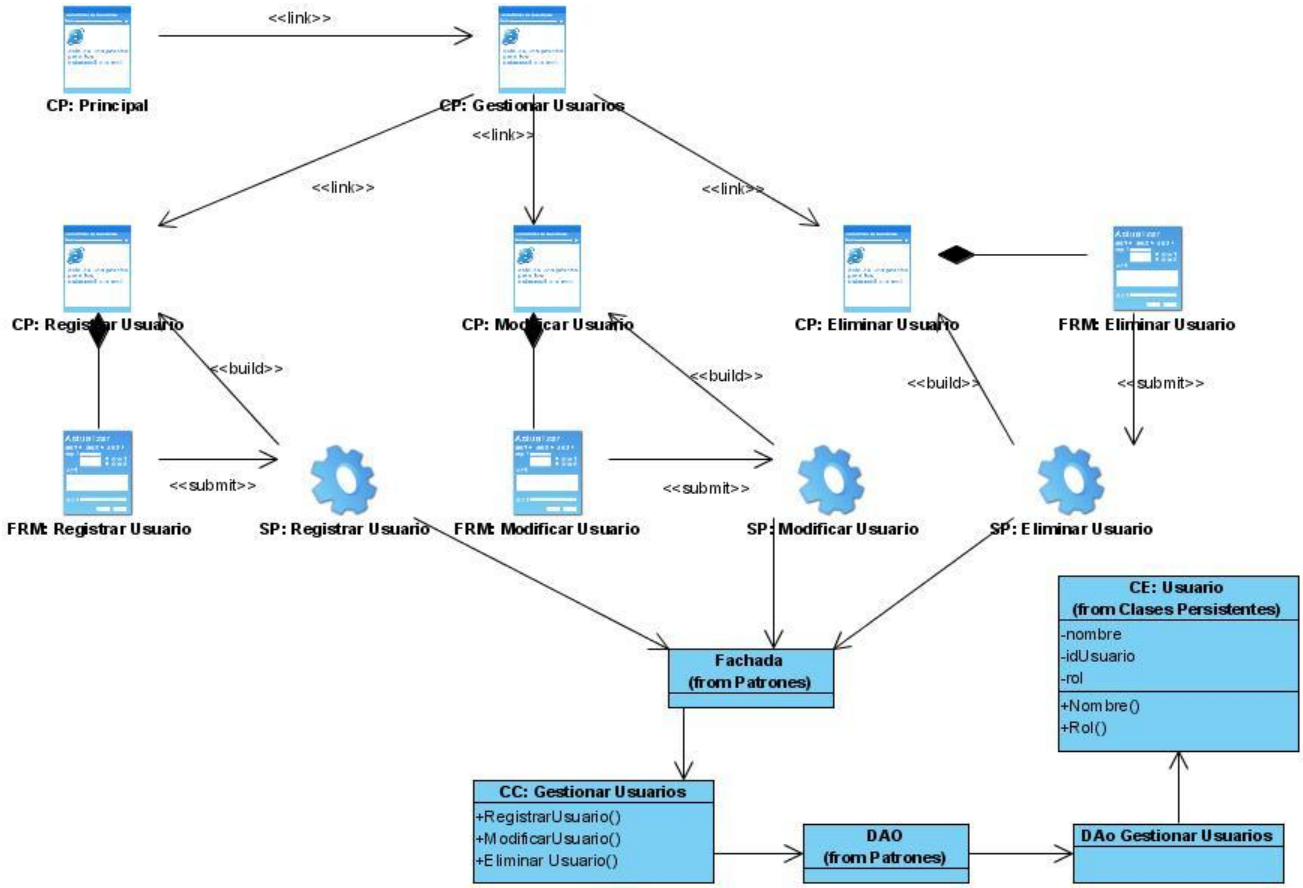


Figura 26 DCD: Gestionar Usuarios.

3.6.2 Descripciones de las clases del diseño.

Tabla 12 DCD: CC_Brindar Aviso.

Nombre:	CC_Brindar Aviso
Tipo de Clase:	Controladora
Descripción:	
Permite al administrador una vez detectada una irregularidad en la utilización de los sistemas informáticos crear un aviso y enviárselo al operador.	

Tabla 13 DCD: CC_Realizar Control.

Nombre:	CC_Realizar Control
Tipo de Clase:	Controladora
Atributo:	**Tipo:**
Reponsabilidad	
Nombre:	Consultar Versión
Descripción:	Permite al administrador solicitar un reporte de las versiones de los sistemas informáticos que están en explotación en la red del ministerio.
Nombre:	Visualizar Reporte
Descripción:	Permite al administrador consultar reportes de control efectuados anteriormente.
Nombre:	Eliminar Reporte
Descripción:	Permite al administrador eliminar los reportes de control que no sean de interés.

Tabla 14 DCD: CC_Consultar Datos.

Nombre:	CC_Consultar Datos
Tipo de Clase:	Controladora
Descripción:	
Permite al operador consultar los datos de un determinado sistema informático con el fin de documentarse más y entender de forma mejor la versión informática que se comenzará a explotar en el ministerio.	

Tabla 15 DCD: CC_ Gestionar Usuarios.

Nombre:	CC_ Gestionar Usuarios
Tipo de Clase:	Controladora
Atributo:	**Tipo:**
Responsabilidad	
Nombre:	Registrar Usuarios
Descripción:	Permite adicionar un usuario dado su nombre de usuario, contraseña y el privilegio.
Nombre:	Modificar Usuarios
Descripción:	Permite modificar los datos de un usuario una vez dado el nombre.
Nombre:	Eliminar Usuarios
Descripción:	El administrador elimina un usuario determinado del sistema retirándole los privilegios con los que contaba anteriormente en la aplicación.

Tabla 16 DCD: CC_ Gestionar Datos de Versiones.

Nombre:	CC_ Gestionar Datos de Versiones
Tipo de Clase:	Controladora
Atributo:	**Tipo:**
Responsabilidad	
Nombre:	Introducir Datos de versiones

Descripción:	El administrador introduce los datos de una determinada versión con el fin que los operadores conozcan mejor la versión a explotar.
Nombre:	Modificar Datos de versiones
Descripción:	El administrador modifica los datos que fueron entrados al sistema.
Nombre:	Eliminar Datos de versiones
Descripción:	El administrador elimina los datos que describen a las versiones de los sistemas informáticos, ya sea por error o porque la versión fue eliminada del sistema.

Tabla 17 DCD: CC_Autenticar.

Nombre:	CC_Autenticar
Tipo de Clase:	Controladora
Descripción:	
Permite al usuario poder acceder al sistema y trabajar en la aplicación según los privilegios que tenga asignado a la hora de ingresar su clave.	

Tabla 18 DCD: CC_ Gestionar Anuncios.

Nombre:	CC_ Gestionar Anuncios
Tipo de Clase:	Controladora
Atributo:	**Tipo:**
Responsabilidad	

Nombre:	Insertar Anuncio
Descripción:	El administrador desea insertar en el sistema algún anuncio para orientar a los operadores de las nuevas versiones a explotar o alguna noticia relevante que necesiten conocer.
Nombre:	Modificar Anuncios
Descripción:	El administrador necesita modificar el anuncio publicado ya sea por tiempo de permanecía en la aplicación o por error a la hora de redactarlo.
Nombre:	Eliminar Anuncio
Descripción:	El administrador necesita eliminar un anuncio determinado, ya sea porque no es necesario que permanezca mas tiempo en el sistema o la información no es la correcta.

3.7 Diseño de la Base de Datos.

En este epígrafe a través del diagrama de clases persistentes y el modelo de datos se representa el diseño de la base de datos, que de una manera lógica representarán las entidades del sistema y sus relaciones que guardarán las informaciones necesarias para la puesta en marcha del Sistema.

3.7.1 Diagrama de Clases Persistentes.

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y el tiempo. Las clases persistentes por lo general tienen como origen las clases clasificadas como entidades porque ellas modelan la información y el comportamiento asociado a algún fenómeno o concepto persistentes.

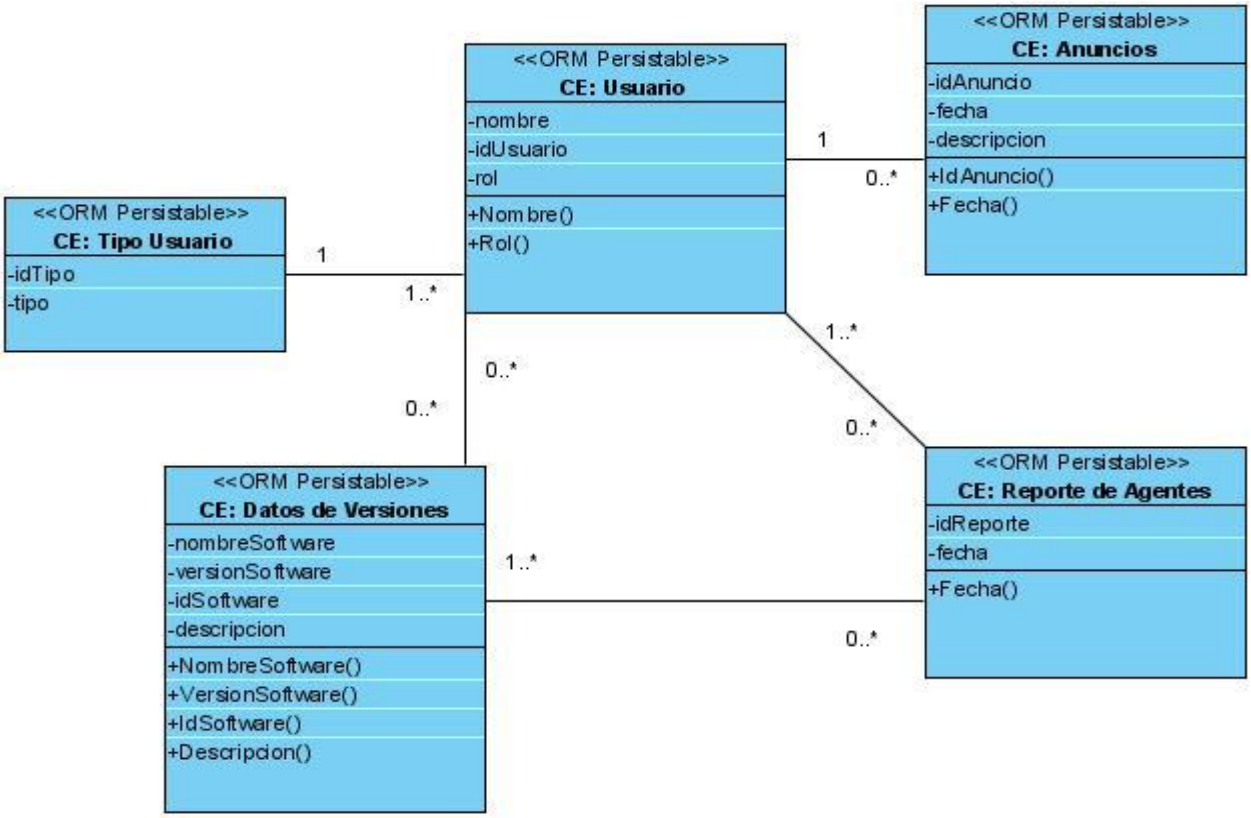


Figura 27 Diagrama de Clases persistentes.

