República de Cuba

Universidad de las Ciencias Informáticas

Facultad 2

Sistema de Gestión de Emergencias de Seguridad Ciudadana (SIGESC).

"Aplicación de Administración del SIGESC."

Trabajo de Diploma

Presentado para optar por el título de

Ingeniero Informático

Autores: Ráiner Cárdenas Alvarez.
Oscar Moncada Flores.

Tutores: Ing. Yordanis Tornés Medina.
Ing. Manfred Ramón Díaz Cabrera.

"Año 53 de la Revolución"
Ciudad de la Habana, Cuba.
24 de Junio de 2011.

Declaración de Autoría.

Por este medio declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas para que haga el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los __ días del mes de _____ del año_____.

Ráiner Cárdenas Alvarez: _____

Firma del Autor

Oscar Moncada Flores: _____

Firma del Autor

Ing. Yordanis Tornes Medina: _____

Firma del Tutor

Ing. Manfred Ramón Díaz Cabrera: _____

Firma del Tutor

Opinión del Tutor del Trabajo de Diploma.

Dedicatoria.

Oscar:

Dedico este trabajo de diploma a mamá y papá, que aunque se encuentren lejos siempre los llevo conmigo.

Ráiner:

A mi madre por ser la razón de mi existir, el tesoro más grande y preciado que tengo, porque nunca se ha dado por vencida conmigo, por ser mi inspiración, mi guía y mi apoyo durante toda mi vida, por apoyarme siempre en mis decisiones aunque no esté de acuerdo con ellas, por quererme tanto y por enseñarme con su ejemplo a enfrentar la vida.

A mi hermana Yaílyn, por asumir responsabilidades de adulto siendo joven y porque a pesar de que la mayor parte del tiempo discrepamos, te quiero mucho y ocupas un lugar muy especial en mi corazón.

Agradecimientos.

De los Autores:

A nuestros tutores, Tornés, Manfred y Baby por guiarnos durante la realización de este Trabajo de Diploma.

Al colectivo de trabajo del proyecto 171, a Daniel y Javier.

A nuestros compañeros tesistas del proyecto, Evelyn y Yadier por levantarnos el ánimo y por tantas noches en vela juntos en el laboratorio que han servido para lograr ver nuestros sueños hechos realidad.

A nuestros amigos de la Universidad y a todas las personas que de una manera u otra han aportado su granito de arena para la realización de esta tesis.

A todos muchas gracias.

Resumen.

Los Centros 171 en Venezuela son los entes coordinadores de los Órganos de Seguridad Ciudadana ante situaciones de emergencia y para automatizar los procesos que se realizan cada día en el Centro poseen un Sistema de Gestión de Emergencias de Seguridad Ciudadana (SIGESC). Dicho sistema informático, diseñado para trabajar sobre el sistema operativo Windows XP, desarrollado con la plataforma de .Net, está conformado por nueve aplicaciones de escritorio, una aplicación web, un servidor de sincronización y un servidor de base de datos.

El SIGESC es considerado un sistema distribuido pues está dividido en componentes de software (aplicaciones) que pueden estar localizados en diferentes ordenadores, comunicarse a través de la red y actuar coordinadamente con el objetivo de satisfacer los requisitos para los que fueron creados. Es por esta razón que se hace muy complejo el despliegue y el mantenimiento en operaciones del mismo, por lo cual se necesita que el SIGESC se convierta en un sistema distribuido administrado. Para ello, después de haber realizado un análisis de las tendencias actuales hacia los Sistemas Distribuidos Administrados se propone desarrollar una aplicación que permita supervisar y administrar centralmente el estado de la configuración y ejecución del SIGESC.

El presente documento propone la fundamentación acerca de las herramientas utilizadas y los elementos necesarios para la construcción de la Aplicación Administración del SIGESC la cual posee un conjunto de funcionalidades que mitigan la complejidad de administración y operación del SIGESC.

Índice

Introducción.

Sentirse seguro es un derecho del que debe disfrutar cada ser humano, sin importar el país en que viva, el credo o su raza; por esta razón garantizar la seguridad ciudadana es uno de los preceptos básicos que defiende la Constitución de la mayoría de las naciones del mundo.

La República Bolivariana de Venezuela para garantizar el orden público, la tranquilidad, la integridad, la libertad y la convivencia dentro de la sociedad, decidió implementar a nivel nacional el Servicio de Atención a Emergencias 171, materializado en la creación de los Centros 171. Estos a su vez, son los entes coordinadores de los Órganos de Seguridad Ciudadana ante situaciones de emergencia y para automatizar los procesos que se realizan cada día en el Centro 171 poseen un Sistema de Gestión de Emergencias de Seguridad Ciudadana (SIGESC). Dicho sistema informático fue diseñado para trabajar sobre el sistema operativo Windows XP, y desarrollado con la plataforma de .Net.

El sistema está conformado por nueve aplicaciones de escritorio, una aplicación web, un servidor de sincronización y un servidor de base de datos. Estas aplicaciones pueden coexistir en una misma estación de trabajo pero en la práctica no es lo que se recomienda para un Centro 171.

Al estar dividido en componentes de software (aplicaciones) que pueden estar localizados en diferentes ordenadores, comunicarse a través de la red y actuar coordinadamente con el objetivo de satisfacer los requisitos para el que fueron creados, el SIGESC es considerado un Sistema Distribuido.

Al poseer dichas características, se hace complejo el despliegue y el mantenimiento en operaciones del mismo, durante los cuales se ha logrado identificar la dificultad de que la preparación del personal encargado de mantener la solución muchas veces no es la requerida, además de no conocer la lógica del negocio en profundidad; todo esto influye directamente en que se puedan cometer diversos errores al configurar la solución, resultado de lo complejo que puede llegar a ser. Producto de esto los equipos de soporte de la solución no poseen la certeza de que el ambiente de configuración y ejecución del sistema es el correcto a la hora de corregir fallos o dar soporte técnico a los problemas que se puedan presentar.

Por tanto el **problema a resolver** es: *¿Cómo mitigar la complejidad de administración y operación del SIGESC como solución de software distribuido?*

Teniendo como **objeto de estudio** de esta investigación *la administración y operación de los sistemas distribuidos.*

El **objetivo general** para darle solución al problema es: *Desarrollar una aplicación que permita supervisar y administrar centralmente el estado de la configuración y ejecución del SIGESC.*

El **campo de acción** es *el estado de la configuración y ejecución del SIGESC*.

Para este fin se proponen varias **tareas a resolver** que se utilizarán de guía de apoyo para darle solución a los problemas identificados:

1. Realización de un estudio del estado del arte de las soluciones distribuidas administrables. Puntualizando en:
    - Sistemas Distribuidos.
    - Sistemas Distribuidos Administrados. Diseño para Operaciones.
    - Administración en la plataforma Microsoft Windows.
    - Tendencias Actuales.
    - Evaluación de la metodología de desarrollo de software y las herramientas necesarias para el desarrollo de la aplicación.
2. Análisis, identificación y descripción de las características de los sistemas distribuidos administrables.
3. Obtención de los requisitos funcionales y no funcionales del sistema.
4. Análisis de la arquitectura del SIGESC.
5. Evolución de los requisitos funcionales al diseño del futuro sistema.
6. Implementación de funcionalidades que permitan monitorear en tiempo real el estado de la configuración del SIGESC en un Centro 171 y verificar y corregir la disponibilidad de los Servidores de Sincronización, Servidor de Aplicaciones Web y Servidor de Base de Datos.

La **Idea a defender:** *Con la implementación del sistema que permitirá supervisar y administrar centralmente el estado de la configuración y ejecución del SIGESC, se mitigará la complejidad de administración y operación del mismo.*

El presente documento consta de los siguientes capítulos:

En el **Capítulo 1:** "Fundamentación Teórica", aborda los conceptos primordiales que se utilizan durante todo el trabajo, las tendencias actuales, metodologías de desarrollo, herramientas CASE y las tecnologías propuestas para el desarrollo de la Aplicación de Administración del SIGESC.

En el **Capítulo 2:** "Características del sistema", se puntualiza el problema, la situación problemática y los procesos que serán automatizados, además aborda los aspectos esenciales del negocio y los requisitos a tener presente para el desarrollo de la aplicación.

En el **Capítulo 3**: "Diseño del sistema", se define el diseño del sistema siguiendo la metodología RUP y se elaboran los diagramas de clases del análisis y del diseño.

En el **Capítulo 4**: "Implementación", se definen el diagrama de componentes y el de despliegue, se especifican pruebas realizadas a la aplicación para comprobar sus funcionalidades.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción.

En el presente capítulo se realiza un análisis sobre los aspectos de los sistemas distribuidos y las características de un sistema distribuido administrado. Se estudian además otros puntos teóricos que son necesarios para la concepción del sistema y se realiza una evaluación de la metodología de desarrollo, el lenguaje de modelado, el lenguaje de programación y la plataforma de desarrollo que se empleará en la construcción de la aplicación.

1.2 Sistemas Distribuidos.

"Un sistema distribuido es aquel en el que los componentes hardware o software, localizados en computadores unidos mediante red, comunican y coordinan sus acciones sólo mediante paso de mensajes.".[1]

El SIGESC, como sistema distribuido, es un sistema informático que se encuentra dividido en componentes de software que se localizan en diferentes ordenadores. Dichos componentes se comunican mediante la red y actúan coordinadamente con el objetivo de satisfacer las necesidades para los que se crearon.

En la Fig.1 se muestran las aplicaciones que conforman el SIGESC 171 en diferentes ordenadores. Las flechas indican el sentido del flujo de información que se establece mediante la red, evidenciándose lo anteriormente planteado.

[1] (Colouris, 2001 p. 2. Capítulo 1.)

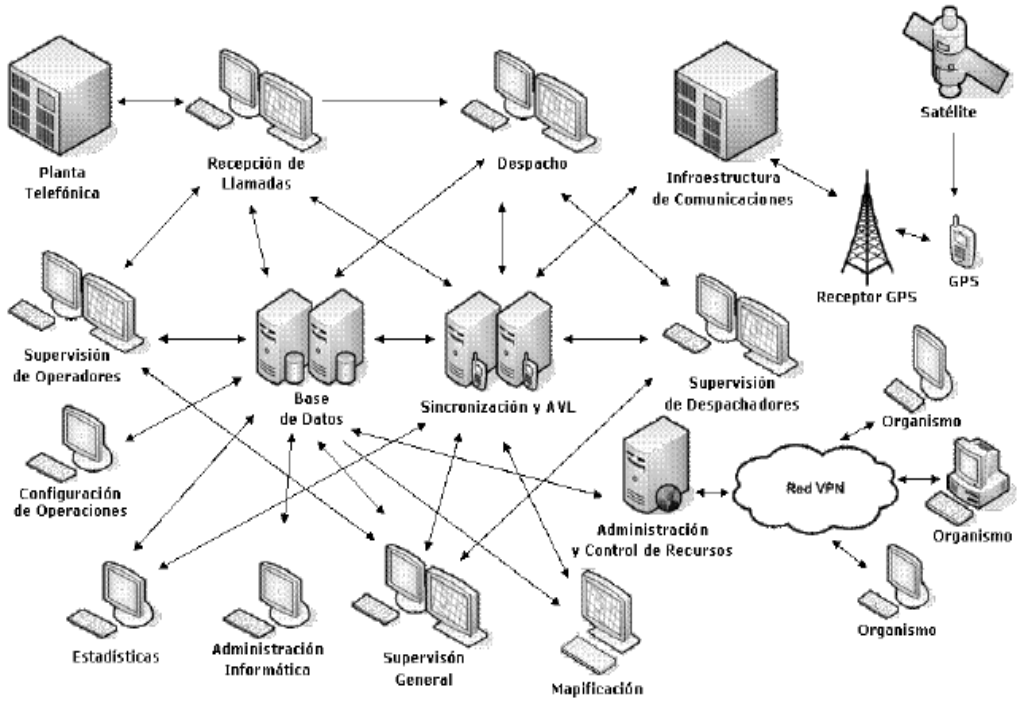

Fig.1.1 Distribución de los componentes del SIGESC.[2]

Los sistemas distribuidos, como el SIGESC, son muy complejos de administrar producto de la distribución computacional de los elementos que lo componen. En un Centro 171 donde existen una gran cantidad de ordenadores destinados para el trabajo con las aplicaciones del SIGESC, estos deben estar configurados correctamente para que dichas aplicaciones funcionen como es debido. El proceso de configuración en un solo ordenador consta de varios pasos descritos en el Manual de Instalación del SIGESC[3], además de las configuraciones que se deben realizar a aplicaciones como Recepción de Llamadas que se le puede asignar una vista de mapa o integrarla con la planta telefónica, si así lo requiere, mediante el Plugin correspondiente y así sucesivamente con cada una de las aplicaciones que necesiten determinada configuración para su correcto funcionamiento. Si se le suma que un Centro 171 pueden existir varias oficinas y cada una de ellas con una determinada cantidad de ordenadores se vuelve complicado las actividades de configuración, supervisión y mantenimiento para mantener el correcto funcionamiento y operación de todos los puntos del Centro.

[2] Tomado de (ALBET Ingeniería y Sistemas, 2009).

[3] Consultar (ALBET Ingeniería y Sistemas, 2007) para más información.

Es por esta razón que se necesita que el SIGESC se convierta en un sistema distribuido administrable mediante el desarrollo de una aplicación capaz de facilitar los procesos de administración del mismo. Por lo que, con el objetivo de identificar las experiencias positivas de empresas inmersas en el desarrollo de soluciones administrables y obtener una solución al problema existente, fue necesario realizar un estudio sobre las tendencias actuales de sistemas distribuidos administrados que existen y que son cimeras en este aspecto.

1.3 Tendencias Actuales hacia los Sistemas Distribuidos Administrados.

Dentro de las grandes compañías desarrolladoras de sistemas de software existe una tendencia que indica que las actividades de despliegue y operaciones necesitan mayor atención por el gran número de componentes de hardware y software heterogéneos que interactúan hoy en dichas soluciones.

Una de las inquietudes fundamentales en cuanto a gastos incurridos por conceptos de automatización de los procesos en las organizaciones, es la propiedad de las tecnologías. Total Cost of Ownership (TCO, Costo Total de propiedad) es un método que ayuda a determinar los costes directos e indirectos y los beneficios relacionados con la compra de equipos o programas informáticos pues pensar que al adquirir un software y explotarlo, el único gasto sería el de comprarlo, es una gran equivocación porque el despliegue y mantenimiento del mismo implican grandes inversiones; por lo que existe un esfuerzo a nivel general por optimizar y organizar las acciones que se realizan durante estas etapas.

Para minimizar estos costos existen dos tendencias principales, las soluciones centradas en el propietario (con el esfuerzo por estructurar eficientemente los procesos operacionales en los entornos corporativos) y las soluciones centradas en empresas desarrolladoras (con la intensión de facilitar la administración de sus soluciones en dichos entornos).

1.3.1 Soluciones centradas en el propietario.

Dentro de los propietarios de soluciones tecnológicas usar diferentes metodologías para estructurar los procesos implicados en el despliegue y operación de tecnologías en general, ha tenido gran aceptación ya que con la organización de las actividades a través de guías y marcos de trabajo se indican cómo evitar y resolver problemas que pueden

ocurrir en un entorno corporativo donde existe desplegada y en operación una solución de software u otro tipo de tecnología.

De estas metodologías, que son un conjunto de conceptos y buenas prácticas que facilitan la administración de los servicios de tecnologías de la información, destacan por su aceptación la Biblioteca de Infraestructura de Tecnologías de la Información, Los Objetivos de Control para la Información y la Tecnología relacionada y el Marco de Trabajo de Microsoft para Operaciones.

1.3.1.1 Biblioteca de Infraestructura de Tecnologías de la Información (ITIL).

Publicada por la Agencia Central de Cómputo y Telecomunicaciones del gobierno del Reino Unido, ITIL, es un conjunto de buenas prácticas y conceptos que se emplea en la gestión de servicios de tecnologías de la información, el desarrollo de dichas tecnologías y las operaciones relacionadas.

Ayuda a las organizaciones que la emplean a garantizar la calidad y eficiencia en las operaciones. Sus métodos están diseñados para servir de guía la cual abarca toda infraestructura, desarrollo y operaciones de TI.

Es uno de los métodos más aceptados para la gestión de servicios en el mundo porque proporciona un conjunto coherente de mejores prácticas, extraídas de los sectores público y privado a nivel internacional.[4]

1.3.1.2 Los Objetivos de Control para la Información y la Tecnología relacionada (COBIT).

Las empresas exitosas reconocen la necesidad de maximizar el valor de las inversiones relacionadas con las tecnologías de la información (TI) y que la necesidad de administrar dichas tecnologías es mayor ahora que nunca.

IT Governance Institute (ITGI, Instituto de Administración de las Tecnologías de la Información) e Information Systems Audit and Control Association (ISACA, Asociación para la Auditoría y Control de Sistemas de Información) proponen un conjunto de conceptos y buenas prácticas conocido como COBIT que brinda una estructura lógica y manejable a través de un dominio y un marco de procesos que:

[4] Consultar (APM Group, 2007-2011) para más información.

- Provee una dirección clara para garantizar que las inversiones en TI de apoyo al negocio.
- Es una manera eficaz de gestionar el cambio.
- Crea valor para el negocio en la alineación con los objetivos de la empresa.[5]

1.3.1.3 Marco de Trabajo de Microsoft para Operaciones (MOF).

Microsoft Corporation propone MOF como una serie de guías destinadas a ayudar a las TI a establecer y aplicar servicios fiables y rentables. "*Es una colección de recomendaciones, principios y modelos que proporciona una guía técnica completa para lograr confiabilidad, disponibilidad y capacidad de soporte técnico y de administración del sistema de producción crítico con productos y tecnologías de Microsoft. Aborda la naturaleza dinámica en constante evolución de los entornos informáticos distribuidos actuales.*"[6]

1.3.2 Soluciones centradas en las empresas desarrolladoras.

Las soluciones centradas en las empresas desarrolladoras se basan en dos tendencias la de equipamiento y la de las iniciativas de diseño y desarrollo de soluciones administrables. La primera surge cuando las empresas que se dedican al desarrollo de software perciben la necesidad de suministrar a sus soluciones herramientas capaces de supervisar el funcionamiento de los sistemas y componentes a través de la automatización de las tareas que antes requerían de un personal encargado y así facilitar la administración de sus soluciones. La segunda presenta dos iniciativas fundamentales: la Iniciativa de Sistemas Dinámicos (DSI) y la Iniciativa de Informática Autónoma, pertenecientes a Microsoft Corporation y a IBM respectivamente.

1.3.2.1 Tendencia de equipamiento.

Las empresas consagradas a la producción de sistemas, con características tan diversas como sistemas operativos, sistemas de gestión de base de datos, e incluso sistemas antivirus han incluido módulos o componentes y en algunos casos grupos de herramientas para la administración y control de dichas soluciones.

[5] Consultar (ISACA, 2011) para más información.

[6] (Microsoft Corporation, 2011 p. MOF)

Dentro de estas empresas se encuentra IBM con el CID para OS/2, Kaspersky Lab con Kaspersky Administration Kit, y Microsoft Corporation con Microsoft Deployment Toolkit 2010 y Microsoft System Center.

1.3.2.1.1 CID para OS/2.

En investigaciones realizadas por IBM Corporation sobre el proceso de instalación y configuración de SO, que en aquel entonces era mediante disquetes y CD-ROM, proyecta como resultado que era muy engorroso pues necesitaba interactuar directamente con personas encargadas a la tarea y que podían cometer errores en el proceso. Es por esta razón que decide equipar a su versión OS/2 (SO que vio la luz en 1992) de un conjunto de capacidades conocidas como CID. Con este método se perseguían diferentes objetivos destinados a mitigar las características no deseadas, que arrojó como resultado la investigación, mediante la **automatización** de los procesos de instalación, configuración y mantenimiento del sistema.

