Universidad de las Ciencias Informáticas.

Facultad 5.

Título: Configurador de tareas para el módulo Aplicaciones del SCADA Guardián del ALBA.

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

Autor

Oscar Calderin Pérez.

Tutor

Ing. Yosell Luis Sehara Driggs.

Co-Tutor

Ing. Arian Antonio Nuñez Alonso.

MSc. Dayany Díaz Corona.

La Habana

Junio de 2012.

"La virtud, como el arte, se consagra constantemente a lo que es difícil de hacer, y cuanto más dura es la tarea más brillante es el éxito"

Aristóteles

DATOS DE CONTACTO

Ing. Yosell Luis Sehara Driggs.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: ylsehara@uci.cu.

Graduado en la UCI en el año 2008, profesor instructor con 3 años de experiencia en el desarrollo de software.

Ing. Arian Antonio Nuñez Alonso.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: anuneza@uci.cu.

Graduado en la UCI en el año 2007, profesor instructor con 4 años de experiencia en el desarrollo de software.

MSc. Dayany Díaz Corona.

Institución: Universidad de las Ciencias Informáticas. (UCI)

Título: Ingeniero en Ciencias Informáticas.

e-mail: ddiazc@uci.cu.

Graduada en la UCI en el año 2007, profesor instructor con 4 años de experiencia en el rol de Analista de Software. Cursó el Diplomado de Calidad e Ingeniería de Software.

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor del trabajo titulado:

Configurador de tareas para el módulo Aplicaciones del SCADA Guardián del ALBA y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Oscar Calderin Pérez

_____

Firma del autor

Ing. Yosell Luis Sehara Driggs

_____

Firma del tutor

Ing. Arian Antonio Nuñez Alonso

_____

Firma del co-tutor

MSc. Dayany Díaz Corona

_____

Firma del co-tutor

Agradecimientos

A la revolución por brindarme la oportunidad de estudiar en una universidad tan maravillosa como la UCI, creo que si volviera a nacer regresaría aquí sin pensarlo dos veces.

A mi familia por quererme tanto y apoyarme siempre en todas mis decisiones.

A mi sobrinito que lo quiero muchísimo.

A todos los amigos que me han apoyado en los bueno y malos momentos, en especial a Enrique, Abel (El Belo), Gabriel (El Kenya), Osmar, Yuniel (El Yuni), Yandy, Arlan y Yoelvys (El Abuelo).

A mis amigas Yinelys, Gicelle, Edelis, Aymeé y Leticia, gracias por estar ahí cada vez que necesité su ayuda.

A Yuya por brindarme su ayuda incondicional para llevar a cabo este trabajo.

A mi tutor y co-tutores por todo el apoyo y la ayuda brindada en la elaboración de este trabajo.

Al proyecto SCADA por los conocimientos adquiridos y brindarme la oportunidad de formarme como un profesional.

A todos mis compañeros de grupo que ya no están en la escuela, Danae, Sulemy, Eileén y Yani.

A todas las amistades que la misión a Venezuela me permitió conocer, Yamila, Alejandro, Dysán, Daryl, Yoendry, Jorge, Yordy, Rosy (La Profe UCI), Yurixay, Alain, Katia, Yoandy, Yaquelin, Lendys y Zamaharys.

A todos los que mencioné y a los que quedaron por mencionar.

…de todo corazón, GRACIAS…

Dedicatoria

A mis padres y mi familia, por su apoyo, dedicación y esfuerzo para que me convirtiera en lo que soy hoy, este trabajo también es de ustedes.

A todas las amistades que van quedando atrás pero también quedan en el corazón.

A los que de una forma u otra me ayudaron y contribuyeron a la realización de este trabajo.

RESUMEN

Un sistema de Supervisión, Control y Adquisición de Datos (SCADA) se encarga de monitorear los eventos que tienen lugar en estaciones remotas. En la Universidad de las Ciencias Informáticas se trabaja en el desarrollo de un sistema de este tipo conocido como SCADA Guardián del ALBA (SCADA GALBA). Debido a la gran complejidad de estos sistemas, su desarrollo fue dividido en módulos. Entre los distintos módulos que lo componen se encuentra el de Aplicaciones, el cual se encarga de agregarle al sistema distintas aplicaciones que le permitan un mayor control de los procesos industriales.

Una de las aplicaciones realizadas por este módulo es el motor de planificación y ejecución que le permite al usuario elaborar sus propias tareas, las cuales ejecutarán un script que le permitirá interactuar con los distintos componentes del SCADA GALBA. Dicho motor no cuenta con una aplicación que le permita gestionar sus tareas, esto provoca que carezca de utilidad y que el SCADA GALBA no se pueda beneficiar con las facilidades que brinda. Por tal motivo con este Trabajo de Diploma se pretende lograr la implementación eficiente de una herramienta, denominada Configurador de tareas, que permita la gestión de tareas para el módulo Aplicaciones fundamentada en el análisis de las principales características que presentan algunos software utilizados en la planificación de tareas.

PALABRAS CLAVE

Aplicaciones, Configurador de tareas, planificación de tareas, SCADA, scripts.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La Universidad de las Ciencias Informáticas (UCI) es un proyecto de la Revolución cuya misión fundamental es formar profesionales calificados en la rama de la Informática, a partir de un modelo pedagógico que vincula dinámicamente el estudio con la producción y la investigación, acorde con las necesidades sociales del país y de otros pueblos hermanos.

En la UCI dicha integración se realiza entorno a un área temática determinada para de esta manera contribuir al desarrollo económico y social del país, a estas áreas temáticas se les denomina Centros de Desarrollo. En la facultad 5 se encuentra el Centro de Informática Industrial (CEDIN) en el cual se desarrolla, bajo paradigmas de software libre, el sistema de supervisión y control para procesos industriales SCADA Guardián del ALBA (SCADA GALBA), dicho software presenta una arquitectura distribuida en la cual se encuentran los módulos de configuración, gestión de archivos y datos, interfaz hombre máquina (HMI por sus siglas en inglés) y aplicaciones, entre otras.

El módulo Aplicaciones surge con la necesidad de agregarle al SCADA GALBA diversas funcionalidades que le permitan lograr cierta homogeneidad con las distintas aplicaciones SCADA que lideran el mercado en la actualidad. La diversidad de estas aplicaciones, debido a la naturaleza heterogénea de los procesos industriales, no permite tener una previsión exacta de qué herramienta, algoritmo de control, optimización, u otras, requerirán para enfrentar los problemas particulares de cada aplicación. Es por ello que la presencia de ejecutores y editores de scripts en los sistemas SCADA de los últimos años, se ha convertido en un requisito importante para los desarrolladores de aplicaciones SCADA.

Por esta razón en dicho módulo se comienza a trabajar en el desarrollo de una herramienta que le permita al usuario elaborar sus propias tareas, y que estas tengan asociadas a ellas un script que contendrá la secuencia de instrucciones a desarrollarse sobre el SCADA GALBA al ejecutarse la tarea. Estas tareas podrán se planifican en forma de calendario, es decir, que se ejecutarán en días y horarios determinados por el usuario, incluso pueden activarse al producirse algún evento del sistema como pueden ser el cambio de un valor, activación de una alarma, entre otras. Como resultado del trabajo realizado en dicho módulo se obtuvo el motor de planificación y ejecución de algoritmos del módulo de Aplicaciones, el cual implementa las funcionalidades antes mencionadas pero que no cuenta con una herramienta que permita gestionar las tareas, este inconveniente trae consigo que

dicho motor carezca de utilidad y no se aprovechen en el SCADA GALBA las diversas ventajas que ofrece.

En dicho contexto surge como **Problema Científico** la siguiente interrogante: ¿Cómo gestionar las tareas en el módulo Aplicaciones del SCADA GALBA?

Para darle respuesta a dicho problema el presente trabajo tiene como **Objeto de Estudio** el diseño e implementación de configuradores de tareas y se define como **Campo de Acción** el desarrollo de configuradores de tareas para sistemas SCADA desarrollados bajo principios de software libre.

Esta investigación tiene la siguiente **Idea a Defender**: Si se desarrolla una herramienta que permita gestionar las tareas para el módulo Aplicaciones del SCADA GALBA se garantizará un eficiente uso del motor de planificación y ejecución dentro del SCADA.

El **Objetivo General** que se desea alcanzar para darle cumplimiento a este trabajo es desarrollar una herramienta que permita gestionar las tareas para el módulo Aplicaciones del SCADA GALBA.

Para cumplir con el objetivo planteado se definieron las siguientes **Tareas de Investigación**:

- Realización de un estudio del arte en planificadores de tareas con el objetivo de obtener una mejor comprensión de sus vistas e interfaces.
- Realización del análisis y diseño del sistema a desarrollar.
- Diseño de las interfaces gráficas de usuario del Configurador de tareas.
- Implementación del Configurador de tareas.

Para el desarrollo de dichas tareas se combinan diferentes **Métodos y Técnicas en la búsqueda y procesamiento de la información**, los fundamentales son:

A nivel teórico:

Análisis histórico – lógico: Utilizado durante la investigación de los antecedentes y las tendencias actuales relacionadas con el desarrollo de configuradores de tareas y en la profundización de los conceptos y términos relacionados con el objeto de estudio y el campo de acción.

Analítico – Sintético: Empleado durante el análisis y la selección de las tecnologías a utilizar en el diseño e implementación de la aplicación.

A nivel empírico:

Revisión de la documentación: Estudio de documentos y otros tipos de bibliografías con el objetivo de obtener la información necesaria para llevar a cabo la investigación.

Experimentos: En la elaboración de prototipos funcionales, para ir comprobando los distintos Casos de Uso.

El presente documento está compuesto por 3 capítulos en los que se expone todo el proceso de desarrollo de este trabajo y está estructurado de la siguiente forma:

En el **Capítulo 1** se profundiza en los principales conceptos asociados al dominio de este trabajo y se realiza un estudio del arte. Además se describen cada una de las herramientas y tecnologías utilizadas para el desarrollo de los componentes tratados en este trabajo.

En el **Capítulo 2** se realiza una propuesta de solución, además se identifican los requisitos funcionales y no funcionales que deben tenerse en cuenta. También se realiza, a partir de los requisitos funcionales, una descripción de los casos de uso significativos y se presentan los prototipos de las principales interfaces gráficas de usuario con las que cuenta la aplicación.

En el **Capítulo 3** se describe el patrón de diseño utilizado y se presentan distintos diagramas generados en las fases de diseño e implementación del sistema, como son el diagrama de clases del diseño, el de componentes y el modelo de despliegue. También se describen las clases significativas y se expone el estilo de código utilizado durante la implementación del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

INTRODUCCIÓN

El siguiente capítulo aborda los elementos teóricos que soportan la presente investigación, quedan definidos los principales conceptos relacionados con el dominio del problema. Se realiza un estudio del arte de herramientas dedicadas a la configuración y planificación de tareas. Además se exponen las principales herramientas y metodologías a utilizar para el desarrollo de la aplicación.

1.1. Sistemas SCADA

En los últimos años el crecimiento de las industrias ha traído consigo un gran desarrollo y extensión de las fábricas y maquinarias que la componen, lo que ha conllevado a un aumento en la complejidad de los procesos industriales. Para lograr un control eficiente de estos procesos surgieron los sistemas de supervisión, control y adquisición de datos, conocidos como SCADA, que es un término tomado del inglés, acrónimo de *Supervisory Control and Data Adquisition*.

Se trata de una aplicación de software especialmente diseñada para funcionar sobre computadoras en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas, programables, entre otros) y controlando el proceso de forma automática desde la pantalla de la computadora. Además dicho sistema provee de toda la información que se genera en el proceso productivo a diversos usuarios, tanto del mismo nivel como de otros supervisores dentro de la empresa: control de calidad, supervisión, mantenimiento, entre otros [1]**.**

1.1.1. Principales funcionalidades de un SCADA

Un sistema SCADA como parte integral de la estructura de las industrias y un recurso importante de información debe cumplir con tres funciones principales: [2]

- **Adquisición de datos** para recoger, procesar y almacenar la información recibida.
- **Supervisión** para observar desde el monitor la evolución de las variables del proceso.
- **Control** para modificar la evolución del proceso, actuando sobre los reguladores autónomos básicos (consignas, alarmas, menús, entre otros.), o directamente sobre el proceso mediante

las salidas conectadas.

1.2. Tarea

En la esfera informática se define como tarea a un “conjunto de actividades básicas que se ejecutan en un sistema operativo” [3], en sistemas SCADA una tarea será una acción o conjunto de acciones que se realizarán sobre el sistema y estará conformada por un conjunto de atributos para su planificación y un script que expresa la secuencia de instrucciones que debe realizar al ejecutarse. Por lo general pueden ser de 4 tipos fundamentales, periódicas, aperiódicas, síncronas y asíncronas.

1.2.1. Tarea periódica

Son aquellas que se configuran para que se ejecuten en base a una planificación, por ejemplo todos los lunes, miercoles y viernes, y se repite cada n semanas hasta una cantidad de repeticiones fijadas o siempre [4].

1.2.2. Tarea aperiódica

Consisten en tareas que no se repiten en el tiempo sólo se activan en fechas y horas determinadas, por ejemplo el jueves 10 de mayo a las 3:00 pm [4].

1.2.3. Tarea síncrona

Son aquellas que se ejecutan con una base de tiempo bien definida, por ejemplo cada 10 min, estas tareas tienen configurado un periodo de repetición y pueden ser por n repeticiones o siempre [4].

1.2.4. Tarea asíncrona

Son las tareas que no tienen una base de tiempo definida, su ejecución está controlada por señales que son activadas por eventos, por ejemplo cuando se activa una alarma [4].

1.2.5. Estados de una tarea

Una tarea tiene los siguientes estados: [5]

1. Creación: el programa es leído del disco y cargado en memoria, se le agrega información adicional para crear la tarea. La nueva tarea cuenta con un identificador para ser reconocida por el sistema operativo (aún no consume recursos).

2. Inicio*:* la tarea se forma en la cola de listos y espera a que se le asignen recursos para comenzar su ejecución: la cola de listos es una fila donde se forman todas las tareas antes de ejecutarse.

3. Ejecución*:* la tarea entra en ejecución y es atendida.

4. Terminación*:* la tarea termina todas sus actividades, libera los recursos y su identificador es borrado.

1.3. Configurador de tareas

Un Configurador de tareas es una herramienta que le permite al usuario administrar sus propias tareas, este tipo de aplicación utiliza internamente un planificador de tareas, el cual es un programa que utiliza un algoritmo de planificación para tomar decisiones de planificación y poderle asignar recursos del sistema a las tareas que lo soliciten [6], es decir que este planificador será el encargado de ejecutar en el momento correspondiente las tareas que hayan sido configuradas previamente en el Configurador de tareas.

1.3.1. Características de los configuradores de tareas

Es conveniente en el desarrollo de una aplicación analizar sus características. Los sistemas dedicados a la configuración de tareas se caracterizan por: [7]

- Interfaz liviana, de fácil accesibilidad y con algoritmos que permitan su actualización automática y constante, permitiendo mostrar el contenido real en cada momento.
- Se construyen según la necesidad y los requisitos.
- Deben permanecer ejecutándose de forma permanente para posibilitar la ejecución de sus trabajos en tiempo y forma.
- Suelen incluir la coordinación de tareas humanas con tareas automáticas.
- Suponen un constante monitoreo del estado de los procesos que tramitan, así como la gestión de los mismos.