3.7.2 Modelo de Datos.

El modelo de datos es necesario para describir la estructura de una base de datos. Se puede definir como un dispositivo de abstracción que permite ver la información de los datos más que su valor concreto. No es más que la representación de un fenómeno de la realidad objetiva a través de los objetos, sus propiedades y las relaciones que se establecen entre ellos.

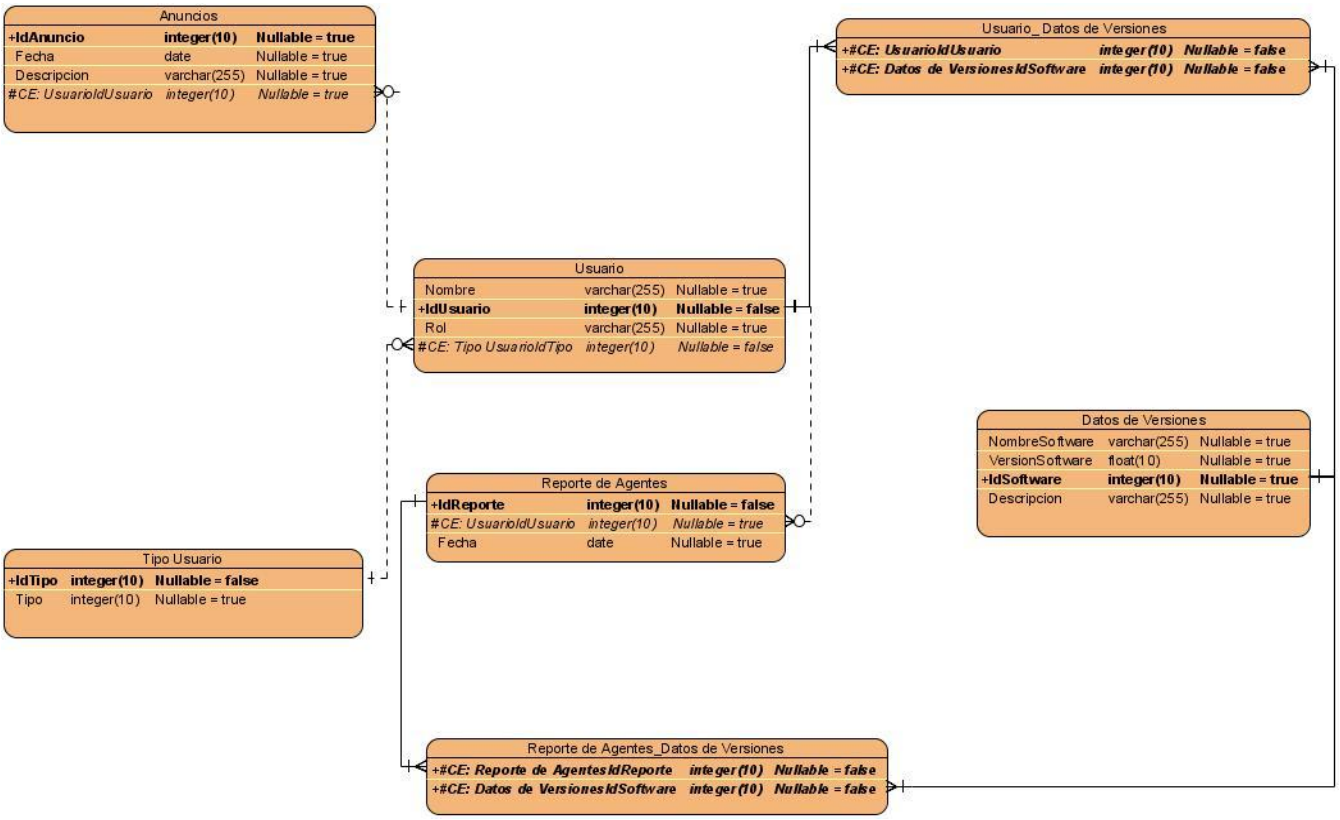


Figura 28 Modelo de Datos.

3.8 Estimación del costo-beneficio.

Estimación por Puntos de Caso de Uso.

La Estimación por Puntos de Caso de Uso es un método de estimación de esfuerzo de un proyecto de desarrollo de software a partir de los casos de uso.

3.8.1. Desarrollo del método “Puntos de Casos de Uso”.

La estimación mediante el análisis de puntos de casos de uso es un método propuesto originalmente por Gustav Karner de Objectory AB, y posteriormente refinado por muchos otros autores.

Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.

Cálculo de los Puntos de Casos de Uso sin ajustar.

$$UUCP = UAW + UUCW$$

Donde:

UUCP: Puntos de Casos de Uso sin ajustar.

UAW: Factor de Peso de los Actores sin ajustar.

UUCW: Factor de Peso de los Casos de Uso sin Ajustar.

Factor de peso de los actores sin ajustar (**UUCW**).

Este valor se calcula mediante un análisis de la cantidad de actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los actores se establece teniendo en cuenta en primer lugar si se trata de una persona o de otro sistema, y en segundo lugar, la forma en la que el actor interactúa con el sistema, los criterios se muestran en la siguiente tabla:

Tabla 19 Factor de Peso de los Actores sin ajustar.

Tipo de actor	Descripción	Factor de Peso	Actores	Total
Simple	Otro sistema que interactúa con la aplicación a desarrollar mediante una interfaz de programación.	1	0	0
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2	0	0
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	3	9
$UAW = \sum CantidadActores_i * Peso_i$				

Σ UAW = cant actores * peso

UAW = 3*3

UAW = 9

Factor de Peso de casos de uso sin ajustar (UUCW).

Este valor se calcula mediante un estudio de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se define teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción es una secuencia de actividades completa. Los criterios se muestran en la siguiente tabla:

Tabla 20 Factor de Peso de los Casos de Uso sin ajustar.

Tipo de CU	Descripción	Factor de Peso	Cantidad de CU	Total
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	7	35
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	0	
Complejo	El caso de uso tiene más de 8 Transacciones.	15	0	
$UUCW = \sum CantidadCU_i * Peso_i$				

UUCW = ΣcantCU ∗ Peso

UUCW = 7 * 5

UUCW = 35

Finalmente, los **Puntos de Casos de Uso sin ajustar** resultan:

UUCP = **UAW** + **UUCW**

UUCP = 9 + 35

UUCP = 44

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe ajustar éste valor mediante la siguiente ecuación:

UCP = UUCP x TCF x EF

Donde:

UCP: Puntos de Casos de Uso ajustados.

UUCP: Puntos de Casos de Uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

Factor de complejidad técnica (TCF).

Este coeficiente se calcula gracias a la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5 donde 0 significa un aporte irrelevante y 5 un aporte muy importante, en la siguiente tabla se muestra el significado y el peso de cada uno de estos factores:

Tabla 21 Factor de complejidad técnica.

Factor	Descripción	Peso	Valor Asignado	Total
T1	Sistema distribuido.	2	0	0
T2	Tiempo de respuesta.	1	3	3
T3	Eficiencia del usuario final	1	4	4
T4	Funcionamiento Interno complejo	1	3	3
T5	El código debe ser reutilizable	1	3	3
T6	Facilidad de instalación	0.5	4	2
T7	Facilidad de uso	0.5	5	2.5
T8	Portabilidad	2	0	0
T9	Facilidad de cambio	1	3	3
T10	Concurrencia	1	3	3
T11	Incluye objetivos especiales de seguridad	1	3	3
T12	Provee acceso directo a terceras partes	1	0	0
T13	Se requieren facilidades especiales de	1	2	2

	entrenamientos a usuarios.			
Sumatoria				28.5
$TCF = 0.6 + 0.01 * \sum (Peso_i * ValorAsignado_i)$				

- Para los factores E1 al E4, un valor asignado de 0 significa sin experiencia, 3 experiencia media y 5 amplia experiencia (experto).
- Para el factor E5, 0 significa sin motivación para el proyecto, 3 motivación media, 5 alta motivación.
- Para el factor E6, 0 significa requerimientos extremadamente inestables, 3 estabilidad media, 5 requerimientos estables sin posibilidad de cambios.
- Para el factor E7, 0 significa que no hay personal part-time, 3 significa mitad y mitad, 5 significa que todo el personal es part-time.
- Para el factor E8, 0 significa que el lenguaje de programación es fácil de usar, 3 medio y 5 que el lenguaje es extremadamente difícil.

EF = 1.4 - 0.03 x Σ (Pesoi x Valor asignadoi)

EF = 1.4 – 0.03 * 21.5

EF = 1.4 – 0.65

EF = 0.75

Finalmente, Puntos de Casos de Uso ajustados resultan:

UCP = **UUCP** * **TCF** * **EF**

UCP = 44 * 0.89 * 0.75

UCP = 29.37

Calculo del esfuerzo del flujo de trabajo implementación mediante la siguiente ecuación:

E = UCP * CF

Donde:

E: Esfuerzo estimado en horas-hombre.

UCP: Puntos de casos de usos ajustados.

CF: Factor de conversión.

Para el cálculo del **Factor de Conversión (CF)** se cuentan cuántos factores de los que afectan el factor de ambiente (E1…E6) están por debajo del valor medio (3), además de los restantes (E7 y E8) que se encuentran por encima de la media (3).

Si esa cantidad es 2 o menos, se utiliza el factor de conversión 20 Horas-Hombre / Punto de Casos de uso.

Si esa cantidad es 3 o 4, se utiliza el factor de conversión 28 Horas-Hombre / Punto de Casos de uso. Si esa cantidad es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

Por lo tanto se puede decir que es factible el desarrollo del proyecto y:

CF = 20 Horas-Hombre / Punto de Casos de uso

E = 29.37 * 20

E = 587.4 Horas-Hombre

Este método proporciona una estimación del esfuerzo en horas-hombre contemplando solo el desarrollo de la funcionalidad especificada en los casos de uso.

Para una estimación mas completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software.

Para ello se puede tener en cuenta el siguiente criterio, que estadísticamente se considera aceptable. El criterio plantea la distribución del esfuerzo entre las diferentes actividades de un proyecto, según la siguiente aproximación:

Tabla 22 Esfuerzo del proyecto.

Actividad	Porcentaje	Horas/Hombre
Análisis	20.00%	293.70
Diseño	20.00%	293.70

Implementación	40.00%	587.40
Pruebas	10.00%	146.85
Sobrecarga (otras actividades)	10.00%	146.85
Total	100%	1468.5

Obviamente, estos valores no son absolutos sino que pueden variar de acuerdo a las características de la organización y del proyecto, y tomando como entrada la estimación del tiempo calculada a partir de los Puntos de Casos de Uso, se pueden calcular las demás estimaciones para obtener la duración total del proyecto.