1.3.2.1.2 Kaspersky Administration Kit.

Desarrollada por Kaspersky Lab, Kaspersky Administration Kit es una herramienta que facilita el trabajo del administrador pues *"permite **organizar** y **monitorizar** de forma **centralizada** la protección de toda la red corporativa, consolidando diferentes capas de protección en un sistema integrado: desde puestos de trabajo que operan con Windows y Linux, pasando por servidores y dispositivos móviles, hasta servidores de correo y pasarelas de Internet".*[7]

1.3.2.1.3 Microsoft Deployment Toolkit (MDT) 2008.

Desarrollada por Microsoft Corporation para **facilitar** a los administradores de TI la tarea de **despliegue** de sistemas operativos en los distintos equipos de la empresa, de manera **centralizada**. Esta herramienta ayuda a instalar sistemas operativos como Windows XP, Windows Server 2003/2008, Windows Vista, Windows 7 y Windows Server 2008 R2. Facilita la **operación** en entornos distribuidos con tecnologías de Microsoft porque es posible llevar a cabo el despliegue desde cualquier estación de trabajo de la red y permite replicar ficheros y configuraciones.

[7] (Kaspersky Lab, 1997-2011 p. Kaspersky Administration Kit)

1.3.2.1.4 Microsoft System Center.

Mycrosoft System Center es una familia de productos que forman parte de la Iniciativa de Sistemas Dinámicos de Microsoft Corporation, abarcan una gran variedad de soluciones para la administración de las TI y ayudan a los profesionales de TI a **administrar** las tecnologías de la información en entornos de centros de datos, equipos cliente, y dispositivos. Específicamente están dirigidos a **facilitar** el trabajo de los administradores de las TI para administrar una red de sistemas.

Usando estas soluciones de gestión integrada y automatizada, las organizaciones de TI pueden proveer servicios más **productivos** para sus empresas.[8]

1.3.2.2 Tendencia Iniciativas de diseño y desarrollo de soluciones administrables.

Las iniciativas de diseño y desarrollo de soluciones administrables plantean métodos para ayudar en el diseño e implementación de sistemas distribuidos administrables.

La Iniciativa de Informática Autónoma consiste en la auto-administración de los sistemas informáticos y el intento de liberar a los administradores de sistemas de los detalles de la operación y mantenimiento de los mismos y proveer a los usuarios de sistemas que se ejecuten al máximo de sus capacidades todo el tiempo. Pero, aunque esta es la meta para reducir la complejidad de los ambientes distribuidos es una tarea que todavía requiere de varios años de trabajo e investigación.[9]

La Iniciativa de Sistemas Dinámicos es un esfuerzo de Microsoft Corporation por mejorar la administración de las aplicaciones propias y los sistemas distribuidos de la plataforma Microsoft Windows mediante un conjunto de soluciones que simplifican y automatizan el modo en que las empresas diseñan, implementan y utilizan sistemas distribuidos. Para ello propone un diseño de operaciones basado en el uso de tres modelos, (el modelo de salud, el de tareas y el de estado), teniendo en consideración además las características necesarias que debe cumplir cada uno de estos sistemas para que sean considerados administrables.

[8] (Microsoft Corporation, 2010 p. System Center)

[9] Consultar (IBM Corporation, 2001) para más información.

1.4 Diseño para Operaciones. Sistemas Distribuidos Administrados.

1.4.1 Diseño para Operaciones.

Operaciones: *"es el trabajo requerido para mantener y administrar una aplicación. Casi siempre comienza poco tiempo después de haber desplegado, entonces es ejecutado constantemente hasta que la aplicación deja de brindar servicios. Operaciones plantea además, que cada una de las aplicaciones desarrolladas debe ser monitoreada y administrada."*[10]

Microsoft Corporation ha desarrollado tres modelos de base para el diseño de operaciones de un servicio o aplicación: el modelo de Salud, el modelo de Tareas, y el modelo de Estado.

1. **El Modelo de Salud** permite orientar qué tipo de información debe ser proporcionada y cómo el sistema o el administrador debe responder ante la misma, cuando la información del sistema no ayuda al administrador a saber lo que está pasando.
2. **El Modelo de Tareas** enumera las actividades que se realizan en la administración del sistema. Estas pueden ser tareas de mantenimiento realizadas en forma rutinaria, tales como copia de seguridad, las tareas basadas en eventos, o tareas de diagnóstico que se realizan para corregir las fallas del sistema. La definición de estas guías de tareas del desarrollo de herramientas de administración se convierten en la base para la automatización de sistemas distribuidos administrados.
3. **El Modelo de Estado** cataloga el estado y la configuración de una aplicación definiendo el alcance y el tipo para las mismas.

Los Modelos están diseñados para proporcionar un mecanismo normativo e iterativo y así asegurar que la administración está integrada en todos los servicios y aplicaciones y que se alinea con las necesidades del administrador que ejecuta el servicio.

Utilizar el Modelo de Salud constituye un apoyo para determinar el estado de ejecución del SIGESC, en tiempo real, o sea, si está funcionando total o parcialmente o si no está funcionando producto de que uno de sus subsistemas esté fuera de servicio. Para lograrlo, este modelo propone tres Estados de señalización:

[10] (David Chappell & Associates, 2008 p. 2)

- **Verde**: Ejecución normal, Funcionando correctamente.
- **Amarillo**: Funcionando parcialmente. Se pueden mejorar algunas funciones, pero algún problema es detectado. Se pueden haber detectado problemas de seguridad, integridad, rendimiento y dependencia pero como ninguna funcionalidad crítica está afectada, se puede seguir trabajando en el sistema.
- **Rojo**: No puede proveer ninguna funcionalidad porque el sistema colapsa debido a que existe uno o varios subsistemas que no están funcionando y que son críticos como por ejemplo el servidor de base de datos de una aplicación si dejara de funcionar, el sistema entero colapsaría.

El Modelo de Tarea es el que guía durante el proceso de identificación y automatización de los requisitos del sistema a desarrollar. Tiene como objetivos: "*Identificar el conjunto de tareas asociadas con el funcionamiento de una aplicación o servicio*. *Definir el alcance de las tareas*: ¿quién *las llevará a cabo*, cuándo *y cómo? Identificar aquellas tareas posibles a automatizar.*"[11]

El Modelo de Estado está relacionado con la gestión de configuración de una aplicación, que bien puede ser la configuración del Registro, la configuración de interfaz gráfica de usuario, las directivas de grupo, o los parámetros de instalación. Este modelo presenta nuevas iniciativas para exponer y clasificar las opciones de configuración, que les permita ser visibles, accesibles y portátiles para los administradores, y así crear herramientas de gestión que soporten la migración y la copia de seguridad. [12]

1.4.2 Sistemas Distribuidos Administrados.

En el documento "Designing Manageable Applications"[13], Microsoft Corporation plantea que una aplicación es administrable cuando:

1. Es compatible con los ambientes de despliegue de destino.
2. Es configurable de manera dinámica en tiempo de ejecución.
3. Interacciona correctamente con herramientas de operación y es coherente con los procesos de operación de sistemas.
4. Provee visibilidad al respecto del estado de funcionamiento del sistema.

[11] (Microsoft Corporation, 2003 p. 15)

[12] Consultar (Microsoft Corporation, 2003) para más información.

[13] (Microsoft Corporation, 2008)

Una aplicación es compatible con el ambiente de despliegue destino si al concebirla se tienen en cuenta las características de hardware hacia dónde va a ser desplegada por lo que se debe realizar un estudio del mercado, de las características de hardware de los ordenadores promedio que se venden en ese momento y se desarrolla el software teniendo presente dichas características.

Es configurable dinámicamente en tiempo de ejecución cuando se pueden realizar configuraciones sin tener necesidad de detener su funcionamiento.

Interacciona correctamente con herramientas de operación cuando durante la implementación se tiene presente que la aplicación debe exponer instrumentación para permitir operaciones administrativas realizadas por esas herramientas.

Provee visibilidad acerca del estado de salud que presenta el sistema, el cual varía acorde con la habilidad de realización de las operaciones cuando a través del establecimiento de parámetros que representan el correcto funcionamiento del sistema y realizando mediciones, se determina si indicadores del sistema se encuentran dentro de esos parámetros, obteniendo así el estado de funcionamiento global o parcial del sistema.

Para lograr que un sistema distribuido posea las características anteriores, las mismas hay que tenerlas en cuenta desde etapas iniciales del ciclo de desarrollo. Durante el desarrollo de las aplicaciones del SIGESC no se tuvo presente que fuera compatible con los ambiente de despliegue de destino, sino que una vez desarrollado el sistema se le hicieron pruebas para ver que arquitectura de hardware era la indicada y entonces se le informó al cliente de las especificaciones necesarias para el despliegue del mismo. El SIGESC no es configurable en tiempo de ejecución porque para poder realizar alguna configuración a una de sus aplicaciones estas deben detener su funcionamiento. Tampoco permite su administración a través de herramientas destinadas a esa actividad debido a la carencia de instrumentación expuesta porque no se tuvo presente durante la etapa de implementación de las aplicaciones lo cual influye en que no sea compatible con herramientas de operaciones como Microsoft System Center y que no pueda proveer visibilidad del estado de funcionamiento del sistema.

El SIGESC se encuentra en fase de despliegue por lo que se hace muy difícil implementar las características sin modificar la arquitectura existente. Un rediseño de esta última

conllevaría a incluir nuevos elementos o realizar cambios sobre los actuales lo que puede provocar una inestabilidad en el SIGESC, materializándose en errores inesperados o resultados no deseados por lo que habría que dedicar tiempo para garantizar el adecuado funcionamiento del sistema ante las transformaciones.

Los Modelos propuestos para el Diseño de Operaciones se deben tener presentes también desde el inicio del ciclo de vida del software y es por esta razón que los modelos de Tarea y Estado no se pueden utilizar en la conformación de la aplicación que se debe desarrollar, pues afectarían la automatización y configuración del SIGESC. Pero el modelo de Salud permitirá diagnosticar el estado de funcionamiento y configuración del SIGESC proveyendo visibilidad del estado de funcionamiento del sistema y convirtiendo al SIGESC en un sistema distribuido con características limitadas de administración. Por lo que se concluye que la Iniciativa de diseño y desarrollo de soluciones administrables, específicamente la Iniciativa de Sistemas Dinámicos, de las Soluciones Centradas en las Empresas Desarrolladoras, es la tendencia que se adecua al software que posteriormente se diseñará y desarrollará pues se va a equipar al SIGESC con una herramienta capaz de supervisar el funcionamiento de los subsistemas y sus diferentes componentes.

El SIGESC está desarrollado para ejecutarse sobre Windows XP SP2 por lo que es prácticamente posible materializar gran parte de la administración de este sistema a través de los beneficios que expone la utilización del Instrumental de Administración de Windows sobre sistemas, aplicaciones y redes informáticas basadas en Microsoft Windows.

1.5 Administración en la plataforma Microsoft Windows. Instrumental de Administración de Windows (WMI).

Windows Management Instrumentation (WMI, Instrumental de Administración de Windows) es una infraestructura de administración estándar de datos y operaciones sobre sistemas operativos basados en Windows incluida como parte de los sistemas operativos de Microsoft a partir de Microsoft Windows 2000. Fue diseñado con el propósito de facilitar a administradores de sistemas, la administración de grandes y complejos sistemas, aplicaciones y redes informáticas.

A través de una simple y consistente interfaz orientada a objetos esta infraestructura permite monitorear y controlar componentes de sistemas de forma local o remota.

Proporciona además un amplio conjunto de servicios de gestión de sistema integrada en los sistemas operativos Microsoft Windows.

El SIGESC está diseñado casi totalmente para su ejecución sobre la plataforma Microsoft Windows, como consecuencia WMI le proporcionaría amplias características para la gestión de operaciones como objeto de soporte de producto.

El uso de WMI, para la administración de la aplicación a desarrollar, permite realizar tareas como:

- Monitorizar el estado de ejecución de servicios de Windows.
- Llevar a cabo operaciones de gestión local o remota.

La materialización de estas tareas sobre el SIGESC proveen a éste de una constante monitorización de información relevante relacionada con la ejecución de servicios específicos, así como la gestión local y remota de la plataforma Windows sobre la cual se ejecutan la gran mayoría de las aplicaciones del SIGESC.

Para realizar la aplicación, que proveerá al SIGESC 171 de características limitadas de administración, es necesario realizar un estudio de la metodología de desarrollo, el lenguaje de modelado, el lenguaje de programación y las herramientas que serán utilizadas en el desarrollo de la misma.

1.6 Metodología de desarrollo de software. Proceso Unificado de Desarrollo.

Rational Unified Process (RUP, Proceso Unificado de Desarrollo) es una metodología tradicional de software que no establece pasos estrictamente obligatorios sino un conjunto de metodologías que se adaptan a las necesidades de los desarrolladores.

RUP está fundado en 6 primicias claves:

- *Adaptar el proceso* a las necesidades del cliente para establecer una buena comunicación con él.
- *Equilibrar prioridades* porque es importante llegar a un punto de acuerdo entre las partes, el equipo de desarrollo y los clientes, cuando los criterios son diferentes en cuanto a los requisitos y así evitar discrepancias futuras.
- *Demostrar valor iterativamente* pues el proyecto debe realizarse por iteraciones, donde cada iteración es examinada por los inversores para medir la calidad y estabilidad del producto.

- *La Colaboración entre equipos* es primordial ya que se hace necesario que fluya la comunicación en todos los sentidos para que el trabajo no se vea afectado en un equipo donde trabajan de forma conjunta múltiples personas para lograr el desarrollo del software.
- Un *alto nivel de abstracción* permite discusiones sobre diversos niveles y soluciones arquitectónicas que pueden estar acompañadas por las representaciones visuales de la arquitectura mediante un lenguaje de modelado como UML.
- *El control de la calidad* no se debe realizar solo al concluir una iteración sino en cada uno de los aspectos de la realización del software, por lo cual, asegurar la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

Debido a la poca comunicación que existe con el cliente, producto de que este se encuentra en otro país, se debe mantener un control mayor del proceso de desarrollo y además generar una gran cantidad de documentación pues en caso de que los desarrolladores de la aplicación no se encuentren disponibles, el personal que los sustituya en un futuro debe tener documentación suficiente para entender, mejorar y dar soporte a la aplicación en cuestión.

Al integrar RUP con el Lenguaje Unificado de Modelado (UML) constituye la metodología estándar más utilizada para el análisis, documentación e implementación de sistemas orientados a objetos.

1.7 Lenguaje de Modelado. Lenguaje Unificado de Modelado.

Unified Modeling Language (UML, Lenguaje Unificado de Modelado) es el lenguaje más conocido y utilizado en el modelado de sistemas de software. UML al ser un lenguaje gráfico, nos permite visualizar, especificar, construir y documentar el sistema que nos proponemos desarrollar. Ofrece un estándar para detallar un plano del sistema (modelo), incluyendo características conceptuales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguaje de programación, esquemas de bases de datos y componentes reutilizables.

Se puede emplear en el desarrollo de un software como soporte a una metodología de desarrollo de un software como RUP, pero no se especifica que metodología o proceso se debe usar.

1.7.2 Notación para el Modelado de negocio.

Para realizar el modelado de negocio se pueden emplear técnicas y notaciones que ayudan a conocer los objetivos del negocio y plasmarlos en un modelo.

Integrated Definition Methods (IDEF) es una familia de lenguajes de modelado con una amplia gama de usos como el modelado funcional, simulación, análisis orientado a objetos hasta el diseño y adquisición de conocimientos.

De la familia IDEF el más conocido y utilizado es IDEF0 ya que *"es una técnica de modelado común para el análisis, desarrollo, reingeniería, y la integración de los sistemas de información, procesos de negocio, para el análisis de ingeniería de software. IDEF0 se utiliza para mostrar el flujo de datos, sistema de control, y el flujo funcional de los procesos de ciclo de vida."*[14]

Utilizando IDEF0 se puede modelar los procesos del negocio desde el nivel de más alto de abstracción hasta el nivel de detalle que se desee, producto de la estructura jerárquica que posee para representar un proceso. Facilita la comunicación y captura de información y permite analizar, documentar y mejorar los procesos.

La notación IDEF se compone de actividades, entradas, salidas, mecanismos de control y sujetos.

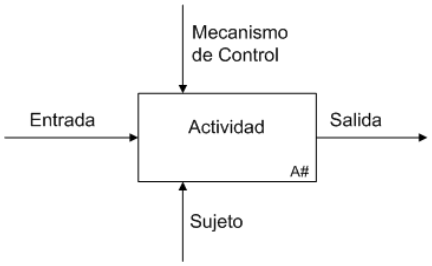

Fig.1.2 Formato Caja de IDEF0. [15]

Cada actividad se representa con un rectángulo cerrado y en la esquina inferior derecha se pone el número de la actividad para seguir un orden. Las actividades deben tener de obligatoriamente Entradas y Salidas y de manera opcional Mecanismos de Control y Sujetos que realizan la actividad.

[14] (Department of Defense. System Management College., 2001 p. 51)

[15] Adaptado de (Deparment of Defense. System Management College., 2001 p. 51).

La entrada se representa como una conexión que entra a la actividad por la izquierda y es lo inicia la actividad. La salida se representa como una conexión que sale de la actividad por la derecha y es el resultado que se obtiene después de haber realizado la actividad.

Un mecanismo de control es una conexión que entra a la actividad por la parte superior y es el que indica las regulaciones que determinan si una actividad se realiza o no.

Un sujeto se representa por una conexión que entra a la actividad por la parte inferior.

Dentro de las ventajas que ofrece IDEF se encuentra que es flexible ante cambios, explica los procesos más complejos de forma fácil, permite descomponer una actividad como un proceso a su vez e identifica posibles procesos redundantes o defectuosos y así organizar el negocio y luego informatizarlo.

Es de vital importancia seleccionar también la plataforma desarrollo para la realización de la solución propuesta pues está incluye todas las herramientas necesarias para automatizar las funcionalidades requeridas.

1.8 Plataformas de desarrollo.

Una plataforma de desarrollo es el entorno de software en el cual se desenvuelve la programación de una aplicación. Ejemplos típicos incluyen: arquitectura de hardware, sistema operativo (SO), lenguajes de programación y sus librerías de tiempo de ejecución.

1.8.1 Plataforma .Net v1.1.

La plataforma .NET es el resultado de un conjunto de tecnologías desarrolladas por Microsoft Corporation con el propósito de crear una plataforma sencilla y potente. Es una capa de software que se coloca entre el SO y el programador y que abstrae los detalles internos del SO. Dicha plataforma es multilenguaje, 100% orientada a objetos y presenta un Modelo de Programación único para todo tipo de aplicaciones y dispositivos de hardware. Se integra fácilmente con aplicaciones existentes desarrolladas en plataformas Microsoft y con aplicaciones desarrolladas en otras plataformas.

.Net está compuesta por un Entorno de Ejecución (Runtime), Bibliotecas de Funcionalidad (Class Library), Lenguajes de Programación, Compiladores, Herramientas de Desarrollo (IDE & Tools).