1.3.2. Ventajas y desventajas de los configuradores de tareas

Dentro de las ventajas de los sistemas para configurar tareas encontramos: [7]

- Automatización de tareas.
- Sistema de ejecución de procesos.
- Centralización de la información de los procesos.
- Administración humana de tareas maquinales.
- Reducción de costos en cuanto a recursos humanos destinados con anterioridad para estas funciones.
- Garantía de reportes fiables de los procesos gestionados.

Al igual que en cualquier programa informático, los configuradores de tareas cuentan con desventajas, entre ellas se encuentran: [7]

- En caso de falta de energía eléctrica pueden quedar procesos sin ejecutarse en el momento señalado.
- Un excesivo número de procedimientos a ejecutarse automáticamente puede congestionar el ordenador, ya que su trabajo se base en hilos del sistema.

1.3.3. Planificadores de tareas en sistemas SCADA

Entre las actividades propias de la automatización de un proceso industrial o productivo se encuentran el control de procesos, el mantenimiento, el manejo de situaciones anormales o fallas, entre otras, de las cuales se tiene que el mantenimiento constituye uno de los factores más importantes para garantizar la efectividad y la eficiencia de los procesos productivos, la disminución de los costos, así como también la disponibilidad de la función de los equipos e instalaciones con la finalidad de atender, adecuadamente y a tiempo, el proceso productivo ó de servicios, con confiabilidad y seguridad [8] [9]**.**

Por esta razón los sistemas de mantenimiento, que son los encargados de gestionar el mantenimiento de una planta industrial, se han convertido en una herramienta apropiada para utilizar en los sistemas SCADA. Estos sistemas de mantenimiento están compuestos por varios módulos, entre los que destacan, el Planificador y el Ejecutor. En este caso el Planificador se encarga de generar los planes de mantenimiento y planifica el lapso de tiempo en el cual se deben realizar las tareas de planificación. El Ejecutor es el que se ocupa de poner en ejecución las tareas planificadas

para el mantenimiento e imparte la información resultante al resto del sistema. El uso de estos sistemas trae como ventajas para los sistemas SCADA que se minimicen los efectos de las fallas presentes en los procesos productivos.

1.4. Software dedicados a la configuración y planificación de tareas

En el presente epígrafe se analizan algunas herramientas utilizadas en la actualidad para la configuración y planificación de tareas, vale destacar que no se está buscando un software para planificar tareas, sino que se analizarán un grupo de estos software para de ellos tomar sus principales funcionalidades así como las características fundamentales en su interfaz gráfica de usuario para tomar el resultado de este análisis como punto de partida del presente trabajo.

1.4.1. Advanced Task Scheduler

Es un planificador de tareas multifuncional que permite la ejecución de tareas y archivos por lotes, abrir documentos y páginas de Internet, mostrar recordatorios emergentes, reproducir sonidos, apagar y reiniciar el ordenador, apagar solo el monitor, detener procesos en ejecución, establecer y cerrar conexiones de marcado de forma automática. Es una herramienta propietaria que funciona en la plataforma Windows.

Ofrece un completo lote de herramientas de planificación que le permiten ejecutar tareas planificadas de forma automática una sola vez, cada minuto, cada hora, diariamente, mensualmente, anualmente, en un período de tiempo especificado tras el inicio del ordenador o mediante algunos eventos como la pulsación de una tecla de acceso directo, por el bajo uso del ordenador o por conexión de marcado establecida / terminada, entre otras [10].

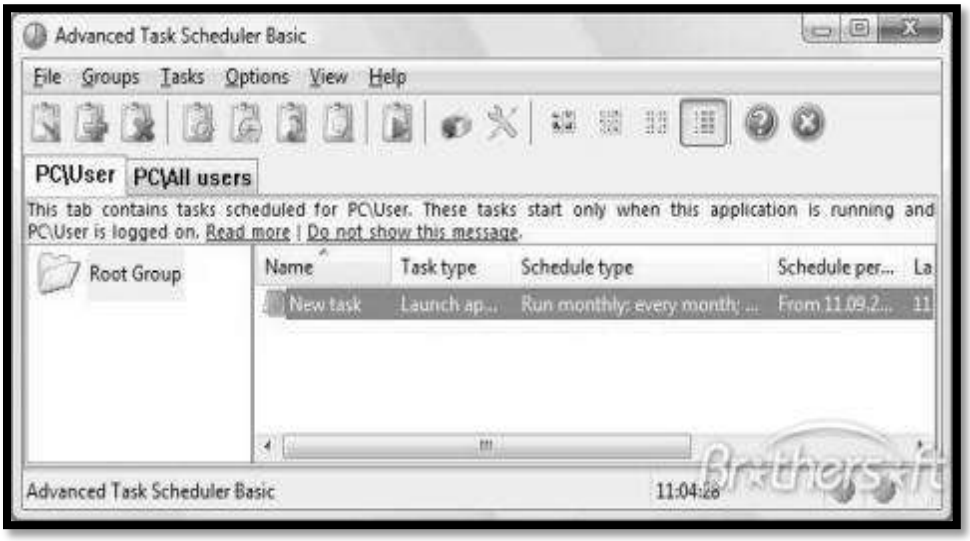


Fig. 1: Advanced Task Scheduler

1.4.2. Task Coach

Es una aplicación de código abierto que permite gestionar tareas, básicamente asignarles una fecha de inicio y fin, de manera que podamos controlar su cumplimiento desde un panel de tareas que se incluye y nos permite hacer un rápido seguimiento de todas ellas. Compatible con la mayoría de los sistemas operativos.

No solo se podrán registrar tareas, sino también sub-tareas, además también se pueden editar e incluso eliminarlas. Cada tarea tiene un asunto o tema, una descripción, una prioridad establecida, una fecha de vencimiento y una fecha de recordatorio que puede ser asignada opcionalmente. Las tareas pueden repetirse en forma diaria, semanal o mensual, dependiendo de cuales sean los requisitos del usuario [11].

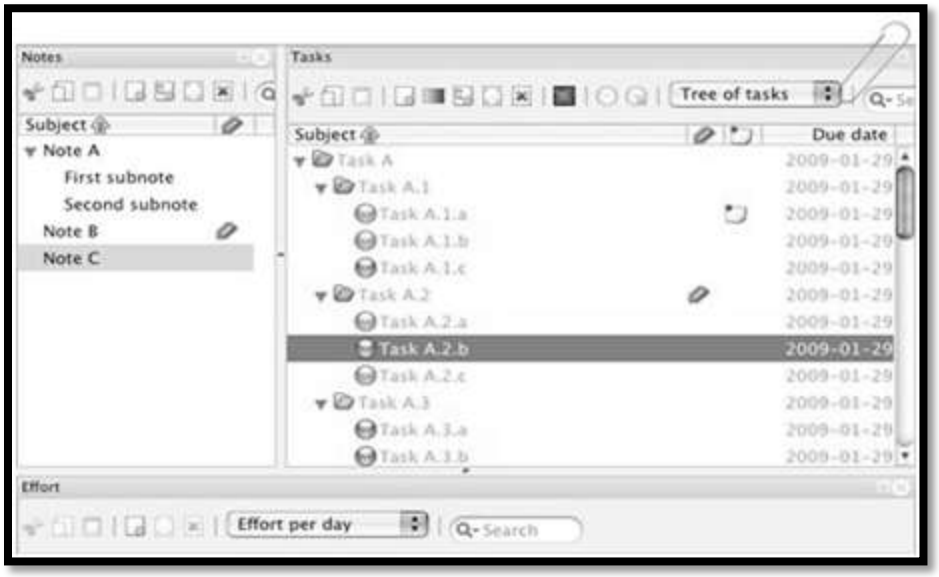


Fig. 2: Task Coach

1.4.3. Automize

Es un planificador de tareas fácil de utilizar y que permite automatizar una gran variedad de tareas, incluyendo las transferencias FTP, monitoreo de sitios web, Telnet, sincronización de carpetas, entre otras. Ofrece notificaciones vía e-mail para todas las tareas basadas en el código de salida de la tarea. Para los usuarios avanzados ofrece la opción de *Scripting*, permitiéndoles desarrollar las escrituras de sus propias tareas [12].

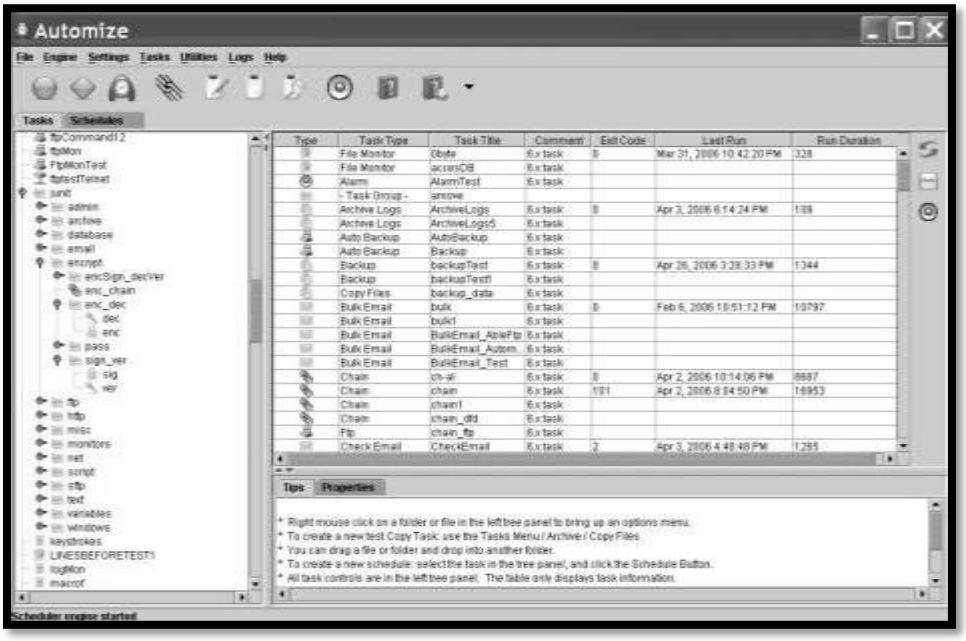


Fig. 3: Automize

Con el análisis de estos software se ha llegado a la conclusión que tienen características en común que pueden ser tomadas en consideración a la hora de desarrollar la aplicación, por ejemplo, los 3 software le brindan al usuario opciones para el filtrado de las tareas. Cuentan con un área donde se muestran todas las tareas que se han configurado en el sistema así como otros paneles donde se muestran detalles específicos de la tarea seleccionada. En cuanto a sus interfaces gráficas de usuario (GUI) todos coinciden en una GUI sencilla que cuenta con una barra de herramienta en la parte superior donde se les brinda a los usuarios accesos rápidos a las acciones que puede llevar a cabo sobre la tarea, ya sea ejecutarlas, pausarlas, eliminarlas, entre otras.

1.5. Descripción de tecnologías

En el presente epígrafe se realiza una descripción de las herramientas y tecnologías empleadas para el desarrollo y documentación de la aplicación.

1.5.1. Sistema Operativo. GNU / Linux

GNU es un acrónimo recursivo que se traduce como "**GNU No es Unix**". Este proyecto, iniciado por Richard Stallman en 1983, pretende desarrollar un sistema operativo completamente libre equivalente o compatible con UNIX.

Linux fue adoptado como núcleo del sistema GNU, y junto a él y otros muchos programas, constituyen el actual sistema operativo GNU / Linux. El resultado de la combinación GNU y Linux es un sistema operativo libre, multiplataforma, multiusuario y multitarea. GNU / Linux es completamente configurable y optimizable, posee un funcionamiento muy rápido ya que es capaz de utilizar todas las posibilidades de hardware del sistema [13].

Una distribución de software libre es una variante de ese sistema operativo (SO) que contiene una recopilación de programas y ficheros, organizados y preparados para su instalación que satisfacen las necesidades de un grupo específico de usuarios. Además del núcleo de Linux, las distribuciones incluyen diversas bibliotecas y herramientas del proyecto GNU.

1.5.1.1. Distribución de Linux: Debian GNU / Linux

Debian es un SO libre que utiliza el núcleo de Linux pero que la mayor parte de sus herramientas básicas provienen del proyecto GNU; de ahí el nombre de GNU / Linux. Esta distribución le ofrece al usuario más de 25000 paquetes, que no son más que programas pre compilados distribuidos en un formato que hace más fácil su instalación.

Nace como una apuesta de separar en sus versiones el software libre del software no libre. Su software no se vende directamente, sino que lo pone a disposición de los usuarios en Internet, aunque si le permite a empresas o personas distribuir comercialmente su software mientras se respete su licencia. Actualmente es la única distribución importante de GNU / Linux que es mantenida solamente por voluntarios, lo que le permite la actualización de los paquetes de software y la participación abierta de toda aquella persona que desee colaborar [14]**.**

Se decidió usar esta distribución porque cuenta con un desarrollo muy estable, por lo que los paquetes que se desarrollen en él y deseen ejecutarse utilizando otra distribución siempre serán estables.

1.5.2. Biblioteca gráfica de desarrollo en sistemas GNU / Linux. Qt

Qt es una biblioteca multiplataforma, software libre (bajo la licencia LGPL[1]) ampliamente usada para desarrollar aplicaciones con una interfaz gráfica de usuario así como también para el desarrollo de programas sin interfaz gráfica como herramientas para la línea de comandos y consolas para servidores. Utiliza el lenguaje C++ de forma nativa, adicionalmente puede ser utilizado en otros lenguajes de programación a través de *bindings*[2]. Funciona en todas las plataformas y tiene un amplio apoyo [15]**.**

Esta biblioteca cuenta con una amplia comunidad de desarrollo que le brinda soporte y posee muchas funcionalidades de alto rendimiento. Para desarrollar en Qt solo es necesario aprender una sola API[3] para escribir aplicaciones que se ejecutarán en cualquier lado, ventaja que nos garantiza su portabilidad, además de que Qt posee un amplio conjunto de *widgets*[4] estándares y permite desarrollar controles personalizados. La calidad en tiempo de ejecución es alta y no depende de ninguna otra biblioteca, aparte de sí misma.

1.5.3. QScintilla2

QScintilla2 es una variante para Qt del componente de edición Scintilla, este es un código fuente libre utilizado para desarrollar editores de texto. Numerosos editores utilizan Scintilla en su desarrollo tal es el caso del Anjuta, Code::Blocks, CodeLite y Notepad++. Posee una licencia que permite su uso en cualquier proyecto libre o producto comercial. Además de las funcionalidades que se encuentran en un editor de texto estándar, QScintilla2 posee algunas características muy útiles especialmente para programas dedicados a la edición y depuración de código fuente. Incluye soporte para el completamiento de código, resaltado de sintaxis para varios lenguajes de programación, indicador de errores, marcadores de líneas, numeración de líneas en el margen, entre otras. Funciona en cualquier

[1] Licencia creada por la Fundación de Software Libre y orientada principalmente a los términos de distribución, modificación y uso de software libre.

[2] Adaptación de una biblioteca para ser utilizada en otro lenguaje de programación distinto de aquél en el que ha sido creada.

[3] Grupo de rutinas que provee un sistema operativo, una aplicación o una biblioteca, que definen cómo invocar desde un programa un servicio que éstos prestan.