El **E** = 1468.5 Horas-Hombre

Estimando que:

Cada mes tiene 4 semanas, cada semana 5 días laborables, y cada día 6 horas laborables nos quedaría como promedio 120 horas laborables, entonces

ET = E / horas laborables

ET = 1468.5 / 120

ET = 12.2375 mes- hombres

Si:

Tiempo = Esfuerzo Total (ET) /Cantidad de Hombres (CH)

Tiempo = 12.2375/ 2

Tiempo = 6.11875

Esto quiere decir que con 2 hombres trabajando en el proyecto el mismo se desarrolla en aproximadamente 6 meses.

3.8.2 Costo del Proyecto.

Se asume como salario promedio mensual $170.00

Cantidad de hombres (CH) = 2

Tiempo: Tiempo total del proyecto.

CHM = **CH** * **SalarioPromedio**

CHM = 2 * 170

CHM = 340

Costo = **CHM** * **Tiempo**
Costo = 340 * 6.11875

Costo = 2080.375

De los resultados obtenidos se interpreta que con 2 hombres trabajando en el proyecto por un tiempo de aproximadamente 6 meses y consumiendo los recursos antes mencionados, se estima un total de $2080.375.

3.9 Conclusiones.

En este capítulo se abordó el tema de la arquitectura y patrones que serán utilizados en el desarrollo del sistema, se hizo una estimación de esfuerzo total del proyecto a partir de los casos de uso con los que se cuenta, se realizó el modelo de datos y las clases persistentes y se entendió de una mejor forma el sistema que será realizado.

CONCLUSIONES.

Con la realización del trabajo se demuestra la necesidad que existe de llevar a cabo un control sistemático en las versiones de productos informáticos que se utilizan actualmente en el ministerio.

Para presentar una propuesta que solucionará el problema existente se realizó un estudio de las herramientas a utilizar, y se llegó a la conclusión que para realizar un trabajo efectivo se deberían utilizar las siguientes tecnologías, para la interfaz de usuario se utilizaría Microsoft SharePoint, como lenguaje de programación C#, como sistema gestor de base de datos se utilizará Oracle DB y para el modelado se escogió Visual Paradigm.

Se realizaron entrevistas para profundizar en el sistema a desarrollar definiéndose los actores que intervendrían en el sistema, se realizaron los diferentes diagramas de análisis y diseño, así como los requerimientos funcionales con que contará el sistema COVER.

Se obtuvo un prototipo de interfaz que responde a la necesidad del sistema y abre un camino para la implementación de este.

Lo que se propone en el trabajo de diploma, ofrecerá ventajas en el control de versiones en las distintas unidades del ministerio desplegadas por todo el país, así como distintos servicios con el fin de tener un control estricto por parte de los administradores del sistema que son los que tendrán los privilegios y serán también los que operarán el sistema central cuando sea creado.

RECOMENDACIONES.

- Llegar a completar el sistema pasando a su próxima fase, implementar una primera versión del software para obtener resultados funcionales.
- Una vez implementado el sistema capacitar al personal que lo utilizará directamente.

BIBLIOGRAFÍA.

1. SISTEMAS DE CONTROL DE VERSIONES. [En línea] [Citado el: 15 de Enero de 2009.] http://blog.g2peru.com/2008/12/19/sistemas-de-control-de-versiones.

2. **Franco, M y Catrin, L.** *Uso práctico de CVS para control de versiones: Conceptos y prácticas recomendadas.*

3. **BEN COLLINS-SUSSMAN, BRIAN W. FITZPATRICK, C. MICHAEL PILATO.** *Version Control with Subversion.*

4. PROTOCOLO CMIP. [En línea] http://www.todoexpertos.com.

5. **M., HUIDOBRO J.** *SNMP. Un protocolo simple de gestión.* .

6. **Hernández, Danaysa Macías.** *Sistema Integrado de Gestión Estadística.* Ciudad de la Habana : UCI, 2007.

7. **Booch, Grady y Rumbaugh, James,Jacobson, Ivar.** *EL LENGUAJE UNIFICADO DE MODELADO.* Ciudad Habana : s.n.

8. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.*

9. *Assistance, Office IT and Servers User. Deployment for Office SharePoint Server 2007.* Ciudad Habana : s.n.

10. **Gascón, Manuel de la Herrán.** Administración y Optimización de Bases de Datos Oracle. [En línea] http://www.oracle.com .

11. **Corporation, Oracle.** *Oracle Database Documentation Library.* 2005.

12. **Corporation, Microsoft.** MSDN. [En línea] [Citado el: 4 de Febrero de 2009.] http://msdn.microsoft.com/es-es/library/z1zx9t92(VS.80).aspx.

13. **Dapena, MSc. Martha D. Delgado.** *Definición del modelo del negocio y del dominio utilizando Razonamiento Basado en Casos.* Ciudad Habana : s.n.

14. **Informatica, Ministerio del Poder Popular para las Telecomunicaciones y la.** MeRinde. [En línea] [Citado el: 4 de Febrero de 2009.] http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=19&Itemid=103.

15. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar.** *EL LENGUAJE UNIFICADO DE MODELADO.* Ciudad Habana : s.n.

16. **Adilen, García, Arias.** *Análisis y Diseño del Módulo de Docencia del Centro de Salud Mental.* Ciudad de la Habana : s.n., 2008.

17. **García, Javier Sevilla Peña y Abduly, Díaz.** *Sistema de Gestión de Información de la Facultad 8.* Ciudad de la Habana : s.n., 2007.

18. Generator FD. [En línea] http://www.generatorfd.com/Arquitectura.aspx.

19. **R.Orfali, D.Harkey y Edwards, J.** *Arquitectura Cliente/Servidor.* 2005.

20. **Welicki, León E. y García, Fernando Cano.** *Desarrollo Multilenguaje con Patrones de Diseño en la plataforma .NET.*

21. **Lago, Ramiro.** http://www.proactiva-calidad.com. *http://www.proactiva-calidad.com.* [En línea] 2008. [Citado el: 8 de Febrero de 2009.] http://www.proactiva-calidad.com/java/patrones/DAO.html. 13.

22. **Larman, Craig.** *UML y Patrones.*

23. *http://www.generatorfd.com/Arquitectura.aspx.* [En línea] http://www.generatorfd.com.

24. **León E. Welicki, Fernando Cano García.** *Desarrollo Multilenguaje con Patrones de Diseño en la plataforma .NET.*

25. **García, Javier Sevilla Peña y Díaz, Abduly.** *Sistema de Gestión de Información de la Facultad 8.* Ciudad de la Habana : s.n., 2007.

26. **COLLINS-SUSSMAN, BEN, FITZPATRICK, BRIAN W. y PILATO, C. MICHAEL.** Version Control with Subversion. [En línea] [Citado el: 17 de Enero de 2009.] http://svnbook.red-bean.com.

27. **Stansberry, Glen.** Smashing Magazine . [En línea] [Citado el: 2009 de Enero de 17.] http://www.smashingmagazine.com/2008/09/18/the-top-7-open-source-version-control-systems/.

28. Selenic Consulting. [En línea] [Citado el: 2009 de Enero de 18.] http://www.selenic.com/mercurial/wiki/SpanishUnderstandingMercurial.

29. Bazaar. [En línea] [Citado el: 2009 de Enero de 18.] http://doc.bazaar-vcs.org/bzr.dev/es/mini-tutorial/index.html#aprendiendo-m-s.

30. Todo Expertos. [En línea] [Citado el: 2 de Febrero de 2009.] http://www.todoexpertos.com/categorias/tecnologia-e-internet/redes-de-computadores/respuestas/135528/cmip.

31. **HUIDOBRO, J. Manuel.** Colegio Oficial/Asociación Española de Ingenieros de Telecomunicación. [En línea] [Citado el: 2 de Febrero de 2009.] http://www.coit.es/publicac/publbit/bit102/quees.htm.

32. **2007, Microsoft Office SharePoint Server.** Microsoft. [En línea] [Citado el: 4 de Febrero de 2009.] http://office.microsoft.com/es-es/sharepointserver/HA101656533082.aspx.

33. Universidad Autónoma del Estado de Morelos. [En línea] [Citado el: 4 de Febrero de 2009.] http://www.uaem.mx/posgrado/mcruz/cursos/miic/oracle3.ppt.

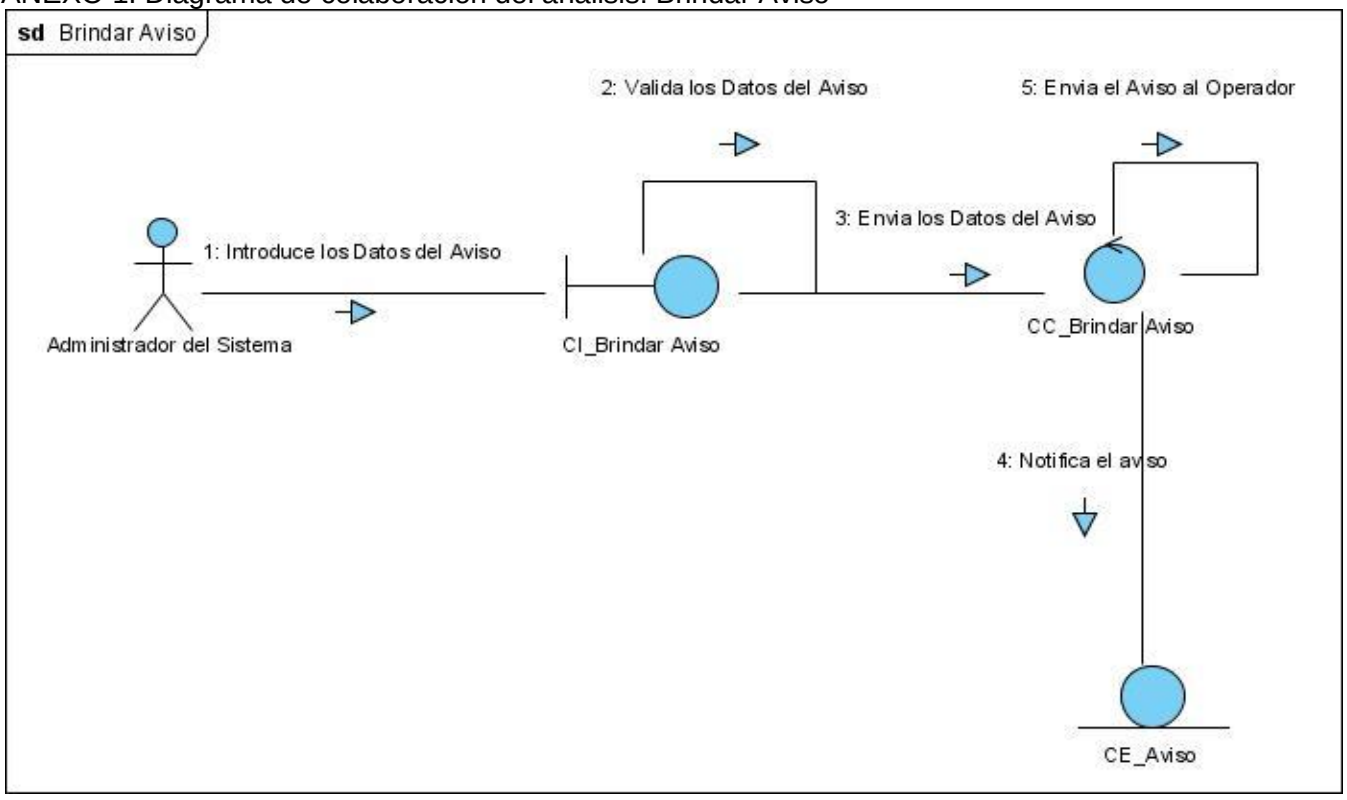
34. **UML, Visual Paradigm for.** Visual Paradigm. [En línea] [Citado el: 6 de Febrero de 2009.] http://www.visual-paradigm.com/product/vpuml/.