Microsoft Corporation define la plataforma .NET como un entorno para la construcción, desarrollo y ejecución de aplicaciones, con comunicación segura y la capacidad de modelar una amplia gama de procesos de negocio. Consiste en tres partes fundamentales: el Common Language Runtime (entorno de ejecución), las Framework Classes (clases de la plataforma) y Frameworks de desarrollo y tecnologías. [16]

La plataforma de .Net, también le brinda al desarrollador:

- Facilidad de extensión pues propone una gran cantidad de clases que pueden ser reutilizables.
- Usabilidad ya que no es necesario tener conocimientos de avanzada para hacer uso de ella y ofrece la documentación necesaria para aprender.
- Integración de manera sencilla con tecnologías como WMI.
- Soporte de ADO.NET[17] para conectarse a Oracle.
- Compatibilidad con sistemas operativos como Windows XP SP2.
- Un framework que ofrece facilidades para generar, implementar y probar aplicaciones de escritorio. [18]

1.8.2 .NET Framework SDK v1.1.

.NET Framework SDK[19], es un grupo de herramientas de desarrollo de software publicado por Microsoft Corporation para la creación de aplicaciones con la plataforma .NET, ya sean servicios Web o aplicaciones de escritorio tradicionales.

"Consiste en un conjunto de herramientas, ejemplos de código, documentación, compiladores, encabezados y bibliotecas que pueden usar los desarrolladores para crear aplicaciones que se ejecuten en los sistemas operativos con modelos de programación nativos (Win32) o administrados (.NET Framework)". [20]

Una de las ventajas de utilizar .NET es que se puede integrar con WMI lo cual constituye una gran ventaja para el desarrollador pues brinda facilidades al implementar tareas de

[16] Adaptado de (Microsoft Corporation, 2009 p. Microsoft .NET)

[17] Estándar de acceso a Bases de Datos.

[18] (Microsoft Corporation;, 2011 p. Microsoft .NET Framework SDK versión 1.1).

[19] Siglas en Inglés: Software Development Kit.

[20] (Microsoft Corporation;, 2011 p. Kit de desarrollo de software de Microsoft Windows)

administración, ya sean de aplicaciones, redes informáticas y así poder monitorear y controlar componentes de sistemas de forma local o remota, etc.

1.8.3 Instrumental de Administración de Windows y la Plataforma .NET.

WMI en .NET Framework se basa en la tecnología WMI original y permite el mismo desarrollo de aplicaciones y proveedores, además de las ventajas que ofrece la programación en .NET Framework.

El espacio de nombres de .NET Framework **System.Management**, permite desarrollar aplicaciones, en cualquier lenguaje escrito para la plataforma de .NET, que pueden obtener datos empresariales y automatizar tareas administrativas mediante WMI. También favorece el desarrollo de aplicaciones que proporcionen datos a clases WMI con el espacio de nombres **System.Management.Instrumentation**.

El objetivo del espacio de nombres **System.Management.Instrumentation** es agrupar las clases y componentes que permiten reducir al mínimo el trabajo necesario para crear una aplicación de gestión. El espacio de nombres también hace que sea fácil de exponer hechos y datos y las relaciones entre el uso de objetos de administración.

1.7.3.1 Arquitectura de WMI .NET.

Las clases del espacio de nombres **System.Management** interactúan con el administrador de objetos WMI para enviar y recibir datos en el repositorio WMI, así como objetos dinámicos suministrados por proveedores.

Elementos de arquitectura. Niveles de WMI.

La arquitectura de WMI consta de los siguientes tres niveles:

- **Los componentes de software cliente** realizan operaciones mediante WMI, como leer detalles de administración, configurar sistemas y realizar suscripciones a eventos.
- **El administrador de objetos** es un intermediario entre los proveedores y los clientes WMI que proporciona ciertos servicios esenciales, entre otros, la publicación y la suscripción estándar de eventos, el filtrado de eventos y el motor de consultas.
- **Los componentes de software de proveedor** capturan y devuelven datos activos a las aplicaciones cliente, procesan llamadas a métodos procedentes de

los clientes y vinculan al cliente con los componentes del sistema que se están administrando.

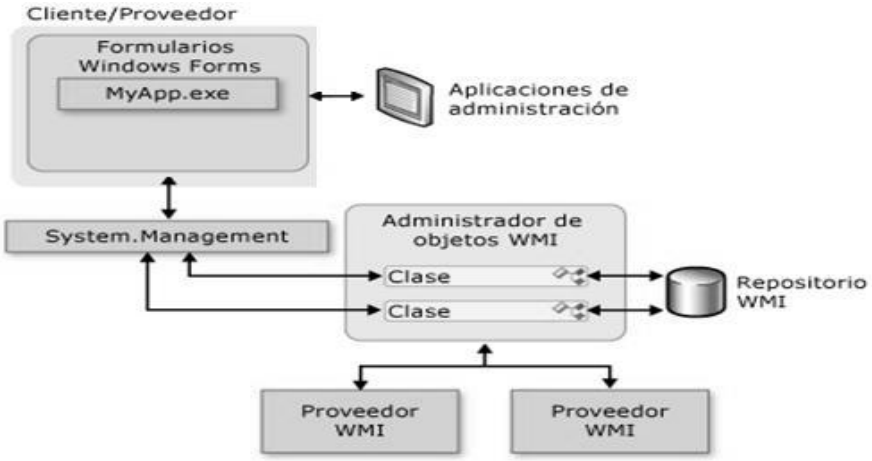

Fig.1.3 Niveles de WMI para aplicaciones de Escritorio.[21]

Ventajas de WMI en .NET Framework.

La escritura de una aplicación cliente o de un proveedor con WMI en .NET Framework proporciona diversas ventajas con respecto a WMI original, sobre todo para los desarrolladores que utilizan C# en lugar de C++, pero para el desarrollo de la aplicación en cuestión, WMI en .NET Framework ofrece como ventaja el acceso a todos los datos WMI. Las aplicaciones cliente tienen el mismo tipo de acceso a datos WMI y pueden realizar las mismas operaciones con éstos que en WMI original.

Limitaciones de WMI en .NET Framework.

Si bien el desarrollo de proveedores y aplicaciones de administración empresarial es más rápido con WMI en .NET Framework, existen algunas limitaciones que afectan principalmente a los proveedores de objetos instrumentados con código administrado. Pero este tipo de limitaciones no afecta el desarrollo de la aplicación Administración del SIGESC porque la utilización de WMI sería para ejecutar procesos remotos y realizar consultas mediante WMI Query Language (WQL, Lenguaje de Consulta WMI).

1.7.3.2 Lenguaje de Consulta WMI (WQL).

Es un subconjunto del estándar de American National Standards Institute Structured Query Language (ANSI SQL) con ligeros cambios semánticos con el apoyo de WMI.

[21] Adaptado de (Microsoft Corporation, 2006 p. Arquitectura de WMI .NET).

Soporta las consultas de datos, de esquemas y de eventos. Pero la que se utilizará es la consulta de datos para recuperar instancias de clases y las asociaciones de datos. Es el tipo más común de consulta en las secuencias de comandos de WMI y aplicaciones.

El Modelo de Salud de la Iniciativa de Sistemas Dinámicos, como se plantea en el acápite 1.4.1 Diseño para Operaciones., permitirá determinar el estado de ejecución del SIGESC lo que se hace posible mediante las consultas de WMI realizadas a las diferentes estaciones a monitorear.

Las clases de WMI que se consultarán para obtener los datos necesarios de un ámbito:

- Win32_Process para la administración de procesos.
- Win32_Environment para verificar la configuración de las variables de entorno.
- Win32_OperatingSystem para consultar datos del SO.
- Win32_PingStatus
- Win32_Directory para administrar carpetas.
- CIM_DataFile para administrar archivos.
- StdRegProv para verificar, en el Registro de Windows, la instalación de programas. Es más rápido que la clase Win32_Product, y además detecta más programas instalados porque esta última no detecta programas que no se hayan instalado con Windows Installer.

1.8.4 Lenguaje de Programación C# v1.0.

C# es un lenguaje de programación orientado a objetos, desarrollado y estandarizado por Microsoft Corporation como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA[22].

Es un lenguaje de alto nivel que evoluciona de los lenguajes C y C++, toma casi todos sus operadores, expresiones y palabras reservadas en cuanto a sintaxis. Es un lenguaje de tipado seguro, moderno, diseñado para programar aplicaciones empresariales de gran utilidad.

C# es el lenguaje principalmente recomendado para programar aplicaciones en .NET. Sin embargo es posible programar en dicha plataforma en muchos otros lenguajes, pero C# ha sido diseñado, particularmente, para ser utilizado en ella. Por estas razones programar

[22] Siglas en inglés: European Computer Manufacturers Association.

usando este lenguaje es más sencillo e intuitivo que hacerlo con otro de los lenguajes que soporta la plataforma, por la cual, suele llamarse a C# como el lenguaje nativo de .NET.

1.8.5 Visual Studio .NET 2003 Enterprise Architect.

Visual Studio .NET es un Integrated Development Environment (IDE, Entorno de Desarrollo Integrado), desarrollado por Microsoft Corporation, para trabajar sobre la plataforma .NET en sistemas operativos de Windows. Soporta varios lenguajes de programación como Visual C++, Visual C#, Visual J# y Visual Basic .NET, y otros muchos lenguajes. Es un IDE muy cómodo que facilita con rapidez la implementación de soluciones para aplicaciones de escritorio, con un alto grado de rendimiento dada las potentes herramientas y componentes asociados. Se considera el mejor IDE para trabajar con la plataforma .Net v1.1. Al instalarlo incluye además, por defecto, la instalación del .NET Framework SDK.

Visual Studio .NET 2003 Enterprise Architect es el único de las cuatro ediciones del 2003 que permite el trabajo con "Enterprise Templates", lo cual le permite al equipo de desarrollo del SIGESC, estandarizar estilos de programación e impulsar restricciones arquitectónicas de organización de código.

1.8.6 Framework.

1.8.6.1 DMAX v2.0.

El Framework DMAX (Desarrollo Modular Basado en una Arquitectura eXtensible), desarrollado por la Universidad de las Ciencias Informáticas, define una estrategia de desarrollo genérica en el dominio de las aplicaciones de escritorio para la gestión empresarial.

DMAX, desarrollado específicamente para soportar la arquitectura del SIGESC, permite la integración con .NET framework v1.1 y agrupa herramientas y funcionalidades que permiten agilizar el trabajo de los desarrolladores, ayudándolos a establecer un estándar de programación y a utilizar restricciones arquitectónicas de organización de código. Provee mecanismos especializados para la interacción entre dominios de aplicación, crear nuevos módulos en nuevos dominios de aplicación, configurarlos, inicializarlos y descargarlos.

Propone una arquitectura de estructura modular que permite el desacople de funcionalidades y define fronteras administrativas de seguridad, auditoría, mecanismos de interacción, configuración y aislamiento dentro del mismo; lo cual beneficia la creación de paquetes (módulos) de implementación totalmente autónomos con un alto grado de cohesión.

Facilita, mediante herramientas, que se desarrollen aplicaciones con una arquitectura multicapa y así garantizar un fácil mantenimiento, especialización de las actividades en el proceso de desarrollo, desarrollo paralelo, independencia de la tecnología de persistencia, entre otras.

Los principios de diseño e implementación que emplea van dirigidos a preservar la legibilidad y estructuración del código con vista a facilitar revisiones, mantenimientos, reutilización o simplemente cambios del equipo de desarrollo.[23]

1.8.6.2 NHibernate v1.1.

NHibernate es un motor de persistencia de datos de Código Abierto para .NET basado en la implementación Hibernate de Java.

Es una herramienta que brinda facilidades para trabajar con procedimientos de almacenado y funciones, facilita además el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación. Proporciona a desarrolladores detallar el modelo de datos y las relaciones existentes. Permite a aplicaciones manipular la información de las Bases de Datos, operando sobre objetos, con todas las características de la POO. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente.

La relación entre los objetos del sistema y las tablas de la BD se realizan a través de archivos XML de mapeo. De esta forma cualquier cambio en la estructura de la base se ajusta en el archivo de mapeo y no en el código. Se mitiga (hasta casi eliminar) el uso de instrucciones SQL sobre el código, las cuales son generadas y administradas por el motor de persistencia.

[23] Consultar (Sánchez Tellez, 2008) para más información.

1.8.7 Sistema Gestor de Base de Datos. Oracle 10g.

La Corporación Oracle ofrece el SGBDR como un producto incorporado a la línea de producción. Además incluye cuatro generaciones de desarrollo de aplicación, herramientas de reportes y utilitarios. Oracle corre en computadoras personales (PC), microcomputadoras, mainframes y computadoras con procesamiento paralelo masivo. Soporta unos 17 idiomas, corre automáticamente en más de 80 arquitecturas de hardware y software distintos sin tener la necesidad de cambiar una sola línea de código.

Oracle10g incluye una versión más optimizada del Procedural Language / Structured Query Language (PL/SQL), lenguaje de programación que se utiliza para programar los procedimientos y funciones de BD en Oracle. Es una extensión de SQL y soporta todas las consultas pues el manejo de los datos que utiliza es el mismo que en SQL, aunque incluye nuevas características como el manejo de variables, estructuras modulares, estructuras de control de flujo y toma de decisiones, y el control de excepciones. El lenguaje PL/SQL está incorporado en el servidor de base de datos y en herramientas de Oracle. Los programas o paquete es de SQL se pueden almacenar en BD como un objeto donde solo podrán ser accedidos por los usuarios que estén autorizados. [24]

1.8.7.1 InstantClient v10.2.

InstantClient es un cliente ligero de conexión que usan las aplicaciones del SIGESC para conectarse a la BD porque permite ejecutar aplicaciones sin necesidad de instalar el cliente estándar de Oracle. La mayoría de las aplicaciones pueden trabajar utilizando la previa instalación de InstantClient de Oracle, mientras se usa mucho menos espacio en disco que con la instalación estándar de Oracle. Los usuarios pueden probar nuevos paquetes de aplicaciones y características de cliente de Oracle rápidamente sin tener que preocuparse de otras instalaciones. Utiliza para la conexión con la BD el protocolo Transparent Network Substrate **(**TNS, Sustrato de Red Transparente) que es una capa de comunicación que utilizan las bases de datos Oracle, donde la información viaja de manera segura porque TNS admite el cifrado y mensajes criptográficos resúmenes para proteger los datos en tránsito. [25]

[24] Consultar (Greenwald, et al.) para más información.

[25] Consultar (Oracle Corporation, 1999).

Dentro de las desventajas de la instalación del InstantClient está que se debe configurar la variable de entorno Path indicándole la dirección del directorio de instalación para poder hacer uso de él. Además de que Oracle en la distribución 10.2 se le olvidó incluir la librería de ejecución Visual C++ del 2003, msvcr71.dll lo que contribuyó a que existieran problemas de conexión entre las aplicaciones y la BD.

1.8.8 Herramienta CASE. Visual Paradigm for UML.

Visual Paradigm for UML (VP-UML) es una herramienta CASE UML diseñada para la ayuda al desarrollo de software y que brinda soporte al modelado visual con UML 2.0. Posee una gran cantidad de módulos para facilitar el trabajo de desarrollo de un software y garantizar la calidad del producto final.

VP-UML soporta los principales estándares de la industria tales como SysML, BPMN, XMI, UML, etc. Ofrece un completo conjunto de herramientas de los equipos de desarrollo de software necesario para los requisitos de la captura, software de planificación, la planificación de controles, el modelado de clases, modelado de datos, etc.

Sus principales características son:

1. Brinda la posibilidad de crear los artefactos necesarios para la confección de un software cumpliendo además con el estándar UML 2.0.
2. Brinda numerosos estereotipos a utilizar lo cual permite que los diagramas se entiendan mejor.
3. Permite redactar especificaciones de casos de uso sin necesidad de utilizar un editor de texto externo.
4. Puede generar código a partir de diagramas para plataformas como .Net, Oracle, así como obtener diagramas a partir de código.
5. Se integra fácilmente con ambientes de desarrollo integrados como Visual Studio.
6. Brinda la posibilidad de documentar todo el trabajo sin tener que utilizar herramientas externas.
7. Disponible en múltiples plataformas como: Microsoft Windows XP.
8. Se puede integrar con SVN que es un sistema de control de versiones, permitiendo el trabajo colaborativo, lo que le permite a múltiples usuarios trabajar sobre el mismo proyecto.
9. Genera la documentación automáticamente en varios formatos como Web o Pdf.

1.9 Conclusiones Parciales.

Después de realizar el estudio del arte de las tendencias actuales hacia los sistemas distribuidos administrados, analizar otros conceptos importantes que son necesarios para la concepción del sistema y realizar una evaluación de las metodologías de desarrollo y herramientas a utilizar se definió que para la realización de la aplicación propuesta se utilizará como metodología de desarrollo de software a RUP, como lenguaje de modelado a UML pues ambas combinan perfectamente constituyendo una metodología robusta que es muy utilizada para el análisis, documentación e implementación de sistemas orientados a objetos. Además se empleará IDEF0 para la representación de los procesos del negocio. Como Plataforma de desarrollo se empleará un conjunto componentes que facilitan el trabajo de desarrollo más fácil como la plataforma .Net v1.1 con el marco de trabajo de .Net (SDK) y su integración con WMI, además utilizar como lenguaje de programación a C#, por ser el lenguaje en que están desarrollados los componentes del SIGESC, para lo cual se utilizará como Entorno de desarrollo a Visual Studio 2003 por ser el mejor IDE que se integra con la versión de .Net que se empleará. Se manejarán otros framework de desarrollo como el DMAX 2.0 y el NHibernate v1.1, también como Sistema Gestor de Base de Dato a Oracle10g y para ejecutar aplicaciones sin necesidad de instalar el cliente estándar de Oracle en las PC clientes, el Instant Client v10.2. Como herramienta CASE el Visual Paradigm Enterprise Edition v3.4 por su fácil integración con el IDE y la metodología seleccionados previamente.

Capítulo 2: Características del Sistema.

2.1 Introducción.

En este capítulo se puntualiza en los procesos que serán automatizados, además se aborda los aspectos esenciales del negocio y los requisitos a tener presentes para el desarrollo del sistema.

Para lograr lo anteriormente planteado, se hace la descripción de la propuesta de solución de este trabajo, describiendo los procesos del negocio que tiene que ver con el objeto de estudio. Se enumeran además los requisitos funcionales y no funcionales que debe tener el sistema que se propone, lo que permite tener una visión general del mismo. Con el objetivo de comprender las especificidades de lo que permitirá hacer el sistema se confeccionará una descripción de todos los casos de uso del mismo.

2.2 Procesos del Negocio.

La modelación de los procesos del negocio se realizará mediante la notación IDEF0.

Los procesos del negocio identificados se muestran en el siguiente diagrama:

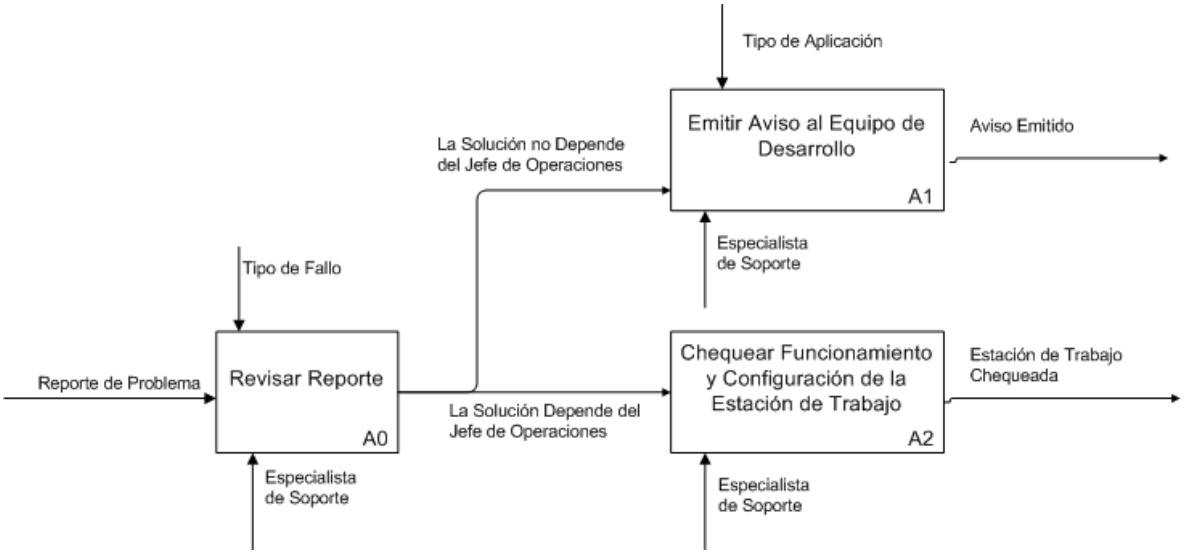

Fig.2.1 Diagrama IDEF0 Principal de procesos del Negocio.