[4] Elemento de una interfaz gráfica de usuario que muestra información con la cual el usuario puede interactuar.

sistema operativo compatible con Qt, dígase Windows, UNIX / Linux y Mac OS / X, y en la mayoría de las versiones de Qt [16] [17].

1.5.4. Lenguaje de programación: C++

En la actualidad, C++ es un lenguaje versátil y potente, que mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además ha eliminado algunas de las dificultades y limitaciones del C original. Entre sus características más notables se encuentran [18]:

- Programación orientada a objetos, permitiendo mayor reutilización de código en un modo más lógico y productivo.
- Portabilidad, prácticamente se puede compilar el mismo código en casi cualquier tipo de ordenador y sistema operativo sin realizar ningún cambio.
- Programación modular: el cuerpo de una aplicación en C++ puede estar compuesto por varios archivos de código fuente que se compilan por separado y luego unidas entre si.
- C++ es compatible con el lenguaje C. Cualquier código escrito en C puede ser incluido en un programa de C++ sin realizar ningún cambio.

1.5.5. Python

El lenguaje de programación interpretado Python fue creado por Guido van Rossum en el año 1991. El principal objetivo que persigue este lenguaje es la facilidad de lectura y de diseño. Es un lenguaje de programación multiparadigma, o sea, provee a los programadores adoptar varios estilos: programación orientada a objetos, estructurada, funcional y otros soportados mediante extensiones. Además usa tipos de datos dinámicos y *reference counting*[5] para la utilización de la memoria. Una característica importante de este lenguaje es la resolución dinámica de nombres (ligadura dinámica de métodos), o sea, enlaza un método con un nombre de variable durante la ejecución del programa. El diseño del lenguaje facilita la extensión, proveyendo la inclusión de nuevos módulos desarrollados en otros lenguajes fácilmente [19].

[5] Técnica para contabilizar las veces que un determinado recurso está siendo referido.

Vale destacar que Python no es utilizado en el desarrollo de la aplicación pero es el lenguaje que compila el motor de Planificación y Ejecución del módulo Aplicaciones, su rápida velocidad de desarrollo y el alto grado de mantenibilidad de su código fueron detalles muy importantes que se tuvieron en cuenta para la decisión del uso del mismo en dicho motor.

1.5.6. IDE Eclipse

Eclipse es un Entorno Integrado de Desarrollo (IDE) multiplataforma y libre, usado para crear aplicaciones clientes de cualquier tipo. Con este IDE se han desarrollado importantes aplicaciones como el afamado IDE de Java llamado *Java Development Toolkit* (JDK) y el compilador (ECJ) que se entrega como parte del Eclipse, y que a la vez es utilizado para el desarrollo del propio Eclipse.

El IDE de Eclipse emplea extensiones (en inglés *plug-in*) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos le permite al entorno soportar otros lenguajes además de Java, como C++ y Python. Existen muchísimos *plugins* disponibles libremente que permiten variadas funcionalidades como añadir control de versiones con Subversion[6], aplicaciones de red como Telnet y sistemas de gestión de bases de datos, entre otras [20]**.**

1.5.7. Metodología de desarrollo. Proceso Unificado de Rational (RUP)

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software. Indican, paso a paso, todas las actividades a realizar para lograr el producto informático deseado, además de las personas que deben participar en el desarrollo de estas actividades y el papel que deben jugar.

El Proceso Unificado de Rational o RUP (*Rational Unified Process*), es un proceso de desarrollo de software que en conjunto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP incluye artefactos (productos tangibles del proceso como por ejemplo, el modelo de Casos de Uso, el código fuente, entre otros) y roles (papel que desempeña una persona en un determinado momento del desarrollo del software, una persona puede desempeñar varios roles a lo largo del proceso) [21]**.**

[6] Sistema de control de versiones, también conocido como svn por ser el nombre utilizado en la línea de comandos.

RUP (véase Fig. 4) divide el proceso de desarrollo en ciclos, teniendo un producto final al concluir cada ciclo, dichos ciclos se dividen en fases (inicio, elaboración, construcción y transición) que finalizan con un hito donde se debe tomar una decisión importante.

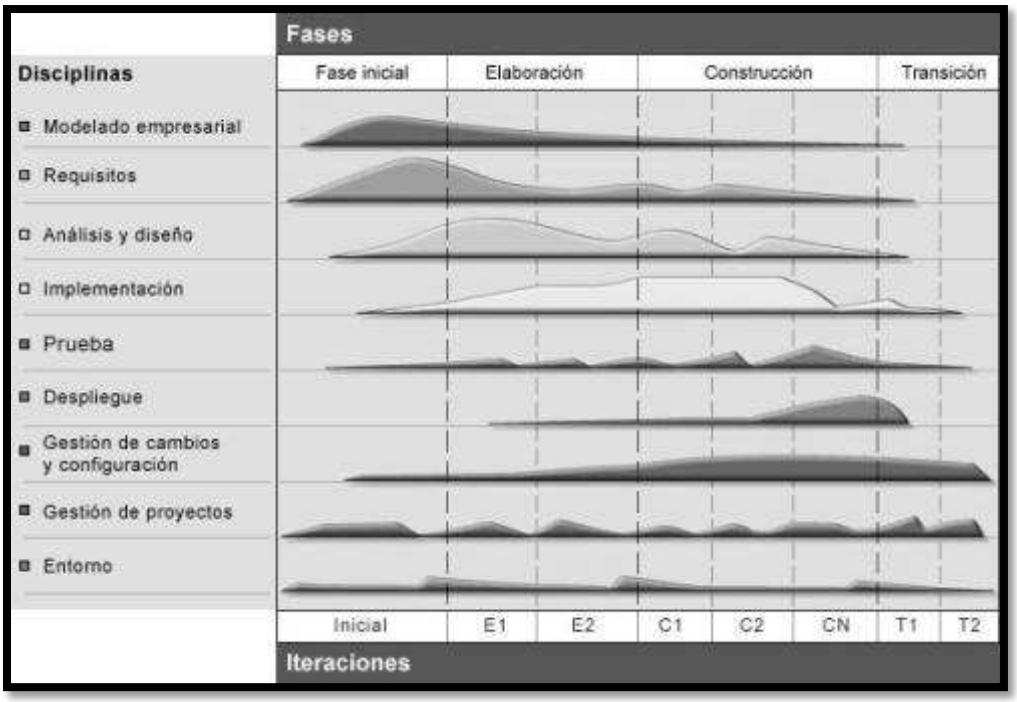


Fig. 4: Fases e iteraciones de RUP

RUP se caracteriza por ser:

- **Dirigido por casos de uso (CU):** Los CU reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requisitos.
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo del proyecto y los usuarios deben estar de acuerdo.
- **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

1.5.8. Herramientas utilizadas para el modelado del sistema

Las aplicaciones informáticas que facilitan el trabajo dentro del ciclo de desarrollo del software son conocidas como herramientas CASE (Ingeniería de Software Asistida por Computadoras) y se emplean para aumentar la productividad del desarrollo de software disminuyendo los tiempos de construcción y el costo de los mismos. La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad en el desarrollo de sistemas de información.

1.5.8.1. Visual Paradigm

Es una herramienta profesional de modelado que soporta el ciclo de vida completo del desarrollo software. Permite representar todos los tipos de diagramas, hacer ingeniería inversa, generar códigos desde diagramas y generar documentación. Es multiplataforma, brinda soporte para varias versiones de UML y ayuda a una rápida construcción de aplicaciones de mejor calidad y a un menor costo. Posibilita la integración con los principales IDE de desarrollo [22]**.**

CONCLUSIONES DEL CAPÍTULO

En el presente capítulo se hizo una breve descripción sobre los sistemas SCADA y sus principales funcionalidades. También se analizaron los principales conceptos relacionados con las tareas y las herramientas dedicadas a la configuración y planificación de tareas. Finalmente, se identificó la metodología, IDE de desarrollo, el lenguaje de programación y la biblioteca gráfica de desarrollo, ofreciendo las definiciones necesarias y precisas para comprender el porqué del uso de las mismas. Todo esto encaminado a obtener los elementos necesarios para dar solución a la investigación.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

INTRODUCCIÓN

En el presente capítulo se presenta la solución técnica del presente trabajo de diploma, contiene la descripción de los requisitos del sistema tanto funcionales como no funcionales, se detallan y justifican los principales actores y se hace una descripción de los principales casos de uso que incluyen todo lo que el sistema debe hacer.

2.1. Solución propuesta

A continuación se describe la solución propuesta, para una mayor comprensión de la misma, se ha separado en 3 sub-epígrafes: Arquitectura, Configurador de tareas y Entregable de la arquitectura de la información.

2.1.1. Arquitectura

La aplicación a desarrollar forma parte del módulo Aplicaciones el cual basa su arquitectura en el estilo arquitectónico 3 capas. La programación por capas es una forma de programar bajo un objetivo principal: que las distintas lógicas presentes se separen y posean estructuras bien planteadas. Por lo general suele plantearse esta visión sobre los 3 niveles o capas:

- Capa de presentación.
- Capa de negocio.
- Capa de datos.

En el caso del módulo al cual nos referimos (véase Fig. 5) la capa de negocio está compuesta por el motor de planificación y ejecución que contiene la lógica del negocio así como el acceso a la capa de datos en la cual se accede a un archivo XML[7] que contiene todas las tareas y condiciones creadas en el sistema. La aplicación que se propone desarrollar en este Trabajo de Diploma está enmarcada dentro de la capa de presentación.

Esta capa es la encargada de que el sistema interactúe con el usuario y viceversa, le presenta la información y obtiene información del usuario, y se comunica únicamente con la capa intermedia o de

[7] Siglas en inglés de eXtensible Markup Language (lenguaje de marcas extensibles).

negocio. A su vez, esta capa reúne todos los aspectos del software relacionados con las interfaces gráficas de usuario y su interacción con los usuarios.

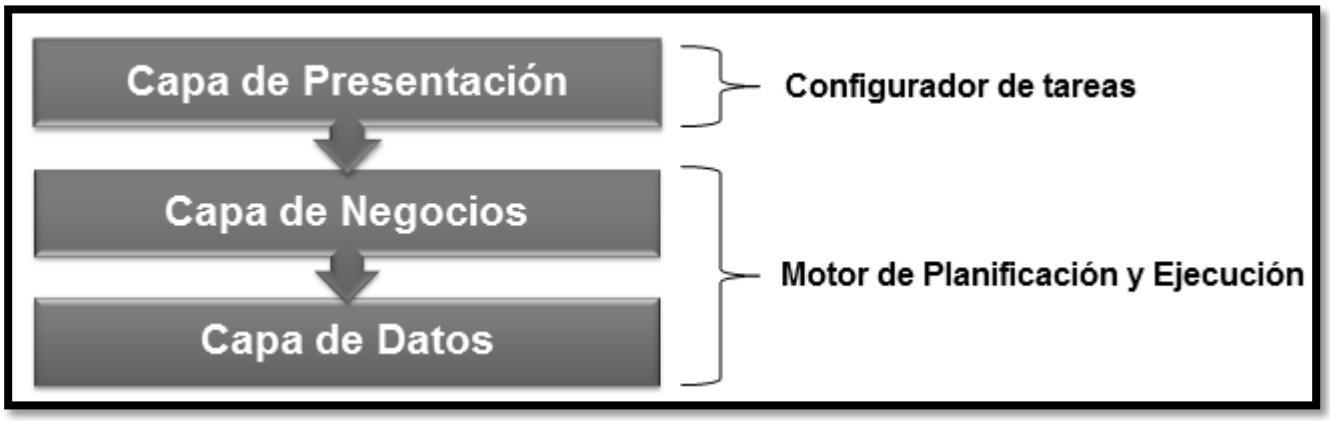


Fig. 5: Arquitectura del módulo Aplicaciones

2.1.1.1. Arquitectura Modelo / Vista del framework Qt

Para el desarrollo de las interfaces gráficas de usuario de esta aplicación se utilizó el *framework*[8] gráfico Qt. Este *framework* propone una arquitectura modelo / vista para gestionar la relación entre la información y la forma de presentársela al usuario.

La arquitectura modelo / vista toma como base el patrón de diseño Modelo – Vista – Controlador (MVC) que se utiliza muy a menudo en el desarrollo de interfaces de usuario. Este está compuesto por 3 tipos de objetos, el modelo se encarga de todo lo que tiene que ver con la persistencia de datos, guarda y recupera la información del medio persistente, la vista es la representación del modelo en forma gráfica disponible para la interacción con el usuario, y el controlador se encarga de manejar y responder las peticiones del usuario, procesando y accediendo a la información del modelo. Antes de MVC, en el diseño de las interfaces de usuario se agrupaban estos objetos, MVC los separa para aumentar la flexibilidad y la reutilización.

Si la vista y los objetos del controlador se combinan el resultado es la arquitectura modelo / vista. Esta arquitectura separa la forma en que se almacenan los datos de la forma en que se le presentan

[8] La traducción de framework (o en plural frameworks) en el idioma español es marco de trabajo.

al usuario, y a la vez proporciona un marco más simple basado en los mismos principios. Esta separación hace que sea posible visualizar los mismos datos en varias vistas diferentes.

La aplicación a desarrollar hace uso de esta arquitectura ya que la interfaz de la misma está compuesta por varios *widgets* de Qt que se rigen por dicha arquitectura, como son los utilizados para visualizar las tareas existentes en el sistema, los logs[9] de estas tareas y las funcionalidades que ofrece la API del SCADA GALBA, entre otros. Por ejemplo, en el caso de la visualización de tareas, a través del motor de Planificación y Ejecución se accede al listado de tareas existentes en el sistema, estas son almacenadas en el modelo, permitiendo configurar los datos para ser mostrado en cualquiera de las diferentes vistas que proporciona Qt.

2.1.2. Configurador de tareas

A partir del marco teórico analizado en el capítulo anterior, se propone el desarrollo de una aplicación que permita ampliar las funcionalidades del SCADA GALBA por medio de la integración al mismo de una herramienta que permita la ejecución y control de tareas. En la aplicación se definen 4 tipos de tareas:

- Tarea periódica.
- Tarea aperiódica.
- Tarea síncrona.
- Tarea asíncrona (estas tareas se ejecutarán al activarse algún evento del sistema, dichos eventos estarán basados en alarmas o en puntos).

La aplicación le permitirá al usuario gestionar estas tareas y a su vez estás interactuarán con el motor del módulo Aplicaciones que es el que se encargará de garantizar la ejecución de cada una de ellas de acuerdo a su planificación. La herramienta contará con un Inspector de Propiedades donde se podrán configurar todas las propiedades de la tarea seleccionada o de la nueva tarea que se desee crear. Dichas tareas también podrán ser filtradas, de acuerdo a varios criterios, como son su tipo y su estado, los cuales pueden ser:

- Planificada.
- No planificada.

[9] Registros del sistema, se refieren a acciones, comportamientos, entre otros, realizados por los sistemas o sobre ellos.

- Esperando confirmación de inicio.
- Iniciada o en ejecución.
- Con errores.
- Iniciada y en espera de interacción con el usuario.