35. **Vilmavis, La Rosa Sordo, Yunet, González Mulet y Yadira, Marrero López.** *Solución Arquitectónica para proyectos de Software del Polo Productivo Bioinformática.* Ciudad Habana : s.n.

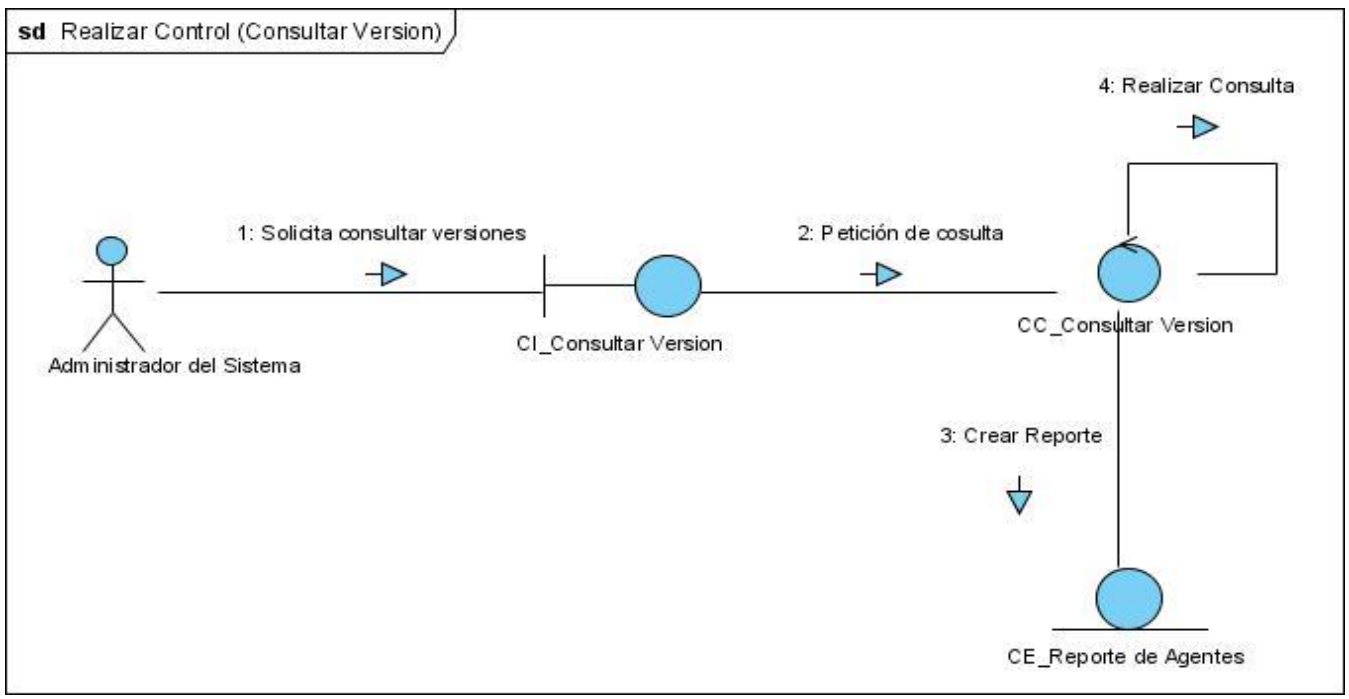
36. **Romero, Mª del Carmen.** Departamento de Tecnología Electrónica. [En línea] [Citado el: 6 de Febrero de 2009.] http://www.dte.us.es/ing_inf/sac/material/gestion-de-redes-2005_4transpaxcara.pdf.

ANEXOS.

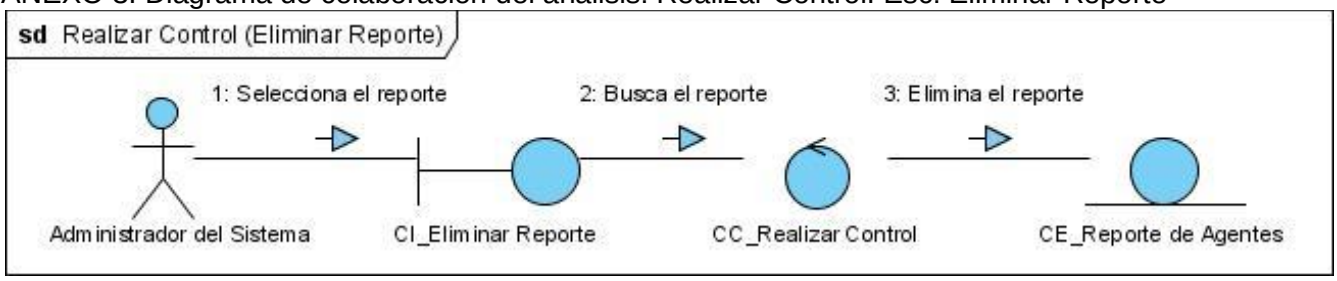
ANEXO 1. Diagrama de colaboración del análisis: Brindar Aviso



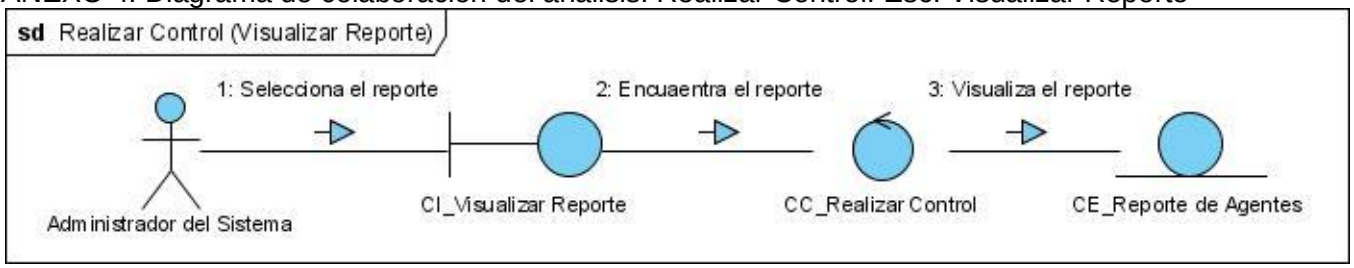
ANEXO 2. Diagrama de colaboración del análisis: Realizar Control. Esc: Consultar Versión



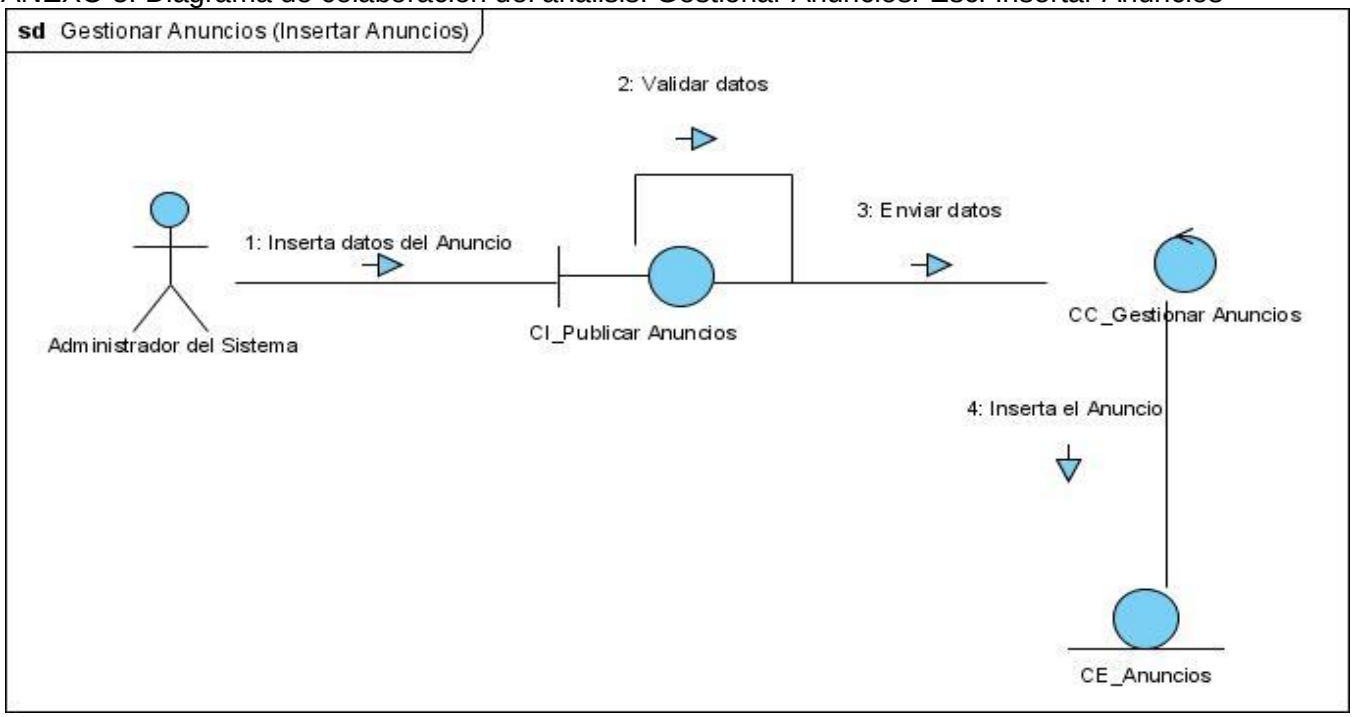
ANEXO 3. Diagrama de colaboración del análisis: Realizar Control. Esc: Eliminar Reporte



