Universidad de las Ciencias Informáticas

Facultad 2

Título: Módulo de control de procesos del Sistema Integral de Análisis de Información.

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autor: Arián Prieto Pack

Tutores: Ing. Lilian Saúco Altuna
Ing. Erik Machado Cano

Co-tutor: MSc. Jorge Luis Olmedo Flores

"Año 54 de la Revolución".

Pensamiento

"Las dificultades son las que enseñan y engrandecen (...)".

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

"[Insertar nombre(s) de autor(es)]"

_____

"[Insertar nombre(s) de tutor(es)]"

_____

DATOS DE CONTACTO

Ing. Lilian Saúco Altuna

Correo: lsauco@uci.cu

Ingeniera en Ciencias Informáticas, graduada en la Universidad de Ciencias Informáticas en el año 2008. Trabaja en la Facultad 2 de dicho centro de enseñanza superior con categoría docente de Instructor. Pertenece al Centro de Telemática, específicamente al proyecto Sistema Integral de Análisis de Información con el rol de líder.

Ing. Erik Machado Cano

Correo: emachado@uci.cu

Ingeniero en Ciencias Informáticas, graduado en la Universidad de Ciencias Informáticas en el año 2008. Trabaja en la Facultad 2 de dicho centro de enseñanza superior con categoría docente de Instructor. Pertenece al Centro de Telemática, específicamente al proyecto Sistema Integral de Análisis de Información con el rol de desarrollador.

RESUMEN

En el presente trabajo se presenta una herramienta informática que permite iniciar y finalizar el **Sistema Integral de Análisis de Información** (SIAI) teniendo en cuenta las relaciones de dependencias que puedan existir entre los procesos informáticos referentes a los diferentes servicios asociados al SIAI de forma general, permitiendo además consultar el estado real de cada proceso involucrado así como sus principales propiedades. Básicamente el trabajo está dividido en 4 capítulos que abarcan desde la investigación realizada sobre los principales sistemas informáticos desarrollados con propósitos similares, hasta la implementación y prueba de la solución propuesta; la cual está basada en la arquitectura Cliente – Servidor, dividiendo lógicamente la solución propuesta en dos sistemas esenciales: una aplicación agente y una gestora. El sistema agente en esencia es un proceso que se ejecuta en segundo plano destinado a obtener la información necesaria acerca de procesos especificados previamente, permitiendo además la realización de las operaciones básicas de control de procesos informáticos (“Iniciar”, “Detener”, “Continuar”, “Finalizar” y “Matar”); y el sistema gestor es una aplicación de escritorio que se encarga de comunicarse y administrar todos los sistemas agentes que sean utilizados con el objetivo de obtener la información gestionada por estos para que pueda ser presentada finalmente al administrador del SIAI. Además es importante señalar que el sistema en su totalidad está desarrollado para ejecutarse en estaciones de trabajo GNU/Linux.

PALABRAS CLAVES

Control de procesos, procesos informáticos, sistema distribuido, Sistema Integral de Análisis de Información (SIAI).

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

La historia de las telecomunicaciones comenzó a desarrollarse en la primera mitad del siglo XIX, con el telégrafo eléctrico (que permitía enviar mensajes con letras y números). Más adelante aparecieron otros avances mucho más significativos como el teléfono (que brindó la posibilidad de comunicarse a distancia utilizando la voz), las radio (que permitieron la comunicación a través de ondas inalámbricas) y principalmente los ordenadores. Pero las innovaciones tecnológicas en este campo nunca se detuvieron, ya que conjuntamente con el desarrollo de las tecnologías se hizo sumamente necesario lograr interconexión entre todos estos dispositivos.

Actualmente existen múltiples medios y vías para comunicarse que continúan en constante evolución, donde la Infraestructura de telecomunicaciones conforma un sector industrial de elevado desarrollo tecnológico y a su vez juega un papel sumamente importante económicamente a nivel mundial. Dentro de los servicios de telecomunicaciones, principalmente los de telefonía fija y móvil, poseen un elevado nivel de adquisición y utilización en la sociedad, de forma tal que mientras más evoluciona la tecnología y la accesibilidad a la misma por parte de los clientes, se incrementan las oportunidades para llevar a cabo acciones fraudulentas, que en esencia no son más que "(…) el uso o adquisición de los productos y/o servicios de telecomunicaciones a través de mecanismos o medios ilegales y sin la intención de pagar por ellos." (1)

Para combatir estas actividades ilegales existen sistemas informáticos de gestión de fraudes (FMS[1]). Estos están compuestos por varias aplicaciones, módulos o subsistemas internos, que durante su funcionamiento deben ejecutarse simultáneamente, por lo que se relacionan entre ellos como procesos informáticos, aunque cada uno tiene vida independiente y funciones específicas.

Cuba ha sufrido pérdidas millonarias por concepto de fraude, implantación, soporte y mantenimiento de los FMS comerciales que utiliza, los cuales son internacionales y en algunos puntos no satisfacen las necesidades reales del entorno cubano. Hoy el país avanza en cuanto al desarrollo de la industria del software, la cual está encaminada a mejorar su ubicación en el mercado a nivel mundial y a contribuir con la informatización de la sociedad cubana. La Universidad de las Ciencias Informáticas (UCI) es una entidad clave en este aspecto, ya que paulatinamente se ha ido convirtiendo en una potencial empresa desarrolladora de software. La misma posee una infraestructura productiva que se encuentra formada por diferentes centros de desarrollo. Uno de ellos es el centro de Telemática (TLM) perteneciente a la facultad

[1] Acrónimo inglés de **Fraud Management System**, traducido al español como Sistema de Gestión de Fraude.

2, el cual brinda servicios y desarrolla productos informáticos para prestigiosas empresas nacionales e internacionales en las ramas de las telecomunicaciones y la seguridad informática. Uno de los grupos desarrolladores perteneciente a este centro se encuentra desarrollando conjuntamente con la Empresa de Telecomunicaciones de Cuba S.A. (ETECSA), el Sistema Integral de Análisis de Información (SIAI): solución cubana de FMS que cumple con los requerimientos básicos para la gestión del fraude e incorpora requerimientos propios del entorno cubano.

Para el correcto funcionamiento del SIAI, encargado de recopilar, ordenar y mantener disponible para su consulta una enorme cantidad de información sobre el uso de los servicios de telecomunicaciones en Cuba, resulta necesario que el conjunto de procesos que se relacionan con el mismo se encuentren funcionando correctamente. Cuando se automatizan procesos de una empresa, generalmente se necesita conocer los estados de las aplicaciones involucradas, en especial aquellos que requieran la intervención humana para la toma de decisiones. Si para determinar el momento de esta intervención fuera necesario mantener personas a tiempo completo monitoreando el flujo de trabajo, la razón de ser y la efectividad del software serían fuertemente cuestionables. Actualmente los administradores del SIAI no cuentan con una herramienta para, de forma centralizada, conocer el estado de sus procesos, así como realizar las operaciones de arranque y parada (startup/shutdown) de los mismos, según necesidad y teniendo en cuenta las relaciones de dependencias entre ellos. Tal escenario conlleva a que ante determinadas fallas del SIAI, la determinación de las causas y acciones correctivas necesarias resulte un proceso tedioso y lento, provocando que durante el tiempo de inestabilidad del sistema pueda aumentar el número de fraudes y por consiguiente retrasar el proceso de detección de los mismos.

Partiendo de la situación problemática descrita anteriormente se plantea como **problema a resolver**: ¿Cómo garantizar el control de los procesos del SIAI por los administradores del mismo, para el apoyo a las investigaciones realizadas por los analistas de fraude de ETECSA?

Como propuesta de solución al problema se propone la informatización del control de los procesos del SIAI, teniendo como **objeto de estudio**: El control y administración de los procesos en los sistemas informáticos; enmarcado en el **campo de acción**: Los sistemas administradores de procesos o tareas para distribuciones de GNU/Linux como sistemas operativos.

Se define como **objetivo general**: Desarrollar una aplicación que permita realizar operaciones de control de los procesos del SIAI teniendo en cuenta las dependencias entre ellos. Desglosándose en las siguientes **tareas**:

- ✓ Elaboración de la fundamentación teórica del trabajo.

- ✓ Estudio de sistemas de control de procesos, para conocer su estado del arte y tendencias a nivel nacional e internacional.
- ✓ Selección de mecanismos de captura de información de los procesos que intervienen en el funcionamiento del SIAI.
- ✓ Selección y estudio de herramientas a utilizar para desarrollar el módulo de control de procesos del SIAI.
- ✓ Estudio e identificación de las relaciones de dependencia entre los procesos que intervienen en el funcionamiento del SIAI.
- ✓ Identificación de los procesos y reglas del negocio.
- ✓ Elaboración del modelo de negocio de los procesos que tienen lugar en el organismo en cuestión, para la comprensión del flujo normal de funcionamiento para controlar los procesos del SIAI.
- ✓ Descripción de los procesos representados en el modelo de negocio.
- ✓ Definición de requerimientos funcionales y no funcionales de la solución de software para controlar los procesos del SIAI.
- ✓ Descripción de los requerimientos funcionales y no funcionales de la solución de software para controlar los procesos del SIAI.
- ✓ Elaboración del modelo de diseño de la solución de software.
- ✓ Implementación de la solución de software para controlar los procesos del SIAI.
- ✓ Realización de pruebas de funcionalidad a la solución de software.
- ✓ Realización de los manuales de usuarios necesarios para guiar a los futuros beneficiarios de la solución de software en un correcto uso y explotación de las funcionalidades del mismo.
- ✓ Elaboración de una adecuada documentación de la solución en todas sus etapas (estudio del estado del arte, modelamiento del negocio, diseño, implementación y prueba).

Para la realización de las tareas se emplearon los siguientes **métodos científicos**:

Métodos teóricos:

- ✓ Analítico-Sintético: Se utilizó este método durante todo el proceso investigativo, ya que permitió descomponer el problema en varias partes, lo que garantizó un mejor entendimiento del mismo para finalmente buscar la relación entre esas partes.
- ✓ Histórico-lógico: Se evidencia principalmente en la primera parte de la investigación donde se realizó la fundamentación teórica, ya que permitió estudiar lo más relevante de los procesos

informáticos, principalmente en los sistemas operativos GNU/Linux, las metodologías de desarrollo de software, herramientas y tecnologías que se utilizaron para la creación de la solución propuesta.

- ✓ Modelación: Fue un método muy importante usado en la parte del modelamiento de los procesos del negocio y en la etapa de análisis y diseño, debido a que brindó la posibilidad de representar diferentes algoritmos o ideas en forma gráfica o de diagramas con el objetivo de lograr un mayor entendimiento, principalmente para la etapa referente a la implementación del sistema.

Métodos empíricos:

- ✓ Entrevista: Método que se puso en práctica cuando se establecieron varias conversaciones con el cliente con el objetivo de obtener la información necesaria acerca del proceso de negocio y los requerimientos del sistema.

A continuación se muestra una breve descripción de los 4 capítulos por los que está compuesto el presente trabajo de diploma:

Capítulo 1: "Fundamentación Teórica", aborda los conceptos y definiciones fundamentales que conforman la base teórica del objeto de estudio, se exponen características relevantes de algunas aplicaciones informáticas que se utilizan para administrar procesos y se analiza el lenguaje, la metodología, las técnicas y herramientas a utilizar para el desarrollo del módulo.

Capítulo 2: "Características del sistema", detalla aspectos como la descripción de los procesos de negocio candidatos a informatizar, presentando los requisitos funcionales y no funcionales que debe cumplir la solución de software que se propone.

Capítulo 3: "Diseño del sistema", describe la arquitectura empleada en el desarrollo del módulo y los patrones de diseño utilizados. Además presenta el diagrama de clases de diseño así como las descripciones de las clases que lo conforman.

Capítulo 4: "Implementación y Prueba", muestra los diagramas de despliegue, de componentes, pruebas realizadas y se obtiene además una versión del producto final.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En el presente capítulo se exponen conceptos necesarios para el desarrollo del Módulo de control de procesos del SIAI. Además se realiza un estudio sobre distintos sistemas distribuidos que controlan procesos informáticos existentes en el mundo y en Cuba específicamente en la UCI, analizando un conjunto de características referentes a las tecnologías y herramientas utilizadas, así como las ventajas y desventajas que poseen con el objetivo de valorar elementos decisivos en la selección de una o varias propuestas adecuadas para la realización del módulo propuesto.

1.2 Conceptos asociados al dominio del problema.

Procesador, CPU[2] o micro: Es el cerebro del computador, es el encargado tanto de ejecutar las aplicaciones como de responder al uso de los periféricos de entrada. (2)

Sistema operativo: Programa o conjunto de programas que median el acceso a los dispositivos físicos y los programas de aplicación, además se encarga de administración y comunicación de los recursos y procesos, así como la administración de los dispositivos entrada, salida y almacenamiento. (3)

Unix: Es un sistema operativo de tiempo compartido, controla los recursos de una computadora y los asigna entre los usuarios. Permite a los usuarios correr sus programas y controla los dispositivos de periféricos conectados. (4)

LIFO[3]: Algoritmo utilizado para implementar la pila (estructura de dato abstracto). (5)

Bloque del control de procesos (PCB[4]): Es una estructura de datos que permite al sistema operativo controlar diferentes aspectos de la ejecución de un proceso. (6)

1.3 Procesos.

1.3.1 ¿Qué es un proceso?

El término “proceso” tiene muchas definiciones, las cuales se asocian en dependencia del contexto en que se haga referencia, aunque en esencia se basa en la acción de avanzar partiendo de un punto de partida inicial, que es sometido a un conjunto de transformaciones teniéndose en cuenta acciones que deben ser cumplidas para obtener un resultado específico. (7)

---

[2] Acrónimo inglés de **Central Processing Unit**, traducido al español como: **Unidad Central de Procesamiento**.

[3]Acrónimo inglés de **Last In First Out,** traducido al español como: **Último en Entrar, Primero en Salir**.

[4]Acrónimo inglés de **Process Control Block**, traducido al español como: **Bloque de Control de Procesos**.

Con respecto a la Informática algunas de las definiciones más claras son:

- ✓ Un programa en ejecución.
- ✓ Una instancia de un programa ejecutado en un computador.
- ✓ La entidad que se puede asignar y ejecutar en un procesador.
- ✓ Una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual y un conjunto de recursos asociados.

También se puede pensar en un proceso como una entidad que consiste en un número de elementos, donde los dos más importantes serían el código del programa y un conjunto de datos como:

- ✓ Identificador: Número identificador único asociado al proceso en cuestión, para distinguirlo del resto de los procesos.
- ✓ Estado: Estado en que se encuentra el proceso.
- ✓ Prioridad: Nivel de prioridad relativo al resto de los procesos.
- ✓ Contador de programa: Dirección de la siguiente instrucción del programa que se ejecutará.
- ✓ Punteros de memoria: Incluye los punteros al código de programa y los datos asociado a dicho proceso, además de cualquier bloque de memoria compartido con otros procesos.
- ✓ Datos de contexto: Estos son datos que están presentes en los registros del procesador cuando el proceso está corriendo.
- ✓ Información de estado entrada/salida (E/S): Incluye las peticiones de E/S pendientes.
- ✓ Información de auditoría: Incluye la cantidad de tiempo de procesador y tiempo de reloj utilizados. (8)

1.3.2 Estados de un proceso en GNU/Linux.

GNU/Linux es un sistema operativo multiproceso por lo que en todo instante mantiene varios procesos en memoria al mismo tiempo. Cuando un proceso tiene que pasar a un estado de espera, el sistema operativo le quita el CPU a ese proceso y se la da a otro proceso que le corresponda o que tenga mayor prioridad. A medida que un proceso se ejecuta, su estado cambia según las circunstancias. Los procesos en GNU/Linux tienen los siguientes estados:

- ✓ **Preparado o listo (R):** Proceso que está listo para ejecutarse, pero se encuentra esperando a que se le asigne un tiempo de CPU.
- ✓ **Ejecutando (O):** Proceso que se encuentra ejecutándose.

- ✓ **Suspendido o dormido (S):** Proceso que no entra en el reparto del CPU, ya que se encuentra esperando algún tipo de evento. En cuanto dicho evento se produce, el proceso pasa a formar parte del conjunto de procesos listos.
- ✓ **Parado (T):** Proceso que tampoco entra en el reparto de CPU, pero no porque se encuentre suspendido esperando algún evento, sino que para pasar al estado de preparado necesita recibir una señal determinada para poder continuar.
- ✓ **Zombie (Z):** Cuando un proceso finaliza, avisa a su proceso padre para que este elimine su entrada en la tabla de procesos. Si por algún motivo el padre no recibe esta comunicación no se elimina de la tabla de procesos, por lo que el proceso hijo queda en estado zombie sin consumir tiempo de CPU, pero si recursos del sistema. (9)

Figura 2. Estados de un proceso en Linux.

1.4 Agentes Virtuales.

1.4.1 ¿Qué es un "Agente Virtual"?

Un Agente Virtual en esencia es un sistema software que posee vida independiente y realiza tareas o funciones específicas cumpliendo con los objeticos para los cuales fue creado, aunque generalmente es controlado por otro sistema informático.

Algunas de las definiciones de varios autores acerca de este término son:

- ✓ *"Un agente es una entidad que percibe y actúa sobre un entorno."* (10)

- ✓ *"Los agentes son sistemas computacionales que habitan en entornos dinámicos complejos, perciben y actúan de forma autónoma en ese entorno, realizando un conjunto de tareas y cumpliendo objetivos para los cuales fueron diseñados."* (10)
- ✓ "Un agente es un sistema situado en alguna parte de un entorno que percibe dicho entorno y actúa en él en beneficio de su propia agenda, el efecto de su actuación se nota en el entorno." (11)

1.5 Sistemas Distribuidos (SD).

En la actualidad los sistemas distribuidos se utilizan cada vez más en la sociedad. La creciente demanda del intercambio de información entre aplicaciones diferentes para dar solución a un problema de negocio, además del uso de recursos compartidos son los motivos fundamentales de su uso.

Algunos conceptos de sistemas distribuidos dados por autores reconocidos en la temática son:

- ✓ *"Sistema en el cual múltiples procesadores autónomos, posiblemente de diferentes tipos, están interconectados por una subred de comunicación para interactuar de una manera cooperativa en el logro de un objetivo global."* (12)
- ✓ *"Un sistema distribuido es aquel en el que los componentes localizados en computadores conectados en red, comunican y coordinan sus acciones únicamente mediante el paso de mensajes."* (13)
- ✓ *"Conjunto de computadores independientes que se muestran al usuario como un sistema único coherente."* (14)

De forma general, un sistema distribuido está compuesto por un conjunto de aplicaciones que se ejecutan en diferentes computadoras geográficamente distribuidas, que incluso pueden encontrarse distantes unas de otras, las cuales se encuentran interconectadas por una red, por lo que intercambian información constantemente para lograr que el sistema funcione de forma correcta y cooperativa. Además si estas máquinas tienen hardware diferente y distintos sistemas operativos, es completamente transparente para los usuarios, por lo que estos perciben un sistema único coherente.

- ✓ **Heterogéneos**: Deberán ser construidos a partir de una variedad de diferentes redes, sistemas operativos, hardware y lenguajes de programación. Los protocolos de comunicación de Internet pueden enmascarar la diferencia en redes, middleware y puede hacer frente a las otras diferencias.
- ✓ **Seguros**: El cifrado se puede utilizar para proporcionar una protección adecuada de los recursos compartidos y de mantener en secreto la información confidencial cuando se transmite en mensajes a través de una red.