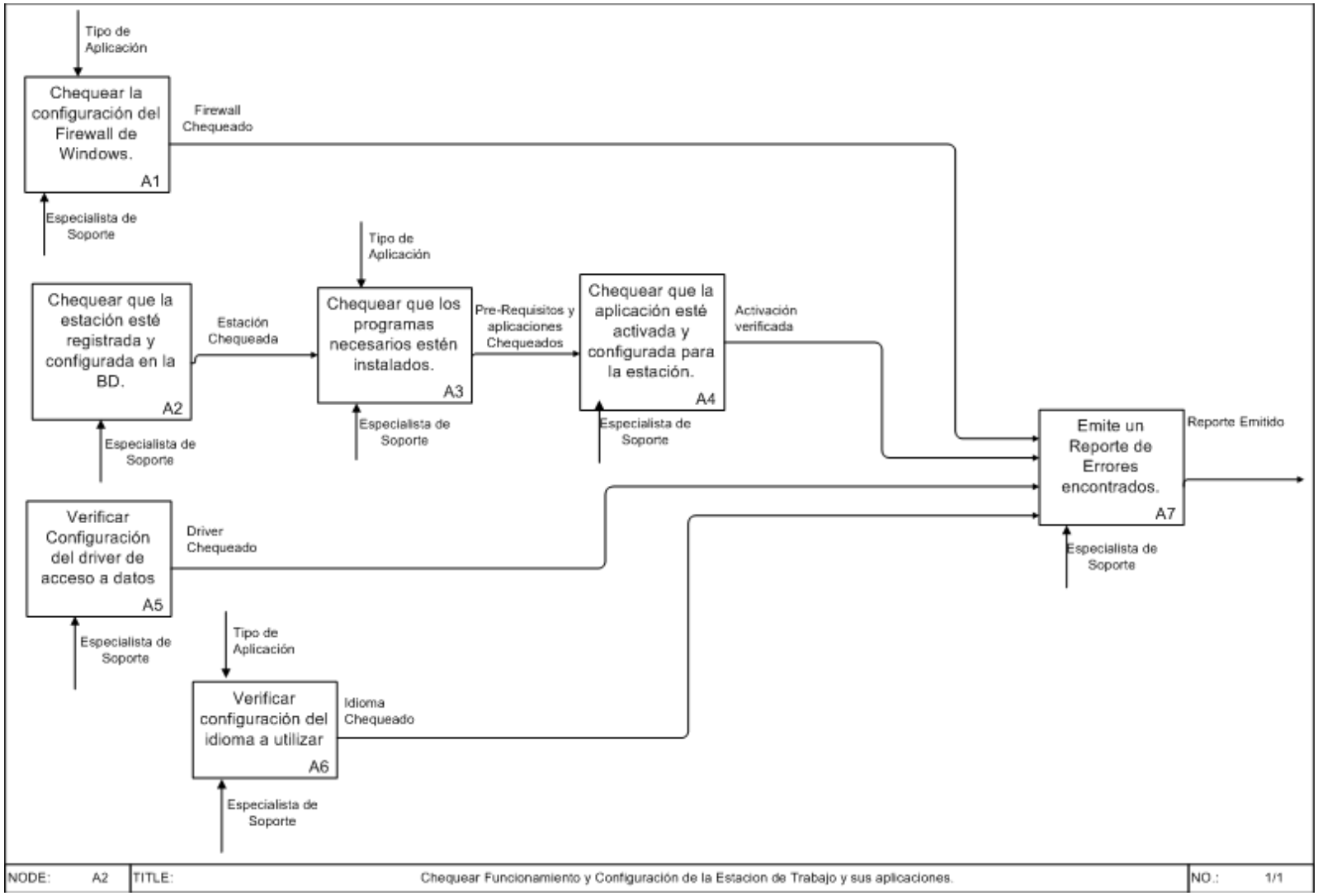

Fig.2.2 Diagrama IDEF0 de procesos del Negocio A2.

Nombre	Justificación
Especialista de Soporte.	Persona encargada de realizar todos los procesos referentes al mantenimiento y operaciones del SIGESC.

Tabla 2.1 Personas que intervienen en los procesos del negocio.

2.2.2 Descripción de los procesos del negocio.

Cuando una aplicación deja de funcionar correctamente, por cualquier razón, el trabajador que ocupa dicha estación realiza un reporte digital donde describe el error que esta ocurriendo, posteriormente, envía dicho reporte por email al Especialista de Soporte.

2.2.2.1 Descripción de los procesos del Negocio Diagrama Principal.

- **Revisar Reporte**: El Especialista de Soporte cuando recibe el reporte de fallo, lo revisa y determina si puede darle respuesta sin implicar al equipo de desarrollo.
- **Chequear Funcionamiento y configuración de la Estación de Trabajo**: Si la solución del fallo ocurrido depende del Especialista de Soporte darle respuesta, dicho equipo se dirige a donde está la PC con problemas y procede a darle solución, verificando una serie de pre requisitos de software y verificando además el estado de configuración de la estación de trabajo que emitió el reporte y de las aplicaciones del SIGESC que en ella deben funcionar.
- **Emitir Aviso al equipo de desarrollo**: En caso que la solución al fallo presentado no depende del Especialista de Soporte se procede a emitir un aviso al equipo de desarrollo donde se le envía un resumen del problema.

2.2.2.2 Descripción de los procesos del Negocio Diagrama A2.

- **Chequear la configuración del Firewall de Windows**: el firewall de Windows debe estar correctamente configurado para que permita la correcta ejecución de todas las aplicaciones del SIGESC, por lo que el Especialista de Soporte verifica que se encuentre activado, pero al mismo tiempo debe tener dentro de sus excepciones una donde el nombre sea el nombre de la aplicación y el puerto sea el especificado para dicha aplicación previamente, la opción TCP debe estar marcada y en la opción 'Cambiar ámbito...' debe tener marcada Sólo mi red (subred).
- **Chequear que la estación esté registrada y configurada en la BD:** para que las aplicaciones del SIGESC funcionen en una estación determinada, esta última debe

estar registrada en la base de datos y el estado de activación debe ser el de "activada". En caso contrario el Especialista de Soporte procede a registrar la estación en la BD o a cambiarle el estado de activación.

- **Chequear que los programas necesarios estén instalados**: Si la solución del problema detectado depende del Especialista de Soporte darle respuesta, se dirige a donde está la PC con problemas y procede verificar que estén instalados los pre requisitos de software y otros programas necesarios para el correcto funcionamiento de las aplicaciones del SIGESC instaladas en dicha estación. En caso que no estén dichos programas instalados en la estación se procede a su instalación inmediata.
- **Chequear que la aplicación del SIGESC esté activada y configurada para la estación:** instalar una aplicación del SIGESC en una estación no es suficiente para que funcione si antes no se activa dicha aplicación en la estación de trabajo y se configura correctamente en caso de que así lo requiera, una vez que esta última esté registrada y activada en BD. En caso contrario el Especialista de Soporte activa la aplicación para que pueda funcionar en la estación previamente.
- **Chequear configuración del driver de acceso a datos:** el Especialista de Soporte verifica que estén copiadas en un directorio local las librerías de acceso a datos, se verifica que la variable de entorno Path del sistema tenga la ubicación del Instantclient_10_2, y se verifica la configuración de la conexión al servidor de bases de datos. En caso de existir algún error el Especialista de Soporte procede a configurar el driver de acceso a datos correctamente.
- **Chequear configuración del idioma a utilizar:** el Especialista de Soporte verifica que exista una variable de entorno del sistema con el nombre=NLS_LANG y valor=SPANISH_VENEZUELA.WE8MSWIN1252, para indicar el idioma a utilizar. En caso de existir algún error el Especialista de Soporte procede a configurar el idioma correctamente.
- **Emitir reporte de errores encontrados:** Cuando el Especialista de Soporte termina de hacer las revisiones de rutina realiza un reporte con todos los errores encontrados y la solución que le dio cada uno de ellos.

2.3 Requisitos Funcionales.

"Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas y de cómo se debe

comportar en situaciones particulares. Los requisitos funcionales del sistema describen lo que el sistema debe hacer."[26]

Los requisitos funcionales que la aplicación debe cumplir son los siguientes:

RF1: Verificar Funcionamiento y Configuración de las Estaciones de Trabajo.

1.1 Verificar que la estación esté registrada y activada en la BD.

1.2 Verificar Instalación de Pre-Requisitos de Software.

1.2.1.1 Instalación Framework DMAX versión 2.0.

1.2.1.2 Instalación Servicios DMAX v2.0.

1.2.1.3 Instalación Microsoft .NET Framework v1.1

1.2.1.4. Instalación SP1 de Microsoft .NET Framework v1.1

1.2.1.5 Instalación Microsoft .NET Framework v2.0.

1.2.1.6 Verificar Instalación del Framework MapInfo MapXtreme v6.7.

1.3 Verificar configuración del driver de acceso a datos.

1.3.1 Verificar presencia del directorio local para las librerías de acceso a datos (Instantclient_10_2).

1.3.2 Verificar adición en la variable de entorno Path del sistema, la ubicación del Instantclient_10_2.

1.3.2 Verificar configuración de la conexión al servidor de bases de datos (Archivo Host).

1.3.2.1 Número IP del servidor de BD.

1.3.2.2 Nombre de la instancia de BD.

1.4 Verificar que exista una variable de entorno con nombre NLS_LANG y valor SPANISH_VENEZUELA.WE8MSWIN1252.

1.5 Verificar Instalación de la o las aplicaciones de escritorio que deba tener instalada la estación.

1.6 Verificar Activación de la o las aplicaciones.

1.7 Verificar Instalación del Plugin Llamada Ericsson MD110. (Solo verificar si la aplicación es Recepción de Llamadas y esta tiene integración con la planta telefónica Sony Ericsson).

1.8 Verificar Configuración del Cortafuegos (Firewall) de Windows para que permita la correcta ejecución de la aplicación.

[26] (Sommerville, 2005 p. 110)

1.8.1 Verificar que el cortafuegos este configurado para que permita la administración remota desde la estación de trabajo donde se esté ejecutando la aplicación Administración del SIGESC.

1.8.2 Activar el Cortafuegos.

1.8.3 Nombre de Excepción.

1.8.4 Puerto.

1.8.5 Protocolo de red.

1.8.6 Ámbito.

1.9 Verificar que exista conexión entre la estación de trabajo y la BD.

RF2: Verificar Funcionamiento de la estación de trabajo Servidor de Sincronización.

2.1 Verificar Funcionamiento y Configuración de las Estaciones de Trabajo. (Todos los puntos del RF1).

2.2 Verificar Instalación del Plugin Sincronización Ericsson MD110. (Para la planta telefónica Ericsson.)

2.3 Verificar el modo de inicio del servicio de Sincronización.

2.3.1 Modo Inicio de Sesión.

RF3: Notificar Sucesos.

3.1 Cuando se detecta un problema, durante el monitoreo de una estación de trabajo, se emite una Notificación con los siguientes datos:

3.1.1 Número IP.

3.1.2 Notificación con el problema encontrado.

RF4: Configuración Remota del driver de acceso a datos.

4.1 Adicionar a la variable de entorno Path del sistema, la ubicación del Instantclient_10_2.

4.2 Copiar en la ubicación del Instantclient_10_2 los archivos necesarios.

4.2 Configuración de la conexión al servidor de BD (Archivo Host).

4.2.1 IP del servidor.

4.2.2 Nombre de la instancia de BD.

RF5: Configurar Remoto el idioma a utilizar mediante la variable de entorno.

5.1 Nombre.

5.2 Valor.

RF6: Configurar las estaciones en la BD del SIGESC.

RF7: Activar una aplicación de manera Remota en una estación de trabajo.

7.1 Aplicación.

7.1.1 Sistema Operativo.

7.1.2 Número IP.

7.1.3 Tipo aplicación.

7.1.4 Oficina.

RF8: Configurar de manera remota el Cortafuegos (Firewall) de Windows para que permita la correcta ejecución de la aplicación.

8.1 Activar el Cortafuegos.

8.2 Nombre de Excepción.

8.3 Puerto.

8.4 Protocolo de red.

8.5 Ámbito.

8.6 Configurar el Firewall para que permita administración remota de la estación de trabajo de donde se efectúa el monitoreo.

RF9: Registrar Reporte de Errores.

9.1 Se registran los siguientes datos de un reporte:

9.1.1 Número IP del Punto.

9.1.2 Oficina.

9.1.3 Descripción del Tipo de Error.

9.1.4 Fecha en que se produce el Error.

RF10: Modificar Reporte de Errores.

10.1 De los datos registrados los datos que se pueden modificar son:

10.1.1 Número IP del Punto.

10.1.2 Oficina.

10.1.3 Descripción del Tipo de Error.

10.1.4 Fecha en que se produce el Error.

R11: Eliminar Reporte de Errores.

R12: Mostrar Reporte de Errores.

12.1 Se realiza una búsqueda de los Reportes de Errores según uno o varios de los siguientes criterios:

12.1.1 Número IP del Punto.

12.1.2 Oficina.

12.1.3 Fecha en que se produce el Error.

12.2 Se muestra como resultado de la búsqueda los Reportes de Errores que hallan coincidido con los parámetros de búsqueda.

RF13: Registrar Credenciales.

13.1 Se registran los siguientes datos:

13.1.1 Número IP del Punto.

13.1.2 Usuario Local del Punto.

13.1.3 Contraseña.

13.2 En caso de que sea un súper usuario solo registra el usuario y la contraseña.

RF14: Modificar Credenciales.

14.1 De los datos registrados los datos que se pueden modificar son:

14.1.1 Número IP de la PC.

14.1.2 Usuario Local del Punto.

14.1.3 Contraseña.

R15: Eliminar Credenciales.

R16: Mostrar Credenciales.

16.1 Se realiza una búsqueda de Credenciales según uno o varios de los siguientes criterios:

16.1.1 Número IP de la PC.

16.1.2 Usuario

16.2 Se muestra como resultado de la búsqueda los usuarios que hallan coincidido con los parámetros de búsqueda.

2.4 Requisitos No Funcionales.

"Los requisitos no funcionales especifican propiedades del sistema, como restricciones del entorno, o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad, fiabilidad."[27]

2.4.1 Software.

La aplicación Administración del SIGESC está diseñado para trabajar sobre la plataforma de Microsoft Windows XP SP2, las aplicaciones de escritorio del SIGESC deben estar funcionando bajo este mismo SO, a excepción de Sincronización que corre bajo Microsoft Windows Server 2003. Ambos sistemas operativos deben tener instalado el Instrumental

[27] (Jacobson, et al., 2000 p. 110)

de Administración de Windows (WMI) y ejecutándose el servicio en cuestión, pues aunque WMI viene por defecto en la instalación de los sistemas operativos de Windows a partir del año 2000, puede darse el caso de que no esté instalado.

El Cortafuegos de las estaciones debe estar activado pero al mismo tiempo debe estar configurado para que permita administración remota y así se puedan monitorear. Para ello se debe ir a Ejecutar\se escribe gpedit.msc+enter\ aparece una ventana que se llama Directiva de grupo\ Configuración del equipo\ Plantillas administrativas\ Red\ Conexiones de Red\ Firewall de Windows\ y se selecciona el perfil de dominio si la estación se encuentra en un dominio sino se elige perfil estándar\ en ambos casos aparece una opción dentro de dicho perfil que se llama "permitir excepción de administración remota"\ Propiedades\Habilitar\Permitir mensajes entrantes no solicitados de\ se introduce el número IP de la estación donde se encuentra funcionando la aplicación de Administración del SIGESC o una subred.

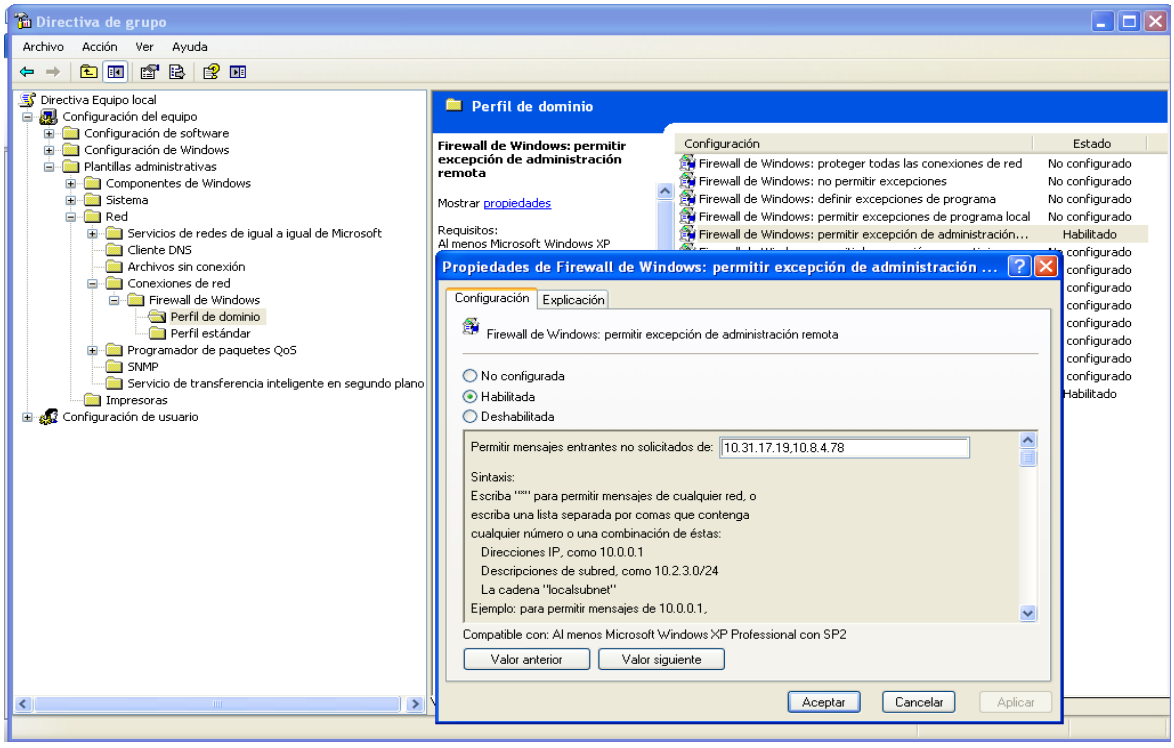

Fig.2.3 Configuración del Cortafuegos.

2.4.2 Seguridad.

Los principios básicos que determinan la seguridad del sistema son los siguientes:

- Seguridad a nivel de estaciones de trabajo, garantizando que la aplicación se ejecute en la estación de trabajo que haya sido autorizada previamente.
- Seguridad y control a nivel de usuarios, contraseñas y roles, garantizando el acceso a la aplicación solo a los usuarios con el rol Administrador Informático.
- Las contraseñas de sesión local, empleadas para entrar a las estaciones de trabajo mediante WMI (para obtener la información necesaria que nos permita verificar el estado de funcionamiento y configuración de las aplicaciones del SIGESC que se ejecutan en las estaciones de trabajo), se almacenan en la BD encriptadas mediante un algoritmo asimétrico.