Dichas tareas tendrán en común que todas contendrán un script especializado creado por el usuario, que indicará la secuencia de acciones a realizarse sobre el SCADA al ejecutarse la tarea, por lo que la aplicación le brindará al usuario un editor de scripts donde este creerá sus propios algoritmos, utilizando el lenguaje de programación Python que es el reconocido por el editor, estos scripts serán capaces de interactuar con los distintos módulos del SCADA de forma tal que se pueda recibir el estado de los puntos y las alarmas y se puedan realizar envíos de comandos. Este editor fue elaborado utilizado las facilidades que brinda la biblioteca QScintilla y le brindará al usuario diversas facilidades como son el completamiento de código y el reconocimiento de las palabras reservadas de dicho lenguaje, entre otras. Además contará con dos paneles, uno mostrará las principales funcionalidades de la API del SCADA GALBA y en el otro se visualizarán los distintos dispositivos del SCADA GALBA con los cuales el módulo de Aplicaciones puede interactuar. Dicho editor también le brindará al usuario las opciones de compilar y ejecutar sus scripts, en caso de que el usuario cometa errores en la elaboración de su script se le mostrarán dichos errores, con el objetivo de que los algoritmos asociados a las tareas estén libres de errores.

2.1.3. Entregable de la arquitectura de la información

Al realizar el estudio de las interfaces gráficas de usuario de los planificadores de tareas vistos en el capítulo anterior se elaboró un prototipo que muestra cómo posicionar la información para que los usuarios no se pierdan en la interfaz, garantizando también con él la accesibilidad y usabilidad de la interfaz, aspectos de gran importancia a la hora de diseñar una interfaz gráfica.

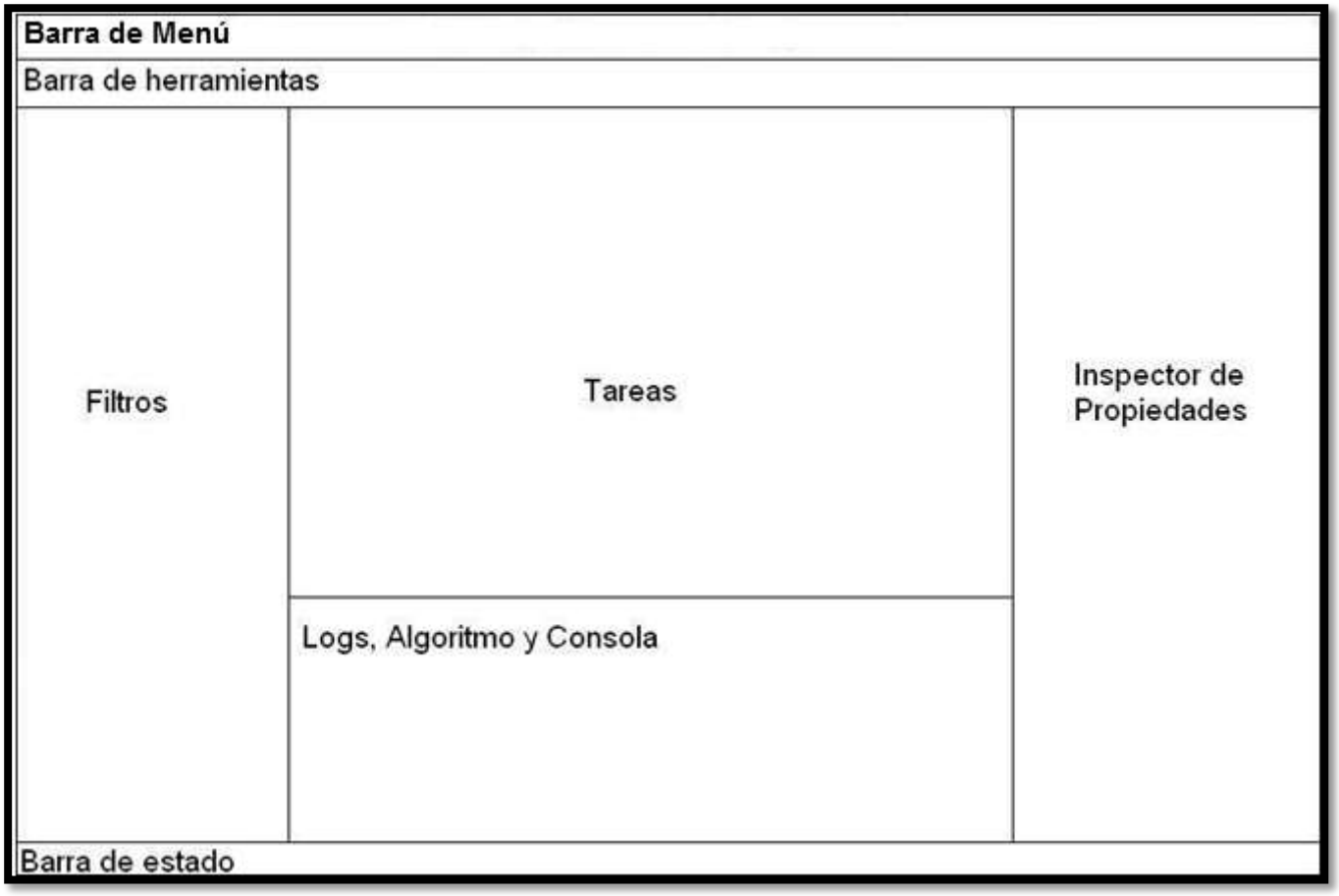


Fig. 6: Estructura de la Interfaz Gráfica de Usuario del Configurador de Tareas

2.2. Especificación de los requisitos del software

Hacer una identificación correcta de los requisitos funcionales que debe cumplir el sistema es muy importante. Un proyecto no puede ser exitoso sin una especificación correcta y exhaustiva de los requisitos [23]. Una vez que se han definido los principales conceptos relacionados con el objeto de estudio y el dominio, se puede comenzar a analizar qué debe hacer la aplicación para que cumpla con los objetivos planteados al inicio de este trabajo. Para ello, se enumeran a través de requisitos funcionales y no funcionales, las acciones y cualidades que el sistema debe poseer.

2.2.1. Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. El propósito fundamental de la captura de requisitos es guiar el desarrollo hacia un sistema correcto. De acuerdo con los objetivos propuestos el sistema debe ser capaz de:

RF 1. Configurar el sistema.

RF 2. Autenticar usuario.

RF 3. Gestionar tarea.

- **RF 3.1.** Crear tarea.
  - **RF 3.1.1.** Crear tarea periódica.
  - **RF 3.1.2.** Crear tarea aperiódica.
  - **RF 3.1.3.** Crear tarea síncrona.
  - **RF 3.1.4.** Crear tarea asíncrona.
- **RF 3.2.** Eliminar tarea.
- **RF 3.3.** Modificar tarea.

RF 4. Gestionar evento.

- **RF 4.1.** Crear evento.
  - **RF 4.1.1.** Crear evento asociado a una alarma.
  - **RF 4.1.2.** Crear evento asociado a un punto.
- **RF 4.2.** Eliminar evento.

RF 5. Visualizar tarea.

- **RF 5.1.** Mostrar propiedades.
- **RF 5.2.** Mostrar logs.
- **RF 5.3.** Mostrar algoritmo.
- **RF 5.4.** Mostrar ejecución en consola.

RF 6. Manejar flujo de tarea.

- **RF 6.1.** Planificar tarea.
- **RF 6.2.** Des-planificar tarea.
- **RF 6.3.** Reiniciar planificación.
- **RF 6.4.** Ejecutar tarea.

RF 6.5. Detener ejecución.

RF 7. Visualizar logs del sistema.

RF 8. Filtrar tareas.

RF 8.1. Filtrar por tipo.

RF 8.2. Filtrar por estado.

RF 9. Visualizar editor de scripts.

RF 10. Gestionar algoritmo.

RF 10.1. Crear algoritmo nuevo.

RF 10.2. Salvar algoritmo.

RF 10.3. Cargar algoritmo.

RF 11. Compilar algoritmo.

RF 11.1. Mostrar errores en el código.

RF 12. Visualizar ayuda de la API.

RF 13. Mostrar funciones de la API.

RF 14. Mostrar dispositivos del SCADA.

2.2.2. Requisitos no funcionales

En el siguiente epígrafe se especifican los requisitos no funcionales que se tuvieron en cuenta para el diseño y desarrollo de los prototipos del sistema.

Los requisitos no funcionales son propiedades o cualidades que debe cumplir el producto. Debe pensarse en ellos como las características que hacen al producto atractivo, usable, rápido y confiable.

Requisitos no funcionales de Usabilidad.

- Las personas que interactúan con el sistema deben haber sido capacitadas en algún curso básico para la interacción con la aplicación.
- La herramienta deberá poseer una interfaz de navegación acorde y funcional, tanto para usuarios expertos, como para los que no tienen conocimientos profundos de informática.

Requisitos no funcionales de Restricciones en el Diseño e Implementación.

- La aplicación será desarrollada en la distribución de GNU / Linux Debian.

- Se debe utilizar el Eclipse como IDE de desarrollo y Qt como biblioteca gráfica para la implementación del sistema así como Qt *Designer* para el diseño de sus interfaces gráficas.
- El sistema será desarrollado siguiendo el paradigma de programación orientada a objetos y bajo la filosofía de software libre.

Requisitos no funcionales de Apariencia o Interfaz Externa.

- El usuario debe tener accesos por varias vías a una determinada información.
- El sistema debe contar con una interfaz sencilla y amigable y que sea agradable al usuario.
- La aplicación debe utilizar como idioma principal el español, solo aparecerán en otro idioma las palabras técnicas que no puedan ser traducidas.
- Los botones expresarán su función, ya sea mediante su texto o la imagen que acompaña a este.

Requisitos no funcionales de Rendimiento.

- La herramienta deberá ser rápida y el tiempo de respuesta debe ser el menor posible, al igual que la velocidad de procesamiento de la información.

Requisitos no funcionales de Documentación.

- El sistema deberá seguir los estándares de documentación del Doxygen.

Requisitos no funcionales de Seguridad.

- El sistema deberá reconocer al usuario a través de su autenticación en el sistema y le permitirá realizar cualquier acción solo si la autenticación es satisfactoria.

Requisitos no funcionales de Disponibilidad

- El sistema debe estar disponible las 24 horas del día para garantizar la ejecución de las tareas en el momento requerido.

2.3. Actor del Sistema

Se considera actor del sistema a toda persona o sistema que interactúa con el sistema y se beneficia con el resultado de dicha interacción. En el sistema que se describe se detectó el siguiente actor:

Tabla 1. Actor del Sistema

Actor	**Descripción**
Operador	Representa el usuario que va a hacer uso del sistema.

2.4. Casos de Uso del sistema

La forma en que los usuarios usan el sistema es representada a través de los casos de uso. Estos últimos son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del actor. Los casos de uso identificados en el presente trabajo son enunciados a continuación:

2.4.1. Diagrama de Casos de Uso del sistema

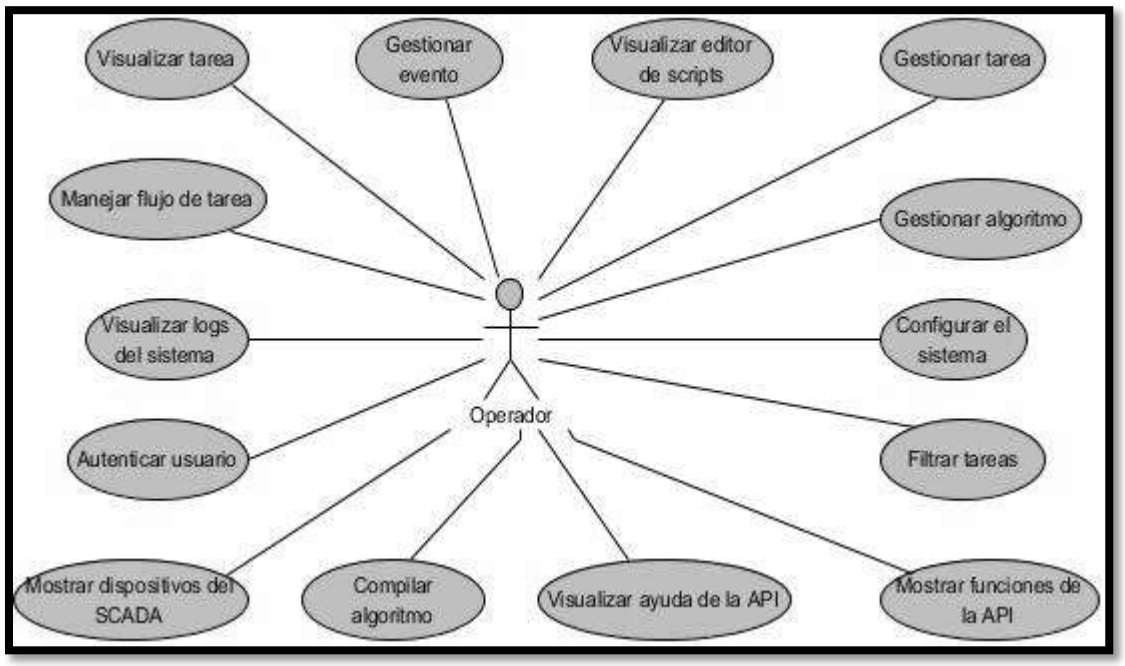


Fig. 7: Diagrama de Casos de Uso del sistema

2.4.2. Descripción de los Casos de Uso significativos del sistema

La descripción de los Casos de Uso muestra la expansión que permite comprender los procesos que se encuentran asociados a cada uno de ellos. A continuación se muestran las descripciones de los Casos de Uso de mayor importancia para la aplicación de acuerdo al criterio del autor.

Tabla 2. Descripción del Caso de Uso: Gestionar tarea

Caso de Uso	Gestionar tarea
Actores	Operador
Resumen	El caso de uso se encarga de la gestión de las tareas, el mismo se inicia cuando el operador desea crear, modificar o eliminar una tarea y termina cuando el sistema lleva a cabo la acción seleccionada por el usuario, y actualiza las tareas configuradas en el panel central del configurador.
Precondiciones	El operador debe estar autenticado en el sistema.
Referencias	RF 3, RF 3.1, RF 3.1.1, RF 3.1.2, RF 3.1.3, RF 3.1.4, RF 3.2, RF 3.3
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal" (Ver Anexo 1)	
Poscondiciones	El operador logra crear, eliminar o modificar una tarea según la opción seleccionada.

Tabla 3. Descripción del Caso de Uso: Gestionar evento

Caso de Uso	Gestionar evento
Actores	Operador
Resumen	El caso de uso se encarga de la gestión de eventos, se inicia cuando el

	operador va a crear una tarea síncrona y le va a asignar el evento asociado a ella, termina el caso de uso cuando el sistema lleva a cabo la acción seleccionada por el usuario.
Precondiciones	El operador debe estar creando una tarea síncrona.
Referencias	RF 4, RF 4.1, RF 4.1.1, RF 4.1.2, RF 4.2
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal" (Ver Anexo 2)	
Poscondiciones	El operador logra realizar sobre el evento la opción seleccionada.

Tabla 4. Descripción del Caso de Uso: Manejar flujo de tarea

Caso de Uso	Manejar flujo de tarea
Actores	Operador
Resumen	El caso de uso se inicia cuando el operador selecciona en la barra de herramientas del configurador de tareas o en el menú Flujo de Tareas la acción que desea hacer sobre la tarea y termina cuando el sistema ejecuta la acción seleccionada por el operador y le cambia el estado a la tarea.
Precondiciones	El operador debe estar autenticado y debidamente autorizado para poder manejar el flujo de una tarea. Deben existir tareas previamente creadas en el configurador de tareas. Debe haber una tarea seleccionada.
Referencias	RF 6, RF 6.1, RF 6.2, RF 6.3, RF 6.4, RF 6.5

Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal" (Ver Anexo 3)	
Poscondiciones	El operador logra realizar sobre la tarea la opción seleccionada.