- ✓ **Calidad de servicio**: No es suficiente con proporcionar acceso a los servicios en sistemas distribuidos. También es importante proporcionar garantías sobre las cualidades asociadas con el acceso de dicho servicio. Ejemplos de tales cualidades pueden incluir parámetros relacionados con el rendimiento, seguridad y fiabilidad.

1.5.1 Sistemas Distribuidos que controlan procesos.

1.5.1.1 Libres.

Nagios

Nagios es un sistema de vigilancia de gran alcance que permite identificar y resolver problemas de infraestructura de tecnología e información antes de que se afecten los procesos de negocio críticos. Actualmente está respaldado por una comunidad de más de un millón de personas de todo el mundo que contribuyen a proyectos de Nagios a través de diferentes maneras, incluyendo el desarrollo, la promoción, capacitación, apoyo y documentación. (15) Es originalmente diseñado para ser ejecutado en GNU/Linux, pero también puede ejecutarse en diferentes variantes de Unix y está licenciado bajo la GPL5 V2.0 publicada por la Fundación de Software Libre. (16) En esencia es una poderosa herramienta de monitorización de redes de código abierto, que utiliza agentes multiplataforma para control de equipos (hardware) y servicios (software) específicos según el interés de los administradores. Básicamente durante su funcionamiento está pendiente de las fallas que se produzcan para intentar solucionarlas automáticamente en caso de ser posible. Además genera alertas cuando se producen comportamientos no deseados definidos previamente por el administrador del sistema, a través de correo electrónico, mensajes SMS o un script personalizado.

Pandora FMS

Pandora FMS es un software de monitorización para detectar problemas y caídas de servicios en servidores, redes y aplicaciones. Está licenciado bajo GPL v2.0 que dispone de una versión específica para empresas, bajo el modelo conocido como “open source”. (17) Utiliza agentes software multiplataforma que proporcionan información acerca del sistema donde están instalados (como: CPU, uso de la memoria, uso de disco, la salida de cualquier comando de consola y los procesos que se encuentran en ejecución.), y agentes hardware (sensores) para monitorizar aspectos sumamente

[5] Acrónimo inglés de **General Public License**.

importantes relacionados con el entorno de trabajo como los consumos energéticos y la temperatura. También permite mostrar gráficos, informes y mapas monitorización a través de la web. Además de que envía notificaciones en caso de que se produzcan anomalías previamente definidas.

JKeeper

Aplicación desarrollada en la UCI con el objetivo de controlar el acceso a los laboratorios de docencia de la universidad. Controla el uso de las sesiones de usuarios, procesos y aplicaciones iniciadas por cada usuario referente a las estaciones de trabajo ubicadas en las subredes pertenecientes a los laboratorios que controla. Está implementado en Java, utiliza agentes para los sistemas operativos de Windows y GNU/Linux. Además posee una vista web administrativa para los administradores, utiliza como gestor de base de datos postgres v9.1, destacándose entre sus funcionalidades principales el bloqueo y control de los procesos referente a cada una de las computadoras conectadas en la red de trabajo.

1.5.1.2 Propietarios.

MangeEngine OpManager

Solución de software para la monitorización de servidores, aplicaciones y dispositivos de red. Posee un elevado nivel de configuración, por lo que está optimizada para empresas de todos los tamaños. Cuenta con un conjunto de aplicaciones internas que poseen una ayuda técnica integrada y permite la administración de activos. Descubre automáticamente toda la red, agrupando los dispositivos de interés en mapas intuitivos y los supervisa en tiempo real en cuanto a rendimiento; ya que permite analizar el uso del tráfico de red. Administra las configuraciones de routers, switches, firewall, puntos de acceso inalámbricos y envía notificaciones en caso de que se produzcan situaciones no deseadas. (18)

I-enable rmf

Solución de gran alcance para supervisar la infraestructura de una red en cuanto a tecnología e información. Monitoriza principalmente el rendimiento y la disponibilidad de las diferentes estaciones o host que pertenezcan a la red de trabajo. Posee una arquitectura basada en web con una interfaz de usuario personalizada y un único servidor rmf que gestiona la disponibilidad y el estado funcional de los servidores, routers, switches, u otros dispositivos como los de almacenamiento o similares. Proporciona informes para permitir un rápido análisis en tiempo real y también cuenta con un sistema automatizado que envía notificaciones por correo electrónico y alertas SMS. (19)

1.5.1.3 Conclusiones del epígrafe.

Es importante señalar que inicialmente se estudiaron las herramientas basadas en software libre más utilizadas a nivel mundial y las desarrolladas en la UCI, con el objetivo de seleccionar alguna de las descritas anteriormente como parte de la solución, pero esta variante fue descartada, producto a que estos sistemas a pesar de que brindan un gran número de funcionalidades referentes a la gestión de procesos informáticos y utilizan agentes virtuales para diferentes plataformas entre las que se encuentra GNU/Linux y Windows, son extremadamente grandes, complejos y están diseñados para el monitoreo de redes, no para control de procesos informáticos en sentido general. Además ninguna de ellas realiza las operaciones de arranque y parada de un sistema informático en su totalidad de forma centralizada, según la necesidad existente y teniendo en cuenta las relaciones de dependencias que puedan existir entre las aplicaciones o subsistemas que se relacionen. Por lo que posteriormente se realizó un estudio de las herramientas propietarias con el objetivo de analizar el principio de su funcionamiento para obtener ideas o sugerencias que puedan ser incluidas en la solución propuesta; que en esencia no es más que el desarrollo del "Módulo de control de procesos del SIAI"; el cual será de gran utilidad, ya que permitirá consultar el estado y las propiedades de los procesos asociados al SIAI, ejecutar las operaciones básicas sobre procesos específicos, realizar las operaciones de arranque y parada del SIAI y gestionar las trazas sobre las diferentes acciones realizadas por diferentes usuarios en dicho módulo.

1.6 Propuesta de metodologías, lenguajes, técnicas y herramientas a utilizar.

1.6.1 Metodología de desarrollo.

El **Proceso Unificado de Software (RUP[6])** es un proceso de desarrollo de software que constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Define claramente quién, cuándo, cómo y qué debe hacerse en el proyecto y se caracteriza por ser un proceso iterativo e incremental que genera una extensa y rigurosa documentación ya que proporciona una guía para encontrar, organizar, documentar, y seguir los cambios de los requisitos funcionales y restricciones a través de una notación de Caso de Uso, permite un seguimiento detallado de cada una de las fases por las que transita el producto y un dominio elevado del estado real del mismo. (20)

[6] Acrónimo inglés de **Rational Unified Process**.

Para el desarrollo del SIAI se establece el uso de esta metodología debido a la larga duración y envergadura del mismo, donde el personal que lo integra es mixto perteneciente a las entidades ETECSA y UCI. La composición del proyecto está integrada principalmente por especialistas (negocio y desarrollo de software) de ETECSA y por parte de la UCI por estudiantes que en su mayoría al culminar los estudios dejan de pertenecer al proyecto, así como personal recién graduado o en período de adiestramiento que al finalizar el mismo tienen la posibilidad de cambiar de centro de trabajo. Se tiene en cuenta la posibilidad de incorporación de nuevos integrantes al equipo de trabajo, no sólo de la UCI sino también de ETECSA; por lo que se hace imprescindible una detallada documentación en cada fase de desarrollo para una mejor comprensión por parte de todos los implicados. Además es importante señalar que se pretende utilizar una variante de RUP modificada que se acomode a las necesidades concretas del mismo. Por lo anteriormente expuesto se considera a RUP como metodología de desarrollo de software a utilizar para la creación del módulo de configuración de procesos.

1.6.2 Lenguaje de Modelado.

Lenguaje Unificado de Modelado (UML[7]) es un lenguaje gráfico que permite visualizar, especificar, construir y documentar el sistema que se pretende desarrollar. Es el lenguaje de modelado de sistemas de software más conocido y utilizado actualmente puesto que sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. (21) Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (como RUP), pero no especifica en sí mismo qué metodología o proceso usar puesto que es independiente del lenguaje que se emplee para la implementación.

Para poder representar correctamente un sistema, desde varias perspectivas, UML ofrece una amplia variedad de diagramas como:

- ✓ Diagrama de casos de uso.
- ✓ Diagramas de paquetes.
- ✓ Diagrama de clases.
- ✓ Diagrama de componentes.
- ✓ Diagrama de despliegue.

Por las características expuestas anteriormente y teniéndose en cuenta además, que sirve para el modelado completo del sistema, facilita el entendimiento entre los analistas del sistema y el cliente, da

[7] Acrónimo inglés de **Unified Modeling Language**.

soporte a la metodología utilizada (RUP), es independiente del lenguaje de programación y brinda rigor en la especificación, se selecciona como lenguaje de modelado a UML.

1.6.3 Técnica de modelación del negocio.

Definición de la Integración para la Modelización de las Funciones (IDEF0[8]): Técnica para representar de manera estructurada y jerárquica las actividades que conforman un sistema o empresa y los objetos o datos que soportan la interacción de esas actividades. IDEF0 es útil para establecer el alcance del análisis, principalmente para un análisis funcional. Además como una herramienta de análisis, IDEF0 ayuda al modelador en la identificación de cuáles son las funciones que se llevan a cabo, lo que se necesita para desempeñar esas funciones, lo que el sistema actual hace bien o hace mal. Algunas de sus características más importantes son:

- ✓ Ser comprensivo, expresivo, capaz de representar gráficamente una amplia variedad de negocio de cualquier empresa a cualquier nivel del detalle.
- ✓ Lenguaje coherente, simple, que prevé la expresión rigurosa y exacta, que promueve la consistencia del uso y de la interpretación.
- ✓ Puede ser generado por una gran variedad de herramientas gráficas en computadores. (22)

La utilización de IDEF0 es una alternativa factible para la descripción detallada del negocio con el objetivo de ahorrar tiempo y esfuerzos, ya que brinda una ágil y sencilla técnica para la modelación del negocio, aprovechando el uso de una notación simple para describir actividades que pueden ser descritas por sus entradas, salidas, controles y mecanismos.

1.6.4 Herramientas CASE.

Cuando se procede a realizar un software se requiere que los procesos sean organizados y completados de forma correcta y eficiente. Las herramientas CASE[9] fueron desarrolladas para automatizar procesos y aumentar la productividad en el desarrollo de software. (23)

[8] Acrónimo inglés de **Integration Definition For Funcion Modeling**.

[9] Acrónimo inglés de **Computer Aided Software Engineering**, traducido al español como "**Ingeniería de Software Asistida por Ordenador**".

Visual Paradigm v5.0.

Visual Paradigm para UML es una herramienta profesional que brinda soporte al ciclo de vida completo del desarrollo de software ya que contiene un grupo de módulos que contribuyen con la confección de un software. Algunas de las características que posee son:

- ✓ **Es profesional**: Genera un conjunto amplio de artefactos utilizados con mucha frecuencia durante la confección de un software. Todos estos, cumpliendo con el Standard UML 2.0.
- ✓ **Es multiplataforma**: Está disponible para diferentes plataformas como Microsoft Windows (98, 2000, XP, Vista o W7), GNU/Linux y Mac OS.
- ✓ **Es amigable**: Se encuentra disponible en varios idiomas y sus componentes se encuentran relacionados ya que cuando es usado algún componente, este surgiere nuevos posibles componentes a utilizar, por lo que no es sumamente necesario localizarlos en la barra donde pueden aparecer un número grande de componentes.
- ✓ **Mejor entendimiento de los diagramas**: Facilita un conjunto de estereotipos.
- ✓ **Integración con distintos Entornos de Desarrollo Integrados** (IDE[10]): Se integra fácilmente con varios IDEs como el Eclipse y el Visual Estudio.
- ✓ **Facilidades para redactar especificaciones de casos de uso**: Es posible crear plantillas para las especificaciones de casos de uso y describirlos.
- ✓ **Generación de documentación**: Brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas. (24)

Microsoft Office Visio 2007.

"Office Visio 2007 proporciona una amplia gama de plantillas, diagramas de flujo de procesos empresariales, diagramas de redes, diagramas de flujo de trabajo, modelos de bases de datos y diagramas de software, que puede utilizar para ver y racionalizar procesos empresariales, realizar el seguimiento de proyectos y recursos, crear organigramas, generar mapas de redes, confeccionar diagramas para la creación de sitios y optimizar sistemas." (25)

Se utilizará esta herramienta debido a que cuenta con un tipo de diagrama predefinido para modelar los procesos de negocio a través de la notación IDEF0 de manera muy intuitiva, además de que se encuentra disponible para varios idiomas entre ellos el español y permite la realización de los diagramas con una excelente calidad visual.

[10] Acrónimo inglés de **Integrated Development Environment**, traducido al español como: **Entorno de Desarrollo Integrado**.

1.6.5 Lenguaje de Programación.

Python v2.7.

Es un lenguaje de programación potente, simple, fácil de aprender y multiparadigma, que obliga a los programadores a adoptar un estilo de programación particular. Contiene gran cantidad de bibliotecas, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar tareas habituales sin necesidad de reprogramarlas desde cero.

Características de Python:

- ✓ **Lenguaje interpretado o de script**: Es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje de máquina.
- ✓ **Tipado dinámico**: Se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo de valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.
- ✓ **Fuertemente tipado**: No permite tratar a una variable sin tener en cuenta el tipo de dato que almacena, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente.
- ✓ **Sintaxis clara**: Posee una sintaxis muy visual gracias a una notación identada (con márgenes) de obligado cumplimiento.
- ✓ **Multiplataforma**: El intérprete de Python está disponible en varias plataformas (Solaris, GNU/Linux, DOS, Windows, OS/2, Mac OS) por lo que si no se utilizan bibliotecas específicas de cada plataforma el programa podrá correr en todos estos sistemas sin problemas.
- ✓ **Orientado a Objeto**: La orientación a objetos es un paradigma de programación en el que los conceptos relevantes del mundo real para los problemas se trasladan a clases y objetos del programa. La ejecución del programa consiste en una serie de interacciones entre los objetos. Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

Teniendo en cuenta que es uno de los lenguajes de programación con los que más capacitación cuenta el equipo de trabajo del SIAI y por las características anteriormente expuestas se seleccionó como lenguaje de programación. Además se puede decir que brinda una amplia biblioteca de funcionalidades estándar, lo que posibilita que se utilicen la menor cantidad de librerías externas; y la existencia de una amplia colección de librerías libres no requirió el uso de alguna privativa. Un ejemplo de estas librerías es *"(…)*

soappy, la cual proporciona una biblioteca de SOAP[11] muy simple de usar y que es totalmente compatible con la interacción dinámica entre clientes y servidores, mediante la cual se pueden publicar objetos y ejecutar funciones en un servidor remoto." (26)

1.6.6 Entorno de desarrollo Integrado.

Los IDE están compuestos por un conjunto de herramientas de programación y juegan un papel fundamental en el desarrollo de cualquier aplicación informática. (…) proveen un marco de trabajo amigable para los lenguajes de programación y ofrecen facilidades al equipo de desarrollo cuando se implementan las aplicaciones debido a que permiten la corrección de errores comunes que se cometen a diario. (27)

Eclipse Helios.

"Eclipse se está posicionando como, no sólo como un potente IDE de desarrollo Java, sino como una plataforma completa para el desarrollo de aplicaciones." (28)

Para la implementación de la solución de software se decide utilizar esta versión de Eclipse debido a que es un IDE multiplataforma, de código abierto y libre que cuenta con una comunidad activa de desarrolladores. Además se integra perfectamente con el lenguaje de programación seleccionado (en este caso Python) permitiendo un excelente completamiento de código, resaltado de sintaxis, depuración de la ejecución, funcionamiento en varias plataformas, así como diferentes funcionalidades que le facilitan el trabajo al programador como la refactorización del código.

1.6.7 Interfaz gráfica de usuario.

PyQt: Qt para Python.

Qt es una de las mejores librerías gráficas que existen en el mundo del software libre implementadas en el lenguaje de C++. Su carácter multiplataforma hace fácil su despliegue. PyQt es la combinación de Python y Qt, estableciendo una interfaz transparente para acceder desde Python a dichas bibliotecas gráficas; por lo que se hace más sencilla la programación visual. También cuenta con una amplia documentación proveniente de Qt y de Python a la vez. (27)

[11] Acrónimo inglés de **Simple Object Access Protocol:** es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Para el desarrollo del sistema se propone utilizar esta librería gráfica ya que mediante ella se pueden crear interfaces de forma rápida, bastantes sencillas, agradables y con una excelente calidad. Además PyQt brinda una abundante documentación y ejemplos, lo que facilita el trabajo para los desarrolladores.

Qt Designer

Es la herramienta por excelencia para el diseño gráfico del Framework Qt. Se creó con el mismo fin para el que fue confeccionado Qt: agilizar el desarrollo y funcionar en una amplia variedad de plataformas. (27) Permite diseñar interfaces de usuario rápidamente con la funcionalidad de arrastrar y soltar, cuenta con una gran biblioteca de “widgets” que pueden ser personalizados y posibilita la pre visualización en aspecto nativo de las interfaces creadas. (29)

1.7 Conclusiones del capítulo.

En este capítulo se realizó un análisis sobre los principales conceptos y definiciones que se relacionan con el objeto de estudio. Se expusieron algunas particularidades o características más relevantes de los Sistemas Distribuidos más utilizados que existen actualmente tanto a nivel internacional como a nivel nacional incluyendo la Universidad de las Ciencias Informáticas y se definieron y justificaron aspectos muy importantes para la creación del “Módulo de control de procesos del SIAI” como la metodología de desarrollo, el lenguaje y las herramientas candidatas a utilizar.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción.

En el presente capítulo se hace referencia a las características del Módulo de control de procesos del SIAI, donde se describe el análisis y la modelación realizado para la creación de la solución propuesta. Además se detallan los aspectos relacionados con el dominio de la solución, la descripción de los procesos del negocio candidatos a automatizar; además de presentarse los requisitos funcionales y no funcionales, los cuales van a asentar las bases para el posterior diseño de la aplicación. También se definen los actores y los casos de usos del sistema respectivamente identificados con el objetivo de tener una mejor comprensión de forma general.

2.2 Información que se maneja.

La información referente a los documentos del proyecto que se maneja es de uso exclusivo para el SIAI, además de ser confidenciales por lo que se cree conveniente no entrar en detalles.

2.3 Propuesta del sistema.

La no existencia de un mecanismo o una herramienta que contribuya con el control de los procesos que intervienen en el funcionamiento del SIAI, expone como necesidad la creación de una aplicación que permita a los administradores del SIAI, consultar el estado en que se encuentran los procesos asociados al sistema, es decir, conocer si se encuentran activos, determinados procesos pertenecientes a máquinas específicas ubicadas en la red de trabajo del SIAI. Para el caso de los procesos que se encuentren activos, se debe especificar sus principales propiedades referentes al sistema operativo donde se encuentran ejecutándose. También la aplicación deberá ejecutar las operaciones de control básicas sobre los procesos que controla (Iniciar, Detener, Continuar, Finalizar y Matar), así como realizar las operaciones de arranque y parada del sistema completo, lo que significa iniciar o finalizar un conjunto de procesos previamente configurados (con datos especificados por el usuario) y en un orden determinado que se establece teniendo en cuenta las relaciones de dependencias que puedan existir entre ellos. Además debe registrar las trazas de las acciones realizadas en el “Módulo de control de procesos del SIAI”, por lo que debe mostrar datos de interés (hora, fecha, usuario,…) de las últimas acciones realizadas referentes a las operaciones de control de procesos, los cambios de procesos que controla y las operaciones de arranque y parada que se lleven a cabo.