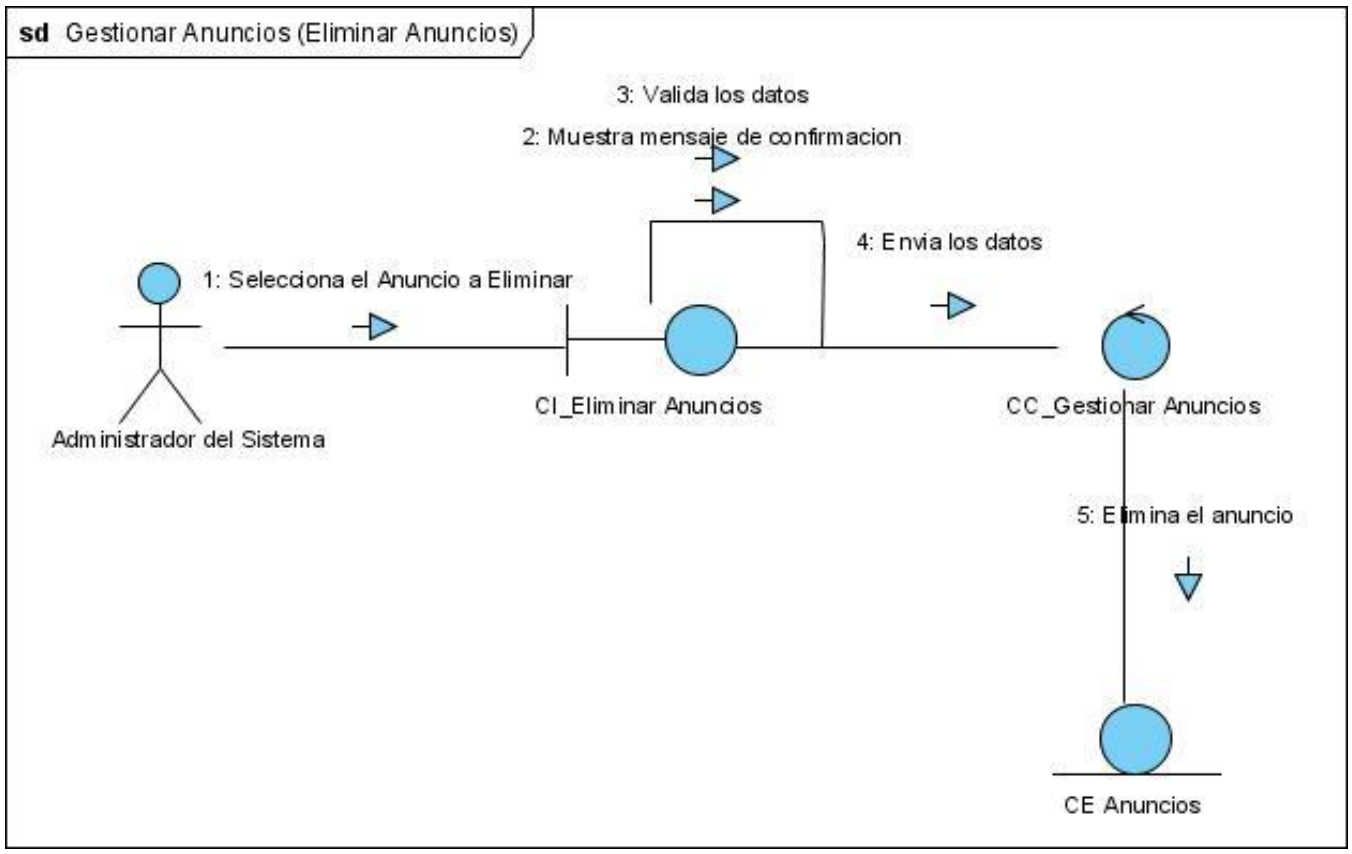
ANEXO 4. Diagrama de colaboración del análisis: Realizar Control. Esc: Visualizar Reporte



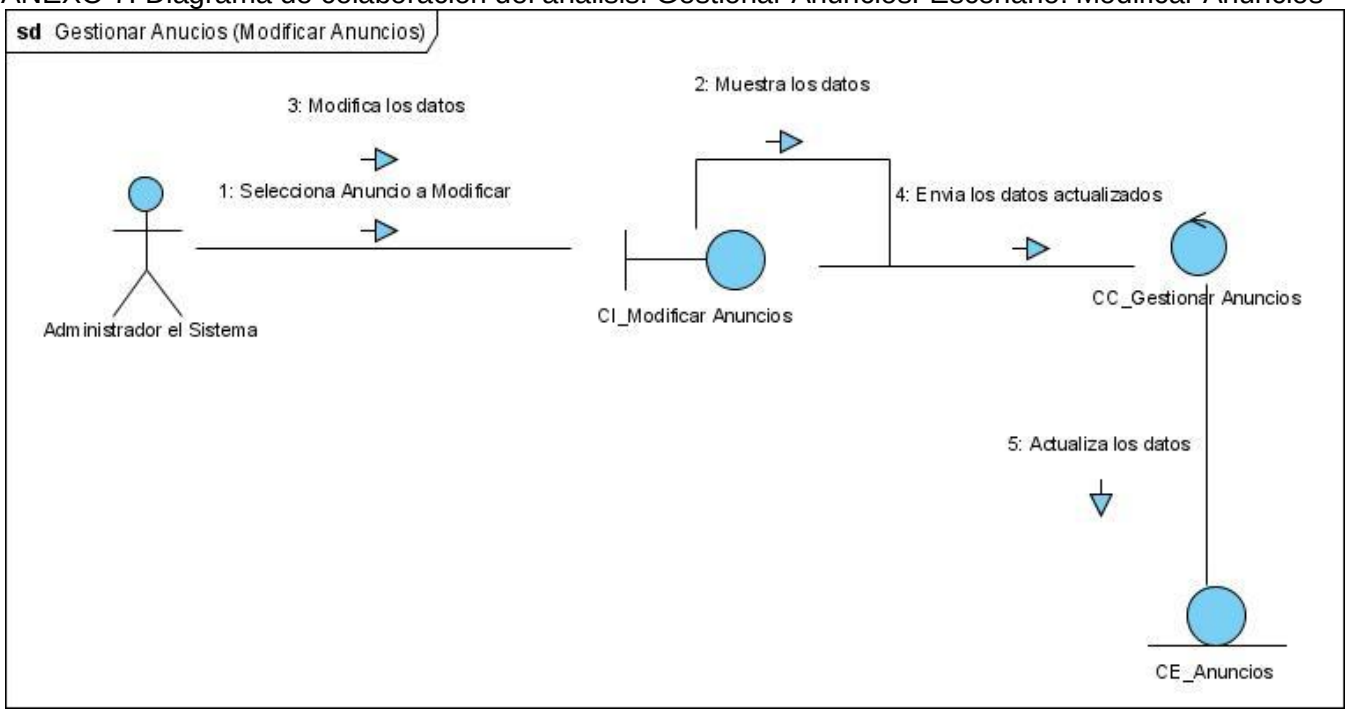
ANEXO 5. Diagrama de colaboración del análisis: Gestionar Anuncios. Esc: Insertar Anuncios



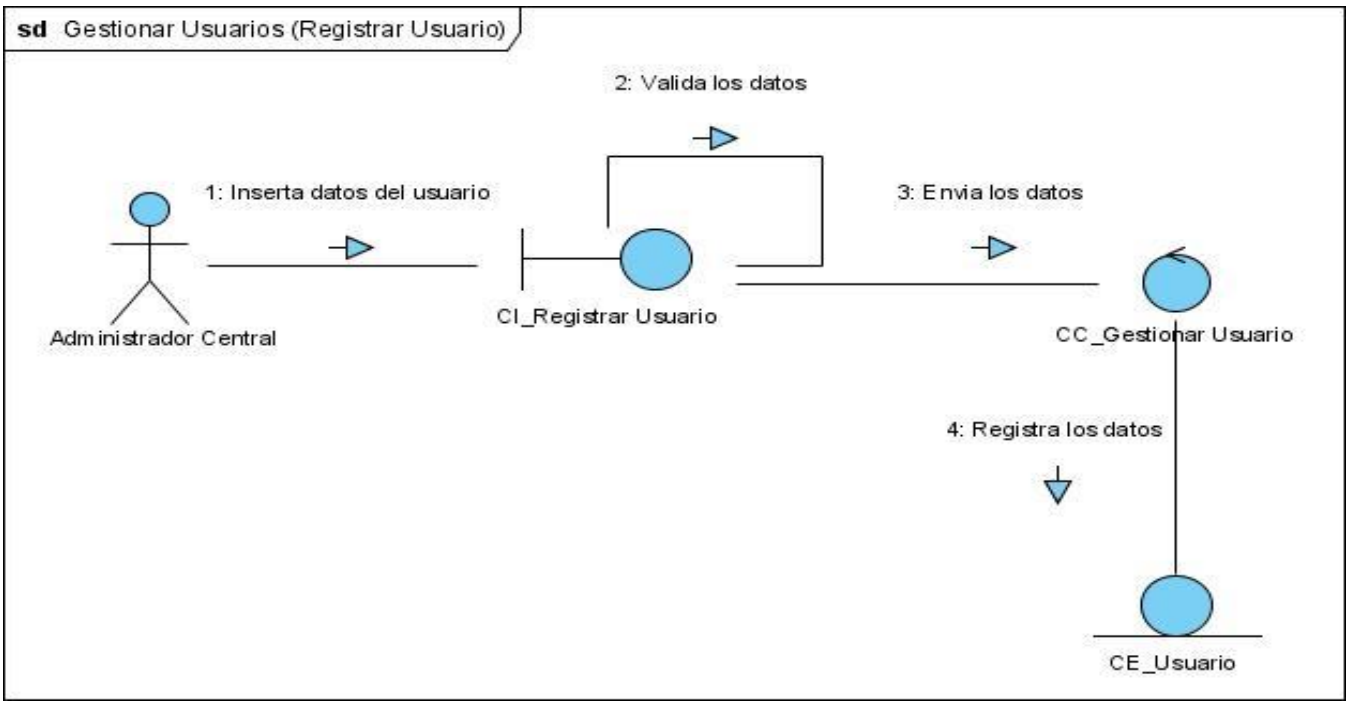
ANEXO 6. Diagrama de colaboración del análisis: Gestionar Anuncios. Escenario: Eliminar Anuncios