Para monitorear cada una de las estaciones de trabajo es necesario establecer una conexión, mediante la red, entre ambas estaciones; la estación donde se encuentre ejecutando la aplicación Administración del SIGESC y la estación a monitorear. Este tipo de conexión requiere que la primera estación se acredite ante la otra mediante credenciales de administrador porque se necesita acceder a la información del Registro de Windows para procesar datos con el objetivo de determinar el estado de configuración de la estación. Dichas credenciales se guardarán en BD mediante un algoritmo de cifrado, pues si alguien accediera a la BD y obtuviera estas credenciales en texto plano estaría comprometida la seguridad del Centro 171.

2.4.2.1 Criptografía.

Para adentrarse en este campo tan grande que comprende la criptografía es necesario conocer los siguientes conceptos:

*"La **criptografía** es la ciencia o técnica que permite proteger la información por medio de la aplicación de un método de cifrado."*

*"**Cifrar**: acción de aplicar técnicas criptográficas, con el objetivo de esconder un mensaje que será enviado por una línea de comunicación insegura, de forma tal que solamente el receptor autorizado pueda leer el mensaje."*[28]

La criptografía resuelve problemas de seguridad como la **confidencialidad**, la **integridad**, donde la **confidencialidad** o **privacidad**, se refiere a que la información sólo pueda ser

[28] (Universidad de las Ciencias Informáticas, 2010 p. 2)

leída por personas autorizadas y la **integridad**, se refiere a que la información no pueda ser alterada en el transcurso de ser enviada.

La criptografía se clasifica en dos ramas fundamentales: Clásica y Moderna:

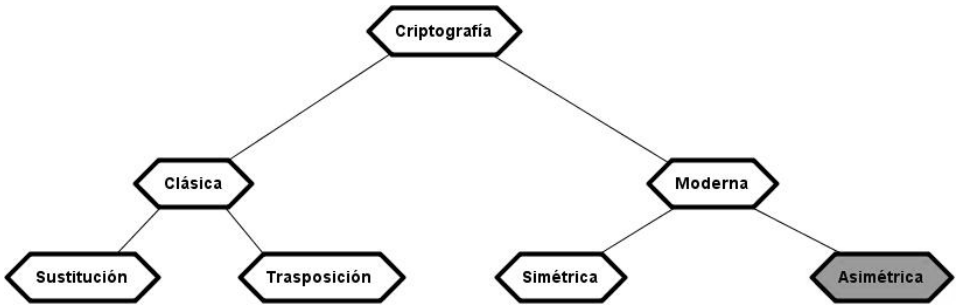

Fig.2.4 Clasificación fundamental de la Criptografía.

Dos técnicas engloban las manifestaciones más relevantes de la criptografía clásica: La **sustitución y l**a **trasposición** pero como todos los algoritmos de sistemas criptográficos clásicos, presentan dificultad en cuanto a la relación complejidad-longitud de la llave y el tiempo necesario para cifrar y descifrar el mensaje.

Estos problemas se resuelven en la criptografía moderna pues surgen una serie de factores que condicionaron el surgimiento de nuevos algoritmos más eficaces:

- La velocidad de cálculo: con la aparición de los computadores se dispuso de una potencia de cálculo muy superior a la de los métodos clásicos.
- El avance de las matemáticas: que permitieron encontrar y definir con claridad sistemas criptográficos estables y seguros.
- Necesidades de seguridad: surgieron muchas actividades nuevas que precisaban la ocultación de datos, con lo que la Criptografía experimentó un fuerte avance.

Producto de los factores planteados surgieron nuevos y complejos sistemas criptográficos, que se clasificaron en los dos tipos o familias principales, los de llave simétrica y los de llave pública (asimétricos).

Los modernos algoritmos de cifrado simétricos mezclan la trasposición y la permutación, mientras que los de llave pública se basan más en complejas operaciones matemáticas.

Los **simétricos**, también conocidos como de llave única, basan su fortaleza en el secreto de una llave k. De modo que, si esta llave se llega a descubrir, hallar la información inicial a partir de la cifrada sería una tarea teóricamente fácil.

Las principales desventajas de los métodos simétricos son la distribución de las llaves, el peligro de que muchas personas deban conocer una misma llave y la dificultad de almacenar y proteger muchas llaves diferentes.

Los asimétricos, también llamados de llave pública, se basan en el uso de dos llaves diferentes, llaves que poseen una propiedad fundamental: una llave puede descifrar lo que la otra ha cifrado.

Fig.2.5 Criptografía Asimétrica.[29]

Las llaves pública y privada tienen características matemáticas especiales, de tal forma que se generan siempre a la vez, por parejas, estando cada una de ellas ligada intrínsecamente a la otra, de tal forma que si dos llaves públicas son diferentes, entonces sus llaves privadas asociadas también lo son, y viceversa.

La principal ventaja de los sistemas de llave pública frente a los simétricos es que la llave pública y el algoritmo de cifrado son o pueden ser de dominio público y que no es necesario poner en peligro la llave privada en tránsito por los medios inseguros, ya que ésta siempre está oculta y en poder únicamente de su propietario. Como desventaja, los sistemas de llave pública dificultan la implementación del sistema y son mucho más lentos que los simétricos.

[29] Tomado de (Universidad de las Ciencias Informáticas, 2010).

Para que un algoritmo de llave pública sea considerado seguro debe cumplir:

- Conocido el texto cifrado no debe ser posible encontrar el texto en claro ni la llave privada.
- Conocido el texto cifrado (criptograma) y el texto en claro debe resultar más caro en tiempo o dinero descifrar la llave que el valor posible de la información obtenida por terceros.
- Conocida la llave pública y el texto en claro no se puede generar un criptograma correcto cifrado con la llave privada.
- Dado un texto cifrado con una llave privada sólo existe una pública capaz de descifrarlo, y viceversa.[30]

El espacio de nombre **System.Security.Cryptography** del framework de .Net v1.1 permite encriptar datos mediante el uso de algoritmos simétricos y asimétricos.

Simétricos	Asimétricos
DES (Estándar de cifrado de datos).	MD5 (Algoritmo de Resumen del Mensaje).
TDES (Triple DES).	SHA1, SHA256, SHA384, SHA512 (Algoritmo de *Hash* Seguro).
RC2 (Rivest Cipher 2).	DSA (Algoritmo de Firma digital).
AES (Estándar de cifrado avanzado o Rijndael).	**RSA.**

Tabla 2.2 Algoritmos de cifrado Framework v1.1.

Para cifrar las credenciales de sesión el uso de algoritmos asimétricos irreversibles es lo más aconsejable pero como estas credenciales se utilizan para autenticarse ante la estación que se está monitoreando, es necesario poder restablecer las credenciales de usuario a su estado original.

De los algoritmos asimétricos que aparecen en la Tabla 2.2 el MD5 y la familia de los SHA son algoritmos irreversibles y por lo tanto no se pueden usar.

El algoritmo DSA no sirve para cifrar información, se usa en firma digital (la firma digital se usa cuando se necesita saber que el mensaje cifrado lo mando la persona que uno espera, o sea para verificar la autoría del mensaje). Una desventaja de este algoritmo es que requiere mucho más tiempo de cómputo que RSA.

[30] Consultar (Universidad de las Ciencias Informáticas, 2010) para más información.

RSA[31] (Rivest, Shamir y Adleman) desarrollado en 1977 es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

RSA basa su seguridad en ser una función computacionalmente segura, ya que si bien realizar la exponenciación modular es fácil, su operación inversa, la extracción de raíces de módulo Ø no es factible a menos que se conozca la factorización del número, llave privada del sistema. RSA es el más conocido y usado de los sistemas de llave pública, y también el más rápido de ellos.[32]

Basándose en lo anteriormente planteado el algoritmo RSA es el recomendado para el cifrado de las credenciales de sesión que se almacenarán en BD y así mitigar los problemas de confidencialidad de la información porque solo podrán ser accedidas por las personas autorizadas. Para ello también en la BD solo se le dará permiso a acceder a la tabla de claves de encriptación al rol de Administrador Informático.

2.5 Descripción de la Solución Propuesta.

El SIGESC 171 es un sistema con un alto nivel de profesionalidad y automatización de los procesos que se llevan a cabo en un CGE 171. Surge con el objetivo de brindar una respuesta rápida al atender las emergencias que se le presenten al pueblo venezolano, coordinando con los órganos de seguridad que sean necesarios.

La aplicación de Administración del SIGESC es una aplicación de escritorio, que solo puede ser usado por los usuarios con el rol de Administrador Informático autenticados en el sistema.

Debe permitirle al Administrador Informático verificar el funcionamiento y configuración de las estaciones de trabajo donde se encuentren las siguientes aplicaciones del SIGESC instaladas:

- Recepción de Llamadas, Despacho, Mapificación, Supervisión de Operadores, Supervisión de Despacho, Supervisión General, Estadística. (Visor de Reportes), Configuración de Operaciones, Administración Informática, Sincronización.

[31] RSA: por las iniciales de los apellidos de sus descubridores.

[32] Consultar (Universidad de las Ciencias Informáticas, 2010).

Se deben registrar reportes con los errores detectados y estos deben poder modificarse, eliminarse o consultarse en el momento que se requiera.

En caso de ocurrir un error con alguna de las aplicaciones mencionadas anteriormente, se debe emitir una alerta automática al Administrador Informático para que este proceda a darle solución.

2.6 Modelo de Casos de Uso del Sistema.

"Un caso de uso (CU) es, simplemente, un texto escrito que describe el papel de un actor que interactúa con el acontecer del sistema."[33]

Los casos de uso son una técnica para especificar el comportamiento de un sistema. El modelo de casos de uso describe lo que hace el sistema para cada tipo de usuario.

En este caso con la aplicación de Administración del SIGESC interactúa el actor que se define a continuación:

Nombre	Justificación
Administrador Informático.	Representa el Rol que se le asigna generalmente al personal encargado de dar soporte al SIGESC. Son los encargados de darle solución a los problemas informáticos que se presenten.

Tabla 2.3 Actor que interviene en Sistema.

Fue necesario realizar un diagrama de paquetes para agrupar los casos de usos por funcionalidades y así conseguir entender mejor el funcionamiento de la aplicación.

[33] (Pressman, 2001 p. 187)

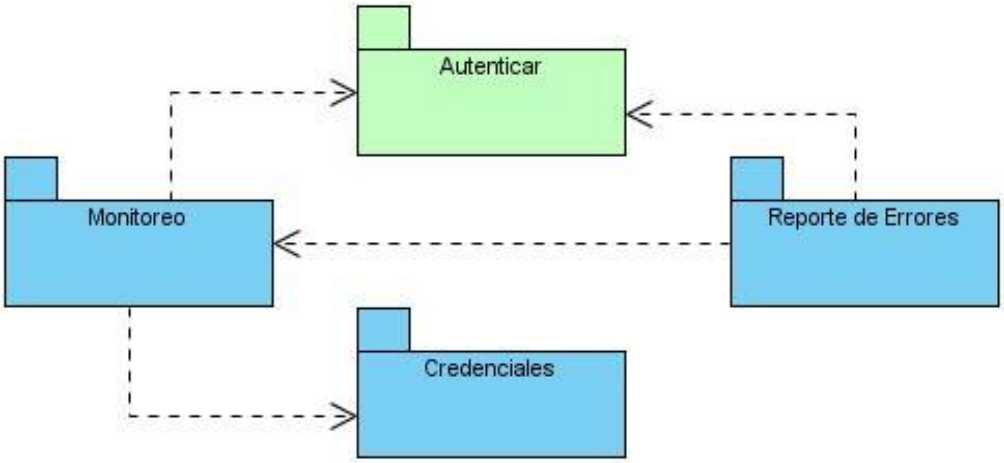

Fig.2.6 Diagrama de Paquetes.

Los diagramas donde se aprecia la relación existente entre actores y los casos de uso se representan a continuación:

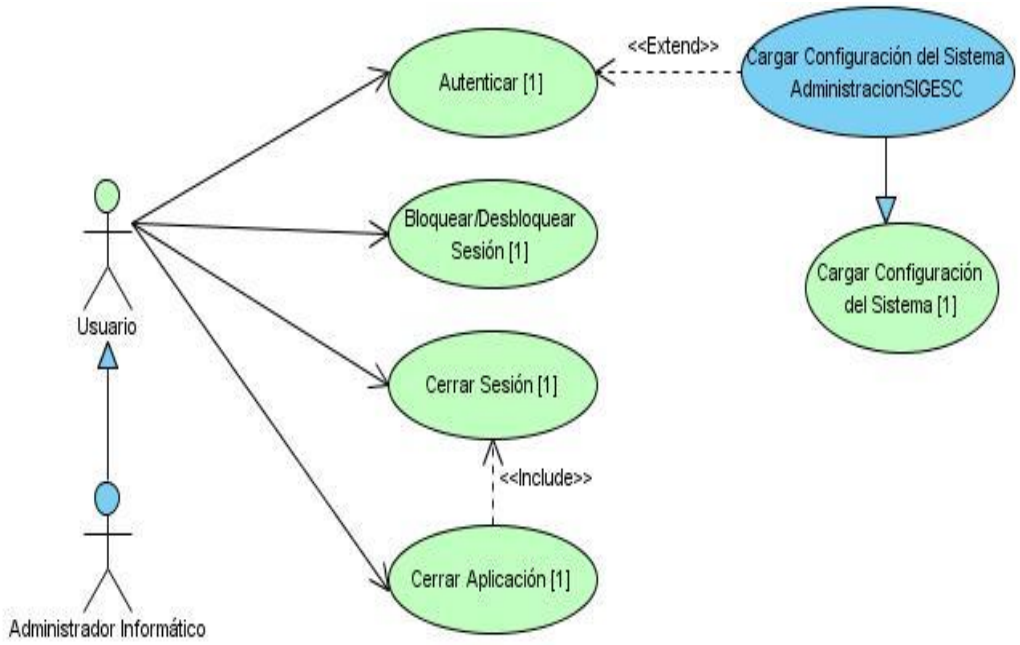

Fig.2.7 Diagrama de Casos de Uso del paquete Autenticar.

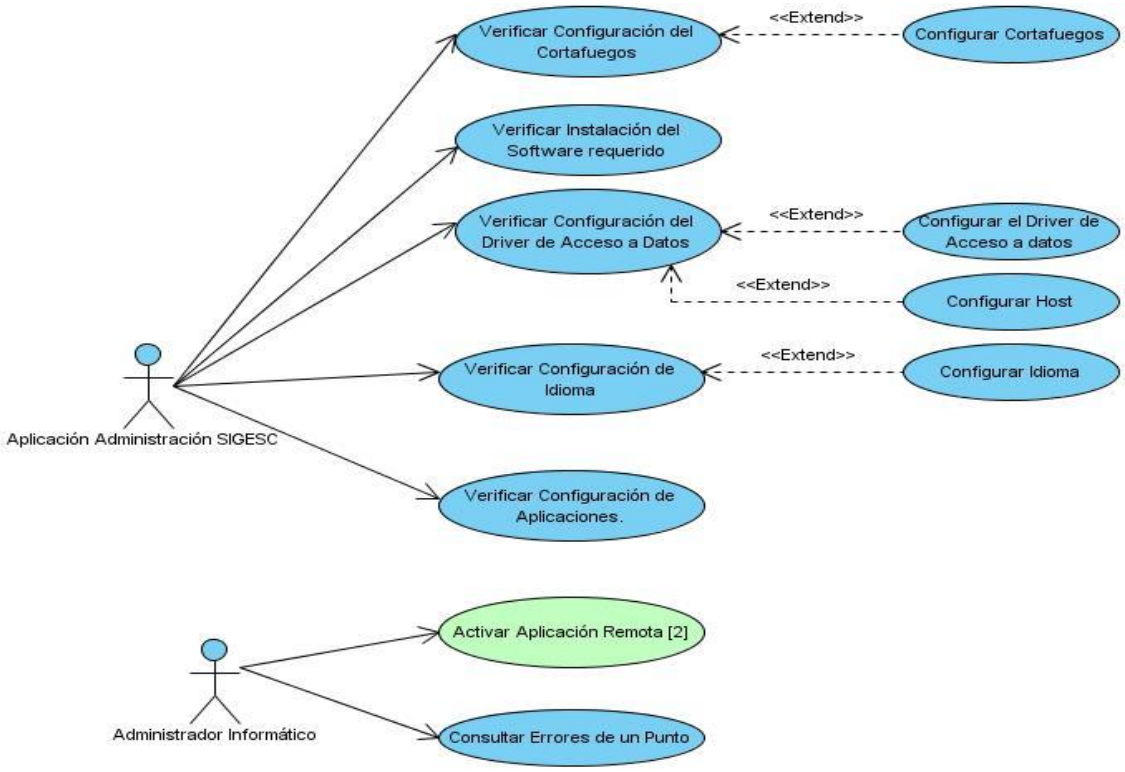

Fig.2.8 Diagrama de Casos de Uso del paquete Monitoreo.

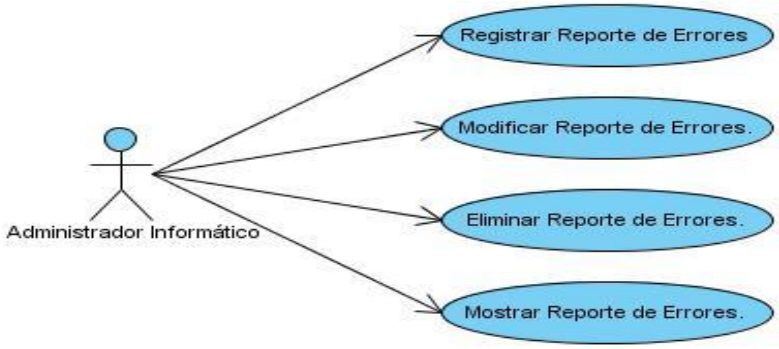

Fig.2.9 Diagrama de Casos de Uso del paquete Reporte de Errores.

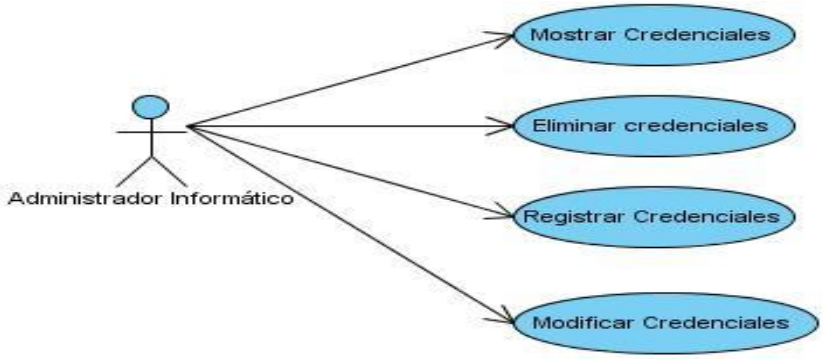

Fig.2.10 Diagrama de Casos de Uso del paquete Credenciales.

Los casos de uso de color verde son parte de otros módulos del sistema, y no se encuentran en los ciclos de desarrollo porque no van a ser implementados como parte de la aplicación. Los números que se encuentran entre corchetes son referencias a los módulos a los que pertenecen dichos casos de uso. Ver tabla 2.3 Referencias.

Código	Título
[1]	00 - Módulo Común.
[2]	12 - Módulo de Configuración de Operaciones.

Tabla 2.4 Referencias.

Se prosiguió a realizar una división de los casos de uso en 2 ciclos de desarrollo como se muestra en la tabla siguiente, donde se aprecia el nombre de cada caso de uso, el ciclo al que corresponde y la justificación de cada uno.