2.4 Modelo de negocio.

En la descripción del modelo de negocio se propone IDEF0 para la captura y modelación detallada de las actividades, entradas, controles, mecanismos y salidas del “Módulo de control de procesos del SIAI”. Tomándose como proceso principal del negocio: Controlar los procesos que intervienen en el funcionamiento del SIAI.

2.4.1 Diagramas de los procesos del negocio.

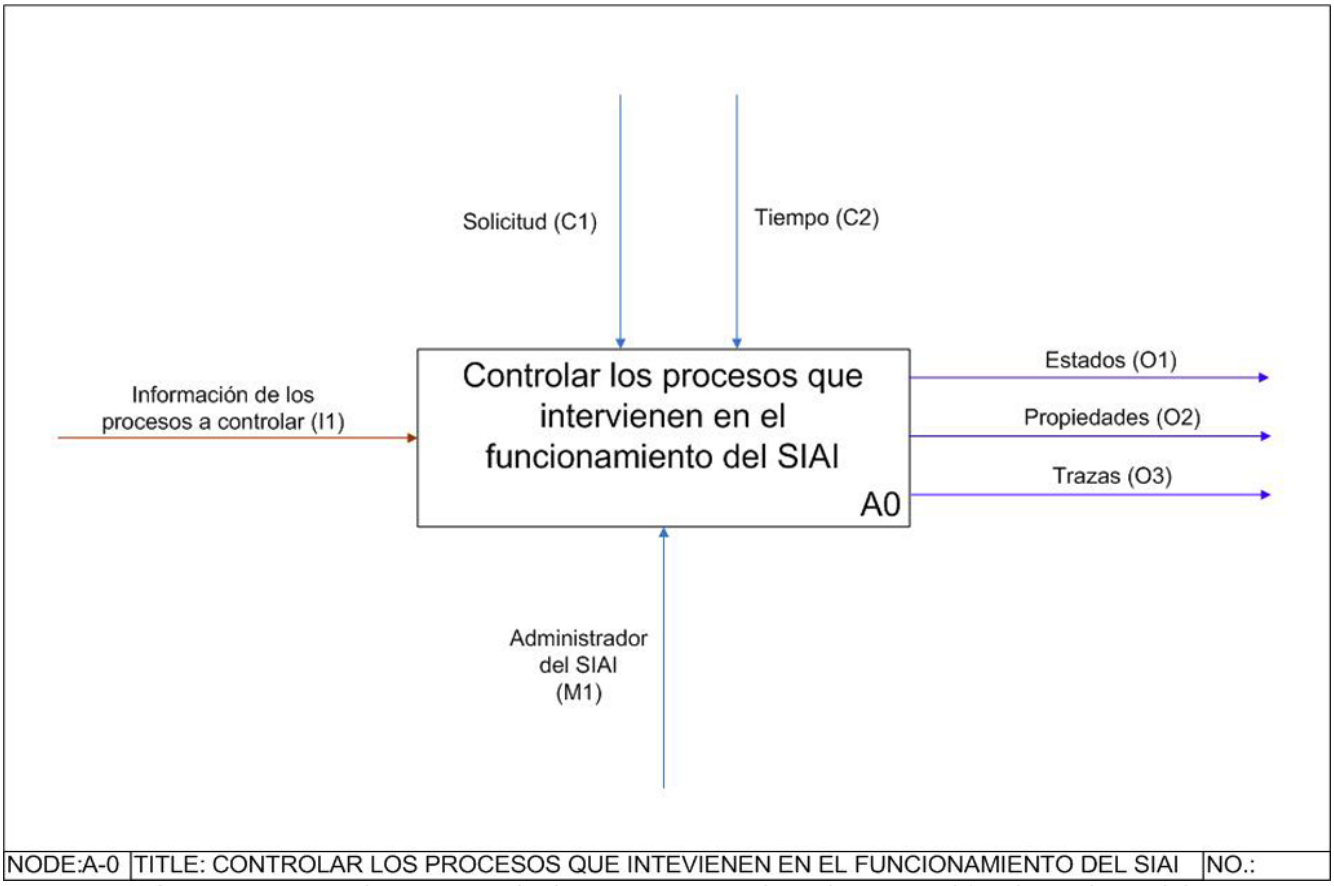


Figura 3. Proceso base: Controlar los procesos que intervienen en el funcionamiento del SIAI.

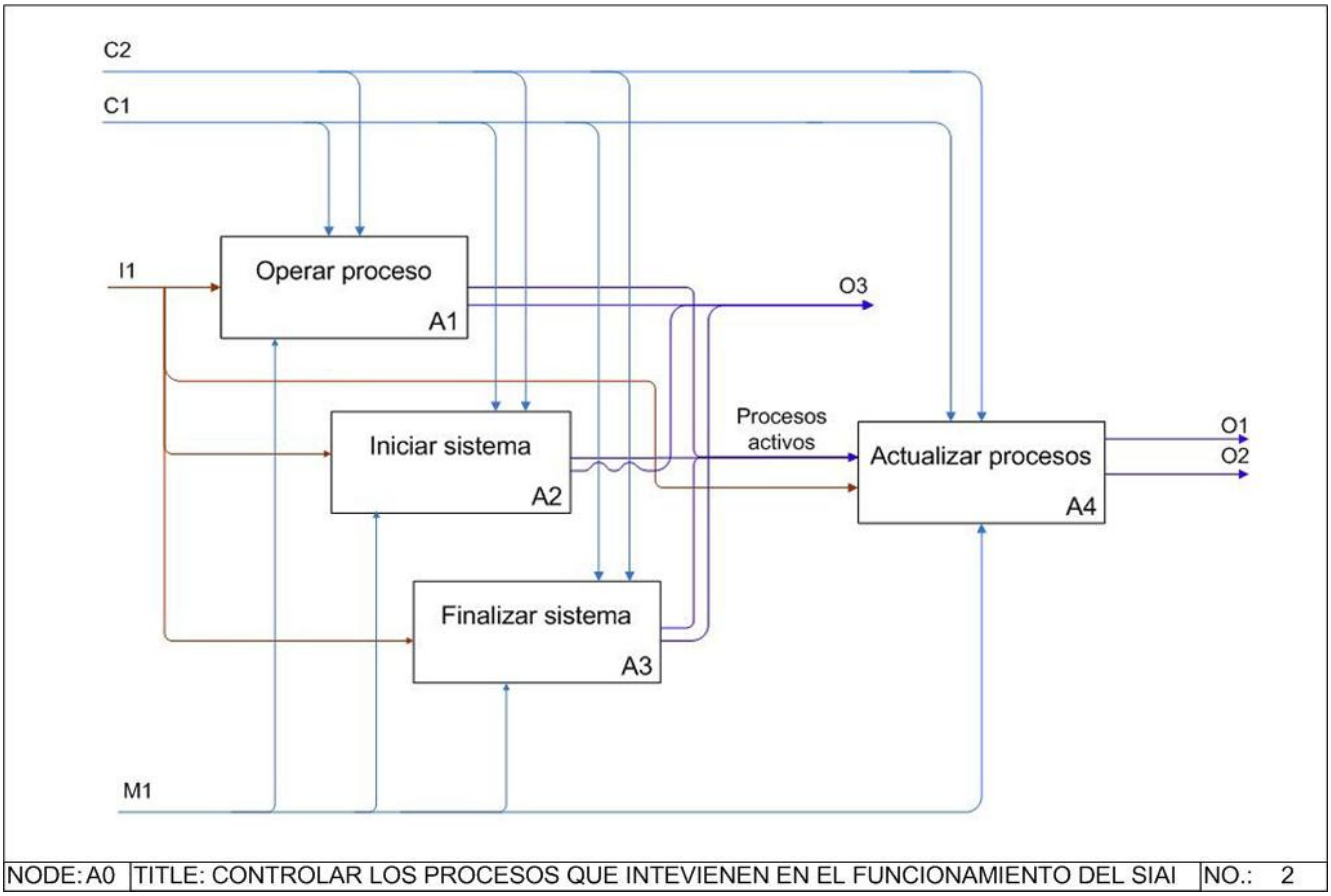


Figura 4. Diagrama detallado del proceso base.

2.4.2 Descripción de los procesos del negocio.

2.4.2.1 Proceso "Controlar los procesos que intervienen en el funcionamiento del SIAI".

Tabla 3. Descripción detallada del proceso.

Ficha del proceso: Controlar los procesos que intervienen en el funcionamiento del SIAI.		
Notación:	A0	
Entrada:	I1	Información de los procesos a controlar. (Incluye nombre, descripción, IP[12], nivel y comando de ejecución)
Controles:	C1	Solicitud.
	C2	Tiempo.

[12] Acrónimo inglés de **Internet Protocol** traducido al español como: **Protocolo de Internet**.

Salidas:	O1	Estados referentes a la existencia real de cada proceso controlado.
	O2	Propiedades principales de los procesos activos como: estado definido por el sistema operativo, usuario asociado, identificador (ID), identificador del padre (PID), tiempo de inicio, tiempo de uso del CPU y el comando de ejecución.
	O3	Trazas referentes a las acciones que se llevan a cabo en el módulo. (Cada traza registra el tipo de traza, la fecha, la hora, el usuario, y los procesos involucrados.)
Mecanismo:	M1	Administrador del SIAI: Personal autorizado para utilizar la aplicación.
Regla del Negocio:	**Estados de un proceso controlado:** "Encendido" si el proceso se encuentra ejecutándose, "Apagado" si el proceso no se encuentra en algún estado de ejecución, o "Sin conexión" si no existe comunicación con el host al que pertenece dicho proceso controlado. **Tipos de trazas:** Se definen como: "traza de inicio", "traza de detención", "traza de continuidad", "traza de finalización", "traza de inicio de sistema", "traza de finalización de sistema" y "traza de configuración".	
Descripción del proceso		
Es el proceso principal que hace referencia al control de los procesos informáticos que intervienen en el funcionamiento del SIAI, tomándose como punto de partida la información necesaria de los procesos referentes a los subsistemas o aplicaciones del SIAI. Esta información es procesada por el administrador del sistema y regulada mediante una solicitud o el tiempo, los cuales rigen el funcionamiento en sentido general. Como resultado de la interacción de todos los factores mencionados se puede actualizar el listado de procesos controlados con los estados reales definidos en las reglas del negocio, teniendo en cuenta la existencia real de cada proceso controlado respecto al sistema operativo en el que se ejecuta. También se obtienen las propiedades de los procesos activos (procesos que se encuentran en algún estado de ejecución) en computadoras específicas pertenecientes a la red de trabajo del SIAI y se registra cada una de las acciones que se lleven a cabo durante el control de los procesos; facilitando de esta forma un mejor seguimiento del funcionamiento del SIAI.		

2.4.2.2 Proceso “Operar proceso”.

Tabla 4. Descripción detallada del proceso.

Ficha del proceso: Operar proceso.		
Notación:	A1	
Entradas:	I1	Información de los procesos a controlar.
Controles:	C1	Solicitud.
	C2	Tiempo.
Salidas:		Procesos activos.
	O3	Traza: acción que se lleve a cabo.
Mecanismos:	M1	Administrador del SIAI.
Descripción del proceso		
Es el proceso encargado de realizar las operaciones básicas de control de procesos “Iniciar”, “Detener”, “Continuar”, “Finalizar”, y “Matar” a los procesos controlados seleccionados, según necesidad e interés del administrador del SIAI. Además es regulado por los controles solicitud y tiempo; y como resultado final se obtiene la información necesaria de los procesos activos del SIAI.		

2.4.2.3 Proceso “Iniciar sistema”.

Tabla 5. Descripción detallada del proceso.

Ficha del proceso: Iniciar sistema.		
Notación:	A2	
Entradas:	I1	Información de los procesos a controlar.
Controles:	C1	Solicitud.
	C2	Tiempo.
Salidas:		Procesos activos.
	O3	Traza: acción que específica el “Inicio del sistema”.
Mecanismos:	M1	Administrador del SIAI.
Descripción del proceso		

Proceso encargado de realizar el arranque o “startup” del SIAI, teniendo en cuenta las relaciones de dependencias que puedan existir entre los procesos controlados que se relacionen como procesos informáticos. El administrador del SIAI es el encargado de ejecutar esta tarea que es regulada mediante una solicitud y el tiempo; devolviendo como resultado final la información necesaria de los procesos activos del SIAI.

2.4.2.4 Proceso “Finalizar sistema”.

Tabla 6. Descripción detallada del proceso.

Ficha del proceso: Finalizar sistema.		
Notación:	A3	
Entradas:	I1	Información de los procesos a controlar.
Controles:	C1	Solicitud.
	C2	Tiempo.
Salidas:		Procesos activos.
	O3	Traza: acción que específica el “Finalización del sistema”.
Mecanismos:	M1	Administrador del SIAI.
Descripción del proceso		
Proceso encargado de realizar la parada o “shutdown” del SIAI, teniendo en cuenta las relaciones de dependencias que puedan existir entre los procesos controlados que se relacionen como procesos informáticos. El administrador del SIAI es el encargado de ejecutar esta tarea que es regulada por los una solicitud y el tiempo; devolviendo como resultado final la información necesaria de los procesos activos del SIAI.		

2.4.2.5 Proceso “Actualizar procesos”.

Tabla 7. Descripción detallada del proceso.

Ficha del proceso: Actualizar procesos.		
Notación:	A4	
Entradas:	I1	Información de los procesos a controlar.
		Procesos activos.
Controles:	C1	Solicitud.
	C2	Tiempo.

Salidas:	O1	Estados referentes a la existencia real de cada proceso controlado.
	O2	Propiedades.
Mecanismos:	M1	Administrador del SIAI.
Descripción del proceso		
Proceso encargado de obtener el estado actual de cada proceso controlado y la información referente a los procesos activos del SIAI. El administrador del SIAI es el encargado de realizar esta operación que se regula mediante una solicitud y el tiempo, por lo que devuelve como resultado final un listado con los procesos activos del SIAI, de los cuales se pueden verificar sus principales propiedades y actualiza el estado real de cada proceso perteneciente al listado de procesos controlados.		

2.5 Especificación de los requisitos del software.

2.5.1 Requerimientos funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. De forma general estos requisitos indican lo que debe hacer el sistema. Los requerimientos funcionales que se identificaron para el desarrollo del módulo fueron los siguientes:

RF1 Configurar puerto de conexión con el agente virtual.

Funcionalidad que permite establecer el puerto de conexión utilizado para garantizar la comunicación con la aplicación agente especificándole:

- ✓ puerto - Campo obligatorio y de longitud variable que puede contener solamente números.

RF2 Verificar la conexión con el agente.

Funcionalidad que permite comprobar la comunicación con un agente específico indicándole:

- ✓ IP - Campo obligatorio y de longitud variable de hasta 15 caracteres que puede contener números y puntos (“.”).

RF3 Gestionar procesos controlados.

Funcionalidad que a partir del listado de procesos controlados (hace referencia al requisito **RF5** Listar procesos controlados), permite adicionar uno nuevo o seleccionar alguno de ellos para ser modificado o eliminado:

RF3.1 Adicionar proceso.

Funcionalidad que permite adicionar un nuevo proceso al listado de procesos controlados, solicitando los siguientes datos:

- ✓ IP - Campo obligatorio y de longitud variable de hasta 15 caracteres que puede contener números y puntos (“.”). Debe ser indicada en el formato XXX.YYY.YYY.YYY; donde XXX permite los valores numéricos 10, 192 o 172, y YYY los valores numéricos en el rango 0-254.
- ✓ Nombre - Campo obligatorio y de longitud variable de hasta 100 caracteres.
- ✓ Orden - Campo obligatorio y de longitud variable sin límites de caracteres.
- ✓ Comando de ejecución (CMD) - Campo obligatorio y de longitud variable sin límites de caracteres.
- ✓ Descripción - Campo opcional y de longitud variable de hasta 100 caracteres.
- ✓ Nivel de dependencia - Campo obligatorio seleccionable que donde se especifican un valores enteros preestablecidos referentes al nivel de dependencia.

RF3.2 Eliminar un proceso.

Funcionalidad que permite eliminar un proceso controlado que haya sido registrado con anterioridad.

RF3.3 Modificar un proceso.

Funcionalidad que permite modificar de un proceso controlado que haya sido registrado con anterioridad su nombre, descripción, IP, nivel de dependencia orden y CMD.

RF4 Guardar configuración de procesos controlados.

Funcionalidad que permite salvar en un fichero la información (nombre, descripción, IP asociado al host que pertenece, nivel, orden y comando de ejecución que registra en el sistema operativo) referente a cada uno de los procesos controlados.

RF5 Listar procesos controlados.

Funcionalidad que permite mostrar un listado de los procesos controlados registrados con anterioridad organizados ascendentemente por el nivel, mostrando de cada uno de ellos su nombre, descripción, IP, nivel de dependencia y CMD.

RF6 Listar procesos activos.

Funcionalidad que permite mostrar un listado de los procesos activos del SIAI, mostrando de cada uno de ellos su nombre, PID, usuario y CMD, los cuales van a estar agrupados por el IP referente al host al que pertenecen.

RF7 Listar trazas.

Funcionalidad que permite mostrar un listado de las trazas registradas con anterioridad organizado cronológicamente, mostrando de cada una de ellas el tipo, la fecha, la hora y el usuario que realizó la acción correspondiente, teniendo en cuenta el filtro de trazas configurado anteriormente (hace referencia al requisito **RF11** Filtrar trazas.).

RF8 Actualizar procesos.

Funcionalidad que permite actualizar los cambios en los listados de procesos controlados y de procesos activos (hace referencia al requisito **RF5** Listar procesos controlados y al **RF6** Listar procesos activos).

RF9 Administrar procesos controlados.

Funcionalidad que a partir de un proceso seleccionado del listado de procesos controlados (hace referencia al requisito **RF5** Listar procesos controlados), permite realizar las operaciones básicas de control (iniciar, detener, continuar, finalizar y matar); a continuación se describen dichas funcionalidades:

RF9.1 Iniciar proceso.

Funcionalidad que permite iniciar la ejecución de un proceso seleccionado que no se encuentre activo, teniendo en cuenta el IP referente a la estación de trabajo que pertenece y la orden de ejecución, atributos configurados inicialmente al adicionarse al listado.

RF9.2 Detener proceso.

Funcionalidad que permite detener la ejecución de un proceso activo seleccionado, teniendo en cuenta el IP referente a la estación de trabajo que pertenece y el CMD que registra en el sistema operativo, atributos configurados inicialmente al adicionarse al listado.

RF9.3 Continuar proceso.

Funcionalidad que permite darle continuidad a la ejecución de un proceso activo seleccionado, teniendo en cuenta el IP referente a la estación de trabajo que pertenece y el CMD que registra en el sistema operativo, atributos configurados inicialmente al adicionarse al listado.

RF9.4 Finalizar proceso.

Funcionalidad que permite finalizar un proceso activo seleccionado, teniendo en cuenta el IP referente a la estación de trabajo que pertenece y el CMD que registra en el sistema operativo, atributos configurados inicialmente al adicionarse al listado.

RF9.5 Matar proceso.

Funcionalidad que permite matar un proceso activo seleccionado, teniendo en cuenta el IP referente a la estación de trabajo que pertenece y el CMD que registra en el sistema operativo, atributos configurados inicialmente al adicionarse al listado.

RF10 Mostrar información de proceso activo

Funcionalidad que a partir de un proceso seleccionado del listado de procesos activos (hace referencia al requisito **RF6** Listar procesos activos) permite mostrar las propiedades IP, nombre, descripción, estado, hora de inicio, usuario asociado, tiempo de uso del CPU, comando de ejecución, ID, y PID.