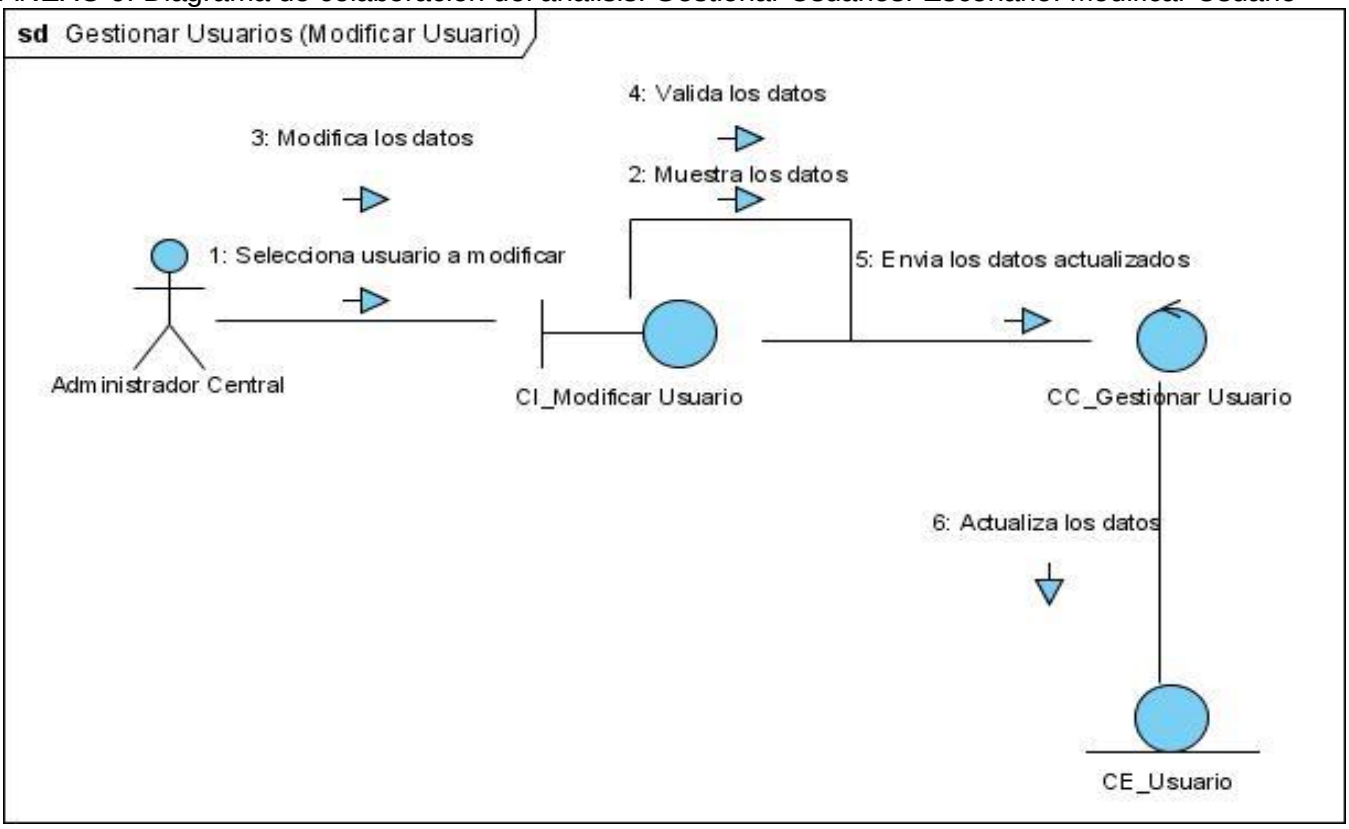
ANEXO 7. Diagrama de colaboración del análisis: Gestionar Anuncios. Escenario: Modificar Anuncios



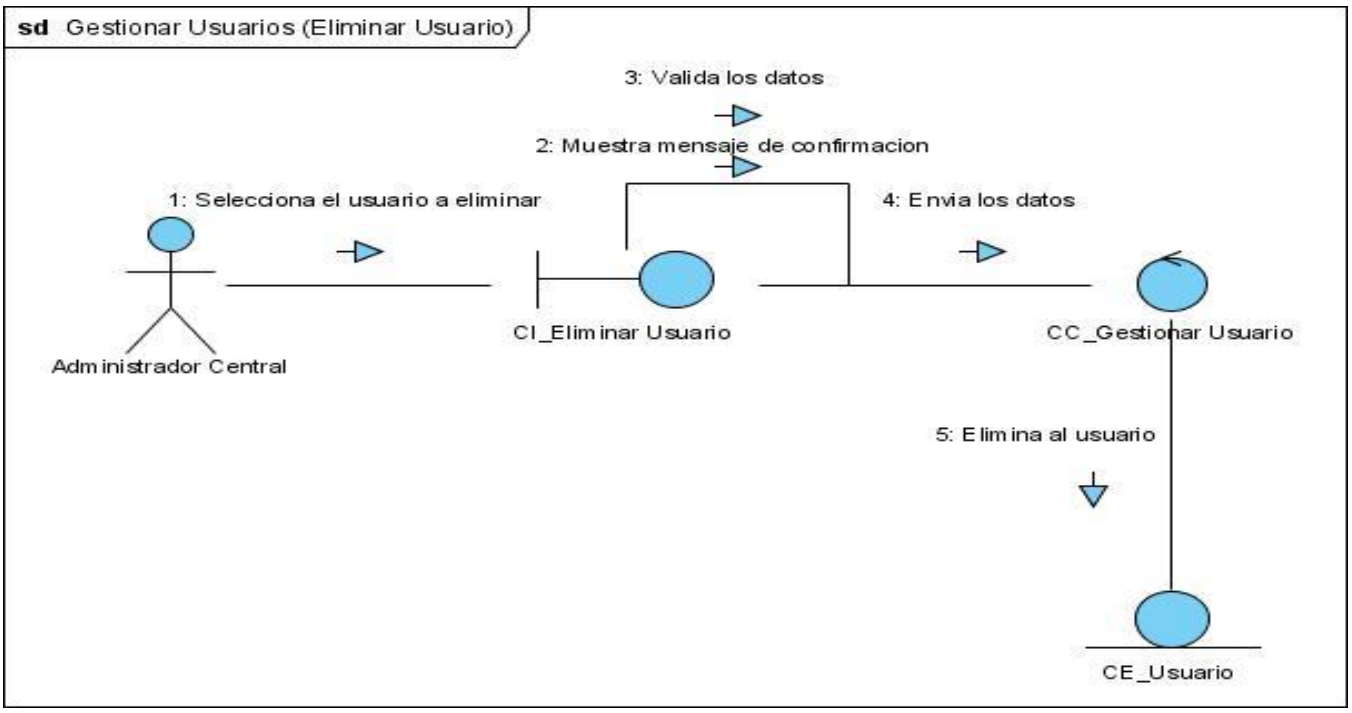
ANEXO 8. Diagrama de colaboración del análisis: Gestionar Usuarios. Escenario: Registrar Usuario



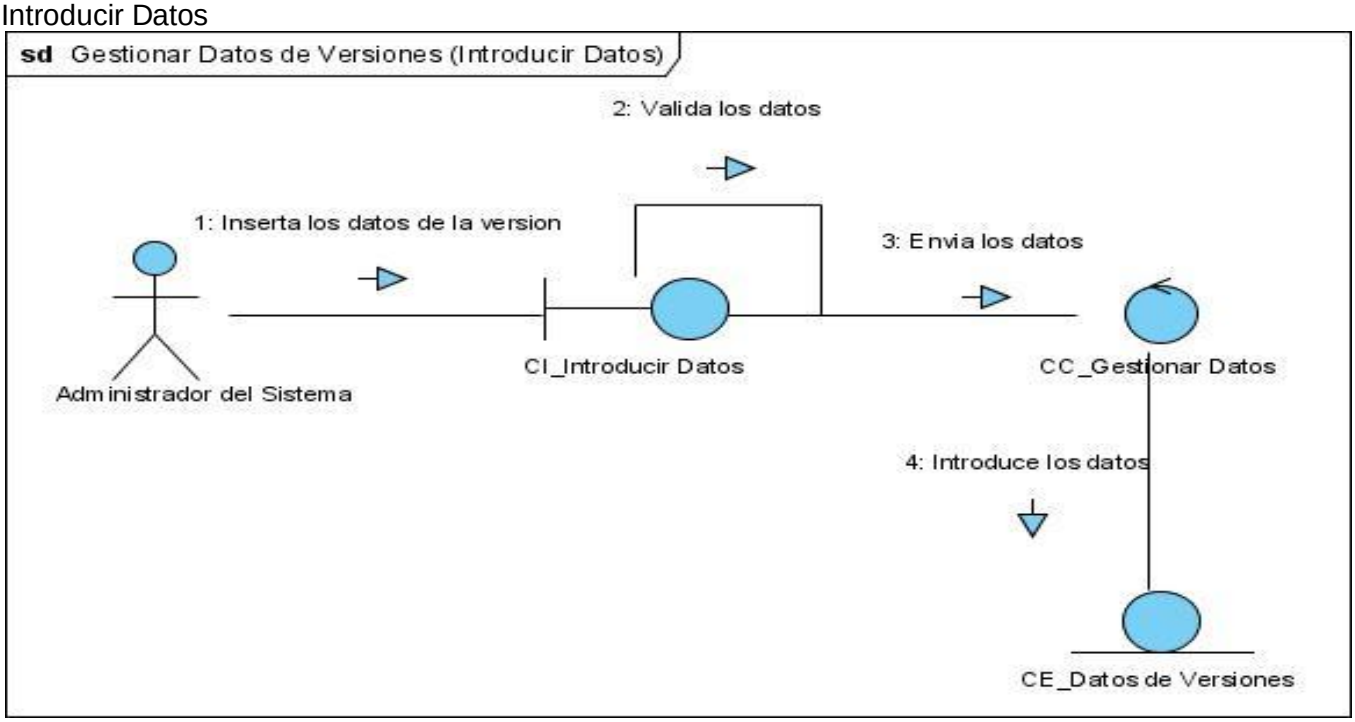
ANEXO 9. Diagrama de colaboración del análisis: Gestionar Usuarios. Escenario: Modificar Usuario