Nombre del Caso de Uso	Ciclo	Justificación de la selección
CU Cargar Configuración del Sistema.	1	Representan los casos de uso que responden a las funcionalidades vitales de la aplicación.
CU Verificar Configuración del Cortafuego.		
CU Verificar Instalación del Software Requerido.		
CU Verificar Configuración de las aplicaciones.		
CU Verificar Configuración del Driver de Acceso a Datos.		
CU Verificar Configuración de idioma.		
CU Registrar Credenciales.		
CU Consultar Errores de un Punto.		
CU Configurar Cortafuego.	2	Representan los casos de uso que sirven de apoyo a la aplicación.
CU Configurar Driver de Acceso a datos		
CU Configurar Host.		
CU Configurar Idioma.		
CU Registrar Reportes.		
CU Mostrar Reportes.		
CU Modificar Reportes.		
CU Eliminar Reportes.		
CU Mostrar Credenciales.		

CU Modificar Credenciales.		
CU Eliminar Credenciales.		

Tabla 2.5 Casos de Uso.

2.7 Expansión de los Casos de Uso.

Mediante la expansión de los casos de uso se puede describir paso a paso la secuencia de eventos que los actores utilizan para completar un proceso a través del sistema.
Las descripciones detalladas de los CU se encuentran en los anexos del 1 al 19.

Nombre del CU: Cargar Configuración del Sistema.	
Propósito:	Cargar los elementos de configuración necesarios definidos para el Módulo de Administración del SIGESC.
Resumen:	Carga de la BD los puntos que están registrados y que se encuentran activos.
Precondiciones:	El usuario debe haberse autenticado en el sistema con anterioridad.
Poscondiciones:	Muestra la interfaz principal con los puntos a monitorear.

Nombre del CU: Verificar Configuración del Cortafuego.	
Propósito:	Verificar la configuración del cortafuego.
Resumen:	Verifica que el cortafuegos este activado y que esté configurado para los puertos correspondientes.
Precondiciones:	No aplica.
Poscondiciones:	Problemas de configuración del cortafuego detectados.

Nombre del CU: Verificar Instalación del Software Requerido	
Propósito:	Verifica la instalación del software necesario para que funcione correctamente la estación de trabajo.
Resumen:	Verifica en el registro de Windows que se encuentren instalados todos los pre-requisitos de software correspondientes y las aplicaciones del SIGESC que deben estar instaladas en la estación.
Precondiciones:	No aplica.
Poscondiciones:	Problemas de instalación de software detectados.

Nombre del CU: Verificar Configuración de las aplicaciones.

Propósito:	Verificar que la aplicación esté configurada.
Resumen:	Verificar la configuración de las aplicaciones del SIGESC.
Precondiciones:	No aplica.
Poscondiciones:	Configuración de aplicaciones chequeada.

Nombre del CU: Verificar Configuración del driver de Acceso a Datos.	
Propósito:	Verificar que el driver de acceso a datos esté configurado.
Resumen:	Se verifica la conexión con BD, en caso de que no exista conexión se procede a verificar la configuración del host y del driver de acceso a datos.
Precondiciones:	No aplica.
Poscondiciones:	No aplica.

Nombre del CU: Verificar Configuración de Idioma.	
Propósito:	Verificar que este configurado el idioma.
Resumen:	Se verifica que la variable de entorno correspondiente esté configurada para el Idioma.
Precondiciones:	No aplica.
Poscondiciones:	No aplica.

Nombre del CU: Configurar Cortafuego.	
Propósito:	Configurar el cortafuego de una estación de trabajo.
Resumen:	Configurar el cortafuego para que la estaciones tengan más seguridad.
Precondiciones:	Cortafuego no configurado o no activado.
Poscondiciones:	Cortafuego configurado.

Nombre del CU: Configurar driver de Acceso a Datos.	
Propósito:	Configurar driver de Acceso a Datos para una estación de trabajo.
Resumen:	Configurar driver de Acceso a Datos.
Precondiciones:	Se detecta que el Driver de Acceso Datos no se encuentra bien configurado.

Poscondiciones:	Driver de Acceso a Datos configurado.

Nombre del CU: Configurar Host.	
Propósito:	Configurar el Host de una estación de trabajo.
Resumen:	Configurar el archivo Host de una estación de trabajo para que pueda tener conexión a BD.
Precondiciones:	Se detecta que el archivo Host no se encuentra configurado.
Poscondiciones:	Archivo Host configurado.

Nombre del CU: Configurar Idioma.	
Propósito:	Configurar el idioma de una estación de trabajo.
Resumen:	Configurar el idioma mediante una variable de entorno.
Precondiciones:	Se detecta que el idioma no se encuentra configurado.
Poscondiciones:	Idioma configurado.

Nombre del CU: Consultar errores de un Punto.	
Propósito:	Consultar los errores detectados en una estación de trabajo.
Resumen:	El caso de uso inicia cuando se selecciona una estación de trabajo y se selecciona la opción Aceptar.
Precondiciones:	Debe haber marcado una estación de trabajo monitoreada.
Poscondiciones:	Descripción del problema.

Nombre del CU: Registrar Reporte de Errores.	
Propósito:	Almacenar en la BD un Reporte de Errores con los datos de la estaciones de trabajo que presentan problemas.
Resumen:	Se almacenan en la BD los datos necesarios para Registrar un Reporte de Errores.
Precondiciones:	Debe haber terminado el monitoreo y entonces si detectó problemas estos se registran en base de datos.
Poscondiciones:	Datos registrados.

Nombre del CU: Mostrar Reporte de Errores.

Propósito:	Mostrar el listado de los Reportes que cumplan con los criterios de búsqueda introducidos por el Administrador Informático.
Resumen:	Se muestra un listado de los Reportes que cumplen con los criterios de búsqueda especificados. Si no se especifica ningún criterio se muestra el listado de Reportes existentes en la BD.
Precondiciones:	No aplica.
Poscondiciones:	Resultados de la Búsqueda.

Nombre del CU: Modificar Reporte de Errores.	
Propósito:	Modificar un Reporte.
Resumen:	Se modifican de un Reporte los datos introducidos inicialmente.
Precondiciones:	Ejecutarse el CU Mostrar Reporte de Errores, y debe existir un Reporte seleccionado.
Poscondiciones:	Reporte Modificado.

Nombre del CU: Eliminar Reporte de Errores.	
Propósito:	Eliminar los datos un Reporte de Errores en la BD.
Resumen:	Eliminar un Reporte de la BD.
Precondiciones:	Ejecutarse el CU Mostrar Reporte de Errores y haber seleccionado un reporte de la lista.
Poscondiciones:	Se elimina el Reporte de la BD y se actualiza el listado de Reportes.

Nombre del CU: Registrar Credenciales.	
Propósito:	Almacenar en la BD los datos de las credenciales de sesión.
Resumen:	El administrador registra los datos de una nueva credencial de sesión, de una estación de trabajo, en el sistema.
Precondiciones:	No aplica.
Poscondiciones:	Credenciales registradas en BD.

Nombre del CU: Mostrar Credenciales.	
Propósito:	Mostrar el listado de las Credenciales de sesión que cumplan con los criterios de búsqueda introducidos por el Administrador

	Informático.
Resumen:	Se muestra un listado de las Credenciales de sesión que cumplen con los criterios de búsqueda especificados. Si no se especifica ningún criterio se muestra el listado de Credenciales existentes en la BD.
Precondiciones:	No aplica.
Poscondiciones:	No aplica.

Nombre del CU: Modificar Credenciales.	
Propósito:	Modificar una Credencial.
Resumen:	Se modifican los datos de una credencial introducidos inicialmente.
Precondiciones:	Debe existir una Credencial seleccionada.
Poscondiciones:	Credencial Modificada.

Nombre del CU: Eliminar Credenciales.	
Propósito:	Eliminar los datos una Credencial de sesión en la BD.
Resumen:	Se elimina una Credencial de sesión de la lista de Credenciales donde se encuentre.
Precondiciones:	Debe existir una credencial seleccionada.
Poscondiciones:	Credencial eliminada.

2.8 Conclusiones.

En este capítulo se realizó una descripción detallada de la solución propuesta, obteniéndose a partir del análisis de los procesos del negocio que dieron paso posteriormente a los requisitos funcionales del sistema. Dichos requisitos fueron representados mediante diagramas de casos de uso y se detallaron cada uno de ellos.

Una vez realizado estos pasos se puede comenzar a construir el sistema, tratando de que este cumpla objetivamente con las funcionalidades descritas anteriormente.

Capítulo 3: Diseño del Sistema.

3.1 Introducción.

Este capítulo muestra un análisis de la Arquitectura Modular del SIGESC pues constituye la base para la creación de los diagramas de clases del diseño, utilizando los patrones establecidos, que se emplearán en la construcción del sistema. Las clases se encuentran agrupadas en paquetes que se formaron acorde a los casos de uso formados en el capítulo anterior.

3.2 Arquitectura Modular del SIGESC.

El SIGESC se encuentra dividido en diferentes módulos, cada uno de ellos presenta una arquitectura multicapa, específicamente de tres capas. La lógica de agrupación por capas definidas se realizó por las funcionalidades de: Presentación, Negocio y Acceso a Datos como muestra la Fig. 3.1

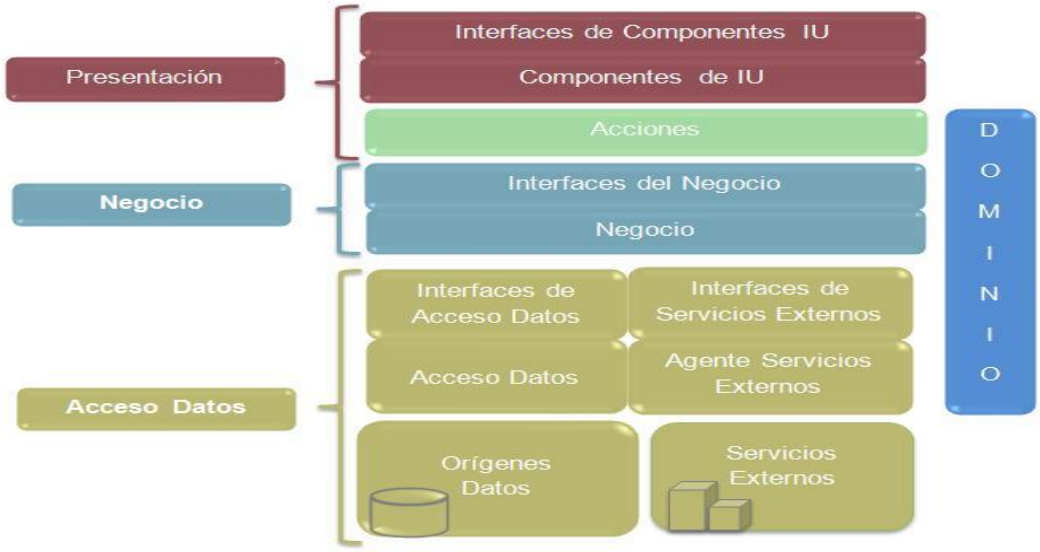

Fig.3.1 Arquitectura de un Módulo.[34]

La capa de Presentación se encuentra dividida en dos grupos de funcionalidades: las "Interfaces de Usuario (IU)" y las "Acciones".

- Las "Interfaces de Usuario" presentan información a los usuarios y acepta entradas o respuestas del usuario para usarlas en el sistema. Estas no desarrollan ningún procesamiento de negocio o reglas de validación de negocios.

[34] Adaptado de (Sánchez Tellez, 2008 pág. 24).

- Las "Acciones" representan peticiones que se le hacen al negocio. Estas peticiones pueden tener una IU asociada o no, las que tienen IU asociada se les denomina clase controladora de esa IU.

La capa de Negocio constituye el núcleo del módulo porque encapsula la lógica de implementación fundamental para la automatización de los procesos del negocio en cuestión. Su modelación requiere de especial atención por ambas partes: los desarrolladores y el cliente pues esta capa es determinante en el éxito del sistema.

La capa de Acceso Datos encapsula la lógica de implementación necesaria para gestionar los datos utilizados en uno o muchos procesos de negocio y abstraer así la forma en que los datos persisten o son obtenidos.
Aislar esta lógica en una capa independiente está dado por varias razones, dentro de ellas la especialización de las actividades en el proceso desarrollo para parte del equipo de desarrollo solo se ocupe de la lógica de acceso a datos, aumentando el rendimiento y la calidad del trabajo. El aislamiento de la tecnología de acceso a datos, mejora futuros mantenimientos del sistema y aumenta grandemente la cohesión de la capa de negocio que no tiene que incluir en su lógica implementaciones de acceso a datos. Desarrollo paralelo de las capas.

Dominio de un Módulo

Cada módulo del SIGESC requiere del paso de datos entre las capas. Estos datos suelen encapsularse en entidades informativas que no son más que clases, siendo su papel fundamental la información que almacena y no el comportamiento de esta. El conjunto de todas las clases entidades es denominado Dominio. [35]

Una vez definida la arquitectura surge la interrogante de ¿Cómo lograr la integración entre capas?

Los mecanismos que se utilicen para la integración entre capas tienen que tener como objetivo principal el de resolver de una forma u otra un conjunto de problemáticas:

[35]El acápite de Arquitectura fue Tomado y Adaptado de (Sánchez Tellez, 2008 pp. 22-24).

- Lograr que las capas queden totalmente desacopladas con el objetivo de mejorar la capacidad de mantenimiento de la aplicación, facilitar el desarrollo paralelo y lograr una mayor cohesión de las funcionalidades de la capa.
- Que el desacople no provoque problemas de rendimiento.
- No provocar dependencia a otras tecnologías.
- Facilidad de uso e implementación. Si la integración entre capas requiere un esfuerzo significativo puede provocar atrasos y la posibilidad de cometer errores aumentaría considerablemente. (Sánchez Tellez, 2008 p. 24)

La solución que propone (Sánchez Tellez, 2008) para el SIGESC es mediante el uso de varios patrones de diseño.

3.3 Patrones de diseño.

"Los patrones de diseño son soluciones de diseño a los problemas recurrentes en la construcción de software. Son a menudo mal interpretados como aplicables sólo a la programación en los grandes sistemas, pero en realidad, se pueden aplicar a la solución de problemas en la programación pequeña como en la implementación de estructuras de datos o algoritmos simples. Los patrones de diseño se pueden combinar en los componentes que resuelven grandes problemas."[36]

En (Larman, 2003) *se propone el uso de dos grupos de patrones de diseño, los patrones GRASP y los GOF.*

Los patrones GRASP (General Responsibility Assignment Patterns, por sus siglas en inglés) *"describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones"*[37], lo que constituye una de las tareas fundamentales en el diseño de un software. También se les conoce como buenas prácticas de programación, y muchas veces son utilizados sin que el diseñador o programador tenga conciencia de ello. Los patrones GRASP que se utilizan son: Experto, Creador, Controlador, Bajo Acoplamiento, Alta Cohesión.

Los patrones GoF (Gang of Four en inglés, Pandilla de los 4), se agrupan en tres categorías atendiendo las funciones que realizan: "Creacionales: Abarcan los procesos de

[36] (Kaisler, 2005 p. 12)

[37] (Larman, 2003 p. 163)

creación de objetos. Estructurales: Tratan con la composición de las clases y objetos. De comportamiento: Caracterizan el modo en que las clases u objetos interactúan y distribuyen responsabilidades."[38]

En el diseño del Administración del SIGESC fueron utilizados los siguientes patrones GOF:

Patrones de Comportamiento:

- **Comando** (Comand): las clases de la capa de acciones son un ejemplo del uso de este patrón.

Patrones Estructurales:

- **Fachada** (Facade): Cada capa mostrará una interfaz de acople o fachada con el conjunto de funcionalidades que necesite la capa inmediata superior.
- **Puente** (Bridge): El puente brinda un mecanismo dinámico, utilizando reflexión, para obtener objetos. Desacopla una abstracción de su implementación lo que permite modificarlas independientemente.
- **Proxy** (Proxy): El proxy (proxy remoto) es el encargado de abstraer la forma en que se obtiene la fábrica concreta utilizando un "Puente".

Patrones Creacionales:

- **Fábrica Abstracta** (Abstract Factory): En la fachada se definirán fábricas abstractas, por familias de objetos, con el objetivo de definir (no implementar) como se van a agrupar los objetos de dicha capa para ser creados por una fábrica concreta que forma parte de la capa en cuestión.[39]

También se utilizó el patrón para el acceso a datos, conocido como DAO.

DAO (Data Access Object**,** Objeto de Acceso a Datos): Es una solución al problema del diferencial de impedancia entre un programa de aplicación orientado a objetos y una base de datos relacional, abstrayendo y encapsulando todos los accesos a la fuente de datos. Usando DAO significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación.[40]

---

[38] (Gamma, et al., 1994 p. Introducción)

[39]Tomado de (Sánchez Tellez, 2008 p. 25).

[40] Tomado de (Sánchez Tellez, 2008 p. 69).

3.4 Diagrama de paquetes del diseño.

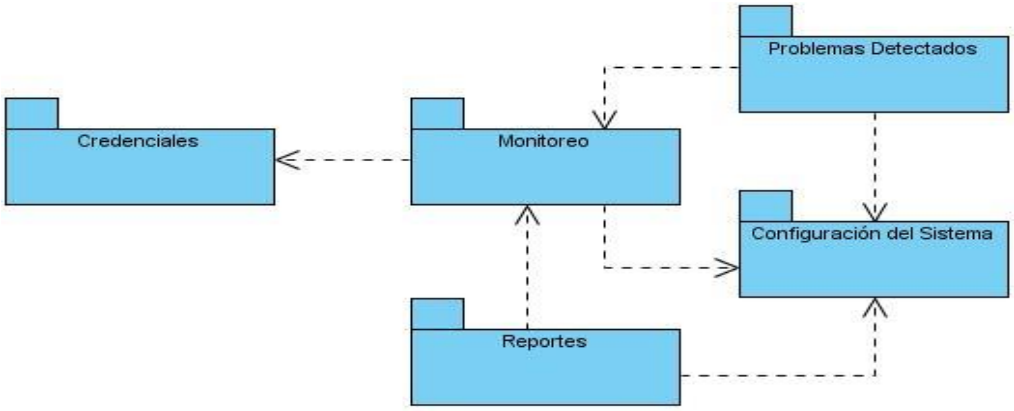

Fig.3.1 Diagrama de paquetes del diseño.

3.5 Diagramas de clases del diseño.

Un *diagrama de* clases es *"un diagrama que describe un conjunto de objetos que comparten las mismas responsabilidades, relaciones, operaciones, atributos y semánticas".*[41]

Contienen normalmente los siguientes elementos:

- Clases, asociaciones y atributos.
- Interfaces y operaciones.
- Colaboraciones.
- Relaciones de dependencia, generalización y asociación.
- Métodos.
- Navegabilidad.