RF11 Filtrar trazas.

Funcionalidad que a partir del listado de trazas (hace referencia al requisito **RF7** Listar trazas) permite establecer la(s) condición(es) de filtrado a aplicar sobre las mismas; salvando en un fichero los tipos de trazas que se desean mostrar y actualizando finalmente el listado de trazas teniendo en cuenta cambios realizados.

RF12 Mostrar leyenda de íconos.

Funcionalidad que permite mostrar la leyenda de íconos del sistema, especificando en cada caso el ícono y su correspondiente descripción de funcionalidad o significado.

RF13 Mostrar ayuda.

Funcionalidad que permite mostrar información de ayuda al usuario del sistema sobre las funcionalidades generales.

2.5.2 Requerimientos no funcionales.

2.5.2.1 RNF Seguridad.

La seguridad del sistema se debe garantizar en todo momento, la cual debe velar por tres aspectos fundamentales:

Confidencialidad: La información manejada por el sistema estará protegida de acceso no autorizado. Solo el personal con los privilegios necesarios podrá acceder a la aplicación conectándose vía SSH[13] al servidor donde se encuentra corriendo, por lo que se requiere de una cuenta de usuario válida.

Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos, por lo que anualmente se realizarán varias auditorías.

Disponibilidad: Al personal autorizado se le garantizará el acceso al sistema mediante una cuenta de usuario con los privilegios necesarios para poder operar y consultar los datos deseados en el momento establecido.

[13] Acrónimo inglés de **Secure Shell**, traducido al español como: Intérprete de órdenes segura.

2.5.2.2 RNF Usabilidad.

La aplicación puede ser utilizada por cualquier persona que posea conocimientos elementales en el trabajo con sistemas informáticos. Motivo por el cual se establecen algunos requisitos adicionales para optimar el trabajo con el sistema por parte de los usuarios como son:

Idiomas.

El objetivo primordial es garantizar la facilidad de uso por personas que hablen el idioma español.

Entrenamiento a usuarios.

Impartir curso sobre el manejo de la herramienta para controlar los procesos del SIAI es una opción para mejorar las habilidades de los usuarios finales en el uso de esta aplicación.

Documentación de usuarios.

La documentación de usuarios se reflejará en el “Manual de usuarios”, que incluirá la documentación detallada de cada una de las funcionalidades del sistema.

2.5.2.3 RNF Fiabilidad.

La fiabilidad incluye los requisitos que garantizan la confianza del usuario en las respuestas del sistema en un sentido técnico, es decir, que la funcionalidad del sistema no se vea afectada por factores ajenos al sistema como los son los factores técnicos. Hace referencia a la capacidad para tolerar errores y recuperarse de los mismos.

RNF Disponibilidad del sistema.

Una vez desplegado el sistema, el mismo deberá estar disponible siempre que se necesite ejecutarlo, solamente no estará disponible cuando se realice el mantenimiento total del producto y en caso de que se produzca algún fallo inesperado durante el período de recuperación.

RNF Tiempo medio de reparación.

Si el fallo involucra hardware debe tener un tiempo medio de reparación de veinticuatro horas. En los casos que involucra software debe ser en dependencia del error, aunque generalmente debe ser menor a las tres horas.

2.5.2.4 RNF Eficiencia.

La eficiencia del sistema se encuentra relacionada con tiempos de respuesta estimados, requeridos y esperados para la comunicación entre la aplicación gestora y la agente, teniendo en cuenta la cantidad de información que se envíe y la cantidad de agentes pertenecientes a diferentes computadoras con las que se comunique.

RNF Tiempo de respuesta del agente virtual.

El tiempo de respuesta del agente virtual depende de la complejidad de la petición realizada por el sistema gestor, donde por lo general no debe exceder los 10 segundos, aunque también hay que tener en cuenta otros factores muy importantes como el cableado utilizado para la red y los equipos de interconexión.

2.5.2.5 RNF Soporte.

El soporte incluye los requisitos que especifican las acciones a tomar una vez que se ha terminado el desarrollo del software, con motivos de asistir a los clientes de este; así como lograr su mejoramiento progresivo y evolución en el tiempo.

RNF Portabilidad.

El código basado en Python es portable a cualquier plataforma de sistema operativo, pero el sistema actualmente no cuenta con agentes para alguna plataforma diferente a GNU/Linux.

RNF Estándares de codificación.

Para reforzar el soporte o mantenimiento del sistema a construir, se deben seguir las normas o pautas de codificación establecidas por el grupo de trabajo del SIAI, específicamente para el lenguaje de programación Python. Las mismas se especifican en el documento ubicado en EXPEDIENTE DE PROYECTO V2.02 (SIAI)\0. GENERALES\PAUTAS DE CODIFICACIÓN\Estándar de codificación Python.doc.

RNF Entorno de pruebas.

Debido a que el proceso de desarrollo se realizará en el Departamento de Antifraude de ETECSA se hace necesario que todas las pruebas se ejecuten es dicha entidad por el personal autorizado.

RNF Reusabilidad.

Como valor adicional al cumplimiento de las funciones del sistema de controlar los procesos que intervienen en el funcionamiento del SIAI, el mismo deberá ser capaz de prestar servicio a otros sistemas sin que esto implique modificaciones o redefiniciones en dicho componente.

2.5.2.6 RNF Interfaz.

Los requisitos no funcionales de interfaz definen las interfaces que deben ser soportadas por la aplicación. Entre ellas las de usuario, hardware y software.

RNF Interfaces de usuario.

Las interfaces de usuarios a manera general deben ser simples de usar, que contribuyan en el aporte de un estilo profesional. A continuación se describe la apariencia del producto que se debe lograr:

- ✓ Interfaz de usuario amigable y fácil de entender sin mucho entrenamiento por parte del cliente.
- ✓ Colores suaves e información legible que evite cualquier pérdida por parte de un usuario sin conocimiento de la aplicación.
- ✓ El usuario puede acceder a las diferentes opciones sin tener que navegar por muchos puntos intermedios.

RNF Interfaces de hardware.

Esta sección incluye los requisitos que describen las interfaces de hardware que deberán ser garantizadas; incluyendo la estructura lógica, rendimiento y el comportamiento esperado.

A continuación se especifican los elementos de hardware de que se deben disponer.

1. Estación de trabajo donde se encuentra ejecutándose la aplicación agente:
   - ✓ Conexión 100 mbps.
   - ✓ CPU con microprocesador a 2.0 GHz o más.
   - ✓ Memoria RAM[14] de 512 MB o más.
2. Estación de trabajo donde se encuentra ejecutándose la aplicación gestora:
   - ✓ Conexión 100 mbps.
   - ✓ CPU con microprocesador a 3.0 GHz o más.
   - ✓ Memoria RAM de 2 GB o más.

[14] Acrónimo inglés de **Random Access Memory**, traducido al español como: Memoria de acceso aleatorio.

RNF Interfaces de software.

Esta sección incluye los requisitos que describen las interfaces de software que deberán ser garantizadas; incluyendo sistema operativo y/o programa(s) específicos.

A continuación se especifica el software del que se debe disponer para las estaciones de trabajo donde se encuentra ejecutándose la aplicación agente y gestora respectivamente:

- ✓ Sistema operativo Linux, kernel 2.6.24 o superior.
- ✓ Python 2.7.x o superior instalado.

2.6 Definición de los Casos de Uso.

2.6.1 Definición del actor del sistema.

Tabla 8. Descripción detallada del actor del sistema.

Actor	Descripción
Administrador del SIAI	Es el encargado de controlar los procesos que intervienen en el funcionamiento del SIAI, realizar las operaciones básicas de control sobre procesos específicos, e iniciar y finalizar el sistema completo.

2.6.2 Diagrama de caso de uso del sistema.

A partir de los requisitos funcionales identificados previamente y el actor definido, se realizó el diagrama de casos de usos del sistema referente a la **Figura 5** con el objetivo de lograr un mejor entendimiento de las acciones que pueden ser realizadas por el personal que interactúa con la aplicación. Los casos de usos presentados simbolizan las principales funcionalidades del sistema.

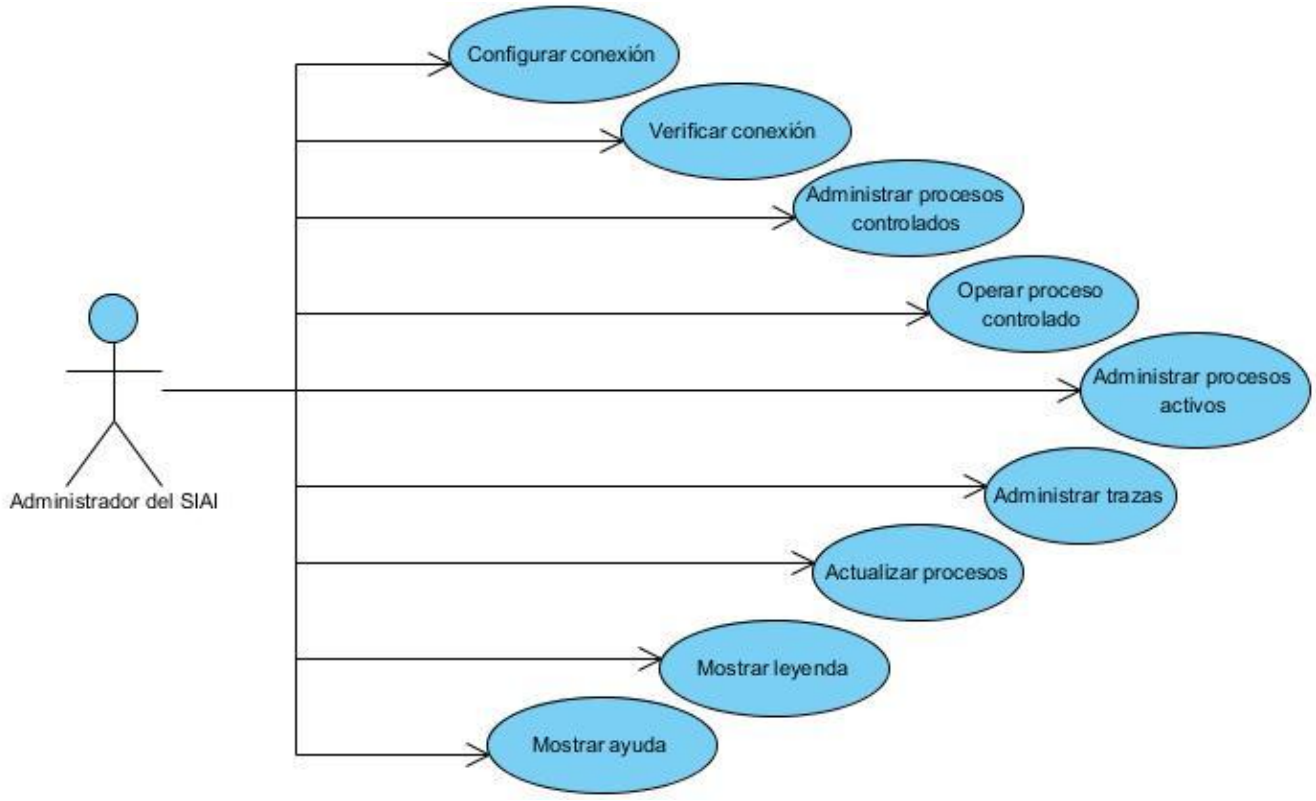


Figura 5. Casos de uso del sistema.

2.6.3 Descripción textual de caso de usos.

La descripción textual de los casos de usos se realiza con el objetivo de ilustrar la interacción del actor con el sistema, permitiendo un mejor análisis y comprensión de cada funcionalidad. A continuación se muestra en la **Tabla 9** la descripción de uno de los CU críticos. Las restantes descripciones de los CU pueden ser encontradas en el Anexo1.

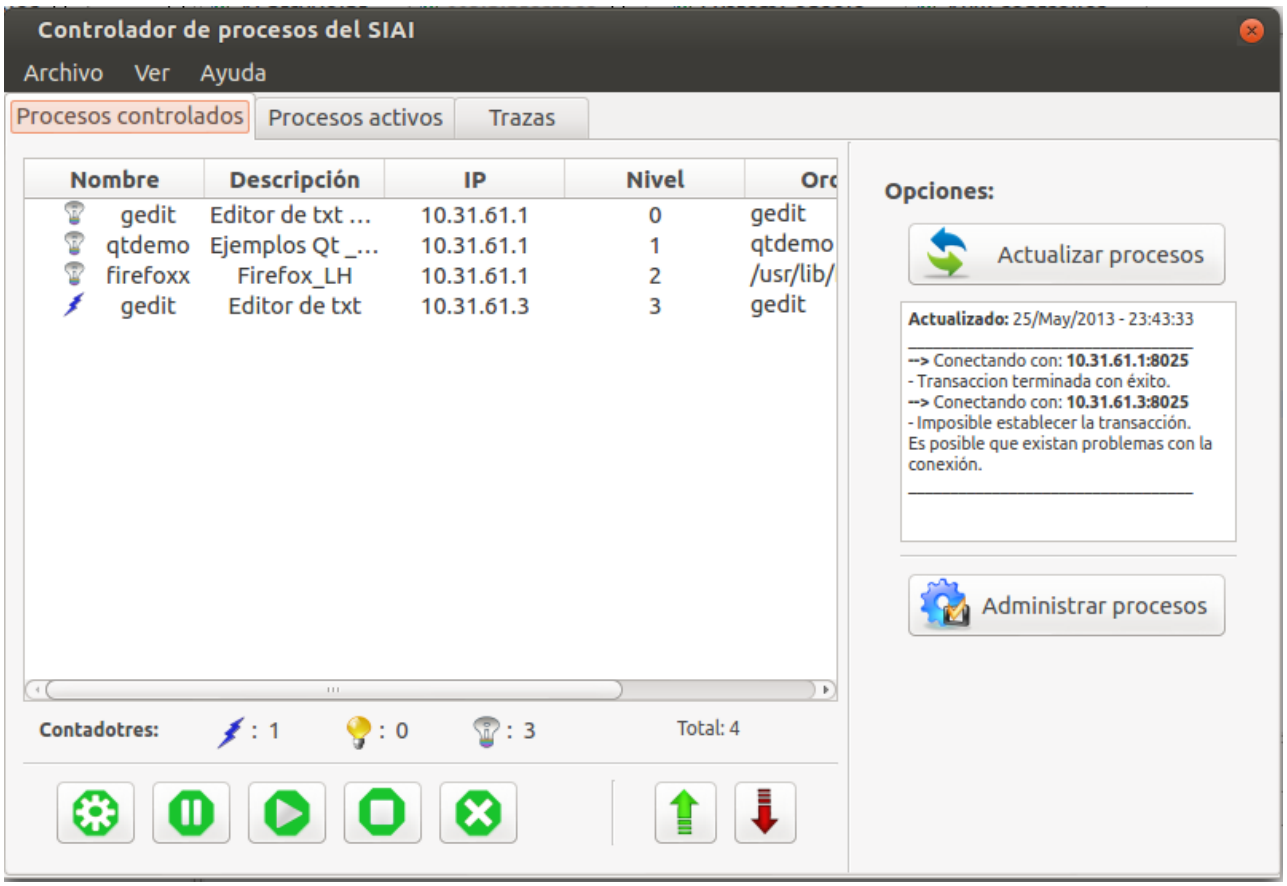
CU Operar proceso controlado.

Tabla 9. Descripción textual CU Operar proceso controlado.

Caso de Uso:	Operar proceso controlado.
Actores:	Administrador del SIAI (Inicia)
Resumen:	El caso de uso se inicia cuando el administrador decide operar sobre los procesos controlados o procesos que intervienen en el funcionamiento del SIAI.

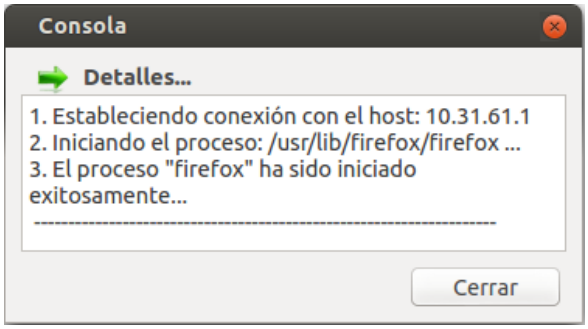
Precondiciones:	
Referencias	RF9
Prioridad	Crítico

Flujo Normal de Eventos	
Sección "Operar proceso controlado"	
Acción del Actor	**Respuesta del Sistema**
1. Accede a la sección de *"Procesos controlados"*.	2. Muestra la interfaz principal listando los procesos controlados configurados previamente, permitiendo realizar las opciones referentes al **panel principal** y a la **barra de operaciones** (Ver manual de usuario): ✓ Iniciar proceso. (Ir a la sección Iniciar proceso) ✓ Detener proceso. (Ir a la sección Detener proceso) ✓ Continuar proceso. (Ir a la sección Continuar proceso) ✓ Finalizar proceso. (Ir a la sección Finalizar proceso) ✓ Matar proceso. (Ir a la sección Matar proceso) ✓ Iniciar sistema. (Ir a la sección Levantar sistema.) ✓ Finalizar sistema. (Ir a la sección Finalizar sistema.) ✓ Actualizar procesos. (Se invoca al CU Actualizar procesos, ver descripción del mismo) ✓ Administrar procesos. (Se invoca al CU Administrar procesos controlados, ver descripción del mismo)

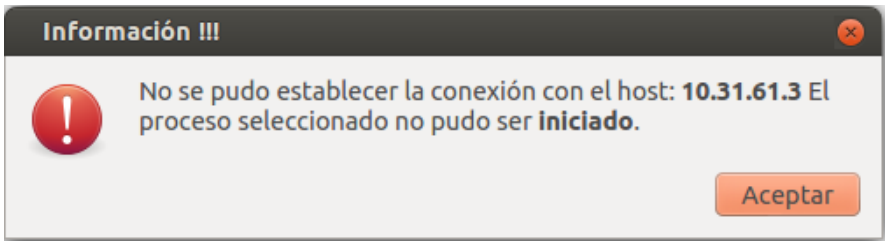


Flujo Normal de Eventos	
Sección "Iniciar proceso"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona el proceso que desea iniciar en el listado de procesos controlados. Selecciona la opción "*Iniciar*".	2. Muestra el siguiente mensaje de confirmación: "El comando **<orden>** va a ser iniciado en el host: **<ip>**. ¿Está seguro que quiere continuar con la operación?".
3. Selecciona la opción "Aceptar".	4. Verifica la conexión con la estación agente.
	5. Muestra una interfaz para ir mostrando los detalles de la operación.

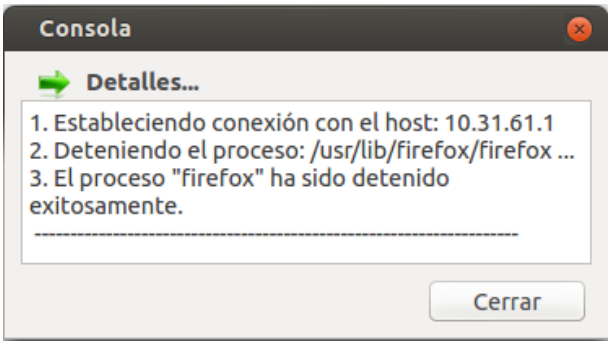
	6. Inicia el proceso seleccionado.
	7. Verifica la existencia del proceso iniciado.
	8. Registra la traza referente a la acción realizada.