Tabla 5. Descripción del Caso de Uso: Gestionar algoritmo

Caso de Uso	Gestionar algoritmo
Actores	Operador
Resumen	El caso de uso se encarga de la gestión de algoritmos, se inicia cuando el operador selecciona la opción en el menú Archivo o desde la barra de herramientas del editor de scripts la acción que desea llevar a cabo sobre el algoritmo y termina cuando el sistema ejecuta la acción solicitada por el operador.
Precondiciones	El usuario debe estar autenticado y autorizado para entrar al editor de scripts.
Referencias	RF 10, RF 10.1, RF 10.2, RF 10.3
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal" (Ver Anexo 4)	
Poscondiciones	El operador logra realizar sobre el algoritmo la opción seleccionada.

Tabla 6. Descripción del Caso de Uso: Compilar algoritmo

Caso de Uso	Compilar algoritmo
Actores	Operador
Resumen	El caso de uso se inicia cuando el operador selecciona la opción Compilar en el menú Proyecto o desde la barra de herramientas del editor de scripts, el caso de uso termina cuando el sistema compila el algoritmo y le muestra al operador los errores cometidos, en caso de tenerlos, en el panel inferior del editor.
Precondiciones	Debe existir un algoritmo en el editor.
Referencias	RF 11, RF 11.1
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal" (Ver Anexo 5)	
Poscondiciones	El operador logra compilar el algoritmo elaborado.

Tabla 7. Descripción del Caso de Uso: Visualizar tarea

Caso de Uso	Visualizar tarea
Actores	Operador
Resumen	El caso de uso se inicia cuando el operador selecciona una tarea en el configurador de tareas, el caso de uso termina cuando el sistema le muestra al operador las propiedades de dicha tarea, así como sus logs, el código de su algoritmo y los detalles de su ejecución en consola.

Precondiciones	Hay una tarea seleccionada en el configurador.
Referencias	RF 5, RF 5.1, RF 5.2, RF 5.3, RF 5.4
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal" (Ver Anexo 6)	
Poscondiciones	El operador logra conocer de la tarea seleccionada sus propiedades, logs, código o detalles de la ejecución en consola, según la opción seleccionada.

2.5. Prototipos de las interfaces gráficas de usuario significativas del sistema

Los prototipos de interfaz de usuario nos ayudan a comprender y especificar las interacciones entre los actores humanos y el sistema durante la captura de requisitos. No solo nos ayuda a desarrollar una interfaz gráfica mejor, sino también a comprender mejor los casos de uso. A la hora de especificar la interfaz de usuario también pueden utilizarse otros artefactos, como los modelos de interfaz gráfica y los esquemas de pantalla [24]**.** A continuación se muestran los prototipos de las interfaces gráficas más significativas de la aplicación.

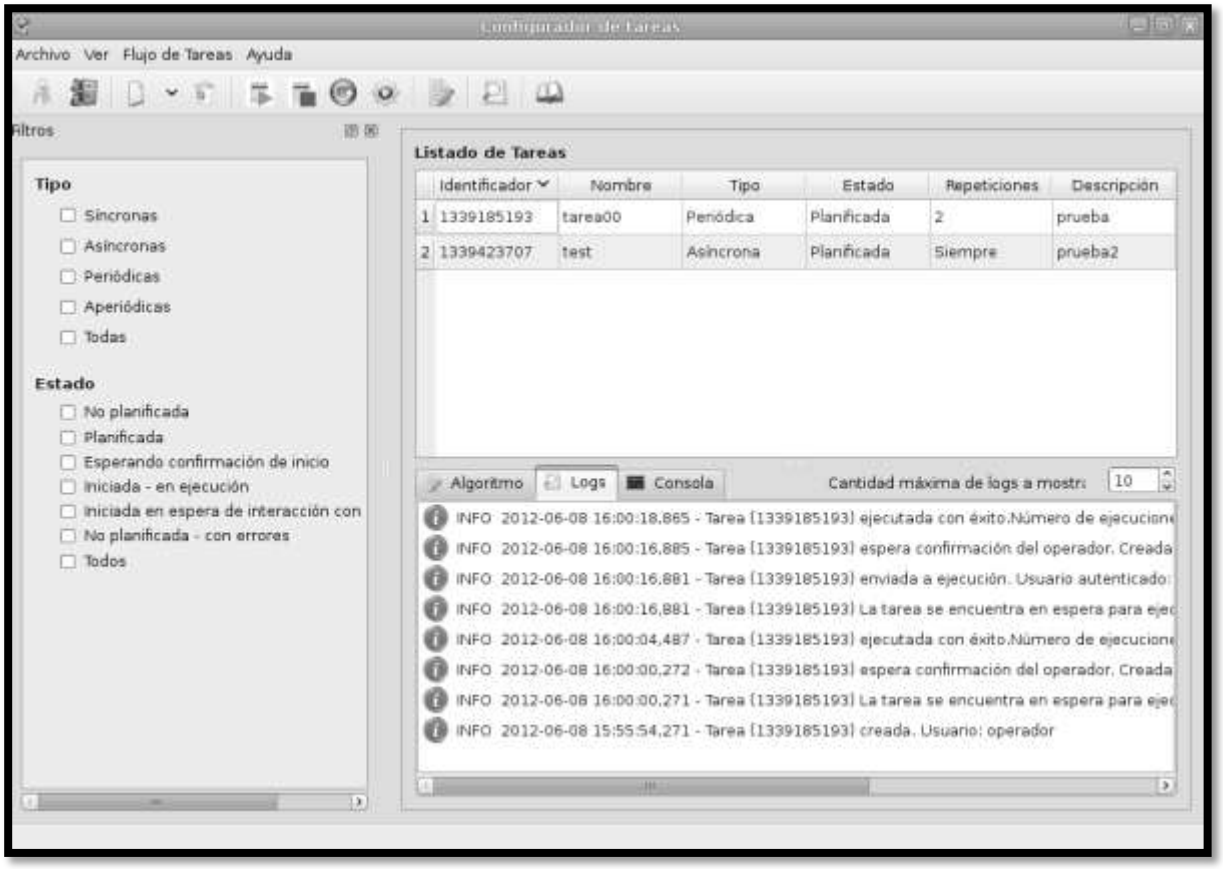


Fig. 8: Interfaz Gráfica de Usuario del Configurador de Tareas.

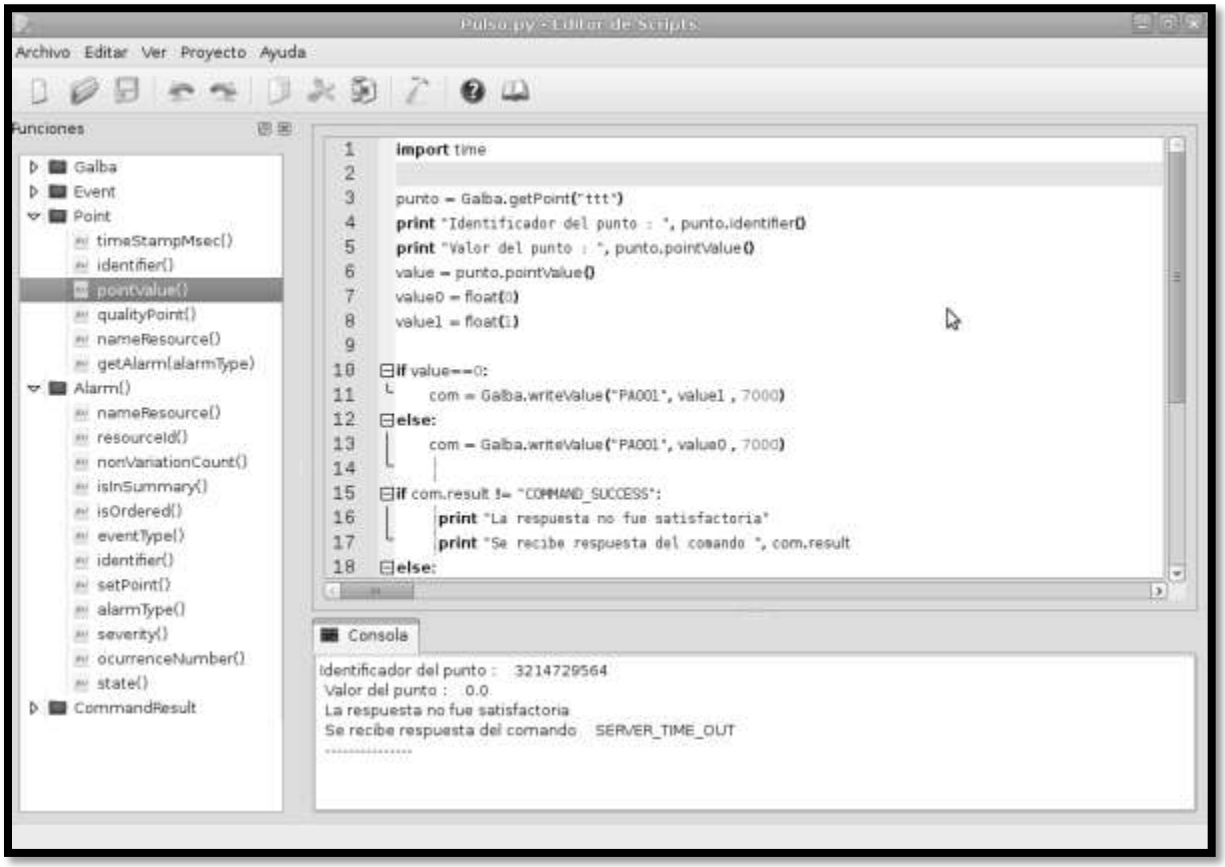


Fig. 9: Interfaz Gráfica de Usuario del Editor de Scripts.

CONCLUSIONES DEL CAPÍTULO

En el presente capitulo se han descrito las principales funcionalidades que deben estar presentes en la solución. Con este fin se definieron los requisitos funcionales, se agruparon en casos de usos, y fueron descritos dichos casos de uso para lograr un mejor entendimiento de los procesos que tendrán lugar en la aplicación a desarrollar. Además se diseñaron las interfaces gráficas necesarias para satisfacer todas las funcionalidades de la aplicación.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

INTRODUCCIÓN

En el presente capítulo se explican los aspectos relacionados con el diseño e implementación del sistema exponiéndolo a través de los diagramas de clases y de componentes del sistema, además se describen las clases significativas del sistema, el modelo de despliegue, los estilos de código utilizados y se explica el patrón de diseño utilizado en el desarrollo de la aplicación.

3.1. Diagrama de clases del diseño

El diagrama de clases es el diagrama principal del diseño para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencias. (**Ver Anexo 7**)

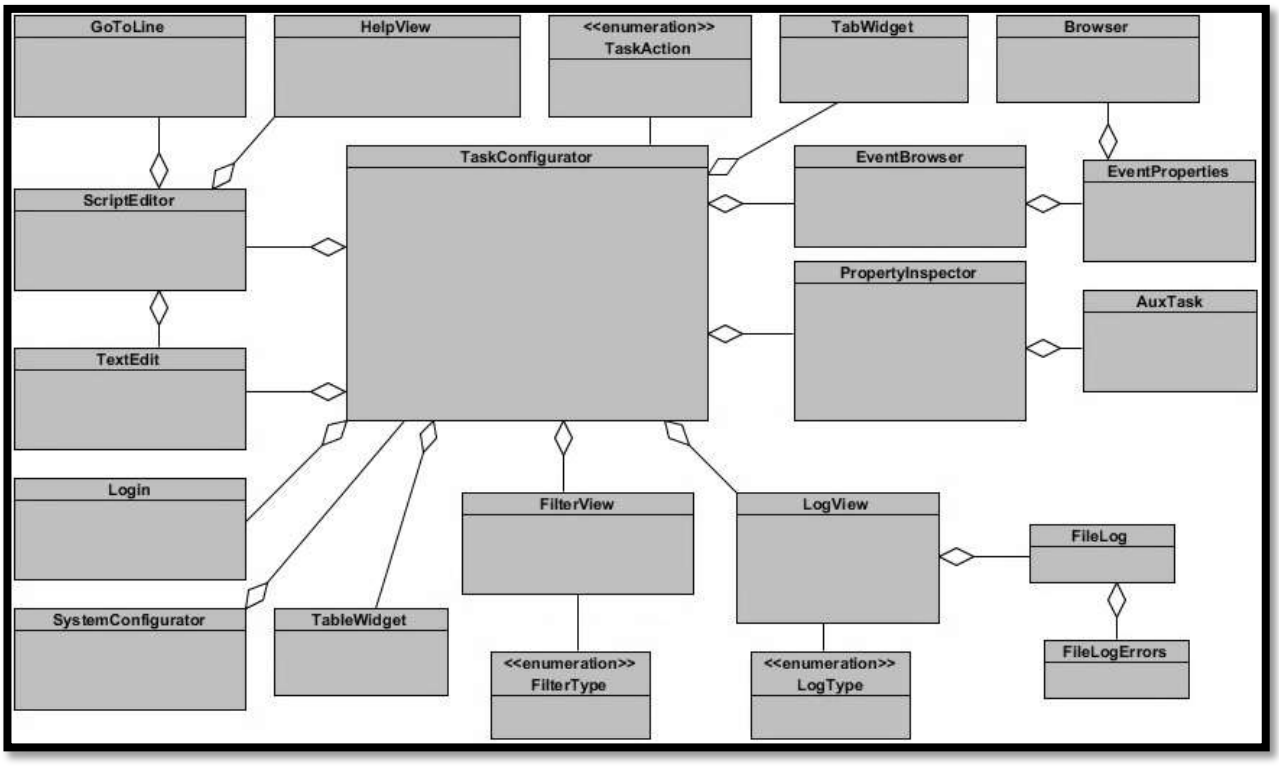


Fig. 10: Diagrama de clases

3.1.1. Descripción de las clases significativas

A continuación se realiza una descripción de las principales clases con las que cuenta el sistema con el objetivo de familiarizarse mejor con la función que realiza cada una de ellas.

Entre las principales clases que conforman la aplicación se encuentra la clase TaskConfigurator que es la entidad base de la interfaz gráfica del Configurador de tareas, ella contiene todos los atributos y funcionalidades necesarias para lograr una correcta gestión de las tareas.

Tabla 8. Clase "TaskConfigurator"

Nombre: TaskConfigurator	
Controladora	
Atributo	**Tipo**
filterView	FilterView*
scriptEditor	ScriptEditor*
eventsBrowser	EventsBrowser*
propertyInspector	PropertyInspector*
logsView	LogView*
loginWin	Login*
tabWidgetDetail	TabWidget*
tableWidget	TableWidget*
Para cada funcionalidad	
Nombre	execute(Task* aTask)
Descripción	Ejecuta la tarea especificada.
Nombre	addNewTask(Task* t)
Descripción	Adiciona la nueva tarea.
Nombre	modifyTask(Task* t)
Descripción	Modifica la tarea especificada.
Nombre	deleteTask(Task* task)
Descripción	Elimina la tarea especificada.
Nombre	pauseTask(Task* task)
Descripción	Pausa la tarea especificada.
Nombre	restarTask (Task* task)
Descripción	Reinicia la tarea especificada.
Nombre	configurePropertyInspector()
Descripción	Configura el inspector de propiedades.