[41] (Rational Software Corporation., 2003)

A continuación se muestran los diagramas de clases correspondientes a cada paquete:

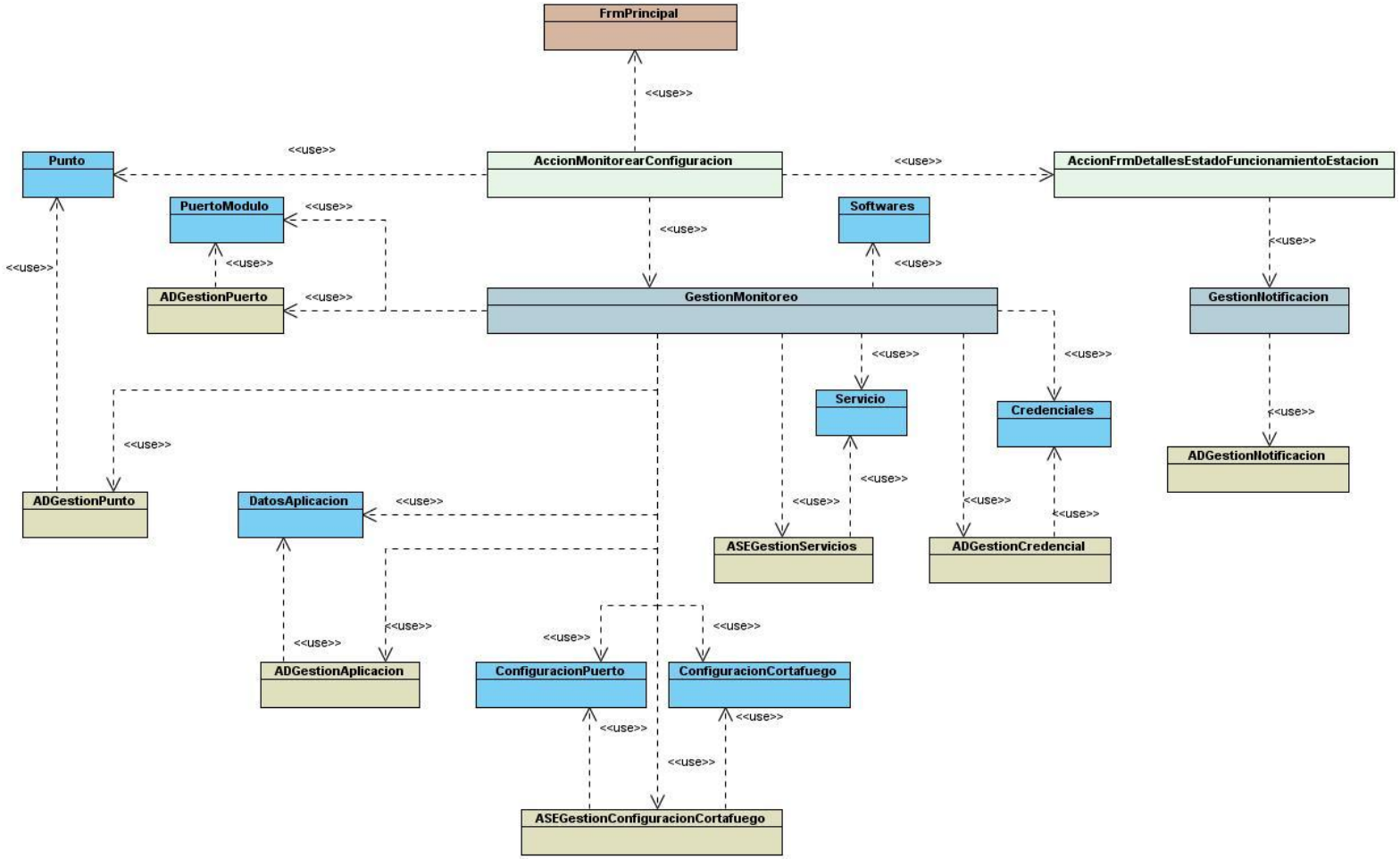

Fig.3.2 Diagrama de clases del diseño del paquete Monitoreo.

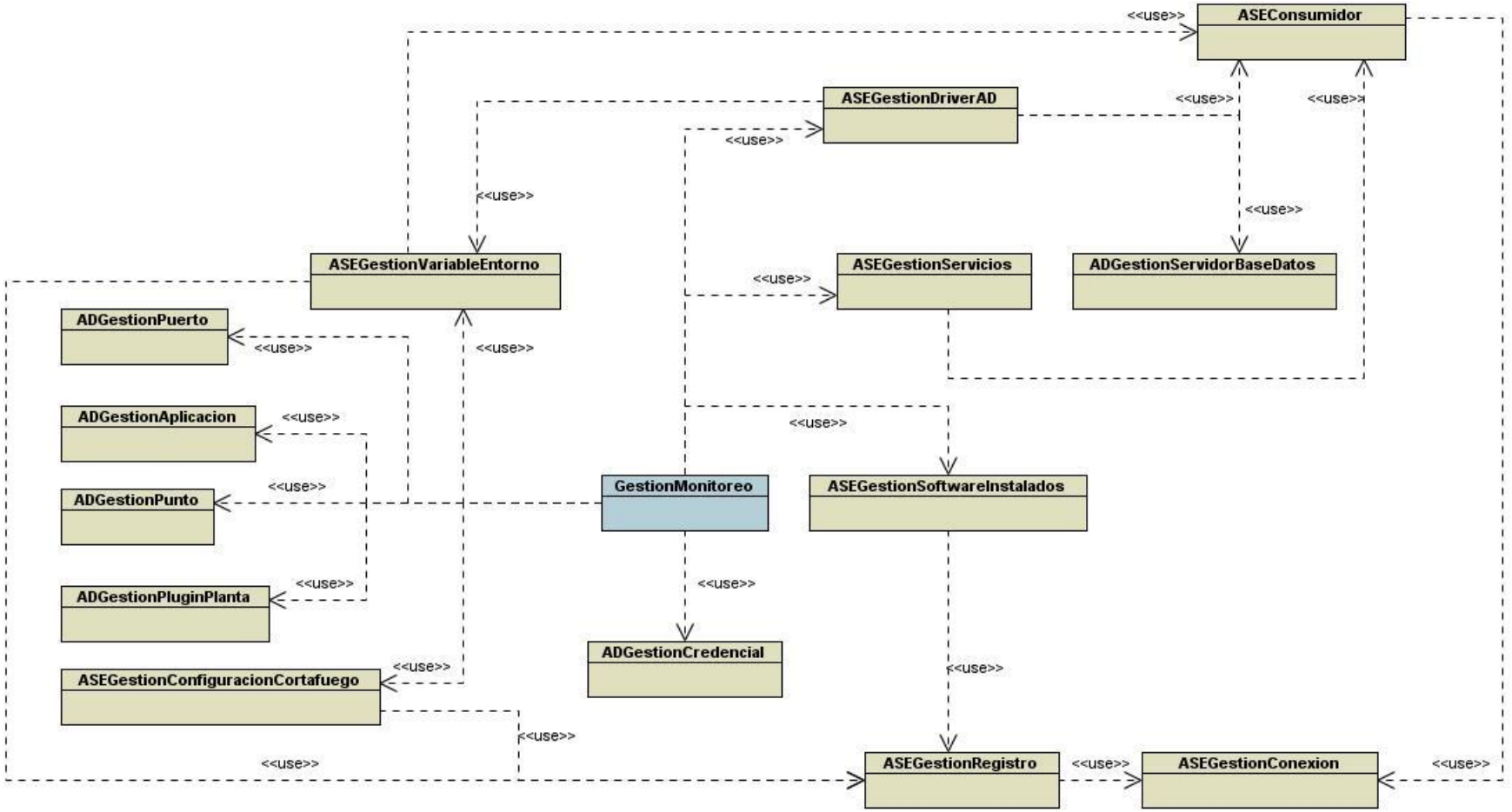

Fig.3.3 Diagrama de clases del diseño del paquete Monitoreo.

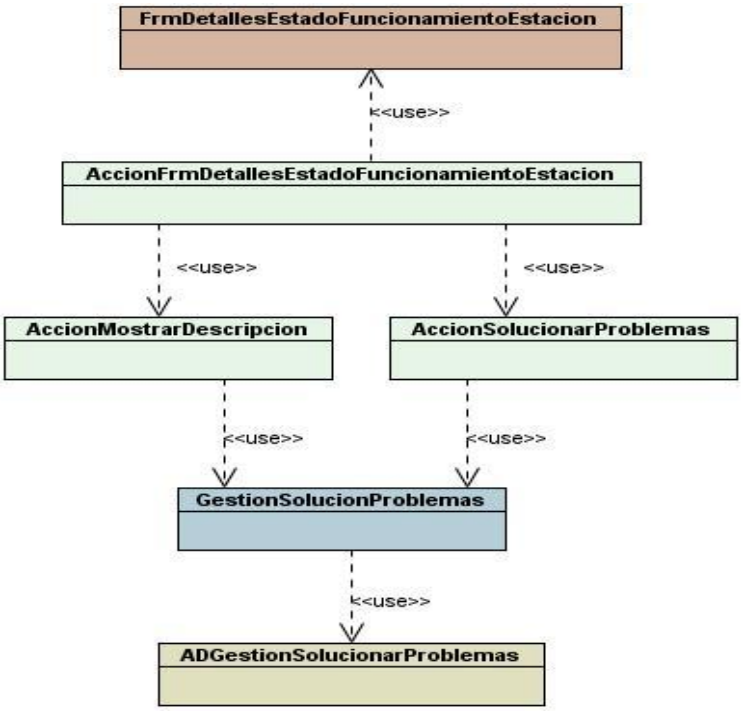

Fig.3.4 Diagrama de clases del diseño del paquete Problemas Detectados.

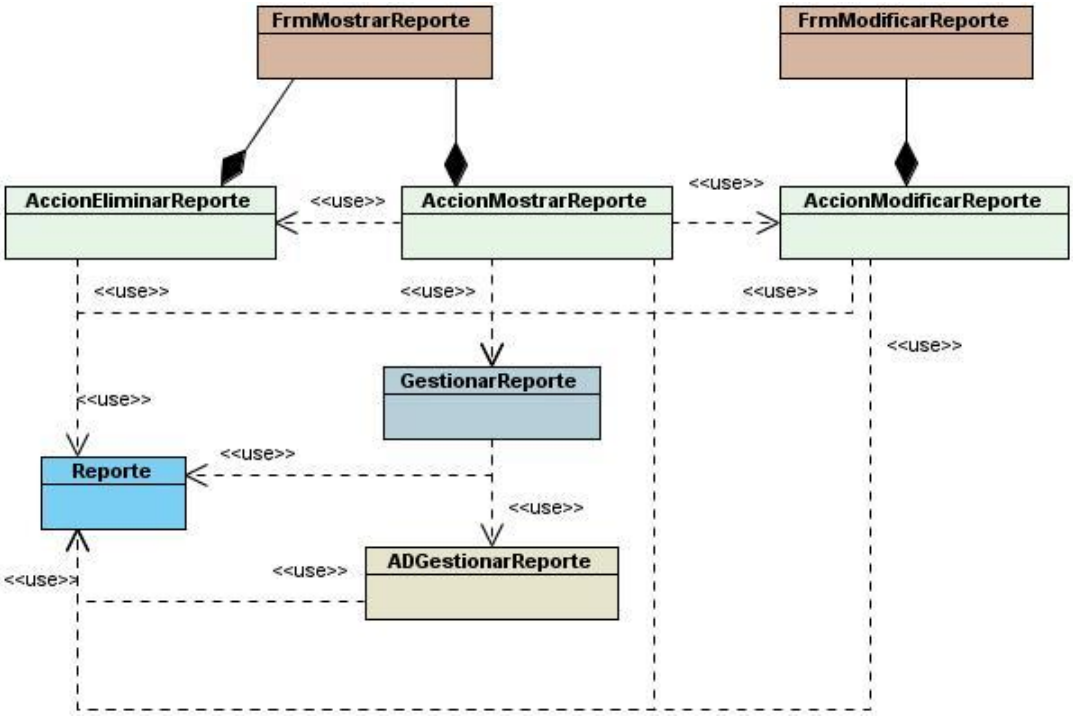

Fig.3.5 Diagrama de clases del diseño del paquete Reportes.

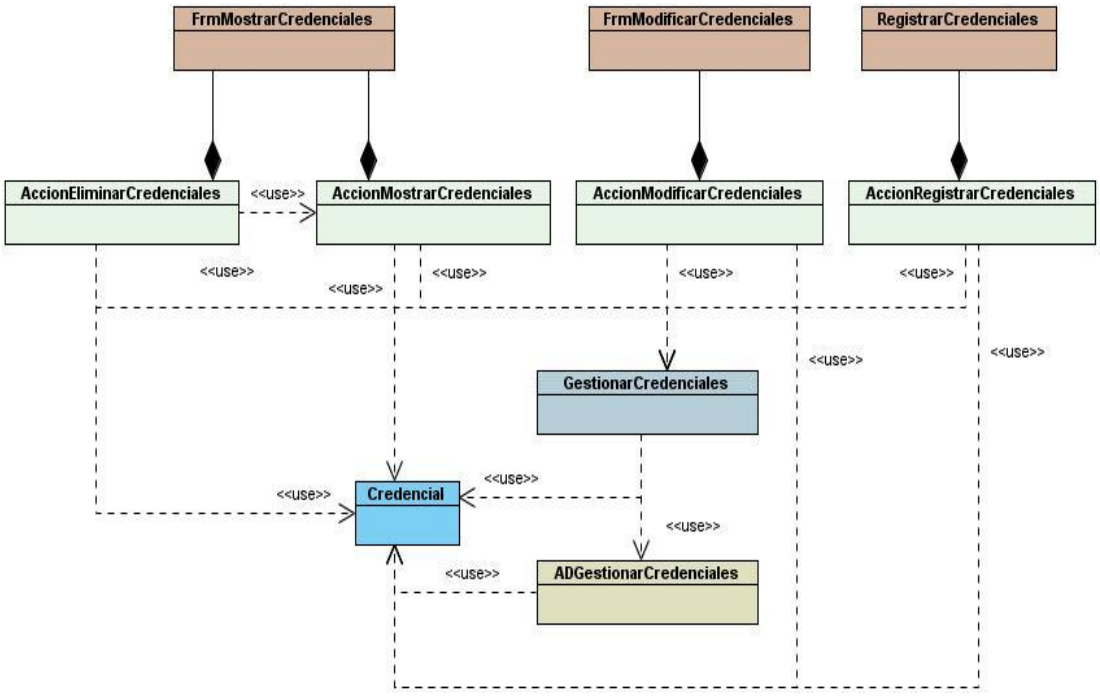

Fig.3.6 Diagrama de clases del diseño del paquete Credenciales.

3.6 Diagrama Entidad-Relación.

El Diagrama Entidad-Relación (DER) *"permite que un ingeniero del software identifique objetos de datos y sus relaciones mediante una notación grafica. En el contexto del análisis estructurado, el DER define todos los datos que se introducen, se almacenan, se transforman y se producen dentro de una aplicación."*[42]

Para la correcta implementación de la aplicación es necesario agregar entidades a la BD que respondan a las necesidades de las funcionalidades propuestas. La entidad de color verde ya está en la BD pero para un mejor entendimiento es necesario incluirla en el diagrama porque se relaciona con las nuevas entidades.

[42] (Pressman, 2001 p. 201)

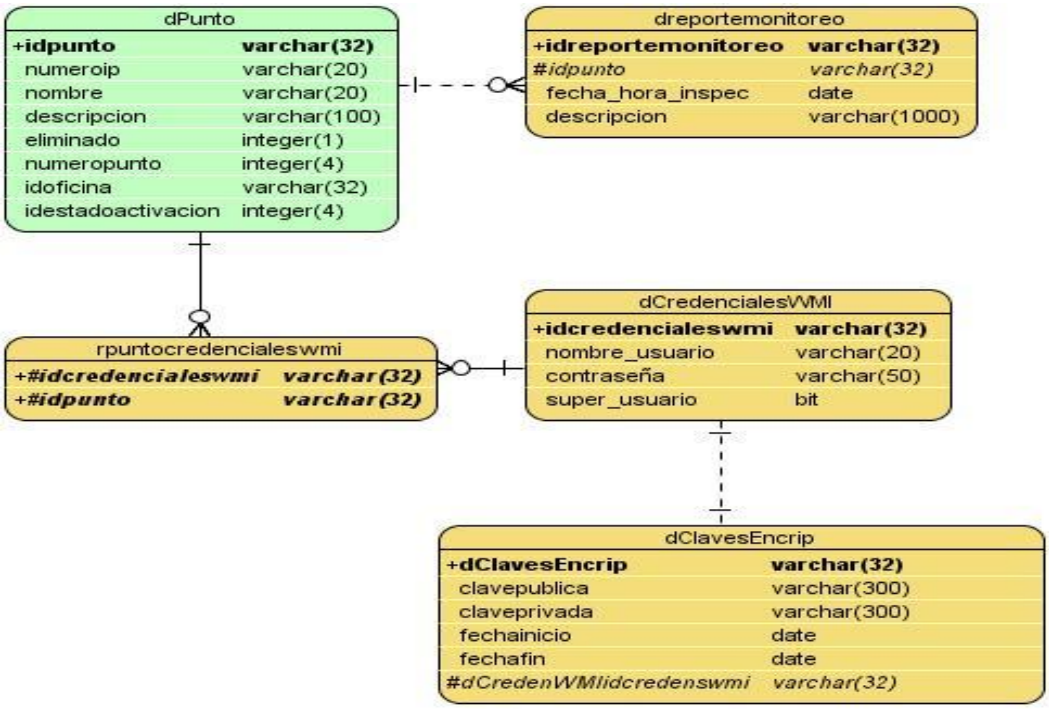

Fig.3.7 Diagrama Entidad-Relación.

3.7 Conclusiones.

En este capítulo se representa el diagrama de paquetes del diseño, los diagramas de clases del diseño correspondiente a cada paquete y se identifican los principales patrones de diseño utilizados. También se representa el Modelo de Datos. Todos estos elementos constituyen la base para la fase de implementación. Las clases expandidas del diseño se encuentran a partir del Anexo 40.

Capítulo 4: Implementación del Sistema.

4.1 Introducción.

En el presente capítulo se muestra el diagrama de implementación de la aplicación Administración del SIGESC y se brinda una descripción de cada uno de los componentes que en él se interrelacionan.

4.2 Modelo de Implementación.

"El modelo de implementación describe cómo los elementos del modelo del diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de implementación describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros."[43]

4.2.1 Diagrama de Componentes.

"Un diagrama de componentes muestra las organizaciones y dependencias entre componentes."[44]

Dicho diagrama se utiliza en la representación de las dependencias de compilación entre ficheros de código fuente, las dependencias en tiempo de ejecución entre ficheros de aplicación y las relaciones entre elementos de implementación y elementos del diseño.