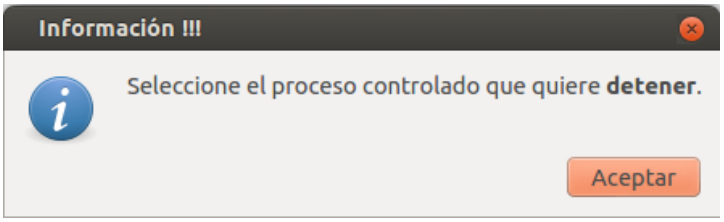
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Muestra el siguiente mensaje: “Seleccione el proceso controlado que quiere iniciar.”.
3.1. Selecciona la opción “*Cancelar*”.	3.2. Desmarca el proceso controlado seleccionado.
	4.1 Muestra el siguiente mensaje: “No se pudo establecer la conexión con el host: **<ip>**. El proceso seleccionado no pudo ser iniciado.”.

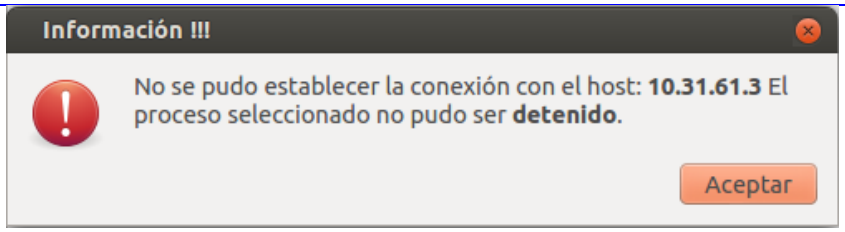


Flujo Normal de Eventos	
Sección "Detener proceso"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona el proceso que desea detener en el listado de procesos controlados. Selecciona la opción "*Detener*".	2. Muestra el siguiente mensaje de confirmación: "El proceso que registra el CMD: **<cmd>** va a ser detenido en el host: **<ip>**. ¿Está seguro que quiere continuar la operación?".
3. Selecciona la opción "*Aceptar*".	4. Verifica la conexión con la estación agente.
	5. Muestra una interfaz para ir mostrando los detalles de la operación.
	6. Detiene el proceso seleccionado.
	7. Verifica la existencia del proceso detenido.
	8. Registra la traza referente a la acción realizada.

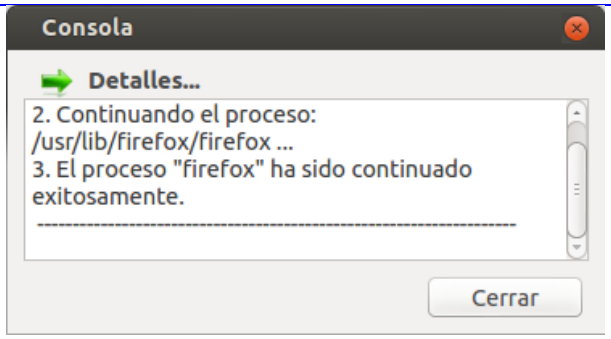


Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Muestra el siguiente mensaje: “Seleccione el proceso controlado que quiere detener.”.
3.1. Selecciona la opción “*Cancelar*”.	3.2. Desmarca el proceso controlado seleccionado.
	4.1 Muestra el siguiente mensaje: “No se pudo establecer la conexión con el host: **<ip>**. El proceso seleccionado no pudo ser detenido.”.

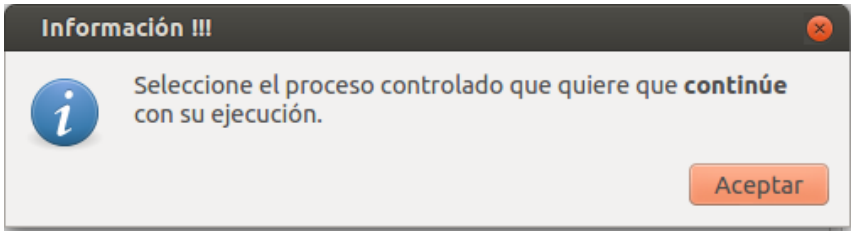


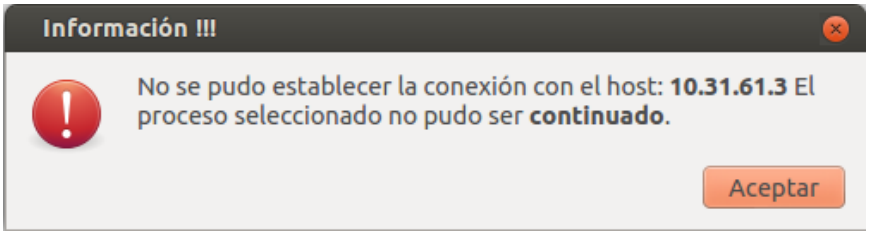


Flujo Normal de Eventos	
Sección "Continuar proceso"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona el proceso que desea que continúe con su ejecución en el listado de procesos controlados. Selecciona la opción "*Continuar*".	2. Muestra el siguiente mensaje de confirmación: "El proceso que registra el CMD: **<cmd>** va a ser continuado en el host: **<ip>**. ¿Está seguro que quiere continuar la operación?".
3. Selecciona la opción "*Aceptar*".	4. Verifica la conexión con la estación agente.
	5. Muestra una interfaz para ir mostrando los detalles de la operación.
	6. Continúa la ejecución del proceso seleccionado.
	7. Verifica la existencia del proceso continuado.
	8. Registra la traza referente a la acción realizada.



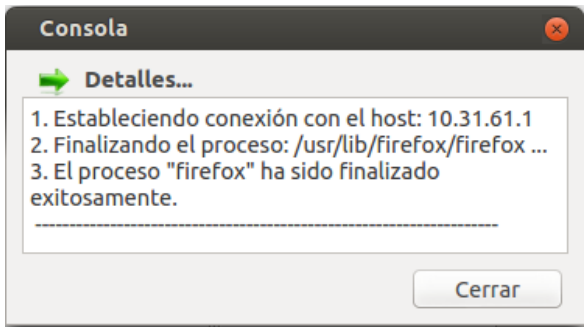
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Muestra el siguiente mensaje: “Seleccione el proceso controlado que quiere que continúe con su ejecución.”.
3.1. Selecciona la opción “*Cancelar*”.	3.2. Desmarca el proceso controlado seleccionado.
	4.1 Muestra el siguiente mensaje: “No se pudo establecer la conexión con el host: **<ip>**. El proceso seleccionado no pudo ser continuado.”.



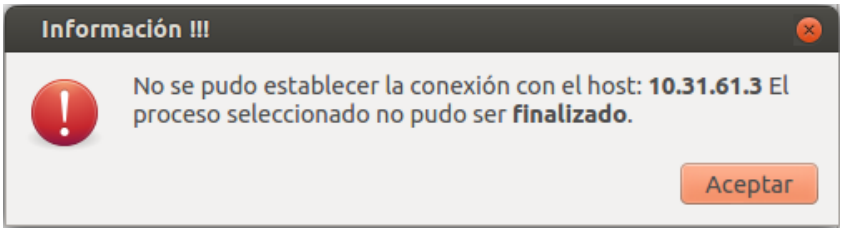


Flujo Normal de Eventos

<table>
<tr><th colspan="2">Sección “Finalizar proceso”</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>1. Selecciona el proceso que desea finalizar en el listado de procesos controlados.
Selecciona la opción “Finalizar”.</td><td>2. Muestra el siguiente mensaje de confirmación: “El proceso que registra el CMD: <cmd> va a ser finalizado en el host: <ip>. ¿Está seguro que quiere continuar la operación?”.</td></tr>
<tr><td>3. Selecciona la opción “Aceptar”.</td><td>4. Verifica la conexión con la estación agente.</td></tr>
<tr><td></td><td>5. Muestra una interfaz para ir mostrando los detalles de la operación.</td></tr>
<tr><td></td><td>6. Finaliza el proceso seleccionado.</td></tr>
<tr><td></td><td>7. Verifica la existencia del proceso finalizado.</td></tr>
<tr><td></td><td>8. Registra la traza referente a la acción realizada.</td></tr>
<tr><td colspan="2"> 

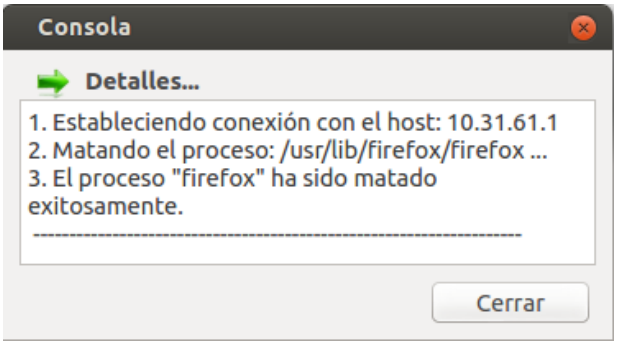

</td></tr>
<tr><th colspan="2">Flujos Alternos</th></tr>
</table>

Acción del Actor	Respuesta del Sistema
	2.1 Muestra el siguiente mensaje: “Seleccione el proceso controlado que quiere que finalizar.”.
3.1. Selecciona la opción “*Cancelar*”.	3.2. Desmarca el proceso controlado seleccionado.
	4.1 Muestra el siguiente mensaje: “No se pudo establecer la conexión con el host: **<ip>**. El proceso seleccionado no pudo ser continuado.”.

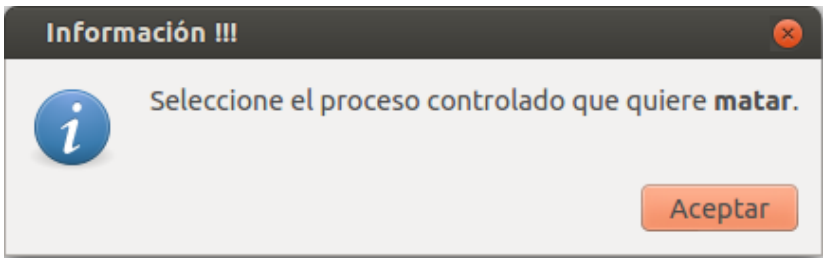


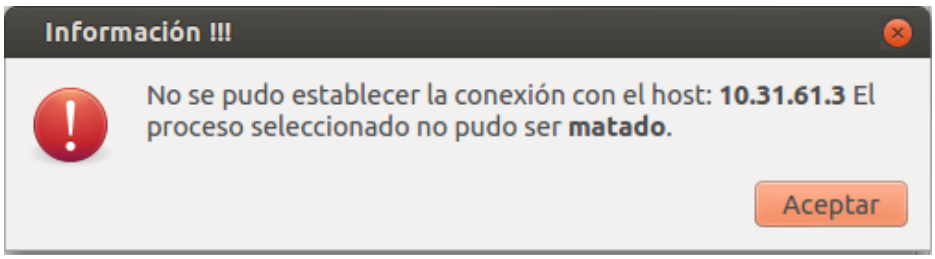
Flujo Normal de Eventos	
Sección “Matar proceso”	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona el proceso que desea matar en el listado de procesos controlados. Selecciona la opción “*Matar*”.	2. Muestra el siguiente mensaje de confirmación: “El proceso que registra el CMD: **<cmd>** va a ser matado en el host **<ip>**. ¿Está seguro que quiere continuar la operación?”.
3. Selecciona la opción “*Aceptar*”.	4. Verifica la conexión con la estación agente.

	5. Muestra una interfaz para ir mostrando los detalles de la operación.
	6. Mata el proceso seleccionado.
	7. Verifica la existencia del proceso matado.
	8. Registra la traza referente a la acción realizada.



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Muestra el siguiente mensaje: “Seleccione el proceso controlado que quiere que matar.”.
3.1. Selecciona la opción “*Cancelar*”.	3.2. Desmarca el proceso controlado seleccionado.
	4.1 Muestra el siguiente mensaje: “No se pudo establecer la conexión con el host: **<ip>**. El proceso seleccionado no pudo ser matado.”.

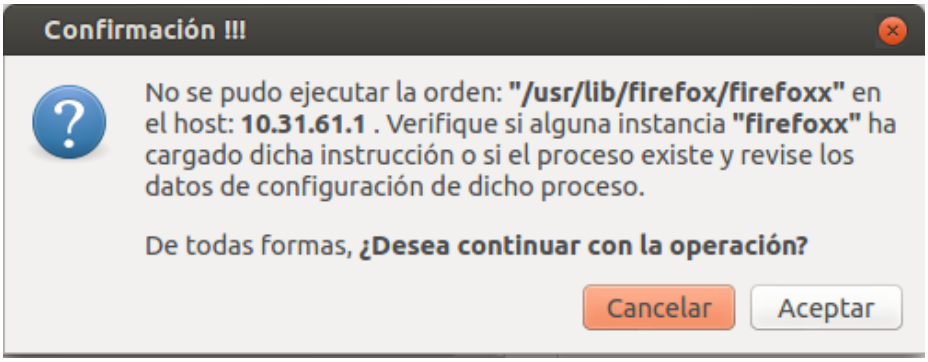


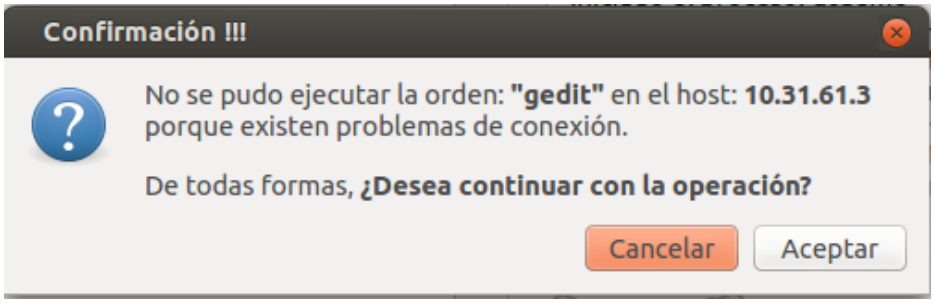


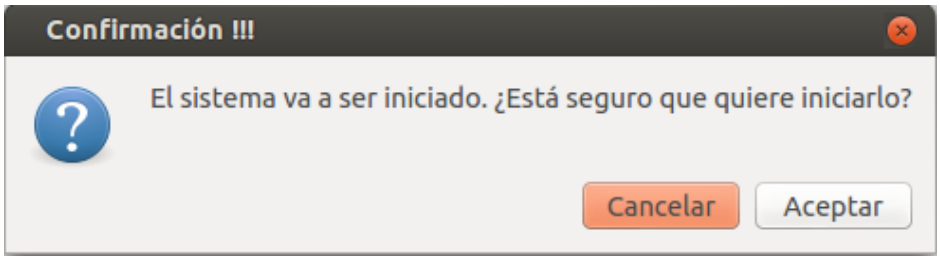
Flujo Normal de Eventos	
Sección "Iniciar sistema"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción "*Iniciar sistema*".	2. Muestra el siguiente mensaje de confirmación: "El sistema va a ser iniciado. ¿Está seguro que quiere iniciarlo?". Brindando la posibilidad de acceder a las opciones: ✓ *"Aceptar"* ✓ *"Cancelar"*
3. Selecciona la opción *"Aceptar"*.	4. Establece el orden de inicio del conjunto de procesos controlados teniendo en cuenta los niveles de dependencias existentes.
	5. Muestra una interfaz indicando el listado de los procesos y los detalles de la operación.
	6. Verifica la conexión con las respectivas estaciones agentes referentes a los procesos involucrados por cada

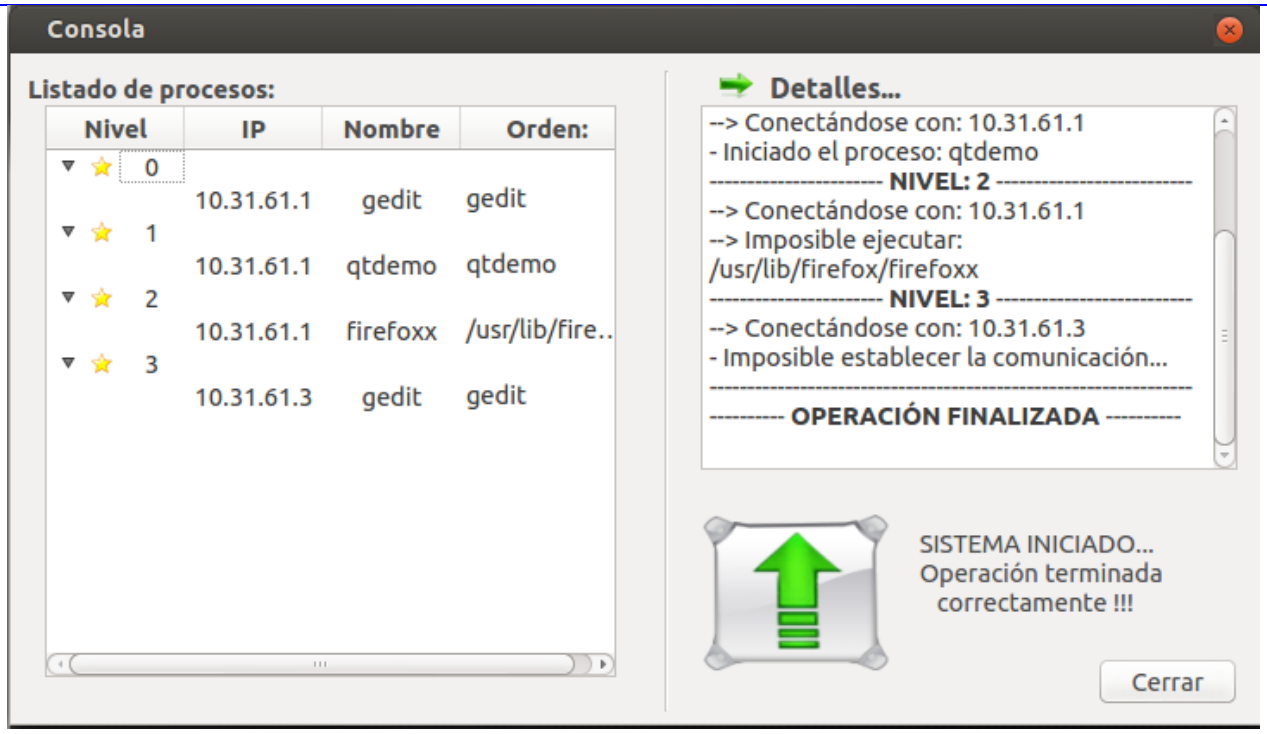
	nivel.
	7. Si no se encuentra alguna estación agente: Muestra el/los siguiente(s) mensaje(s): “No se pudo ejecutar la orden: **“<cmd>”** en el host **“<IP>”**, porque existen problemas de conexión. De todas formas, ¿Desea continuar con la operación?” Brindando la posibilidad de acceder a las opciones: ✓ *“Aceptar”* ✓ *“Cancelar”* Sino: continúa con la acción 9.
8. Selecciona la opción “*Aceptar*”.	9. Inicia el conjunto de procesos controlados según el orden establecido en el nivel correspondiente.
	10. Verifica que los procesos involucrados se hallan iniciado correctamente.
	11. Si al menos un proceso falla: Muestra el siguiente mensaje: “No se pudo ejecutar la orden: **“<cmd>”** en el host **“<IP>”**. Verifique si alguna instancia **‘’<nombre>”** ha cargado dicha instrucción o si el proceso existe. De todas formas revise los datos de configuración de dicho proceso. ¿Desea continuar con la operación?” Brindando la posibilidad de acceder a las opciones: ✓ *“Aceptar”* ✓ *“Cancelar”* Sino: continúa con la acción 14.
12. Selecciona la opción “*Aceptar*”.	13. Retorna a la acción 9.