La clase ScriptEditor se encarga de la gestión de scripts, es la entidad base de la interfaz gráfica del Editor de Scripts y en ella se incluyen todos los atributos y funcionalidades necesarios para lograr una correcta edición y compilación de los scripts.

Tabla 9. Clase "ScriptEditor"

Nombre: ScriptEditor	
Entidad	
Atributo	**Tipo**
helpView	HelpView*
Editor	TextEdit*
currentFile	QString
Para cada funcionalidad	
Nombre	initEditor(QString dir)
Descripción	Inicia el editor.
Nombre	save()
Descripción	Salva el script.
Nombre	Execute()
Descripción	Ejecuta el código del script.
Nombre	createTreeData()
Descripción	Muestra los dispositivos del SCADA.
Nombre	createTreeFunctions()
Descripción	Muestra las funciones de la API.
Nombre	showErrores(QString errors, QString accion)
Descripción	Muestra los errores en compilación o ejecución.
Nombre	Compile()
Descripción	Compila el código del script.

La clase EventProperties se encarga de la creación de eventos, es la entidad base de la interfaz gráfica del configurador de eventos, contiene todos los atributos y funcionalidades necesarios para lograr una correcta configuración de los mismos.

Tabla 10. Clase "EventProperties"

Nombre: EventProperties	
Entidad	
Atributo	**Tipo**
window	Browser*
eventBrow	EventsBrowser*
Para cada funcionalidad	
Nombre	AddEventByPoint()
Descripción	Añade un evento por punto.

Nombre	AddEventByAlarm()
Descripción	Añade un evento por alarma.
Nombre	alarmName(int num)
Descripción	Muestra el nombre de la alarma de acuerdo al identificador especificado.
Nombre	activationAlarmValue()
Descripción	Devuelve el valor final de activación de la alarma.

La clase PropertyInspector se encarga de la edición de las propiedades de las tareas, es la entidad base de la interfaz gráfica del Inspector de Propiedades y contiene todos los atributos y funcionalidades necesarios para la correcta edición de las mismas.

Tabla 11. Clase "PropertyInspector"

Nombre: PropertyInspector	
Entidad	
Atributo	**Tipo**
task	Task*
code	string
Para cada funcionalidad	
Nombre	fillProperties()
Descripción	Muestra las propiedades de la tarea en uso.
Nombre	desblok(bool acc)
Descripción	Desbloquea el inspector de propiedades.
Nombre	configure(TaskType type, QString operation)
Descripción	Configura el inspector para mostrar los atributos necesarios para cada tipo de tarea.
Nombre	idByNameEvent(string nameEvent)
Descripción	Devuelve el id de un evento según su nombre

La clase FilterView se encarga del filtrado de tareas, es la entidad base de la vista de filtros y contiene todos los atributos y funcionalidades necesarios para brindarle al usuario un filtrado de tareas por estado o por tipo.

Tabla 12. Clase "FilterView"

Nombre: FilterView	
Entidad	
Atributo	**Tipo**
actFilters	ActiveFilters
Para cada funcionalidad	
Nombre	updateActiveFilterList(FilterType type , int state)

Descripción	Actualiza la lista de filtros activa.
Nombre	stateFilterChange(int state)
Descripción	Activa un filtro de estado.
Nombre	typeFilterChange(int type)
Descripción	Activa un filtro de tipo.

3.2. Patrones de diseño. Singleton

Un patrón de diseño describe una estructura común recurrente de comunicar componentes que resuelven un problema de diseño general en un contexto particular [25]. A continuación se describe el patron de diseño Singleton el cual fue utilizado en el desarrollo de esta aplicación.

El patrón Singleton o Instancia única es útil para limitar el máximo número de instancias de una clase a solo una. En este caso, si más de un objeto necesita utilizar una instancia de la clase Singleton, esos objetos comparten la misma instancia de esta clase. En la aplicación dicho patrón es utilizado en las clases TaskConfigurator y FileLog.

La clase TaskConfigurator corresponde a la interfaz principal del Configurador de tareas y se encarga de la creación y edición de tareas, debido a esto se hace necesario el uso de este patrón para garantizar que de todas las secciones de código del módulo Aplicaciones se acceda a una única instancia del Configurador.

A su vez, la clase FileLog es utilizada para generar las trazas del módulo Aplicaciones. Debido a que este módulo hace uso de hilos se hace necesario controlar el acceso a los archivos de trazas desde un único lugar. En este caso el patrón Singleton permite que las trazas se escriban desde un único objeto, de manera que se logra una escritura secuencial y correcta sobre el archivo de trazas independientemente del hilo en que se está escribiendo.

3.3. Implementación del sistema

La fase de implementación en el desarrollo de un producto de software, en ella se ponen en práctica todas las descripciones y arquitecturas propuestas en la fase de análisis y diseño, es el complemento del trabajo de las fases que lo preceden dentro del proceso unificado de desarrollo de software. Ofrece una materialización precisa de los requisitos y en ella se obtienen como resultado componentes de código que se compilan e integran en versiones ejecutables.

3.3.1. Modelo de despliegue

El modelo de despliegue es un modelo de objeto que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre nodos de cómputo. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño [26]**.**

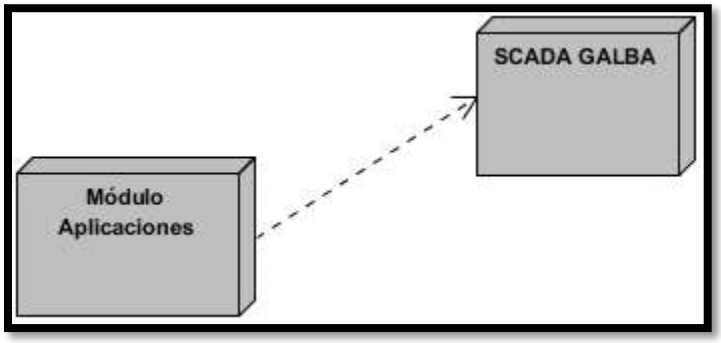


Fig. 11: Modelo de despliegue

Descripción del modelo de despliegue

Como se observa en la Fig. 11, uno de los nodos de este modelo es la PC donde se está ejecutando el módulo Aplicaciones y el otro nodo es la PC donde está el servidor del SCADA GALBA.
Aplicaciones se comunica con el SCADA GALBA a través del middleware mediante una conexión tcp / ip.

3.3.2. Diagrama de componentes del sistema

Los diagramas de componentes describen las relaciones los elementos que conforman un sistema. Es un modelo de implementación y permite visualizar la organización de los componentes. Estos representan todos los elementos del software que entran en la fabricación como: archivos, librerías o documentos [27]**.**

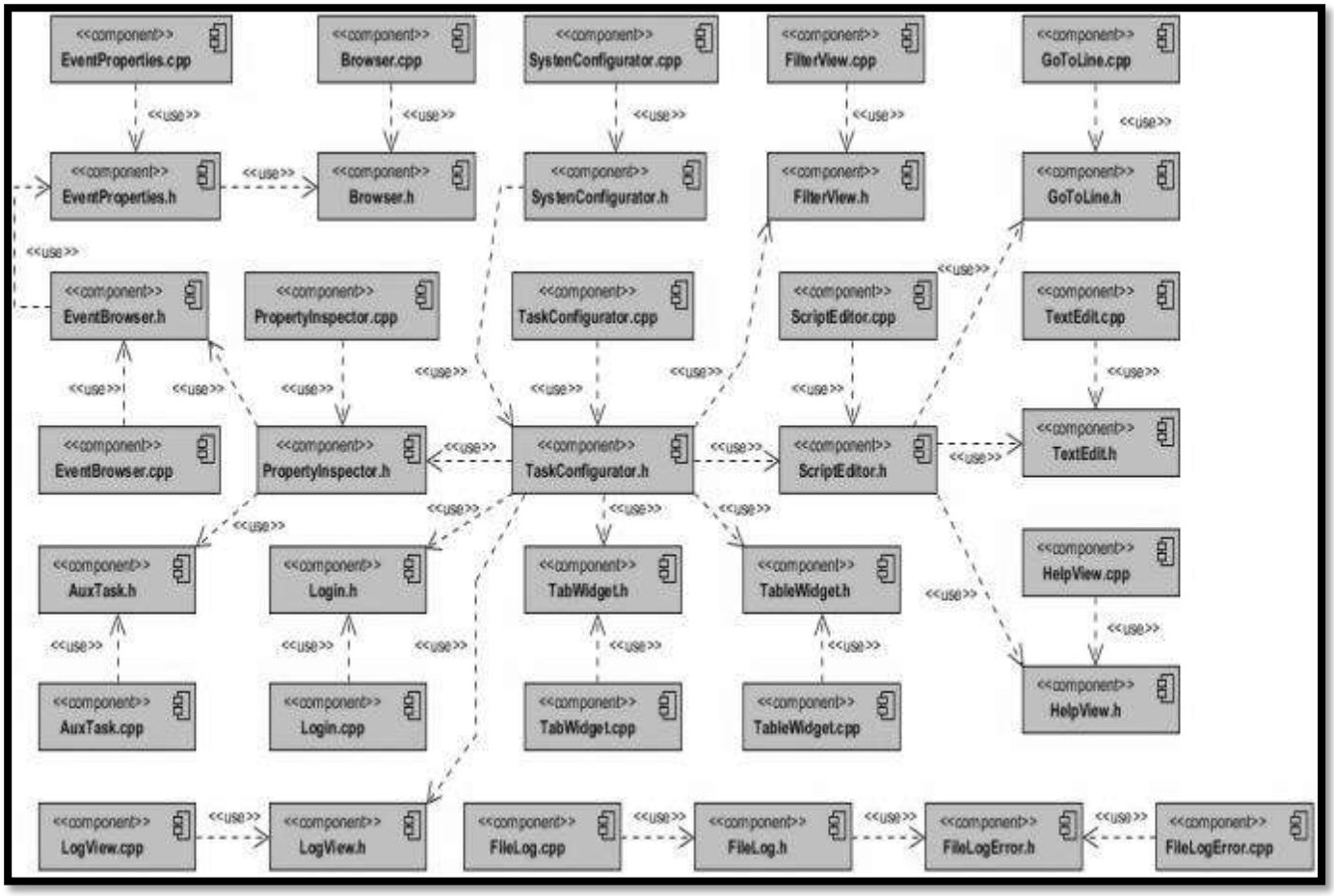


Fig. 12: Diagrama de componentes

3.3.3. Estilo de código

Nombres

- Los nombres de las clases son sustantivos.
- Dado que los nombres generalmente son el producto de concatenar varias palabras, se debe emplear mayúscula para el inicio de cada palabra y minúscula para el resto, con excepción de la primera letra del nombre, la cual debe ser en minúscula.
- En el caso de las clases se utiliza la estructura anterior, con la excepción de que la primera letra del nombre debe ser en mayúscula.
- Los nombres de las constantes deben contener solo letras mayúsculas.
- Los nombres de los métodos son frases que incluyen verbos.

- Los nombres de los atributos y parámetros son frases con sustantivos.

Manejo de errores

- Se pueden manejar los errores mediante mecanismos de excepciones o mediante valores de retorno, aunque esto debe ser uniforme dentro de un mismo objeto.

Documentación y comentarios

- Emplear oraciones completas al documentar.
- Documentar mientras se programa.
- Documentar cualquier cosa que no sea obvia en el código.
- Documentar cada rutina agregando: nombre del desarrollador, fecha, parámetros de entrada y descripción general del algoritmo.
- Al modificar el código se deben actualizar todos los comentarios y documentación asociada.
- Usar "//" para los comentarios de líneas simples y "/* */" para los comentarios de bloques de código.

Codificación

- Se establece un tamaño de identación estándar de 4 una tabulación, equivalente a 4 espacios.
- Alinear secciones del código.
- Alinear verticalmente llaves de apertura y cierre.
- Usar espacios antes y después de los operadores que el lenguaje de programación permita.
- Emplear select – case en sustitución de if anidados sobre las mismas variables.
- Emplear i, j, k, l, p, q, r para contadores en ciclos.
- Comentar siempre las llaves que cierren.
- Empelar al máximo operadores de tipo: +=, -=, /=, *=, ++, --, etc.
- Evitar colocar más de una sentencia por línea.

3.4. Validación de la solución propuesta

La realización de pruebas es una actividad en la cual un sistema o componente es ejecutado bajo ciertas condiciones o requisitos específicos, cuyos resultados son observados, registrados y evaluados. La validación del configurador de tareas se llevo a cabo por el equipo de pruebas del

SCADA GALBA mediante la ejecución de varios casos de prueba [28] [29] [30] [31] [32] [33] [34] [35] [36], en estos se detectaron algunos errores los cuales fueron corregidos y actualmente la aplicación funciona correctamente.

CONCLUSIONES DEL CAPÍTULO

El capítulo estuvo dirigido a presentar el diseño propuesto, se mostró el diagrama de clases del sistema así como la descripción de las clases de mayor relevancia dentro de dicho diagrama. También quedan plasmados el diagrama de componentes, el modelo de despliegue y se exponen los estilos de códigos utilizados durante la implementación de la aplicación. Además se justifica el uso del patrón de diseño Singleton en el desarrollo de la aplicación.

CONCLUSIONES GENERALES

Con el desarrollo del Configurador de tareas se da cumplimiento a los objetivos de este trabajo, pues da camino a la obtención de un sistema en el que se aplican los resultados de todo el proceso investigativo realizado a lo largo de las etapas de este trabajo, lográndose los siguientes resultados:

- Se desarrolló una aplicación que permite gestionar las tareas para el módulo Aplicaciones.
- Se logró ampliar las funcionalidades del SCADA GALBA al integrarle una herramienta que le permite crear y ejecutar tareas.

RECOMENDACIONES

Los objetivos de este trabajo de diploma han sido logrados, pero a lo largo de su desarrollo, han ido surgiendo ideas que podrían llevarse a cabo en un futuro, de forma que se logre una aplicación más útil y efectiva, para lo cual se recomienda:

- Modificar o crear el Configurador de tareas como un *plugin* para el actual HMI del SCADA GALBA.
- La confección de la ayuda de la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

1. Ingener Way. [En línea] [Citado el: 11 de 10 de 2010.] http://www.ingenerway.com/SCADA.htm.

2. Xuletas. [En línea] [Citado el: 12 de 10 de 2010.] http://www.xuletas.es/ficha/opc-2/.

3. **J, Guevara L. P y J. J. Medel.** *Introducción a los Sistemas de Tiempo Real.* México : s.n., 2003.

4. *Desarrollo SCADA Nacional. Glosario de Términos.* 2009.

5. **MORENO, ING. OSCAR ABIMAEL MORALES.** *FORMALIZACIÓN DEL CONCEPTO DE PLANIFICADOR DE TAREAS EN TIEMPO REAL CONCURRENTES.* México : s.n., 2009.

6. **G, Buttazzo.** *Hard real-time computing systems.* 1997.

7. **Salazar, Rismary Cabrera Estévez y Leonardo Samada.** *Módulo Administrador de Tareas del Proyecto Sistema de Gestión Fiscal.* Ciudad de La Habana : s.n., 2010.