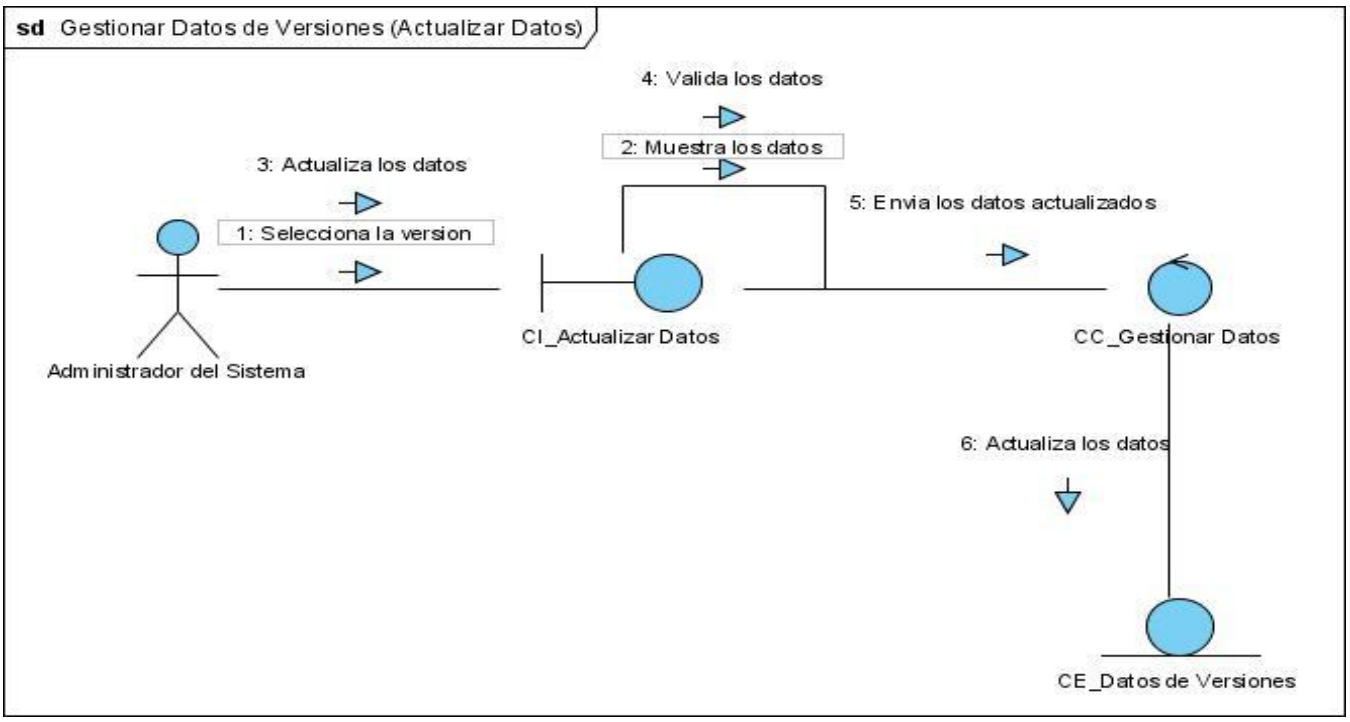
ANEXO 10. Diagrama de colaboración del análisis: Gestionar Usuarios. Escenario: Eliminar Usuario



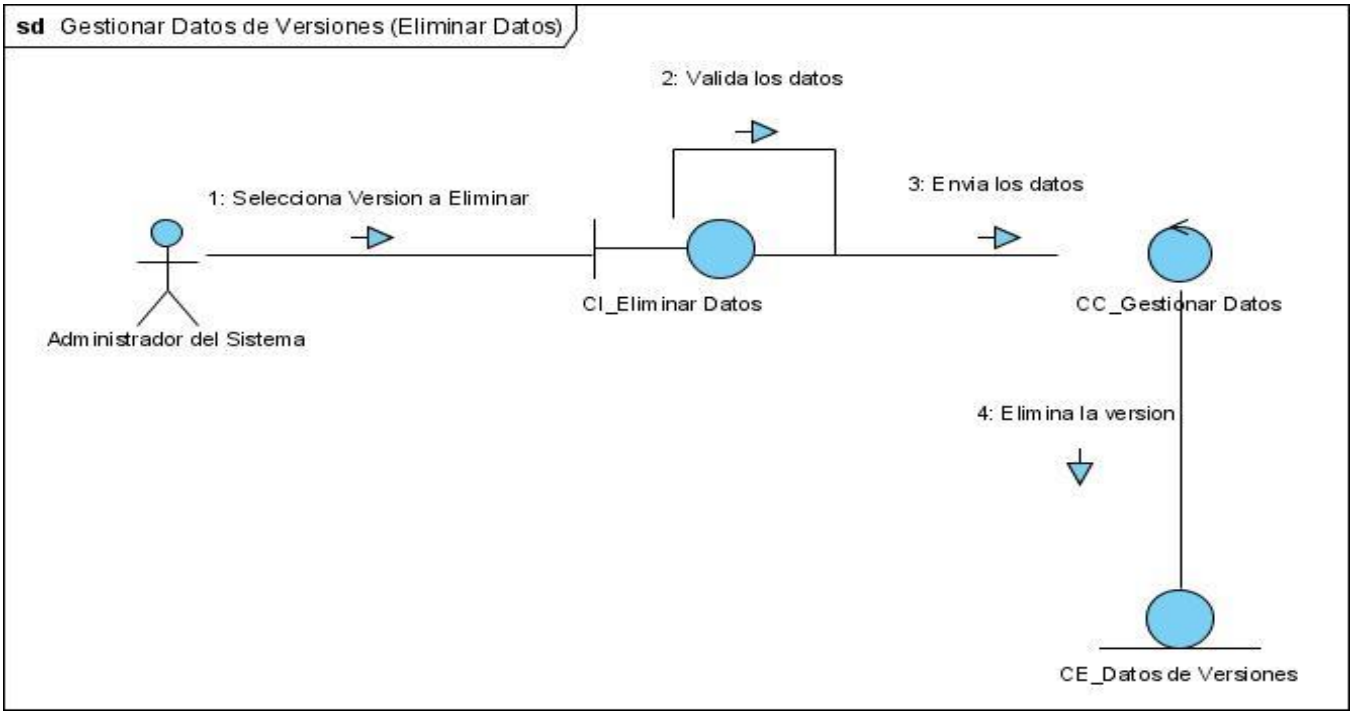
ANEXO 11. Diagrama de colaboración del análisis: Gestionar Datos de Versiones. Escenario: Introducir Datos



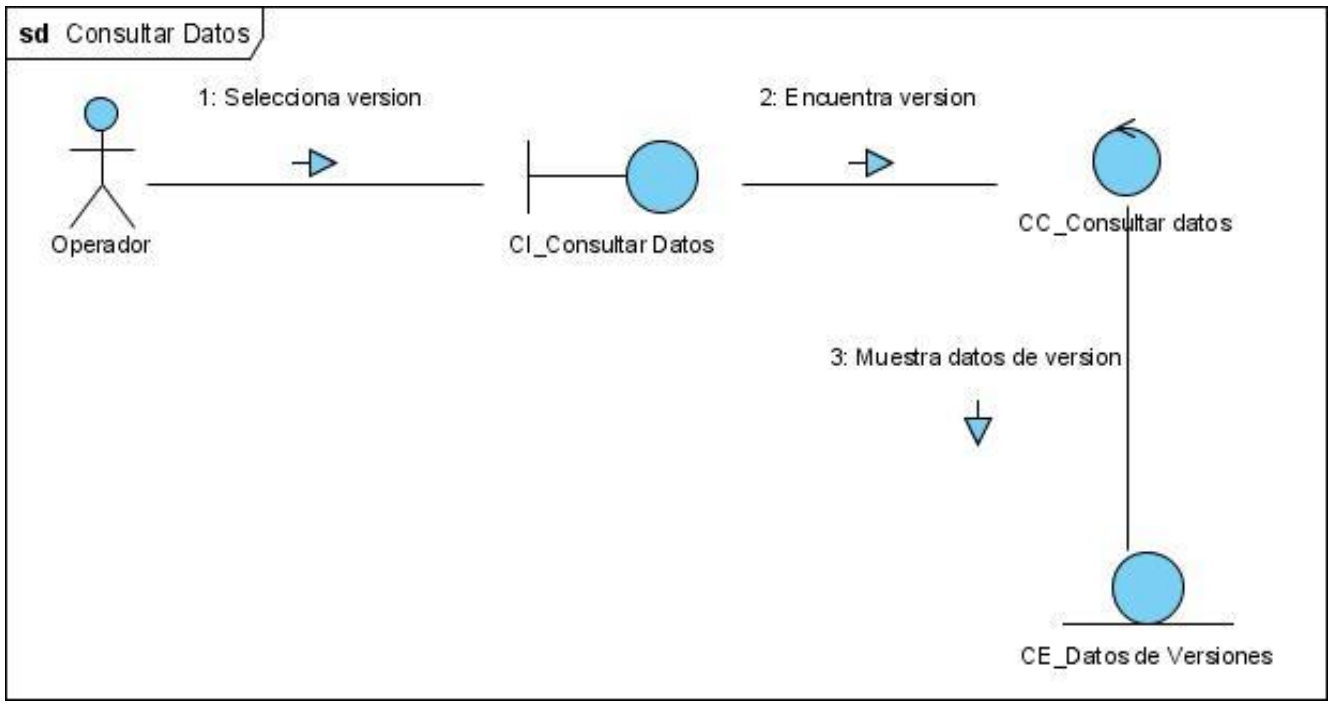
ANEXO 12. Diagrama de colaboración del análisis: Gestionar Datos de Versiones. Escenario: Actualizar Datos



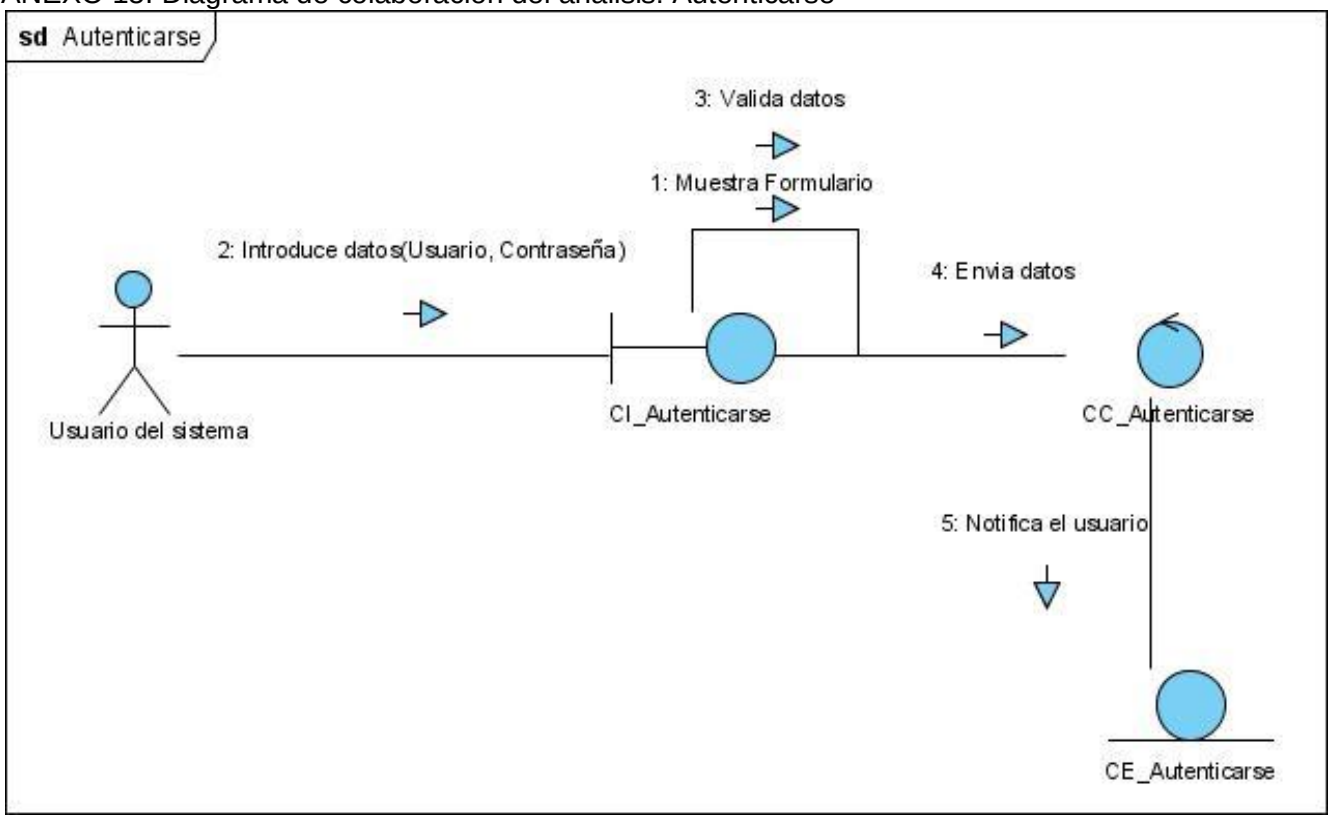
ANEXO 13. Diagrama de colaboración del análisis: Gestionar Datos de Versiones. Escenario: Eliminar Datos