	14. Registra la traza referente a la acción realizada.

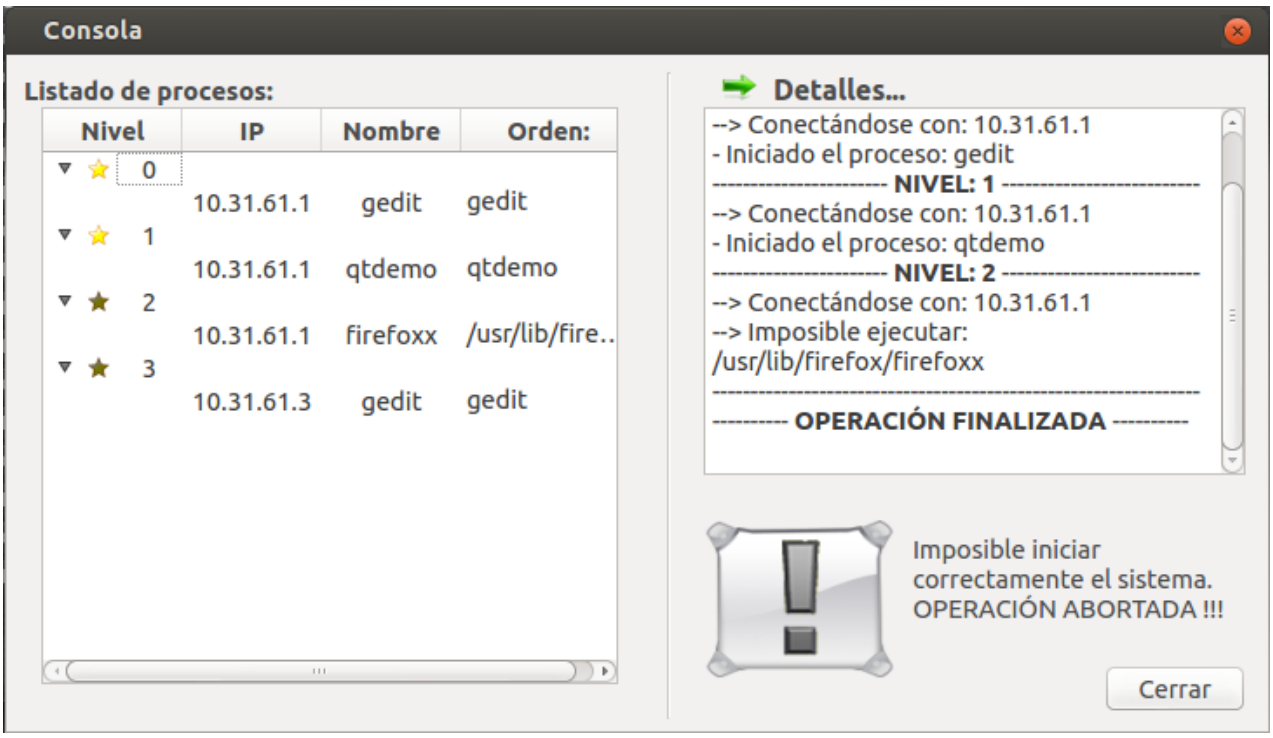








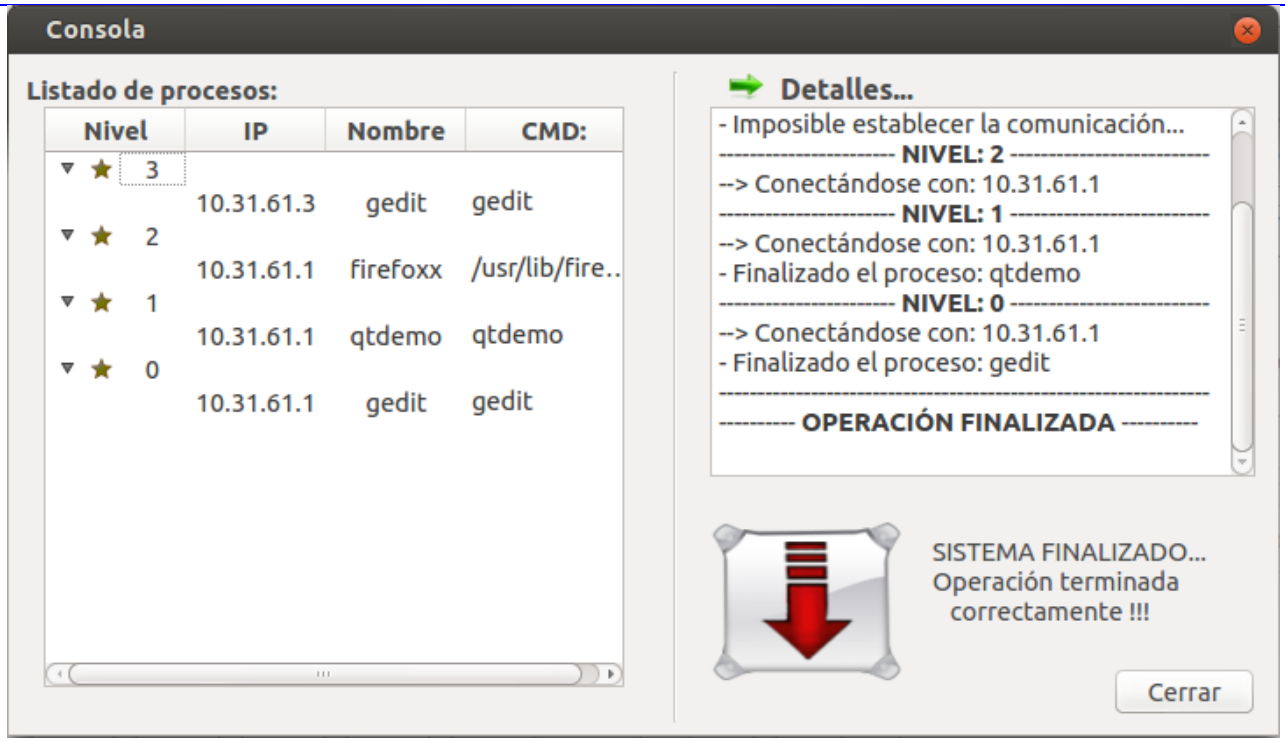
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3.1. Selecciona la opción "*Cancelar*".	3.2. Retorna a la acción 2 de la sección "Operar proceso controlado".
8.1 Selecciona la opción "*Cancelar*".	8.2 Aborta la operación manteniendo los procesos iniciados hasta el momento.
12.1 Selecciona la opción *"Cancelar".*	12.2 Aborta la operación manteniendo los procesos iniciados hasta el momento.



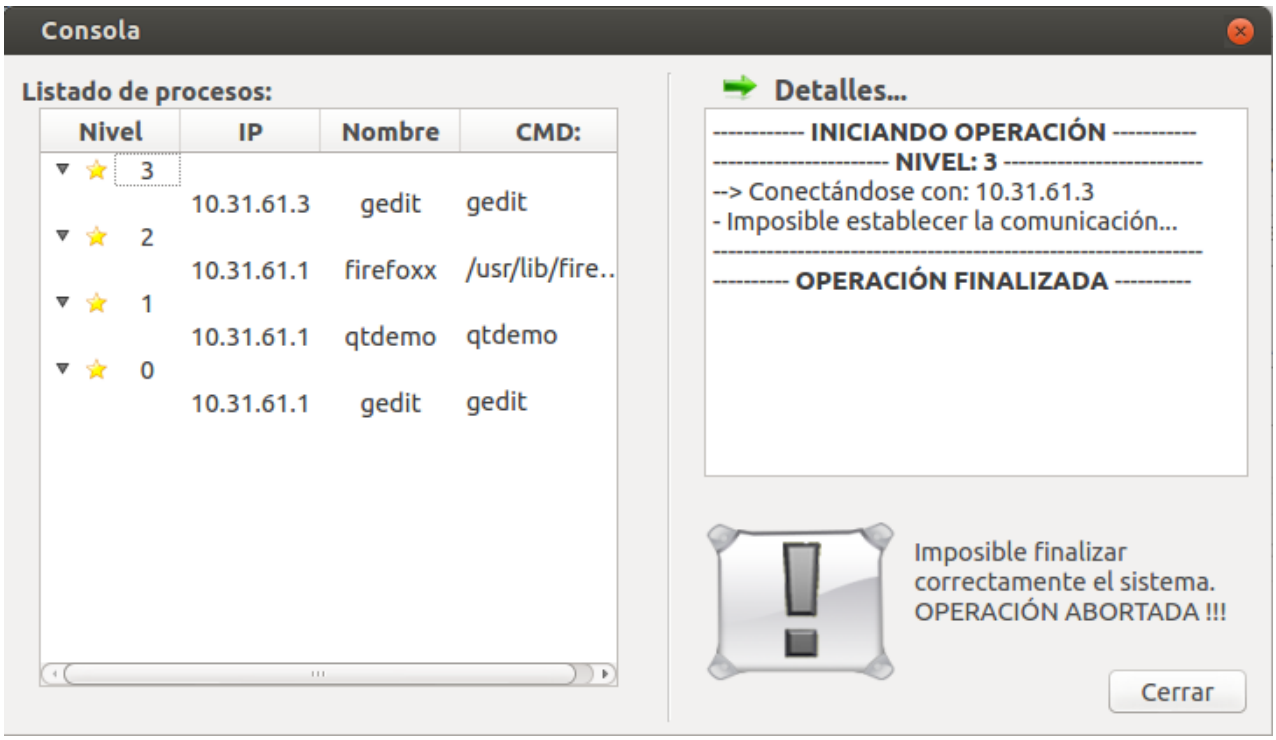
Flujo Normal de Eventos	
Sección "Finalizar sistema"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción "*Finalizar sistema*".	2. Muestra el siguiente mensaje de confirmación: "El sistema va a ser finalizado. ¿Está seguro que quiere finalizarlo?". Brindando la posibilidad de acceder a las opciones: ✓ *"Aceptar"* ✓ *"Cancelar"*
3. Selecciona la opción *"Aceptar"*.	4. Establece el orden de finalización del conjunto de procesos controlados teniendo en cuenta los niveles de dependencias existentes.

	5. Muestra una interfaz indicando el listado de los procesos y los detalles de la operación.
	6. Verifica la conexión con las respectivas estaciones agentes referentes a los procesos involucrados por cada nivel.
	7. Si no se encuentra alguna estación agente: Muestra el/los siguiente(s) mensaje(s): “No se pudo finalizar el proceso con CMD: **“<cmd>”** en el host **“<IP>”**, porque existen problemas de conexión. De todas formas, ¿Desea continuar con la operación?” Brindando la posibilidad de acceder a las opciones: ✓ *“Aceptar”* ✓ *“Cancelar”* Sino: continúa con la acción 9.
8. Selecciona la opción “*Aceptar*”.	9. Finaliza el conjunto de procesos controlados según el orden establecido en el nivel correspondiente.
	10. Verifica que los procesos involucrados se hallan finalizados correctamente.
	11. Si al menos un proceso falla: Muestra el siguiente mensaje: “No se pudo finalizar el proceso con CMD: **“<cmd>”** en el host **“<IP>”**. Verifique que el sistema agente tenga permisos para finalizarlo y rectifique los datos configurados referentes al proceso controlado. De todas formas, ¿Desea continuar con la operación?” Brindando la posibilidad de acceder a las opciones:

	✓ *“Aceptar”* ✓ *“Cancelar”* Sino: continúa con la acción 14.
12. Selecciona la opción “*Aceptar*”.	13. Retorna a la acción 9.
	14. Registra la traza referente a la acción realizada.



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3.1. Selecciona la opción *"Cancelar"*.	3.2. Retorna a la acción 2 de la sección "Operar proceso controlado".
8.1 Selecciona la opción *"Cancelar"*.	8.2 Aborta la operación manteniendo los procesos finalizados hasta el momento.
12.1 Selecciona la opción *"Cancelar"*.	12.2 Aborta la operación manteniendo los procesos finalizados hasta el momento.



Poscondiciones:	Teniendo en cuenta la selección del usuario: 1. Se inicia un proceso controlado seleccionado perteneciente al respectivo listado. 2. Se detiene un proceso controlado seleccionado perteneciente al respectivo listado, que haya sido iniciado anteriormente. 3. Se le da continuidad a un proceso controlado seleccionado perteneciente al respectivo listado, que haya sido detenido anteriormente. 4. Se finaliza un proceso controlado seleccionado perteneciente al respectivo listado, que haya sido iniciado anteriormente. 5. Se mata un proceso controlado seleccionado perteneciente al respectivo listado, que haya sido iniciado anteriormente. 6. Se inicia el sistema teniendo en cuenta los procesos controlados configurados previamente. 7. Se finaliza el sistema teniendo en cuenta los procesos controlados

	configurados previamente.

2.7 Conclusiones del capítulo.

En el presente capítulo se abordaron los aspectos básicos que van a fomentar la propuesta del sistema realizada, partiendo de un análisis y descripción de las funcionalidades que deberá poseer. Se enfatizó en cómo se lleva a cabo el proceso de “Controlar los procesos que intervienen en el funcionamiento del SIAI”, para posteriormente detallar los requisitos funcionales del sistema así como sus cualidades o propiedades (requisitos no funcionales). También se especificó detalladamente cada funcionalidad mediante la descripción textual de casos de uso, con la finalidad de crear bases sólidas para el posterior diseño e implementación del sistema.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción.

En este capítulo se describen los patrones de diseño, estándares de arquitectura para de esta formar presentar el modelo de clases del diseño, el cual va ser una vista de cómo se llevará a cabo la implementación del sistema siguiendo los patrones definidos con el objetivo de estructurar y organizar la implementación. Todo el compendio de información que a continuación se muestra, va ayudar a desarrollar de forma flexible y sobre todo organizada el “Módulo de control de procesos del SIAI”. De manera general se describe cómo el sistema será realizado a partir de las funcionalidades previstas y las restricciones impuestas, por lo que se indica con precisión lo que se debe programar.

3.2 Arquitectura y patrones de diseño.

3.2.1 Arquitectura Cliente - Servidor

La arquitectura Cliente – Servidor representa una división lógica, donde la capacidad de proceso se reparte entre ambos, esto trae como ventaja un diseño mucho más claro del sistema. Esta arquitectura consiste básicamente en la comunicación entre dos nodos, un nodo cliente que realiza peticiones al nodo servidor, y este a su vez es capaz de recibir y procesar dicha información. El hecho de que la división cliente - servidor sea lógica, trae como resultado distintas características dentro de los sistemas cliente – servidor, variando en dependencia de los servicios que se quieran brindar con dicho sistema, pero hay que tener presente que la arquitectura básica es siempre la misma.

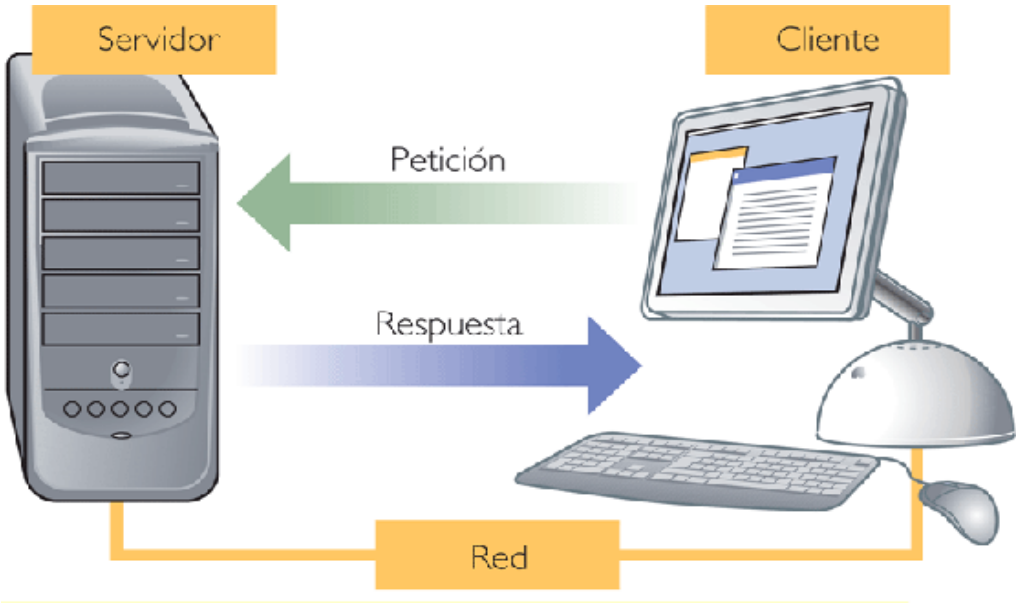


Figura 6. Arquitectura “Cliente - Servidor”.

Comunicación entre los sub-módulos agente y gestor.

El cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio). Englobando la posibilidad de que un cliente pueda hacer un gran número de peticiones a varios servidores y un servidor puede responder las solicitudes o peticiones de varios clientes. Considerando la lógica de la aplicación que se desea desarrollar y analizando que el modelo cliente-servidor provee usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones, se decidió utilizar este estilo arquitectónico.

En la actualidad existen implementaciones de tecnologías que brindan servicios orientados a la arquitectura Cliente – Servidor en múltiples lenguajes como C++, Java o Python, pero la creciente utilización de este tipo de arquitectura ha dado como resultado el surgimiento de diferentes protocolos de comunicación, los cuales permiten llamar funciones remotas con el objetivo fundamental de garantizar el tráfico de información entre dos sistemas. Entonces ¿qué tecnología emplear?, teniendo en cuenta fundamentalmente el lenguaje de programación seleccionado, se decidió utilizar una librería denominada **“**python-soappy**”**, la cual utiliza el protocolo SOAP (protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML[15]), empleando como

[15] Acrónimo inglés de **eXtensible Markup Language** traducido al español como **Lenguaje de Marcas Extensible**.

protocolo de transporte HTTP[16] para el envío y recibo de las peticiones. Además SOAP se considera un protocolo abierto y adaptable ya que permite la interoperabilidad entre distintos lenguajes, arquitecturas y sistemas operativos garantizando la invocación de funciones remotas de manera estructurada y organizada (utiliza HTTP y XML, HTTP+XML= SOAP) brindando la posibilidad de especificar diferentes parámetros de entrada en caso de que se requieran.

Específicamente en el "Módulo de control de procesos del SIAI", el sistema agente en esencia es un proceso que se ejecuta en segundo plano en cada una de las estaciones de trabajo pertenecientes a la subred del SIAI, el cual tiene como función principal publicar o brindar diferentes servicios, que son consultados por el sistema gestor, aplicación encargada de gestionar toda esta información, que finalmente es mostrada al usuario después de ser procesada teniendo en cuenta que ambas aplicaciones deben estar conectadas entre sí mediante una red de computadoras.

3.2.2 Patrones de diseño aplicados.

Experto en información (Experto)

El patrón experto en información es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).

Aplicación en el sistema:

Se utiliza en todo el desarrollo de la aplicación, delegando las responsabilidades a los componentes que poseen la información necesaria para realizar determinadas acciones, conservando el encapsulamiento. Por ejemplo, en el Sistema Gestor a clase "ManagerClient" y en el Sistema Agente la clase "AgentServer" son las encargadas de establecer la comunicación entre ambos sistemas.