8. **J. Aguilar, J. Calderón, V. Bravo, O. Cárdenas, J. González, D.Hernández, L. León, N. Pérez, D. Pernía, A. Ríos, F. Rivas y O.Terán.** *Levantamiento de Requerimientos y Determinación de los Paradigmas de la Plataforma de Automatización.* Mérida, Venezuela : s.n., 2004.

9. **J. Aguilar, H. Arciniegas, A. Bianchi, J. Calderón V. Bravo, O.Cárdenas, J. González, D. Hernández, L. León, N. Pérez, D. Pernía, A.Ríos, F. Rivas y O. Terán.** *Conceptualización de la Arquitectura de Automatización Industrial para Producción.* Mérida, Venezuela : s.n., 2003.

10. [En línea] [Citado el: 12 de 10 de 2011.] http://www.freedownloadmanager.org/es/downloads/Planificador_de_Tarea_Anticipado_21847_p/.

11. [En línea] [Citado el: 12 de 10 de 2011.] http://www.aplicacionesempresariales.com/task-coach-un-asesor-en-las-tareas.html.

12. [En línea] [Citado el: 11 de 10 de 2011.] http://freecode.com/projects/automize.

13. **Hernández, Yanet Campo Pérez y Juan Luis Vento.** *Herramienta para la Gestión de Riesgos en el Polo de Hardware y Automática.* La Habana : s.n., 2009.

14. DEBIAN. [En línea] [Citado el: 12 de 11 de 2010.] http://www.es.debian.org.

15. INFORAPID Portal de conocimiento. [En línea] [Citado el: 1 de 5 de 2012.] http://es.inforapid.org/index.php?search=GObject.

16. [En línea] [Citado el: 15 de 11 de 2011.] http://www.riverbankcomputing.co.uk/static/Docs/QScintilla2/index.html.

17. [En línea] [Citado el: 15 de 11 de 2011.] http://www.scintilla.org/.

18. [En línea] [Citado el: 25 de 11 de 2011.] http://cplusplus.com/info/description/.

19. [En línea] [Citado el: 12 de 6 de 2012.] www.python.org.

20. ECLIPSE. [En línea] [Citado el: 5 de 11 de 2011.] http://www.eclipse.org.

21. RUP. [En línea] [Citado el: 5 de 11 de 2011.] https://export.writer.zoho.com/public/jorge_luis10/metodolog%C3%ADa-rup3/fullpage.

22. JasperForge. [En línea] [Citado el: 17 de 1 de 2011.] http://jasperforge.org/projects/jasperreports.

23. **LARMAN, C.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999.

24. **Jacobson, J. Rumbaugh y I.** *El proceso unificado de desarrollo de software.* La Habana : s.n., 2000.

25. **E. Gamma, R. Helm, R. Johnson y J. Vlissides.** *Design Patterns - Elements of Reusable Object-Oriented Software .* 1995.

26. **Larman, C.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 1999.

27. buenas tareas. [En línea] [Citado el: 9 de 5 de 2012.] http://www.buenastareas.com/ensayos/Diagrama-De-Componentes/2744940.html.

28. *ESCMGA-SGA-0151-51_DCP_ConfigurarModuloDeAplicaciones.* 2011.

29. *ESCMGA-SGA-0151-52_DCP_CrearTareaDeAplicaciones.* 2011.

30. *ESCMGA-SGA-0151-53_DCP_EjecutarTareaDeAplicaciones.* 2011.

31. *ESCMGA-SGA-0151-54_DCP_EliminarTareaDeAplicaciones.* 2011.

32. *ESCMGA-SGA-0151-55_DCP_ManejarFlujoDeTareas.* 2011.

33. *ESCMGA-SGA-0151-56_DCP_ModificarTareaDeAplicaciones.* 2011.

34. *ESCMGA-SGA-0151-57_DCP_PlanificarTareasDeAplicaciones.* 2011.

35. *ESCMGA-SGA-0151-58_DCP_FiltrarTareasDeAplicaciones.* 2011.

36. *ESCMGA-SGA-0151-59_DCP_GestionarAlgoritmo.* 2011.

GLOSARIO DE TÉRMINOS

Alarma: Evento generado por un dispositivo o una función que señala la existencia de una condición anormal a través de un cambio discreto audible o visible, o ambas, que requiere su atención inmediata.

API: Grupo de rutinas que provee un sistema operativo, una aplicación o una biblioteca, que definen cómo invocar desde un programa un servicio que éstos prestan.

Arquitectura de la Información: Disciplina y arte encargada del estudio, análisis, organización, disposición y estructuración de la información en espacios de información, y de la selección y presentación de los datos en los sistemas de información interactivos y no interactivos.

bindings: Adaptación de una biblioteca para ser utilizada en otro lenguaje de programación distinto de aquél en el que ha sido creada.

CASE: Computer Aided Software Engineering, herramientas de ingeniería de software asistida por computadora.

Evento: Conjunto de acciones que proceden de la ejecución o activación de otra acción y que ha sido registrada, la combinación de estas acciones también pueden dar como resultado otro evento en particular o una serie de eventos.

Framework: estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.

GUI: Interfaz Gráfica de Usuario.

HMI: Human Machine Interface (Interfaz hombre máquina).

IDE: Integrated Development Environment (Ambiente de desarrollo integrado).

Interfaz de Usuario: Medio que el usuario utiliza para comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.

Interfaz Gráfica de Usuario: Componente de una aplicación informática que el usuario visualiza y a través de la cual opera con ella. Está formada por ventanas, botones, menús e iconos, entre otros elementos.

LGPL: Licencia creada por la Fundación de Software Libre y orientada principalmente a los términos de distribución, modificación y uso de software libre.

Logs: Registros del sistema, se refieren a acciones, comportamientos, entre otros, realizados por los sistemas o sobre ellos.

Multiplataforma: Término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

Plugin: es un módulo de hardware o software que añade una característica o un servicio especifico a un sistema más grande.

Punto: Los puntos se refieren a las direcciones de memoria que se configuran en el SCADA y que referencian a una variable capturada por un dispositivo de campo, o una variable del proceso. Pueden ser de tipo analógico, digital, texto y vector.

Script: En informática, un script es un conjunto de instrucciones que permiten automatizar tareas creando pequeñas utilidades. Por lo general son interpretadas y editadas como archivos de texto.

Software: es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.

Subversion: Sistema de control de versiones, también conocido como svn por ser el nombre utilizado en la línea de comandos.

Templates: (plantillas) son el mecanismo de C++ para implementar el paradigma de la programación genérica. Permiten que una clase o función trabaje con tipos de datos abstractos, especificándose más adelante cuales son los que se quieren usar.

XML: Siglas en inglés de eXtensible Markup Language (lenguaje de marcas extensibles).

widgets: Elemento de una interfaz gráfica de usuario que muestra información con la cual el usuario puede interactuar.

ANEXOS

Anexo 1. **Descripción ampliada CU "Gestionar Tarea"**

Caso de Uso	Gestionar tarea	
Actores	Operador	
Resumen	El caso de uso se encarga de la gestión de las tareas, el mismo se inicia cuando el operador desea crear, modificar o eliminar una tarea y termina cuando el sistema lleva a cabo la acción seleccionada por el usuario, y actualiza las tareas configuradas en el panel central del configurador.	
Precondiciones	El operador debe estar autenticado en el sistema	
Referencias	RF 3, RF 3.1, RF 3.1.1, RF 3.1.2, RF 3.1.3, RF 3.1.4, RF 3.2, RF 3.3	
Prioridad	Crítico	
Flujo Normal de Eventos		
Sección "Principal"		
Acción del Actor		**Respuesta del Sistema**
a) Si el actor desea crear una tarea ir a la sección "Crear Tarea". b) Si el actor desea modificar una tarea ir a la sección "Modificar Tarea". c) Si el actor desea eliminar una tarea ir a la sección "Eliminar Tarea".		
Sección "Crear Tarea"		

a) Si el actor desea crear una tarea periódica ir a la sección “Crear Tarea Periódica”. b) Si el actor desea crear una tarea aperiódica ir a la sección “Crear Tarea Aperiódica”. c) Si el actor desea crear una tarea síncrona ir a la sección “Crear Tarea Síncrona”. d) Si el actor desea crear una tarea asíncrona ir a la sección “Crear Tarea Asíncrona”.	
Sección “Crear Tarea Periódica”	
1. El actor selecciona la opción “Periódica” en la barra de herramientas del configurador de tareas o desde el menú “Archivo / Nueva”.	2. El sistema le muestra en el panel derecho un inspector de propiedades con los campos necesarios para crear una tarea periódica.
3. El operador completa los campos anteriormente expresados y da clic en el botón “Aceptar” que se encuentra en la parte inferior del inspector de propiedades.	4. El sistema verifica que todos los campos contienen información y que dicha información cumple con el formato requerido (Ver Flujo Alterno 4.1). 5. El sistema muestra la tarea creada en el panel central del configurador de tareas.
Sección “Crear Tarea Aperiódica”	
1. El actor selecciona la opción	2. El sistema le muestra en el panel

"Aperiódica" en la barra de herramientas del configurador de tareas o desde el menú "Archivo / Nueva".	derecho un inspector de propiedades con los campos necesarios para crear una tarea aperiódica.
3. El operador completa los campos anteriormente expresados y da clic en el botón "Aceptar" que se encuentra en la parte inferior del inspector de propiedades.	4. El sistema verifica que todos los campos contienen información y que dicha información cumple con el formato requerido (Ver Flujo Alterno 4.1). 5. El sistema muestra la tarea creada en el panel central del configurador de tareas.
Sección "Crear Tarea Síncrona"	
1. El actor selecciona la opción "Síncrona" en la barra de herramientas del configurador de tareas o desde el menú "Archivo / Nueva".	2. El sistema le muestra en el panel derecho un inspector de propiedades con los campos necesarios para crear una tarea síncrona.
3. El operador completa los campos anteriormente expresados y da clic en el botón "Aceptar" que se encuentra en la parte inferior del inspector de propiedades.	4. El sistema verifica que todos los campos contienen información y que dicha información cumple con el formato requerido (Ver Flujo Alterno 4.1). 5. El sistema muestra la tarea creada en el panel central del configurador de tareas.
Sección "Crear Tarea Asíncrona"	

1. El actor selecciona la opción “Asíncrona” en la barra de herramientas del configurador de tareas o desde el menú “Archivo / Nueva”.	2. El sistema le muestra en el panel derecho un inspector de propiedades con los campos necesarios para crear una tarea asíncrona.
3. El operador completa los campos anteriormente expresados y da clic en el botón “Aceptar” que se encuentra en la parte inferior del inspector de propiedades.	4. El sistema verifica que todos los campos contienen información y que dicha información cumple con el formato requerido (Ver Flujo Alterno 4.1). 5. El sistema muestra la tarea creada en el panel central del configurador de tareas.
Sección “Modificar Tarea”	
1. El operador selecciona en el panel central la tarea que desea modificar.	2. El sistema le muestra en el panel derecho un inspector de propiedades con los campos editables de la tarea seleccionada.
3. El operador ingresa los cambios que desea y da clic en el botón “Aceptar”.	4. El sistema verifica que todos los campos contienen información y que dicha información cumple con el formato requerido (Ver Flujo Alterno 4.1). 5. El sistema muestra la tarea creada en el panel central del configurador de tareas y los nuevos valores de sus propiedades son mostrados en el inspector de propiedades.

Sección "Eliminar Tarea"	
1. El operador selecciona en el panel central la tarea que desea eliminar. 2. El operador selecciona la opción "Eliminar" desde la barra de tareas del configurador de tareas o desde el menú "Archivo".	3. El sistema verifica que haya al menos una tarea seleccionada (Ver Flujo Alterno 3.1). 4. El sistema actualiza la vista de las tareas que se muestra en el panel central del configurador de tareas quitando de él la o las tareas seleccionadas por el operador.
Flujo Alternativo	
	3.1. Si no existe ninguna tarea seleccionada en el configurador de tareas se le muestra al operador el mensaje "Debe seleccionar una tarea".
3.2. El operador selecciona una tarea y retorna al paso 2 del flujo principal.	
	4.1. Si algún campo no contiene información el sistema le muestra al operador el mensaje "Faltan por llenar los siguientes datos:" y a continuación se le listan los campos vacíos.
4.2. El operador acepta el mensaje, llena los campos y da clic en el botón "Aceptar".	4.3. El sistema retorna al paso 4 del flujo principal.
Poscondiciones	El operador logra crear, eliminar o modificar una

	tarea según la opción seleccionada.

Anexo 2. **Descripción ampliada CU "Gestionar Evento"**

Caso de Uso	Gestionar evento
Actores	Operador
Resumen	El caso de uso se encarga de la gestión de eventos, se inicia cuando el operador va a crear una tarea síncrona y le va a asignar el evento asociado a ella, termina el caso de uso cuando el sistema lleva a cabo la acción seleccionada por el usuario.
Precondiciones	El operador debe estar creando una tarea síncrona
Referencias	RF 4, RF 4.1, RF 4.1.1, RF 4.1.2, RF 4.2
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**
a) Si el actor desea crear un evento ir a la sección "Crear Evento". b) Si el actor desea eliminar un evento ir a la sección "Eliminar Evento".	
Sección "Crear Evento"	
a) Si el actor desea crear un evento asociado a una alarma ir a la	

sección “Crear Evento Asociado a Alarma”. b) Si el actor desea crear un evento asociado a una alarma ir a la sección “Crear Evento Asociado a Punto”.	
Sección “Crear Evento Asociado a Alarma”	
1. El operador selecciona la opción “crear un evento nuevo asociado a una alarma”.	2. El sistema le despliega en la ventana los campos necesarios para crear un evento asociado a una alarma.
3. El operador inserta los datos y da clic en el botón “Aceptar”.	4. El sistema verifica que todos los campos contengan información y que estos cumplan con el formato requerido (Ver Flujo Alterno 4.1). 5. El sistema crea el evento y se lo muestra en la vista central del cuadro de diálogo.
Sección “Crear Evento Asociado a Punto”	
1. El operador selecciona la opción “crear un evento nuevo asociado a un punto”.	2. El sistema le despliega en la ventana los campos necesarios para crear un evento asociado a un punto.
3. El operador inserta los datos y da clic en el botón “Aceptar”.	4. El sistema verifica que todos los campos contengan información y que estos cumplan con el formato requerido (Ver Flujo Alterno 4.1). 5. El sistema crea el evento y se lo

	muestra en la vista central del cuadro de diálogo.
Sección "Eliminar Evento"	
1. El operador selecciona el evento que desea eliminar. 2. El operador da clic en el botón "Eliminar".	3. El sistema verifica que exista un evento seleccionado (Ver Flujo Alterno 3.1). 4. El sistema le muestra al operador el mensaje "¿Está seguro que desea eliminar el evento?" y a continuación se le muestra el identificador del evento seleccionado.
5. El operador acepta el mensaje.	6. El sistema elimina el evento seleccionado y actualiza la vista central del cuadro de diálogo eliminando dicho evento de ella.
Flujo Alternativo	
	3.1. Si no hay ningún evento seleccionado el sistema le muestra al usuario el mensaje "Seleccione el evento que desea eliminar".
3.2. El operador selecciona un evento, da clic en el botón "Eliminar" y regresa al paso 4 del flujo principal.	
	4.1. Si algún campo no contiene información el sistema le muestra al operador el mensaje "Faltan por llenar los siguientes datos" y

	a continuación se listan los campos vacíos.
4.2. El operador acepta el mensaje, llena los datos y da clic en el botón "Aceptar".	4.3. El sistema retorna al paso 4 del flujo principal.
Poscondiciones	El operador logra realizar sobre el evento la opción seleccionada.