A continuación se muestra el diagrama de componentes:

[43] (Jacobson, et al., 2000 p. 257)

[44] (Rational Software Corporation., 2003 p. Glosario)

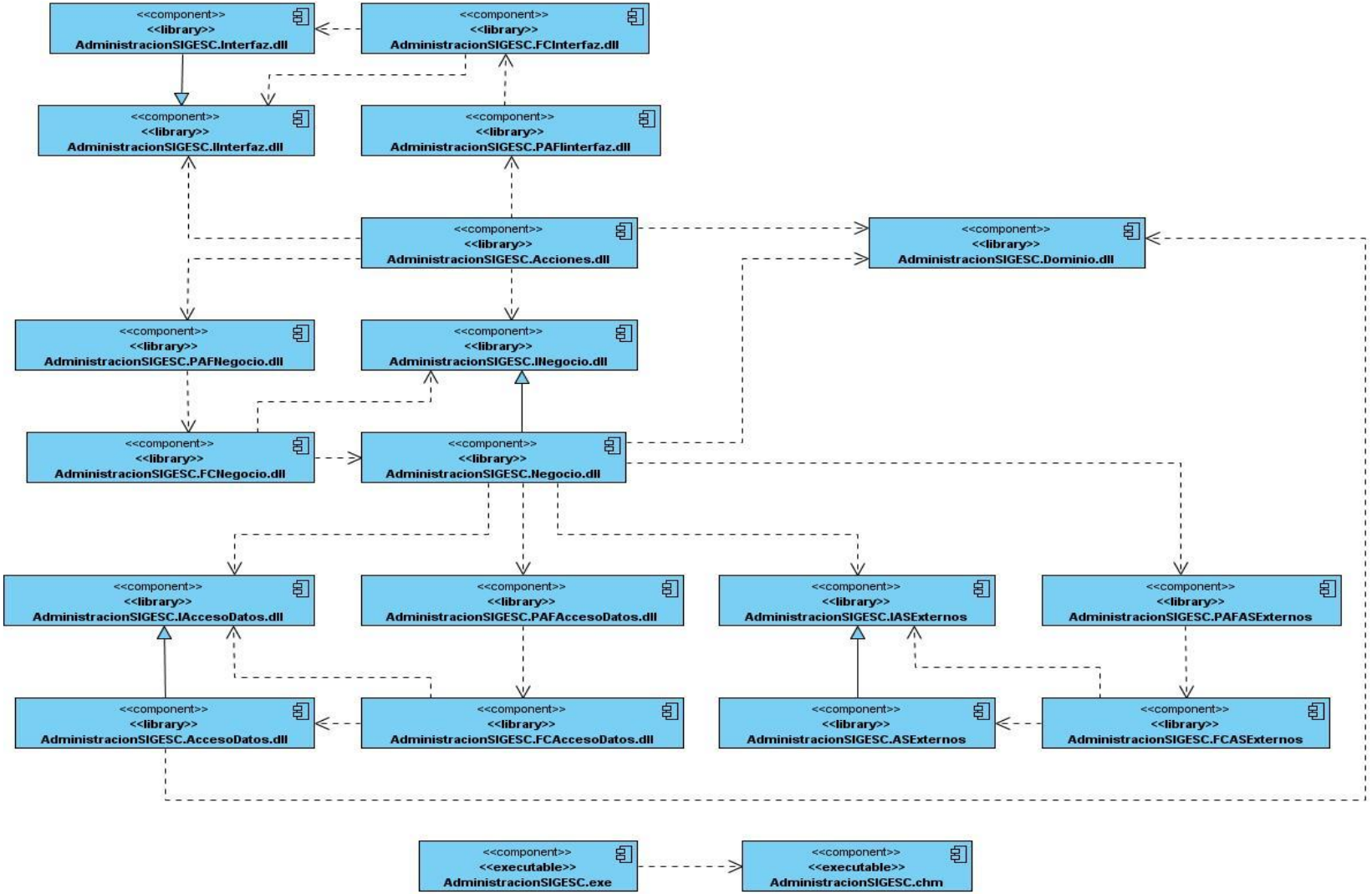

Fig.4.1 Diagrama de Componentes.

4.3 Descripción de los componentes.

La descripción expandida de los componentes se encuentra en los anexos de 20 al 39.

4.3.1 Componente AdministracionSIGESC.exe

Propósito:

Iniciar la llamada a la acción que da comienzo a la ejecución de la aplicación. Contiene incrustados los ficheros donde se definen todas las configuraciones generales de la aplicación, el menú y la seguridad.

Recursos

- **Sistema.xml**: Fichero XML, que almacena la configuración general del sistema, de la aplicación. Se especifica el nombre de la aplicación, el tipo de aplicación, la cultura de trabajo y el vínculo al fichero de ayuda. Además de la existencia de acciones activas, los ensamblados de las acciones y de las fábricas concretas utilizadas. También contiene las acciones iniciales que se activan al ejecutar la aplicación y los comandos o teclas calientes, con la acción que ejecutan respectivamente.
- **MenuPrincipal.xml**: Fichero XML que almacena la configuración de menú de la aplicación. Incluye las acciones a ejecutar en cada una de las opciones.
- **Seguridad.xml**: Fichero XML, que almacena la configuración de la aplicación relacionada con la seguridad. Incluye restricciones del sistema y restricciones de usuario. En el caso de las restricciones de usuario, existen clasificaciones para las acciones. En el caso de la configuración de la aplicación Administración del SIGESC solo existe un rol, el cual tiene permiso a ejecutar todas las acciones.

4.3.2 Componente AdministracionSIGESC.Acciones.dll

Propósito

Contiene las clases que realizan acciones para generar una operación determinada. Estas clases están subscritas a los eventos que se puedan desatar en la interfaz de usuario e invocan a los métodos correspondientes de las clases de negocio para llevar a cabo la acción solicitada.

4.3.3 Componente AdministracionSIGESC.Dominio.dll

Propósito

Contiene las clases entidades del módulo. Estas clases representan a los objetos persistentes y pueden ser usadas verticalmente en todas las capas de la aplicación.

4.3.4 Componente AdministracionSIGESC.FCNegocio.dll

Propósito

Contiene las clases que construyen y devuelven los objetos de la capa de Negocio.

4.3.5 Componente AdministracionSIGESC.INegocio.dll

Propósito

Contiene las clases que definen el comportamiento de los objetos de la capa de Negocio.

4.3.6 Componente AdministracionSIGESC.Negocio.dll

Propósito

Contiene las clases que controlan el negocio de la aplicación. En estas clases se definen todos los algoritmos o pasos necesarios para obtener un resultado satisfactorio a partir de la petición realizada.

4.3.7 Componente AdministracionSIGESC.PAFNegocio.dll

Propósito

Contiene las clases fábricas abstractas de negocio y proxy de negocio. Los proxy se encargan de abstraerse de la obtención de las fábrica concreta de los objetos negocio. Las fábricas abstractas proveen un comportamiento para la creación de familias de objetos interdependientes o interrelacionados, sin especificar sus clases concretas.

4.4 Conclusiones.

En este capítulo se representó el modelo de implementación mediante el diagrama componentes y se describieron además cada uno de los componentes que lo integran. Proporcionando una visión final del sistema como una aplicación ejecutable desarrollada a partir de las especificaciones del Capítulo de Diseño.

Conclusiones Generales.

El Trabajo de Diploma: *"Sistema de Gestión de Emergencia de Seguridad Ciudadana 171.* Aplicación *Administración del SIGESC"*, se estructuró en cuatro capítulos para darle cumplimiento al objetivo general del trabajo que consiste en: *Desarrollar una aplicación que permita supervisar y administrar centralmente el estado de la configuración y ejecución del SIGESC.*

En el Capítulo 1: "Fundamentación Teórica" se realizó un estudio del arte de las tendencias actuales hacia los sistemas distribuidos administrados, se analizaron otros conceptos importantes que fueron necesarios para la concepción del sistema y se realizó una evaluación de la metodología de desarrollo y herramientas a utilizar para la realización de la aplicación propuesta.

En el Capítulo 2: "Características del Sistema" se puntualiza en los procesos que serán automatizados, además se aborda los aspectos esenciales del negocio y los requisitos funcionales y no funcionalesa tener en cuenta para el desarrollo del sistema.Se muestran los diagramas de los casos de uso agrupados en paquetes y se detallan sus funcionalidades, quedando así definidos una serie de elementos de gran importancia para de desarrollo del sistema.

En el Capítulo 3: "Diseño del Sistema"se representó el diagrama de paquetes del diseño, los diagramas de clases del diseño correspondiente a cada paquete y se identificaron los principales patrones de diseño utilizados, constituyendo así, la base para la fase de implementación.

En el Capítulo 4: "Implementación" se representó el modelo de implementación mediante el diagrama componentes y se describieron además cada uno de los componentes que lo integran. Proporcionando una visión final del sistema como una aplicación ejecutable desarrollada a partir de las especificaciones del Capítulo de Diseño.

En síntesis, se cumplió el objetivo general del Trabajo de Diploma realizando el diseño de la aplicación Administración del SIGESC, fundamentado en los requisitos funcionales y no funcionales identificados, proporcionando una visión que dio paso a la implementación del mismo y convertir al SIGESC *en un sistema distribuido con características limitadas de administración.*

Recomendaciones.

Se recomienda:

- Implementar la segunda iteración de la aplicación Administración del SIGESC.
- Valorar la Integración de la aplicación con un Directorio de Dominio para obtener las credenciales con que se establecería la conexión a las estaciones a monitorear.
- Extender el monitoreo hacia los servidores de Base de Datos y Aplicación Web.

Glosario de Términos.

Abreviaturas	Significado
AES	Estándar de cifrado avanzado o Rijndael.
ADO.NET	Es un estándar de acceso a BD.
ANSI SQL	National Standards Institute Structured Query Language.
BD	Base de Datos.
C#	Lenguaje de programación.
CAL	Licencia de Acceso para Cliente de Microsoft Windows.
Capa	Una capa es una agrupación lógica de un conjunto de componentes acoplados entre sí donde cada componente es un conjunto funcionalidades bien definidas.
CID	Configuration, Installation and Distribution.
CLR	Lenguaje Común en Tiempo de Ejecución.
COBIT	Objetivos de Control para la Información y la Tecnología relacionada.
CU	Caso de Uso
DER	Diagrama Entidad-Relación.
DES	Estándar de cifrado de datos.
DLL	Biblioteca de vínculos dinámicos (Dynamic-Link Library).
DMAX	Desarrollo Modular Basado en una Arquitectura Extensible.
DSA	Algoritmo de Firma digital.
DSI	Iniciativa de Diseño Dinámico.
ECMA	European Computer Manufacturers Association. Organización internacional basada en membrecías de estándares para la comunicación y la información.
GRASP	Patrones Generales de Asignación de Responsabilidades.
IBM	International Business Machines.
IDE	Entorno de Desarrollo Integrado
IDEF	Integration Definition Function.
IDEF0	Integrated Definition for Function Modeling.
IEC	Comisión Electrotécnica Internacional.
ISACA	Asociación para la Auditoría y Control de Sistemas de Información.

ISO	Organización Internacional de Normalización.
ITGI	IT Governance Institute.
ITIL	Biblioteca de Infraestructura de Tecnologías de la Información.
IU	Interfaz de Usuario.
MD5	Algoritmo de Resumen del Mensaje.
Módulo	Un módulo es un conjunto de funcionalidades acopladas convenientemente para realizar una lógica de negocio determinada.
MOF	Marco de Trabajo de Microsoft para Operaciones.
MSIL	Lenguaje Intermedio Microsoft.
PL/SQL	Lenguaje Procedimental/Query.
POO	Programación orientada a objetos.
RC2	Rivest Cipher 2.
RSA	Rivest, Shamir y Adleman.
RUP	Proceso Unificado de Desarrollo.
SGBD	Sistema Gestor de Base de Datos.
SGBDR	Sistema Gestor de Base de Datos Relacional.
SHA	Algoritmo de *Hash* Seguro.
SIGESC.	Sistema de Gestión de Emergencias de Seguridad Ciudadana.
SO	Sistema Operativo.
SP	Service Pack.
SVN	Subversion es un sistema de control de versiones de software libre bajo licencia de tipo Apache/BSD y se le conoce también como svn.
TCO	Costo Total de propiedad.
TDES	Triple DES.
TI	Tecnologías de Información.
TNS	Sustrato de Red Transparente. Protocolo de seguridad.
UML	Lenguaje de Modelado Unificado.
WMI	Windows Management Instrumentation. (Instrumental de Administración de Windows).
WQL	Lenguaje de Consulta WMI.
XML	Lenguaje de Marcas Extensible (eXtensible Markup Language).

Referencias Bibliográficas.

1. **Colouris, George. 2001.** *Sistemas Distribuidos: Conceptos y Diseño.* [ed.] Addison Wesley. 3ra edición. 2001. Capítulo 1, pág 2.
2. **ALBET Ingeniería y Sistemas. 2009**. Descripción General del SIGESC. Ciudad Habana: s.n., 2009.
3. **ALBET Ingeniería y Sistemas. 2007.** *Manual de Instalación.* Ciudad Habana. : s.n., 2007. CT-SW-DR-031801.
4. **APM Group. 2007-2011.** http://www.itil-officialsite.com/. [En línea] 2007-2011. [Citado el: 14 de 01 de 2011.] APM Group. 2861902.
5. **ISACA. 2011.** ISACA.org. *The comprehensive IT governance framework that addresses every aspect of IT and integrates all of the main global IT standards.* [En línea] 2011. [Citado el: 15 de 01 de 2011.] http://www.isaca.org/Knowledge-Center/cobit/Documents/CobiT-4.1-Brochure.pdf.
6. **Microsoft Corporation. 2011.** Microsoft Download Center. MOF. [En línea] 2011. ppt. http://download.microsoft.com/download/9/5/e/95e24fbf-5a65-4bd4-af44-44049ad9bed0/MOF%20-%20Presentaci%F3n.ppt.
7. **Kaspersky Lab. 1997-2011.** Kaspersky Lab. [En línea] 1997-2011. [Citado el: 15 de 01 de 2011.] http://www.kaspersky.com/sp/administration_kit.
8. **Microsoft Corporation. 2010.** Microsoft System Center. [En línea] 2010. [Citado el: 16 de 01 de 2011.] http://www.microsoft.com/systemcenter/en/us/overview.aspx.
9. **IBM Corporation. 2001.** *Autonomic Computing: IBM's Perspective of State of Information Technology.* New York : IBM Corporation, 2001.
10. **David Chappell & Associates. 2008.** *What is Applications Lifecycle Management?* San Francisco, California. : s.n., 2008.
11. **Microsoft Corporation. 2003**. Design for Operations. Building Health, Task, and State Models. s.l. : Microsoft Corporation, 2003, p15.p18.
12. **Microsoft Corporation. 2008**. Design for Operations. Designing Manageable Applications. s.l. : Microsoft Corporation, 2008, pp. 1-3.
13. **Department of Defense. System Management College. 2001**. Systems Engineering Fundamentals. [ed.] Virginia. Acquisition University Press Fort Belvoir. Virginia: Department of Defense. System Management College. 2001. 22060-5565. p51.

14. **Microsoft Corporation. 2009.** Microsoft .NET. [En línea] 2009. [Citado el: 23 de 01 de 2011.] http://www.microsoft.com/net/overview.aspx.
15. **Microsoft Corporation;. 2008.** Información general de la biblioteca de clases de .NET Framework. [En línea] 2008. [Citado el: 25 de 04 de 2011.] http://msdn.microsoft.com/es-es/library/hfa3fa08%28VS.80%29.aspx.
16. **Microsoft Corporation. 2011.** Microsoft .NET Framework SDK versión 1.1. [En línea] 2011. [Citado el: 25 de 04 de 2011.] http://www.microsoft.com/downloads/es-es/details.aspx?familyid=9B3A2CA6-3647-4070-9F41-A333C6B9181D&displaylang=es.
17. **Microsoft Corporation. 2011.** Kit de desarrollo de software de Microsoft Windows. [En línea] 2011. [Citado el: 25 de 04 de 2011.] http://msdn.microsoft.com/es-es/windows/bb980924.aspx.
18. **Microsoft Corporation. 2006.** Arquitectura de WMI .NET. [En línea] Microsoft Corporation, 2006. [Citado el: 2011 de 01 de 24.] http://msdn.microsoft.com/es-es/library/ms257361%28v=VS.80%29.aspx.
19. **Sánchez Tellez, Eddy . 2008.** *Especificación de la Arquitectura Base del Sistema de Gestión de Emergencia y Seguridad Ciudadana 171.* Ciudad Habana : Universidad de las Ciencias Informáticas, 2008. p 22-25. p 69.
20. **Greenwald, Rick, Stackowiak, Robert y Stern, Jonathan**. Oracle Essentials. Oracle Database 10g.
21. **Oracle Corporation. 1999.** Net8 Administrator's Guide. [En línea] 1999. [Citado el: 20 de 01 de 2011.] http://www.cs.umbc.edu/portal/help/oracle8/network.815/a67440/ch2.htm.
22. **Sommerville, Ian. 2005**. Ingeniería del Software. [ed.] Miguel Martín-Romo. 7ma. s.l.: PEARSON EDUCACION S.A., 2005, pp110.
23. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo del Software.* [trad.] Salvador Sánchez, y otros. Español. Madrid : Addison Wesley, 2000. ISBN: 84-7829-036-2. p.110, p257.
24. **Universidad de las Ciencias Informáticas. 2010.** Fundamentos Básicos de la Criptografía. *Entorno Virtual de Aprendizaje.* [En línea] 2010. [Citado el: 28 de 05 de 2011.] http://eva.uci.cu/mod/resource/view.php?inpopup=true&id=38433.
25. **Pressman, Roger S. 2001**. Ingeniería del Software. Un enfoque Práctico. [trad.] Darrel Ince. 5ta Edición. s.l.: Mc Graw Hill, 2001, pp187.

26. **Kaisler, Stephen H. 2005.** *Software Paradigms.* New Jersey : Wiley - Interscience, 2005. ISBN 0-471-48347-8. p.12.
27. **Larman, Craig. 2003.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2da Edición en Español. México : Prentice Hall, 2003. pág. 520. ISBN 8420534382. p.163.
28. **Gamma, Erich, y otros. 1994.** *Design Patterns-Elements of Reusable Object-Oriented Software.* s.l. : Addison Wesley Longman, 1994. p(Introducción).
29. **Rational Software Corporation. 2003**. Glosario del Rational Unified Process. 2003. 2003.06.00.65.

Bibliografía.

30. **AjpdSoft, Proyecto. 2011.** Proyecto AjpdSoft. [En línea] 2011. [Citado el: 25 de 05 de 2011.] procedure IPServerBaseDato(resultado out ref_cursor).
31. **lonso Riverón, Yisel, Cruz Navarro, Yaneisy y Tornés Medina, Yordanis. 2008.** *IDEF Una alternativa para modelamiento de negocio con RUP.* Universidad de las Ciencias Informáticas. Ciudad Habana. Cuba : s.n., 2008. pág. 21, Documento Word.
32. **Cabrera Díaz, Manfred Ramón. 2010.** *Arquitectura y Aplicaciones del SIGESC. Introducción a Microsoft .NET.* [Power Point] Ciudad Habana. : s.n., 2010.
33. **Hernández León, Rolando Alfredo y Coello González, Sayda. 2011.** *El Proceso de Investigación Científica.* Hernández León, Rolando Alfredo. Ciudad de La Habana : Editorial Universitaria del Ministerio de Educación Superior, 2011. pág. 110. 978-959-16-1307-3..
34. **Maccas, Angel Miguel.** *Diseño del Software.*
35. **Marquès i Puig, Joan Manuel, Vilajosana i Guillén, Xavier y García López, Pedro A. 2007.** *Arquitecturas, paradigmas y aplicaciones de los sistemas distribuidos.* 2007.
36. **Microsoft Corporation. 2006.** Centro de desarrolladores de Visual C#. [En línea] [Citado el: 24 de 01 de 2011.] http://msdn.microsoft.com/es-es/vcsharp/default.aspx.
37. **Microsoft Corporation. 2006.** Información general sobre WMI .NET. [En línea] Microsoft Corporation, 2006. [Citado el: 24 de 01 de 2011.] http://msdn.microsoft.com/es-es/library/ms257340%28VS.80%29.aspx.
38. **Microsoft Corporation. 2011.** Querying with WQL. [En línea] 2011. [Citado el: 19 de 01 de 2011.] http://msdn.microsoft.com/en-us/library/aa392902.aspx.
39. **Microsoft Corporation. 2011.** Suscriptores de Oracle. [En línea] 2011. [Citado el: 19 de 01 de 2011.] http://msdn.microsoft.com/es-es/library/ms151738%28v=sql.100%29.aspx.
40. **Sánchez Téllez, Eddy.** *DMAX.* [Power Point] Ciudad Habana. : UCI.
41. **Sanchez, María A. Mendoza. 2004.** *Metodologías de desarrollo de software.* 2004.
42. **Visual Paradigm International, Ltd.** Visual Paradigm for UML. [En línea] [Citado el: 22 de 01 de 2011.]
43. **Visual Paradigm International, Ltd. 2010.** *VPM-UML Quick Start.* s.l. : Visual Paradigm International., 2010.