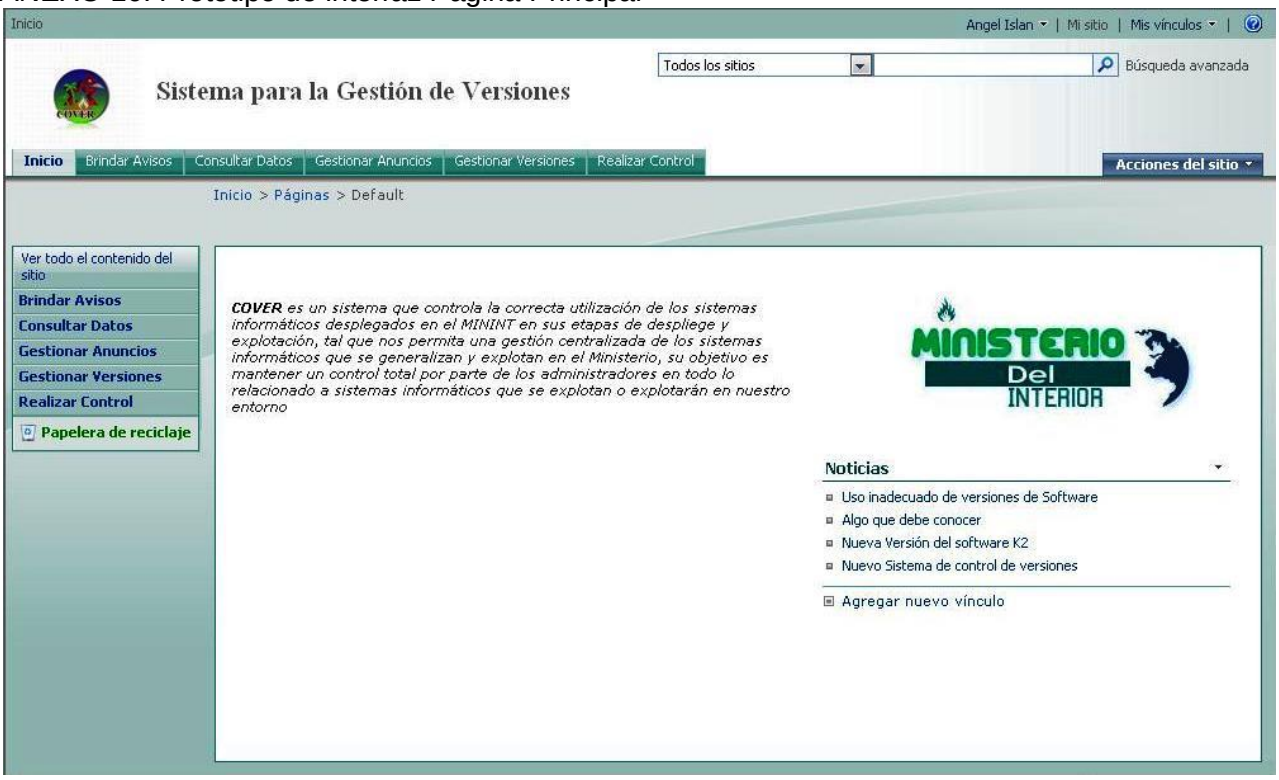
ANEXO 14. Diagrama de colaboración del análisis: Consultar Datos



ANEXO 15. Diagrama de colaboración del análisis: Autenticarse



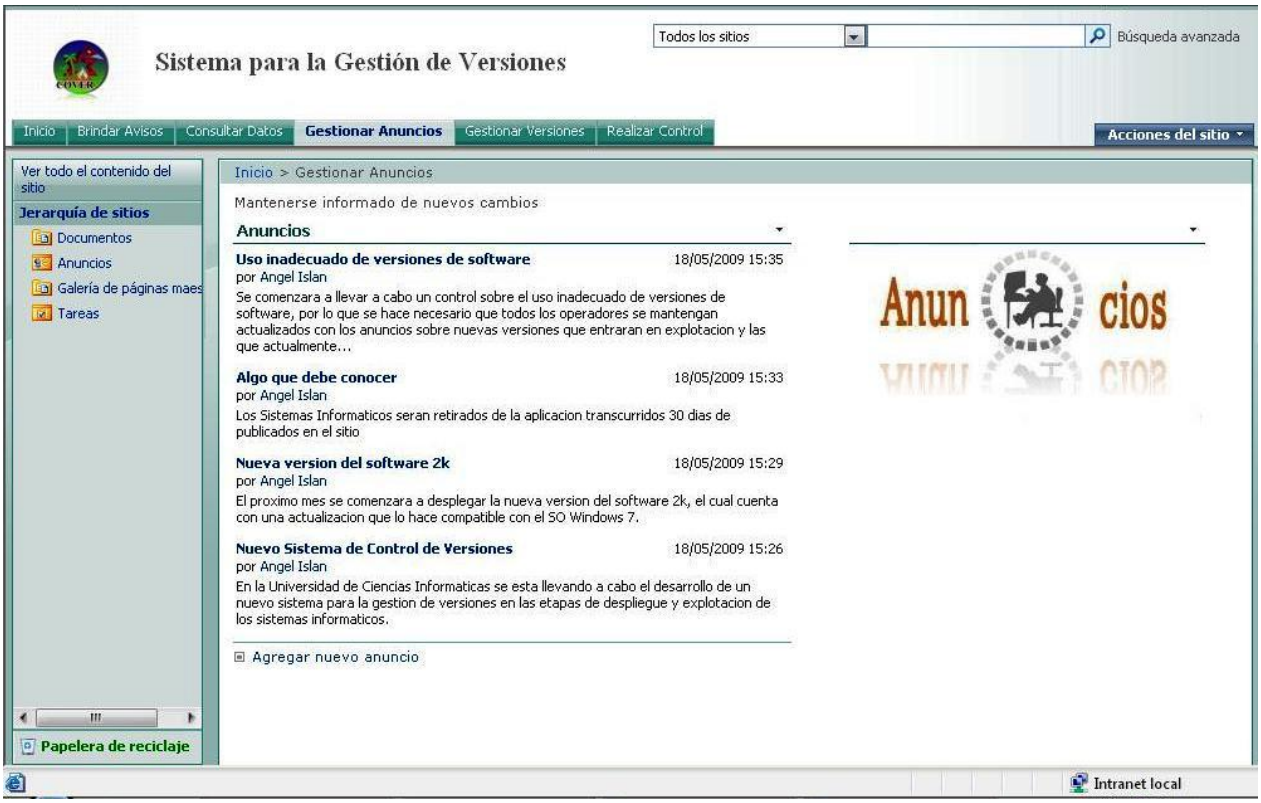
ANEXO 16. Prototipo de interfaz Página Principal



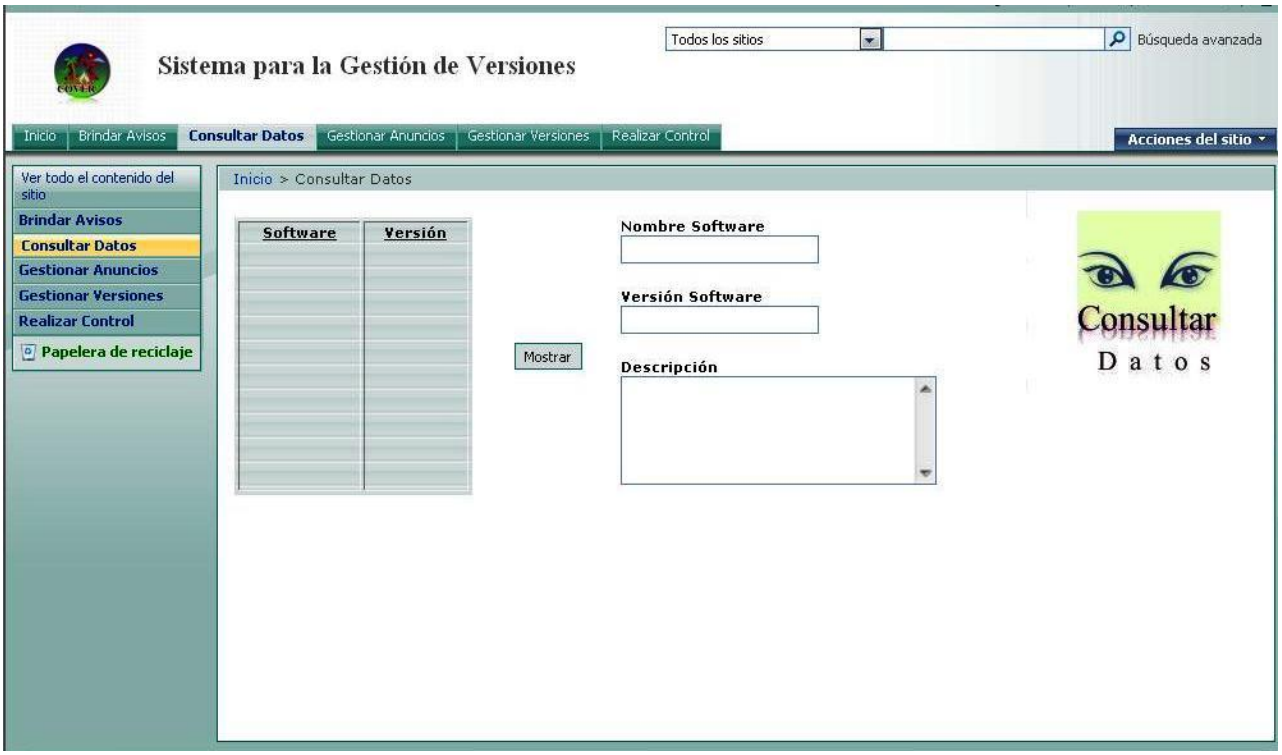
ANEXO 17. Prototipo de interfaz Realizar Control

ANEXO 18. Prototipo de interfaz Gestionar Versión

ANEXO 19. Prototipo de interfaz Gestionar Anuncios



ANEXO 20. Prototipo de interfaz Consultar Datos



ANEXO 21. Prototipo de interfaz Brindar Aviso