Anexo 3. **Descripción ampliada CU "Manejar flujo de tarea"**

Caso de Uso	Manejar flujo de tarea
Actores	Operador
Resumen	El caso de uso se inicia cuando el operador selecciona en la barra de herramientas del configurador de tareas o en el menú Flujo de Tareas la acción que desea hacer sobre la tarea y termina cuando el sistema ejecuta la acción seleccionada por el operador y le cambia el estado a la tarea.
Precondiciones	El operador debe estar autenticado y debidamente autorizado para poder manejar el flujo de una tarea Deben existir tareas previamente creadas en el configurador de tareas Debe haber una tarea seleccionada
Referencias	RF 6, RF 6.1, RF 6.2, RF 6.3, RF 6.4, RF 6.5
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**

a) Si el actor desea planificar una tarea ir a la sección “Planificar Tarea”. b) Si el actor desea des-planificar una tarea ir a la sección “Des-planificar Tarea”. c) Si el actor desea reiniciar una tarea ir a la sección “Reiniciar Tarea”. d) Si el actor desea ejecutar una tarea ir a la sección “Ejecutar Tarea”. e) Si el actor desea detener una tarea ir a la sección “Detener Tarea”.	
Sección “Planificar Tarea”	
1. El operador selecciona la tarea que desea planificar en el panel central del configurador de tareas. 2. El operador selecciona la opción “Iniciar Planificación” en la barra de herramientas del configurador de tareas o en el menú “Flujo de Tareas”.	3. El sistema verifica que exista al menos una tarea seleccionada (Ver Flujo Alterno 3.1). 4. El sistema le muestra al operador un mensaje de confirmación para realizar la opción seleccionada.
5. El operador presiona el botón “Si”	6. El sistema cambia el estado de la tarea a “Planificada” y actualiza la vista de tareas que se encuentra en el panel central del configurador de tareas.
Sección “Des-planificar Tarea”	
1. El operador selecciona la tarea que desea des-planificar en el panel	3. El sistema verifica que exista al menos una tarea seleccionada (Ver

central del configurador de tareas. 2. El operador selecciona la opción “Des-planificar” en la barra de herramientas del configurador de tareas o en el menú “Flujo de Tareas”.	Flujo Alterno 3.1). 4. El sistema le muestra al operador un mensaje de confirmación para realizar la opción seleccionada.
5. El operador presiona el botón “Si”.	6. El sistema cambia el estado de la tarea a “Des-planificada” y actualiza la vista de tareas que se encuentra en el panel central del configurador de tareas.
Sección “Reiniciar Tarea”	
1. El operador selecciona la tarea que desea reiniciar en el panel central del configurador de tareas. 2. El operador selecciona la opción “Reiniciar Planificación” en la barra de herramientas del configurador de tareas o en el menú “Flujo de Tareas”.	3. El sistema verifica que exista al menos una tarea seleccionada (Ver Flujo Alterno 3.1). 4. El sistema le muestra al operador un mensaje de confirmación para realizar la opción seleccionada.
5. El operador presiona el botón “Si”.	6. El sistema cambia el estado de la tarea a “Reiniciada” y actualiza la vista de tareas que se encuentra en el panel central del configurador de tareas.
Sección “Ejecutar Tarea”	
1. El operador selecciona la tarea que	3. El sistema verifica que exista al

desea ejecutar en el panel central del configurador de tareas. 2. El operador selecciona la opción “Ejecutar” en la barra de herramientas del configurador de tareas o en el menú “Flujo de Tareas”.	menos una tarea seleccionada (Ver Flujo Alterno 3.1). 4. El sistema le muestra al operador un mensaje de confirmación para realizar la opción seleccionada.
5. El operador presiona el botón “Si”.	6. El sistema cambia el estado de la tarea a “En ejecución” y actualiza la vista de tareas que se encuentra en el panel central del configurador de tareas.
Sección “Detener Tarea”	
1. El operador selecciona la tarea que desea detener en el panel central del configurador de tareas. 2. El operador selecciona la opción “Detener Planificación” en la barra de herramientas del configurador de tareas o en el menú “Flujo de Tareas”.	3. El sistema verifica que exista al menos una tarea seleccionada (Ver Flujo Alterno 3.1). 4. El sistema le muestra al operador un mensaje de confirmación para realizar la opción seleccionada.
5. El operador presiona el botón “Si”.	6. El sistema cambia el estado de la tarea a “Detenida por el usuario” y actualiza la vista de tareas que se encuentra en el panel central del configurador de tareas.
Flujo Alternativo	
	3.1. Si no existe una tarea seleccionada

	el sistema le muestra al operador el mensaje “Debe seleccionar una tarea”.
3.2. El operador acepta el mensaje y retorna al paso 1 del flujo principal.	
Poscondiciones	El operador logra realizar sobre la tarea la opción seleccionada.

Anexo 4. **Descripción ampliada CU “Gestionar algoritmo”**

Caso de Uso	Gestionar algoritmo
Actores	Operador
Resumen	El caso de uso se encarga de la gestión de algoritmos, se inicia cuando el operador selecciona la opción en el menú Archivo o desde la barra de herramientas del editor de scripts la acción que desea llevar a cabo sobre el algoritmo y termina cuando el sistema ejecuta la acción solicitada por el operador.
Precondiciones	El usuario debe estar autenticado y autorizado para entrar al editor de scripts.
Referencias	RF 10, RF 10.1, RF 10.2, RF 10.3
Prioridad	Crítico
Flujo Normal de Eventos	
Sección “Principal”	
Acción del Actor	**Respuesta del Sistema**
a) Si el actor desea crear un algoritmo nuevo ir a la sección	

"Crear Algoritmo". b) Si el actor desea salvar un algoritmo ir a la sección "Salvar Algoritmo". c) Si el actor desea cargar un algoritmo ir a la sección "Cargar Algoritmo".	
Sección "Crear Algoritmo"	
1. El operador selecciona la opción "Nuevo" en la barra de herramientas del editor de scripts o desde el menú "Archivo".	2. El sistema le muestra al operador el editor en blanco.
3. El operador elabora su algoritmo y cierra el editor.	
Sección "Salvar Algoritmo"	
1. El operador selecciona la opción "Salvar" en la barra de herramientas del editor de scripts o desde el menú "Archivo".	2. El sistema le muestra al operador una ventana para que el operador introduzca el nombre del archivo a guardar.
3. El operador introduce el nombre del archivo y presiona el botón "Aceptar".	4. El sistema guarda el algoritmo en un archivo de extensión ".py" con el nombre entrado anteriormente por el operador (Ver Flujo Alterno 4.1). 5. El sistema activa la opción "Compilar" en la barra de herramientas del editor de scripts.
Sección "Cargar Algoritmo"	

1. El operador selecciona la opción “Abrir” en la barra de herramientas del editor de scripts o desde el menú “Archivo”.	2. El sistema abre una ventana donde el operador selecciona el algoritmo que desea cargar. 3. El sistema abre el archivo seleccionado mostrándose el algoritmo en el editor. 4. El sistema activa la opción “Compilar” en la barra de herramientas del editor de scripts.
Flujo Alternativo	
	1.1. Si existe otro archivo con el mismo nombre el sistema le muestra al operador el mensaje “¿Desea reemplazar el archivo existente?”.
1.2. El operador toma la decisión y retorna al paso 5 del flujo principal.	
Poscondiciones	El operador logra realizar sobre el algoritmo la opción seleccionada.

Anexo 5. **Descripción ampliada CU “Compilar algoritmo”**

Caso de Uso	Compilar algoritmo
Actores	Operador
Resumen	El caso de uso se inicia cuando el operador selecciona la opción Compilar en el menú Proyecto o desde la barra de herramientas del editor de scripts, el caso de uso termina cuando el sistema compila el algoritmo y le muestra al operador los errores cometidos, en caso de tenerlos, en el panel inferior del

	editor.
Precondiciones	Debe existir un algoritmo en el editor
Referencias	RF 11, RF 11.1
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**
1. El operador selecciona la opción "Compilar" en la barra de herramientas del editor de scripts o en el menú Proyecto.	2. El sistema verifica que código del script elaborado por el operador no contenga errores (Ver Flujo Alterno 2.1). 3. El sistema le muestra al operador el mensaje "Script ejecutado".
Flujo Alternativo	
	2.1. Si el código del script elaborado por el operador contiene errores el sistema le muestra al operador los errores cometidos en el panel inferior del editor de scripts.
2.2. El operador corrige los errores cometidos en el script y regresa al paso 1 del flujo principal.	
Poscondiciones	El operador logra compilar el algoritmo elaborado.

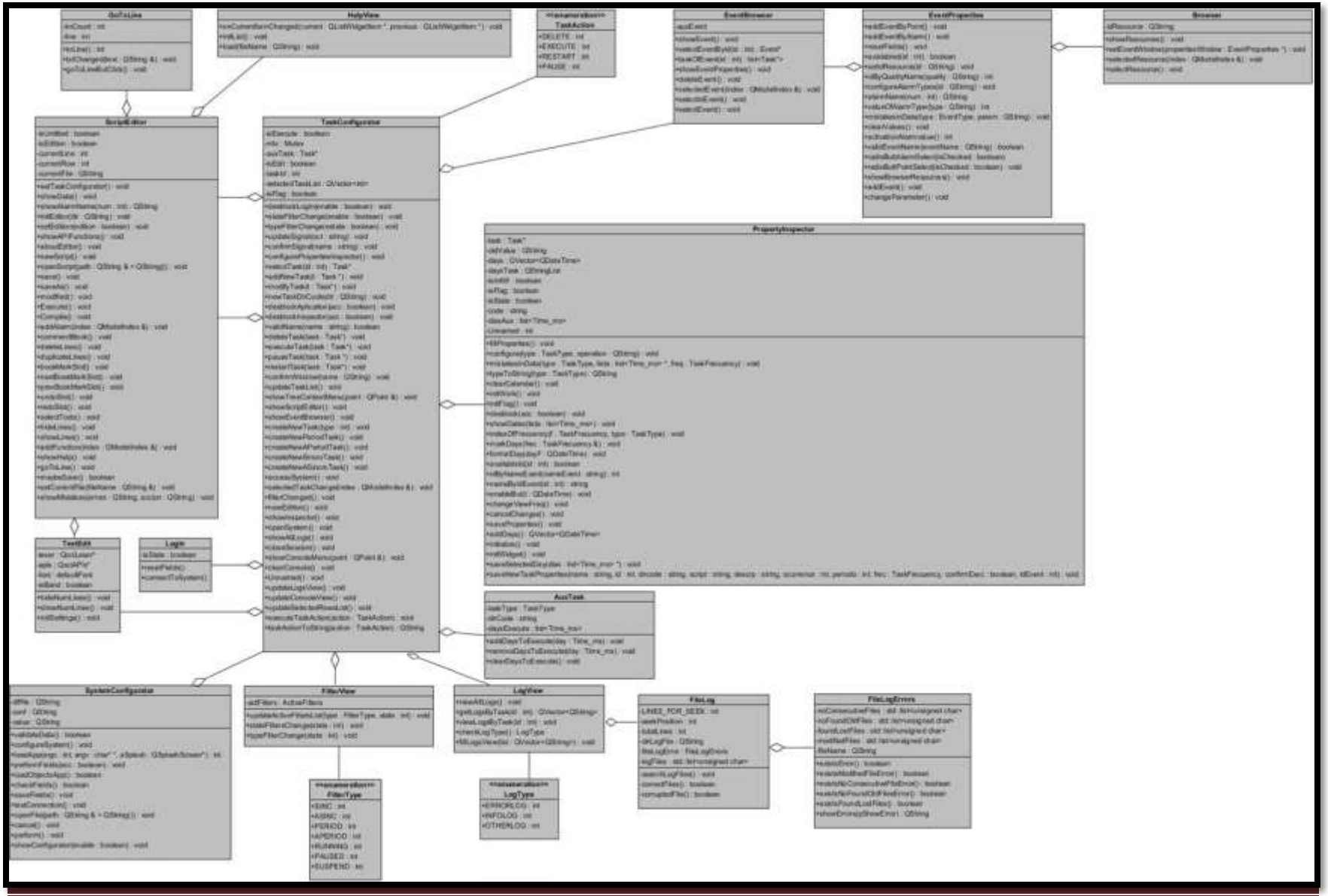
Anexo 6. **Descripción ampliada CU "Visualizar tarea"**

Caso de Uso	Visualizar tarea
Actores	Operador
Resumen	El caso de uso se inicia cuando el operador selecciona una tarea en el configurador de tareas, el caso de uso termina cuando el sistema le muestra al operador las propiedades de dicha tarea, así como sus logs, el código de su algoritmo y los detalles de su ejecución en consola (Ver Anexo 8).
Precondiciones	Hay una tarea seleccionada en el configurador.
Referencias	RF 5, RF 5.1, RF 5.2, RF 5.3, RF 5.4
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**
e) Si el actor desea mostrar las propiedades de una tarea ir a la sección "Mostrar Propiedades". f) Si el actor desea mostrar los logs de una tarea ir a la sección "Mostrar Logs". g) Si el actor desea mostrar el código de una tarea ir a la sección "Mostrar Algoritmo". h) Si el actor desea mostrar los	

detalles de la ejecución en consola de una tarea ir a la sección “Mostrar Ejecución en Consola”.	
Sección “Mostrar Propiedades”	
1. El operador selecciona la tarea de la cual quiere conocer sus propiedades.	2. El sistema le muestra al operador las propiedades de la tarea seleccionada en el inspector de propiedades del configurador de tareas.
Sección “Mostrar Logs”	
1. El operador selecciona la tarea de la cual quiere conocer sus logs.	2. El sistema le muestra al operador los logs la tarea seleccionada en la pestaña Logs que se encuentra en el panel inferior del configurador de tareas.
Sección “Mostrar Algoritmo”	
1. El operador selecciona la tarea de la cual quiere conocer su código.	2. El sistema le muestra al operador el código de la tarea seleccionada en la pestaña Algoritmo que se encuentra en el panel inferior del configurador de tareas.
Sección “Mostrar Ejecución en Consola”	
1. El operador selecciona la tarea de la cual quiere conocer los detalles de su	2. El sistema le muestra al operador los detalles de la ejecución en consola de la

ejecución en consola.	tarea seleccionada en la pestaña Consola que se encuentra en el panel inferior del configurador de tareas.
Poscondiciones	El operador logra conocer de la tarea seleccionada sus propiedades, logs, código o detalles de la ejecución en consola, según la opción seleccionada.

Anexo 7. **Diagrama de Clases Ampliado**


GoToLine
HelpView
<<enumeration>>
TaskAction
EventBrowser
EventProperties
Browser
ScriptEditor
TaskConfigurator
PropertyInspector
TextEdit
Login
AuxTask
SystemConfigurator
FilterView
LogView
FileLog
FileLogErrors
<<enumeration>>
FilterType
<<enumeration>>
LogType