[16] Acrónimo inglés de **Hypertext Transfer Protocol** traducido al español como **Protocolo de Transferencia de Hipertexto**.

Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. Su intención básica es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se favorece el bajo acoplamiento. (30)

Aplicación en el sistema:

Es necesario conocer cuál dentro de la estructura de clases de la aplicación es la idónea para instanciar los datos que brinden la solución esperada. Se utiliza a lo largo de todo el proceso de desarrollo del sistema en sentido general al crear instancias de clases necesarias para su posterior utilización. Por ejemplo, en el Sistema Gestor la clase “ProcessesData” se crea una instancia de la clase “ManagerClient”, para poder procesar los datos enviados por el Sistema Agente.

Alta cohesión y bajo acoplamiento

Los conceptos de cohesión y acoplamiento no están íntimamente relacionados, sin embargo se recomienda tener un mayor grado de cohesión con un menor grado de acoplamiento. De esta forma se tiene menor dependencia y se especifican los propósitos de cada objeto en el sistema.

- **Alta cohesión**

Consiste en que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.

Aplicación en el sistema:

Se utiliza en gran medida para evitar el exceso de responsabilidades sobre una misma clase evitando comportamientos innecesarios, otorgando de esta forma una gran claridad y facilidad para comprender el funcionamiento de la aplicación, permitiendo un alto grado de reutilización de los objetos. Este patrón se evidencia específicamente en las clases “ManagerClient” y “AgentServer” perteneciente a los sistemas gestor y agente respectivamente, ya que básicamente son las clases encargadas de controlar las acciones de acceso a datos.

- **Bajo acoplamiento**

Las clases deben tenerse lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. El acoplamiento es una

medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas.

Aplicación en el sistema:

Se utiliza en gran medida para evitar las dependencias excesivas o alto acoplamiento entre las clases. El sistema en sentido general deberá mantener una estructura reutilizable permitiendo la inserción de nuevas funcionalidades o la modificación de algunas ya existentes. Se evidencia su utilización en el Sistema Gestor en la clase “EntityProcess”, clase bastante utilizada pero con responsabilidades bien definidas para mantener un bajo margen de dependencias.

Controlador

El patrón Controlador sugiere que la lógica de negocios debe estar separada de la capa de presentación, para aumentar la reutilización de código y a la vez tener un mayor control. Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Aplicación en el sistema:

Se utiliza para asignar que clases deben tener la responsabilidad de controlar el funcionamiento de otras clases y del sistema en sentido general. En el Sistema Gestor se evidencia este patrón, ya que es ampliamente utilizado en la clase “MainInterface”, clase que se encargan de controlar los datos provenientes de la capa de presentación para separarlos de la lógica del negocio, además de manejar casi todos los eventos del sistema.

3.3 Diagrama de Paquetes.

Un paquete es un conjunto de cualquier tipo de elementos de un modelo: clases, casos de uso, diagramas de colaboración u otros paquetes (los anidados). El paquete define un espacio de un nombre anidado, de modo que los elementos del mismo nombre, pueden duplicarse dentro de varios paquetes. (30) A continuación se muestra cómo quedarían estructurado los paquetes para cada uno de los sistemas (gestor y agente).

3.3.1 Sistema gestor.

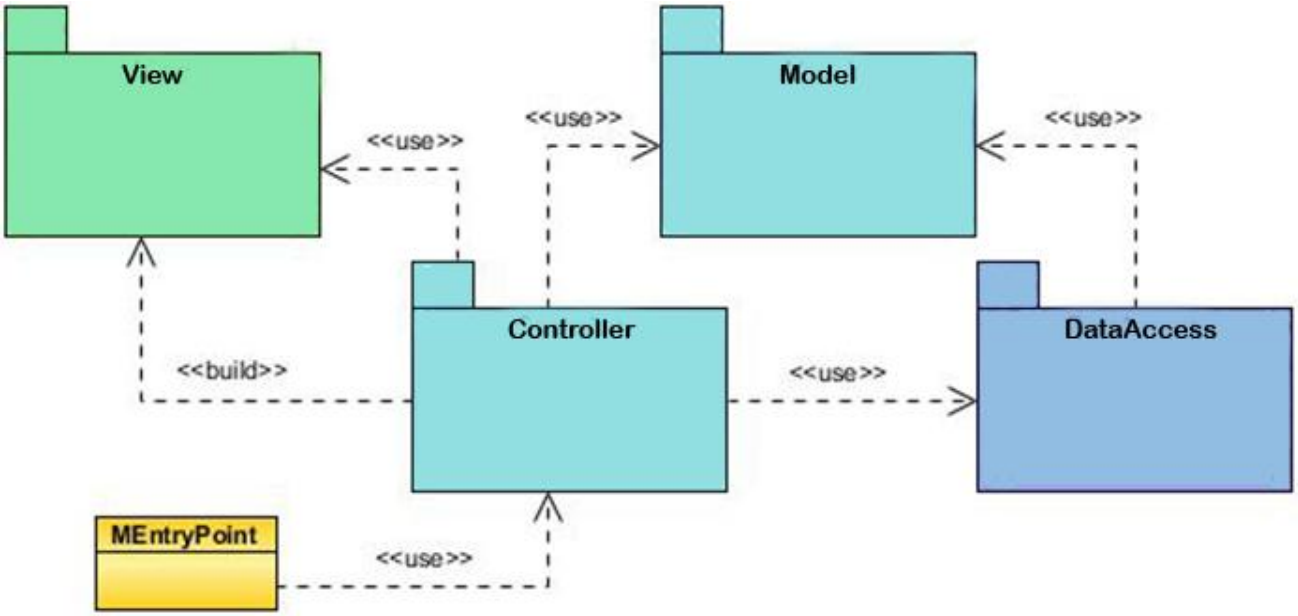


Figura 7. Diagrama de Paquetes del “Sistema gestor”.

3.3.2 Sistema agente.

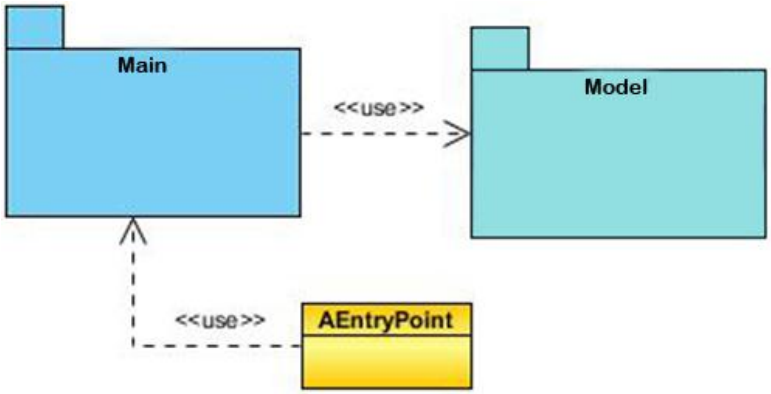


Figura 8. Diagrama de Paquetes del “Sistema agente”.

3.4 Diagramas de clases del diseño.

Un diagrama de clases del diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Estos diagramas son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la

relación entre uno y otro. Describen gráficamente las especificaciones de las clases de software y las interfaces.

Para ver la descripción detallada de cada uno de los diagramas de clase de diseño de los paquetes anteriores, referirse al Anexo 2.

3.4.1 CU Operar proceso controlado.

Sistema gestor.

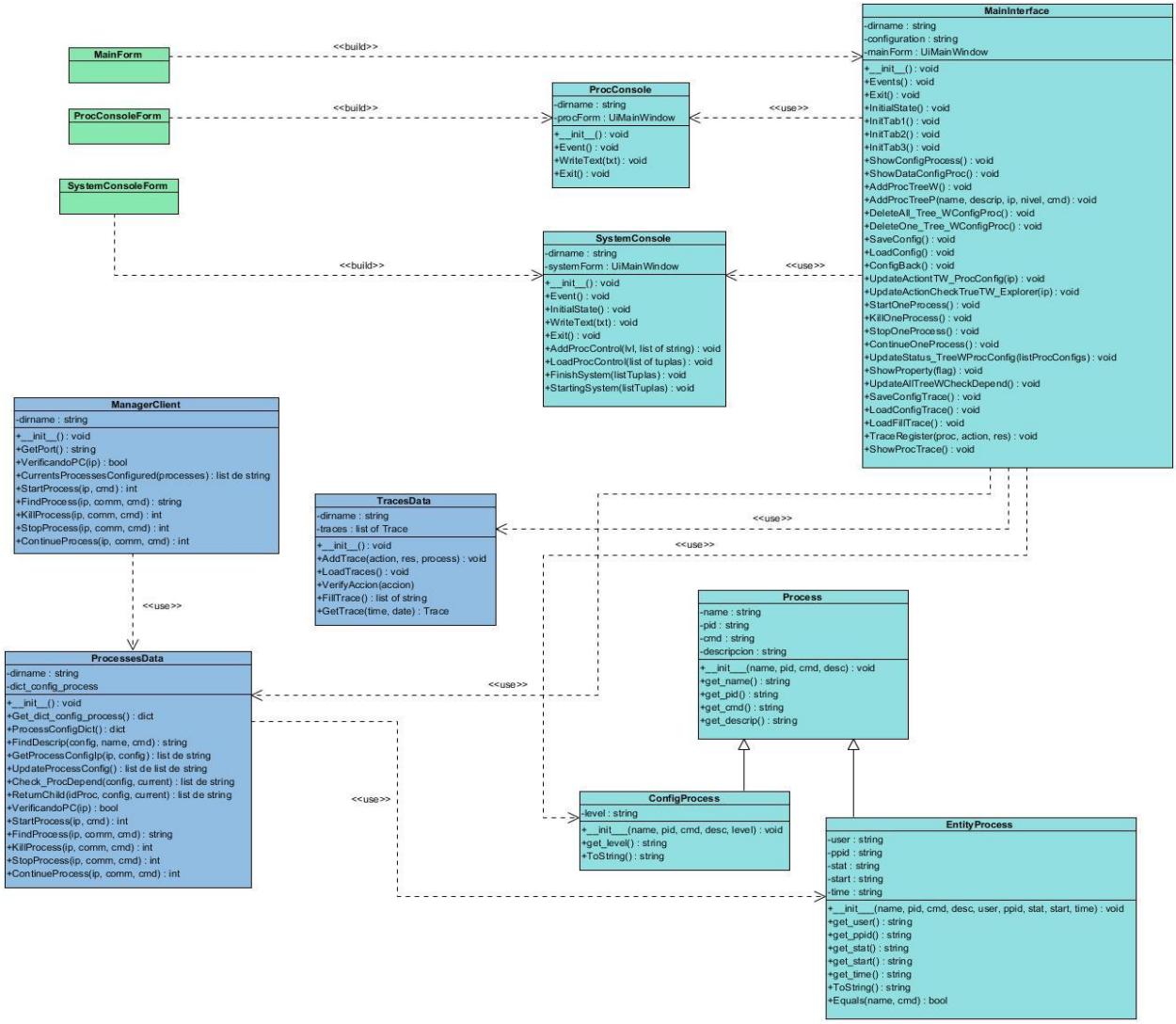


Figura 9. Diagrama de clases del diseño del CU Operar proceso controlado para el “Sistema gestor”.

Sistema agente.

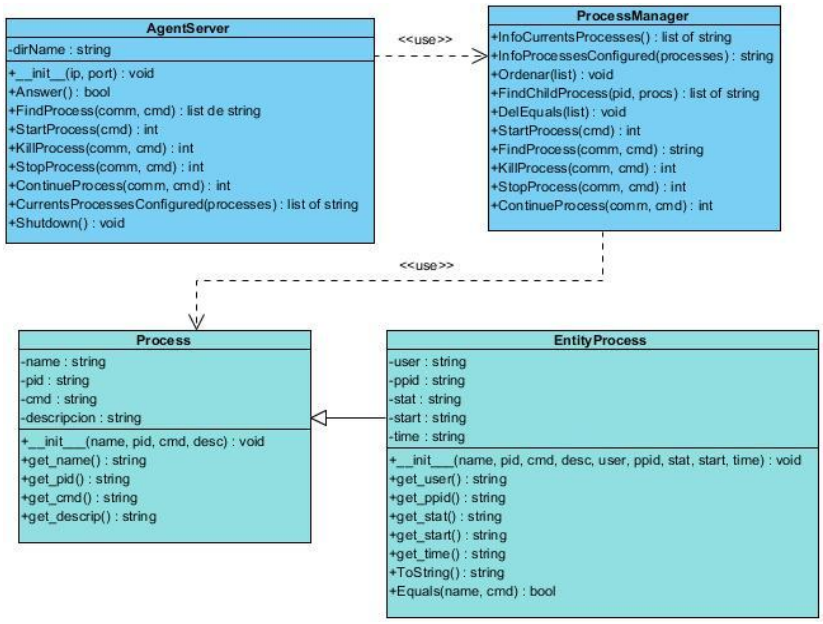


Figura 10. Diagrama de clases del diseño del CU Operar proceso controlado para el “Sistema agente”.

3.5 Conclusiones del capítulo.

En este capítulo quedó plasmada la arquitectura que se seleccionó para la implementación del Módulo de control de procesos del SIAI, además de definir los patrones de diseños candidatos a utilizar teniendo siempre en cuenta las buenas prácticas de programación que permitirán para versiones posteriores una mayor reusabilidad y operabilidad en caso de futuros cambios. Además se expusieron los diferentes diagramas de paquetes y de clases del diseño de los sistemas gestor y agente respectivamente con el objetivo de garantizar un mayor entendimiento y sentar las bases para el éxito de la implementación del sistema propuesto.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS

4.1 Introducción.

En el presente capítulo se generan los artefactos: Diagrama de despliegue, Diagrama de componentes y Casos de prueba, los cuales en gran medida describen la forma en que quedará el sistema después de concluida la fase de implementación, con el objetivo de mostrar la organización y las dependencias lógicas entre un conjunto de componentes de software, como son código fuente, librerías, binarios, ejecutables o documentos utilizados por el sistema. Por último se describen los casos de pruebas realizados al sistema con el objetivo de capturar e identificar los posibles errores que se pudieron haber cometido en la fase de implementación para una pronta corrección de los mismos.

4.2 Diagrama de despliegue.

Los diagramas de despliegue muestran a los nodos procesadores, la distribución de los procesos y de los componentes (…); (30) Es decir, describen la arquitectura física del sistema durante la ejecución, en términos de: nodos procesadores, dispositivos y componentes de software. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

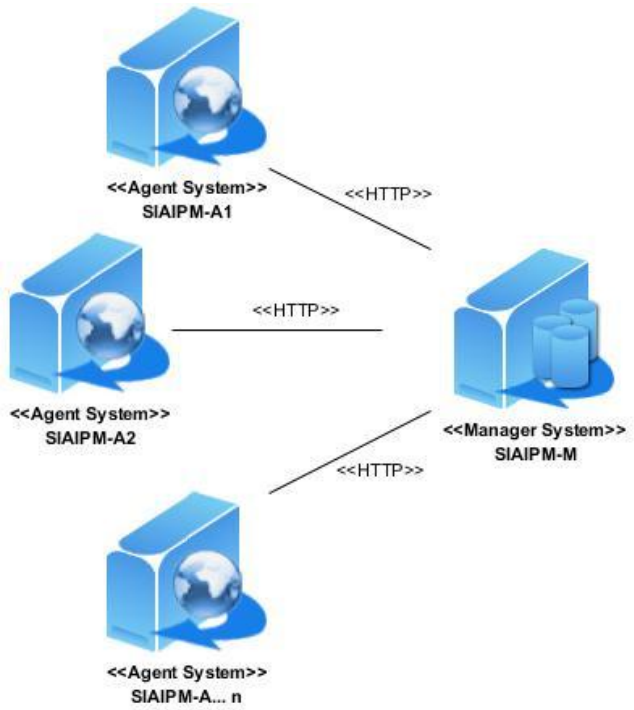


Figura 11. Diagrama del despliegue del **"**Módulo de control de procesos del SIAI**"**.

4.2.1 Descripción de los nodos del Diagrama de despliegue.

A continuación se describen los nodos que componen el diagrama de despliegue:

Tabla 10. Descripción del diagrama de despliegue.

Nodo	Descripción
	Sistema agente que deberá correr en cada host o estación de trabajo perteneciente a la red del SIAI que contenga procesos candidatos a ser controlados.
	Sistema gestor que deberá correr en un host, servidor o estación de trabajo perteneciente a la red del SIAI.

4.3 Diagrama de componentes.

Los diagramas de componentes se emplean para el modelamiento de las vistas estáticas de un sistema en términos de componentes, como código fuente, binarios, librerías o ejecutables, así como de las dependencias lógicas que existen entre ellos. (20)

4.3.1 Diagrama de componentes general del Sistema gestor.

En la **Figura 12** se muestra el diagrama de componentes del Sistema gestor. El paquete "View" incluye los componentes referentes a las interfaces de usuario, las cuales son construidas por sus respectivas clases controladoras pertenecientes al paquete "Controller", que contiene los componentes encargados de obtener la información necesaria para poder desarrollar la lógica del negocio descrita. El paquete "Model" contiene los componentes referentes a las entidades modelos y el de "DataAccess" es el encargado de obtener la información gestionada por el Sistema agente y de enviarla a los respectivos componentes controladores. Además el paquete "Resources" contiene los recursos utilizados por el sistema, el "Config" los ficheros de configuraciones necesarias y un componente aislado cuya función es ser el punto de entrada a la aplicación.

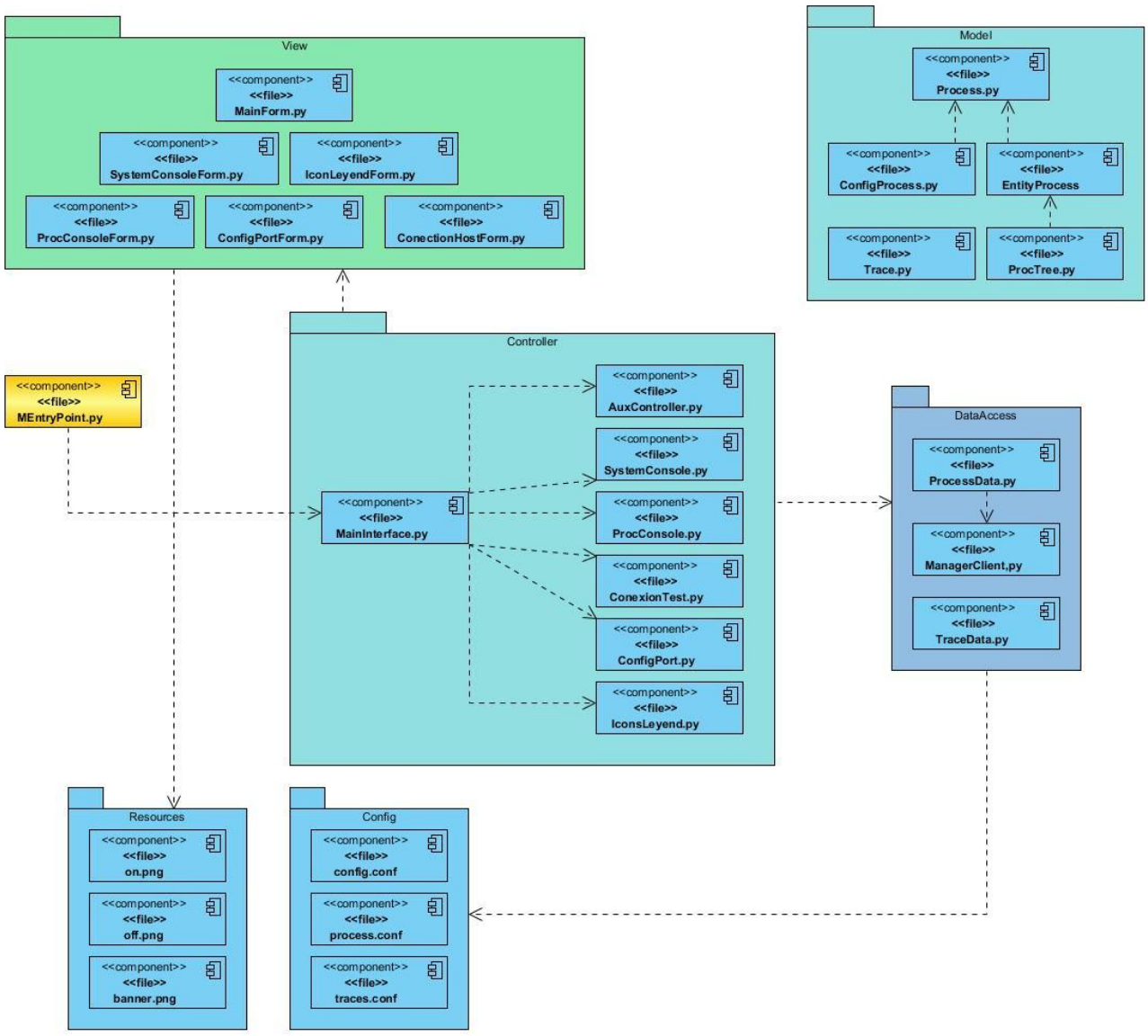


Figura 12. Diagrama de componentes del Sistema gestor del “Módulo de control de procesos del SIAI”.

4.3.2 Diagrama de componentes general del Sistema agente.

En la **Figura 13** se muestra el diagrama de componentes del Sistema agente, el cual cuenta con dos paquetes, en el paquete “Model” recoge los componentes necesarios referentes a las clases modelos utilizadas y el paquete “Main” contiene los componentes principales para el funcionamiento de este submódulo, ya que en esencia este paquete es el que se encarga de gestionar toda la información referente a los procesos controlados y de garantizar el acceso a esta información por parte del Sistema

Gestor. Aunque también se cuenta con otros componentes aislados como el punto de ejecución y el fichero de configuración del sistema.

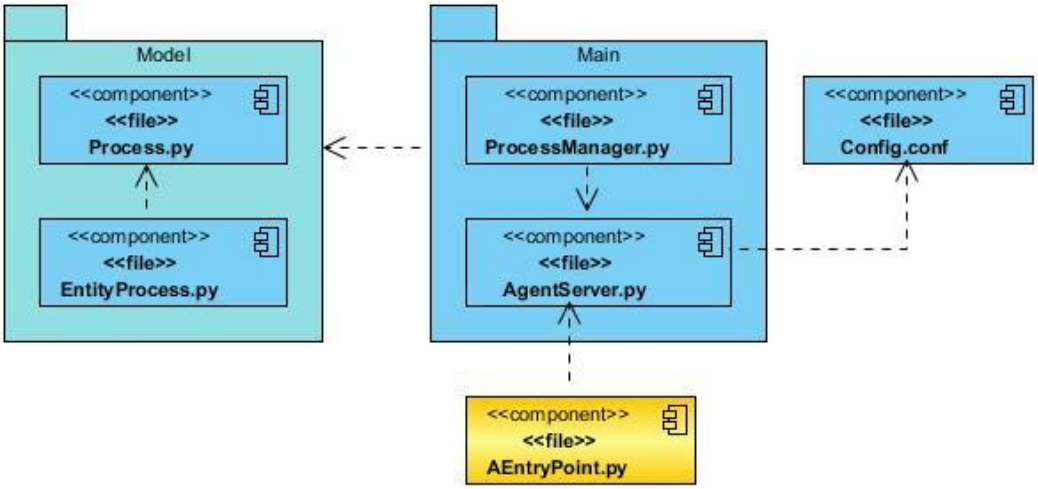


Figura 13. Diagrama de componentes del Sistema agente del "Módulo de control de procesos del SIAI".

4.4 Pruebas.

Dentro de los instrumentos capaces de medir el estado de calidad de un producto se encuentran las pruebas. El proceso de pruebas se dirige fundamentalmente a componentes del software o al sistema de software en general, con el objetivo de medir hasta cuando el software cumple las funcionalidades establecidas por el cliente. Las pruebas del software verifican y revelan la calidad de un producto. Son utilizadas para identificar posibles errores en la implementación, calidad, o usabilidad de un programa de software.

4.4.1 Estrategia de pruebas.

Luego de la implementación de la solución, para la verificación del funcionamiento de la misma fueron realizadas las pruebas. Se definió una estrategia con niveles, tipos de pruebas y métodos correspondientes, que permitieron solucionar errores que presentaba la aplicación y perfeccionar la solución implementada.

La estrategia elaborada hace énfasis en los niveles Unidad e Integración. Relacionados a cada nivel de pruebas, están los tipos de pruebas y estos a su vez se aplican a través de métodos. En la **Tabla 11** se puede apreciar la estrategia aplicada al sistema desarrollado.

Tabla 11. Descripción de la estrategia de pruebas.

Nivel de pruebas	Tipos de pruebas	Métodos
Unidad	Funcionalidad - Función	Caja Blanca
Integración	Funcionalidad - Función	Caja Negra

Niveles de pruebas.

- ✓ **Pruebas de Unidad**: Son el proceso de comprobar los componentes individuales en el sistema. Este es un proceso de prueba de defecto, por lo que sus objetivos es encontrar defectos en estos componentes. Los desarrolladores de los componentes son los responsables de probarlos.
  Existen diferentes objetos que pueden probarse en esta etapa:
  1. Funciones individuales o métodos dentro de un objeto.
  2. Clases de objetos que tiene varios atributos y métodos.
- ✓ **Pruebas de Integración:** El proceso de integración del sistema implica construir este a partir de sus componentes y probar el sistema resultante para encontrar problemas que puedan surgir debido a la integración de los componentes. Los componentes que se integran pueden ser componentes comerciales, componentes reutilizables que han sido adaptados a un sistema en particular o componentes nuevos desarrollados.

Tipos de pruebas.

- ✓ **Función:** Pruebas fijando su atención en la validación de las funciones, métodos, servicios, caso de uso.

Métodos de pruebas.

- ✓ **Prueba de caja blanca:** Comprueba los caminos lógicos del software, proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado.
- ✓ **Pruebas de caja negra:** Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software.

4.4.2 Resultados de las pruebas realizadas.

Pruebas de Unidad.

Esta prueba se realizó específicamente a la clase principal (clase que se encarga de recolectar la información necesaria para llevar a cabo la lógica del negocio) del Sistema agente, ya que es la fuente principal de la información en el sistema en sentido general y era sumamente necesario verificar la validez de los resultados devueltos. Para la ejecución de esta prueba se utilizó **"PyUnit"**, que es el framework estándar para desarrollar los casos de prueba de unidad, para programas desarrollados en el lenguaje Python. El mismo está diseñado para trabajar con cualquier versión de este lenguaje superior a la 1.5.2, además de ser multiplataforma. (31) Con la realización de este tipo de prueba de caja blanca se logró trabajar sobre los errores encontrados, como es el caso de los valores que se esperaban y el que realmente devolvía. Estos errores se consideraron no relevantes debido a que fueron solucionados, dando la posibilidad de brindar una solución final que no presenta dificultades después de la realización de dos iteraciones.

Pruebas de Integración.

Las pruebas de integración se realizaron a la solución desarrollada en cuatro iteraciones. En la primera iteración se probaron detalladamente las funcionalidades del sistema, detectando cinco no conformidades entre las que se destacan problemas referentes a las funcionalidades:

- ✓ Verificar la conexión con el agente perteneciente a una estación de trabajo especificada.
- ✓ Iniciar el sistema y finalizar el sistema, teniendo en cuenta el nivel de dependencia existente entre los procesos controlados configurados.

En la segunda iteración persistieron dos no conformidades referentes a las funcionalidades:

- ✓ Iniciar y finalizar el sistema teniendo en cuenta el nivel de dependencia existente entre los procesos controlados configurados.

En la tercera iteración se mantenía una no conformidad referente a la funcionalidad:

- ✓ Iniciar el sistema teniendo en cuenta el nivel de dependencia existente entre los procesos controlados configurados.

Mientras que en la última iteración se resolvió el problema existente, por lo que no se registraron no conformidades. Resultando finalmente un módulo funcional con las características descritas en los requerimientos del sistema, cumpliendo con los objetivos especificados.

Figura 16. Resultados de las pruebas de las pruebas de integración.

4.4.3 Entorno de pruebas.

Partiendo de los requisitos no funcionales que se definieron para el desarrollo de esta solución el entorno de pruebas utilizado para las aplicaciones agente y gestora respectivamente cuenta con las particulares siguientes:

Tabla 13. Características de las computadoras utilizadas para la realización de las pruebas.

Hardware	Datos
Procesador	Core Dos Duo 2.2 GHz
Memoria	1GB
Tarjeta madre	INTEL-965
Ancho de banda	100 Mbps
Sistema Operativo	Ubuntu 12.04

4.5 Conclusiones del capítulo.

En este capítulo se logró un mejor entendimiento del sistema desde el punto de vista de implementación y pruebas, presentado una descripción general y bien detallada de los artefactos generados. Mediante la realización del diagrama de despliegue se muestra cómo están distribuidos físicamente los dispositivos de software entre los diferentes nodos así como una breve descripción de los mismos. Además se describió como está organizado y relacionado los diferentes componentes que conforman el **"Módulo de control**

de procesos del SIAI” (ambos sub-módulos: el Sistema gestor y el Sistema agente) y se explicaron los detalles referentes a la estrategia de pruebas puesta en práctica conjuntamente con los resultados obtenidos y el entorno de pruebas empleado.

CONCLUSIONES GENERALES.

El resultado principal de este trabajo de investigación es la obtención de un sistema capaz de controlar y gestionar todos los procesos informáticos referentes a los servicios que conforman el SIAI, teniendo en cuenta la necesidad e interés de los administradores del mismo. Se cuenta con una documentación completa y bien detallada que se fue generando desde el comienzo, resaltando principalmente las etapas de diseño e implementación que sobre todo fueron guiadas por la metodología RUP. Con el propósito de darle cumplimiento al objetivo general y basado en la problemática expuesta, se llevaron a cabo satisfactoriamente cada una de las tareas que fueron identificadas al inicio de la investigación.

- ✓ Se realizó un estudio de las tendencias actuales referente a los diferentes sistemas distribuidos que controlan procesos, comenzando por las variantes libres con el objetivo de verificar si se podía utilizar de alguna de ellas como parte de la solución, pero esta variante fue descartada y se procedió a estudiar las opciones privativas para obtener una mejor visión a la hora de desarrollar la solución finalmente propuesta.
- ✓ La utilización de RUP como metodología de desarrollo de software con sus variantes modificadas, permitió guiar el proceso completo de desarrollo sobre todo generando una abundante documentación.
- ✓ Mediante la descripción de los proceso del negocio se identificaron las funcionalidades que debía poseer el sistema, añadiéndose además un conjunto de requisitos no funcionales que adecuan al sistema según las necesidades del cliente.
- ✓ Se propuso un modelo de análisis para describir y comprender como se llevaría a cabo la implementación del sistema teniéndose como punto de partida los requisitos identificados anteriormente. Además con el objetivo de crear un sistema escalable que permita añadir futuras funcionalidades se tuvieron en cuenta la utilización de diferentes estilos arquitectónicos y patrones de diseño que permitieron agilizar el proceso de desarrollo teniendo en cuenta las buenas prácticas de programación.
- ✓ Se presentaron los artefactos correspondientes para mostrar como deberá ser desplegado el sistema en su totalidad para su puesta en ejecución teniendo en cuenta los subsistemas que lo componen.
- ✓ Se definió una estrategia de pruebas haciendo énfasis en dos niveles sumamente importantes: Unidad e Integración. Donde una vez realizadas las pruebas se obtuvieron un conjunto de

inconformidades en la implementación, las cuales fueron corregidas después de varias iteraciones obteniendo finalmente resultados satisfactorios que garantizan la calidad del producto desarrollado.

Luego del análisis anterior se puede afirmar que la presente investigación alcanzó los objetivos propuestos.

RECOMENDACIONES

A partir de los resultados obtenidos con la realización del presente trabajo de diploma se hace necesario que queden plasmadas algunas recomendaciones para futuras versiones del sistema.

- ✓ Incluirle al sistema de forma general más funcionalidades que puedan ser utilizadas como el apagar o encender alguna PC específica a través de la red.
- ✓ Incorporarle un sistema agente o actualizarle el existente para controlar procesos específicos en otras plataformas como Windows en cualquiera de sus variantes.

BIBLIOGRAFÍA

1. **S.A., Empresa de telecomunicaciones de Bogotá.** *ETB.* [En línea] [Citado el: 13 de noviembre de 2012.] http://issuu.com/josarchila/docs/fraude_en_telecomunicaciones/1?mode=a_p.
2. **1.** About.com Computadoras. [En línea] [Citado el: 19 de noviembre de 2012.] http://computadoras.about.com/od/conoce-procesadores/a/Que-Es-Un-Procesador.htm.
3. **2.** SlideShare. [En línea] [Citado el: 19 de noviembre de 2012.] http://www.slideshare.net/jengibre/concepto-de-sistema-operativo-2072384.
4. **3.** SlideShare. [En línea] [Citado el: 19 de noviembre de 2012.] http://www.slideshare.net/aleesqueda/sistema-operativo-unix#btnNext.
5. **4.** (I) Investopedia. [En línea] [Citado el: 20 de noviembre de 2012.] http://www.investopedia.com/terms/l/lifo.asp#axzz2EULsANTp.
6. **Universidad de Oviedo.** Arquitectura y Tecnología de Computadoras. [En línea] [Citado el: 20 de noviembre de 2012.] www.atc.uniovi.es/telematica/2ac/Apuntes-y.../T08-Procesos.pdf.
7. **11.** Master Magazine. [En línea] [Citado el: 21 de noviembre de 2012.] http://www.mastermagazine.info/termino/6377.php.
8. **Stalling, William.** *Sistemas Operativos "Aspectos internos y principios de diseño".* Madrid : s.n., 2005.
9. **Berrocal, Abelardo Jara.** Blog: Ubuntu, electronica y software libre. [En línea] [Citado el: 21 de noviembtre de 2012.] http://pintucoperu.wordpress.com/2007/12/10/los-estados-de-un-proceso-en-unix-y-linux/.
10. **Botti, V.** *Agentes inteligentes: El siguiente paso a la Inteligencia Artificial.* 2000.
11. **Caglayan, Ana.** *Agent Sourcebook.* 1997.
12. **LELANN, G.** *"Motivations, Objectives, and Characterization of Distributed Systems" in Distributed Systems, Architecture and Implementation.* s.l. : Spring-Verlag, 1981. ISBN 3-540-10571-9.
13. **COULOURIS, G. y DOLLIMORE, J.** *"Distributed Systems: Concepts and Design" 3rd. Edition. .* s.l. : Addison-Wesley, 2001. ISBN 0-13-214301-1.
14. **S.Tanenbaum, Andrew y Steen, Maarten Van.** *Distributed Systems - Principles and Paradigms (Second Edition).* s.l. : Pearson Education, Inc, 2006. 0-13-239227-5.
15. **Nagios.** Sitio oficial del Nagios. [En línea] [Citado el: 1 de diciembre de 2012.] http://www.nagios.org/about/overview/.
16. **Barth, Wolfgang.** *Nagios: System And Network Monitoring.* 2006. ISBN 1-59327-070-4.
17. **FMS, Pandora.** Pandora FMS - flexible monitoring system. [En línea] [Citado el: 1 de diciembre de 2012.] http://pandorafms.com/.
18. **OpManager.** ManageEngine: Powering it ahead. [En línea] [Citado el: 1 de diciembre de 2012.] http://www.manageengine.com/network-monitoring/.
19. **3i Infotech.** 3i Infotech (Innovation . Insight . Integrity). [En línea] [Citado el: 2 de diciembre de 2012.] http://www.3i-infotech.com/content/IT_infrastructure/ienablermf_overview.aspx.
20. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Pearson Adisson-Wesley, 2000.
21. **6.** Object Management Group - Unified Modeling Language. [En línea] [Citado el: 21 de noviembre de 2012.] http://www.uml.org/.

22. **Álvarez Romero, Eduardo y Pueyo, Daniel.** *Integration Definition For Funcion Modeling (IDEF0).* 2004-2005.
23. **7.** EncuRed. [En línea] [Citado el: 21 de noviembre de 2012.] http://www.ecured.cu/index.php/Herramienta_CASE.
24. **8.** Visual Paradigm. [En línea] [Citado el: 22 de noviembre de 2012.] http://www.visual-paradigm.com/product/vpuml/.
25. **Microsft Office.** Office. [En línea] [Citado el: 2012 de noviembre de 26.] http://office.microsoft.com/es-es/visio-help/descripcion-general-de-microsoft-office-visio-2007-HA010165640.aspx.
26. **Python.** Sitio oficial de Python. [En línea] [Citado el: 2 de diciembre de 2012.] http://pypi.python.org.
27. **Ordoñes Leyva, Yoanni y Avilés Vázquez, Ernesto.** *Herramienta informática de Minería de Uso de la Web sobre los registros de navegación por Internet.* Universidad de las Ciencias Informáticas, Ciudad de La Habana : s.n., 2010.
28. **García, Alejandro Pérez.** [En línea] [Citado el: 2012 de noviembre de 26.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=eclipseHelios#6.%20Conclusiones|outline.
29. **Qt.** Sitio oficial de Qt. [En línea] 2 de diciembre de 2012. http://qt.digia.com/Product/Developer-Tools/.
30. **Craig, Larman.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999.
31. **Purcell, Steve.** Python Unit Testing Framework. *PyUnit.* [En línea] [Citado el: 23 de abril de 2013.] http://pyunit.sourceforge.net/pyunit.html.
32. **Lisandra Valdés Pera, Haydee Sánchez Berrillo.** *Propuesta de Técnica para la Comunicación entre Agentes.* La Hababa, Universidad de las Ciencias Informáticas : s.n., 2008.
33. **Zabbix.** Sitio oficial de Zabbix. [En línea] 1 de diciembre de 2012. http://www.zabbix.com/.
34. **9.** Dia Proyect. [En línea] [Citado el: 22 de noviembre de 2012.] http://projects.gnome.org/dia/??.
35. **Eclipse.** Sitio Oficial de Eclipse. [En línea] [Citado el: 2 de diciembre de 2012.] http://www.eclipse.org/org/.
36. **James Rumbaugh, Jacobson, Ivar y Booch, Grady.** *El lenguaje unificado de modelado. Manual de referencia.* 2000. 8478290370.
37. **Marquina, Ernesto.** *Guía de Patrones, Prácticas y Arquitecturas.* 2008.
38. **Facultad de Informática - Universidad Politécnica de Madrid.** *Patrones del "Gang of Four".*
39. **masadelante.com. [En línea] [Citado el: 22 de marzo de 2013.] http://www.masadelante.com/faqs/socket.**
40. Code Project. [En línea] [Citado el: 2013 de marzo de 23.] http://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design.
41. Larman, Craig. *UML y Patrones.* Naucalpan de Juhez : Prentice Hall, 1999. pág. 16. ISBN 970-1 7-0261-1.