Universidad de las Ciencias Informáticas

Facultad 5

Implementación de un modelo para la configuración de un sistema SCADA.

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

Autor: Yosell Luis Sehara Driggs

Tutor: Ing. Rene López Banacaldo

Ciudad de la Habana, Julio de 2008.

"Año 50 del triunfo de la Revolución"

Declaración de autoría

Por este medio declaro ser autor de este trabajo y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Firma del Autor	Firma del Tutor
(Yosell Luis Sehara Driggs)	(Ing. Rene López Banacaldo)

Datos de contacto

Ing. Rene López Banacaldo

Graduado de Ingeniero Informático en el 2004 y profesor adiestrado con 4 años de experiencia docente en la Universidad de las Ciencias Informáticas (UCI) y 4 en el desarrollo de software.

Agradecimientos

A todos mis compañeros del proyecto SCADA y a los que me han ayudado a realizar este trabajo.

A la Universidad de las Ciencias Informáticas por las oportunidades que me ha brindado.

Gracias a mi familia por sus consejos, la educación que me han dado y el apoyo en cada una de mis decisiones.

Dedicatoria

A toda mi familia en especial a mi querida hermana Yailín, a mi hermano Luis Manuel y a mis padres Mariela y José.

Resumen

En los últimos años el crecimiento de las industrias ha conllevado a la extensión de las fábricas y maquinarias que las componen, lo que ha traído como consecuencia 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 Acquisition.*

Debido a su magnitud y complejidad, los sistemas SCADA generalmente se desarrollan en módulos. Entre los módulos que componen estos sistemas se encuentra el de interfaz hombre-máquina (HMI), que es el encargado de presentar la información en forma de sinópticos gráficos. El módulo HMI debe contar con una herramienta que permita a los usuarios configurar el sistema para determinado proceso y mostrar esta configuración en un ambiente de ejecución.

Este trabajo tiene como objetivo implementar un modelo que permita mediante el HMI de un SCADA configurar el sistema para procesos particulares. Para alcanzar esta meta se debe crear una estructura que soporte dicha configuración, la que estará compuesta por conceptos fundamentales en el control de procesos tales como: variables, alarmas, dispositivos de control, entre otros.

Palabras clave

Configuración, control, edición, HMI, módulo, SCADA.

Índice de contenido

Índice de figuras

Índice de tablas

INTRODUCCIÓN

Con el desarrollo y crecimiento de las industrias surgió la necesidad de automatizar, monitorear y controlar los procesos a distancia. El desarrollo tecnológico y de la informática han traído soluciones para estas necesidades, desde sistemas que solo permitían monitoreo hasta alcanzar otros que permiten a los operadores ejecutar acciones directas sobre los procesos en cualquier lugar de las instalaciones.

Para los procesos de automatización generalmente se utiliza un SCADA. El termino SCADA usualmente se refiere a un sistema central que monitorea y controla un sitio completo o un sistema que se extiende sobre una gran distancia (kilómetros / millas). La instalación de un sistema SCADA necesita un *hardware* de señal de entrada y salida, sensores y actuadores, controladores, HMI, redes, comunicaciones, base de datos entre otros. (Rodríguez)

Debido a la importancia de los sistemas SCADA, el Polo de Hardware y Automática de la Universidad de las Ciencias Informáticas (UCI) asume la tarea de conformar el proyecto nombrado SCADA. Este proyecto es asumido por estudiantes y profesores de la UCI contando además con el apoyo de especialistas con experiencias en automatización. Los integrantes de este proyecto tendrán la responsabilidad de crear un sistema de este tipo, bajo paradigmas y plataformas de software libre, al cual en el presente documento se le nombra SCADA HA.

Generalmente un sistema SCADA está compuesto por diferentes módulos o subsistemas, algunos de ellos son los de procesamiento de datos, base de datos, comunicación, manejadores y HMI. En el SCADA HA el módulo HMI estará dividido en dos sub-módulos, el ambiente de configuración y el ambiente de ejecución. El ambiente de configuración engloba las utilidades relacionadas con la creación y edición de los elementos que compondrán el sistema y permite la creación de una aplicación para un proceso en particular, mientras que el ambiente de ejecución o *run-time* está encargado de mostrar al operador la aplicación creada.

El sub-módulo de configuración del módulo HMI del SCADA HA debe poseer un modelo que permita configurar el sistema para procesos particulares. Para lograr lo anteriormente dicho, es preciso que este cuente con una estructura que contenga los elementos más comunes en el control de procesos, como son: las variables, las alarmas, los despliegues, los dispositivos de control entre otros; y las funcionalidades que permitan la configuración de esta estructura.

En el contexto expuesto anteriormente surge como **problema científico** la siguiente interrogante: ¿Cómo implementar la estructura y funcionalidades de un modelo que permita la configuración de un sistema SCADA apoyándose en un HMI?

Este trabajo tiene como **objeto de estudio** las formas y métodos para la configuración de los sistemas SCADA, mientras que el **campo de acción** lo constituyen las herramientas de configuración de estos sistemas de control.

Como **Idea a Defender** se define:

Si se implementa la estructura y funcionalidades para un modelo que permita mediante una HMI la configuración de un sistema SCADA, la herramienta de configuración del SCADA HA dispondrá de las funcionalidades necesarias para dar respuesta a las necesidades del sistema.

Para dar solución al problema anteriormente planteado se ha trazado como **objetivo**:

Implementar un modelo que mediante un HMI permita la configuración de un sistema SCADA.

Definiéndose los siguientes **Objetivos Específicos** para lograr un mayor nivel de detalles en la investigación:

1. Implementar la estructura que soportará la configuración del sistema.
2. Implementar las vistas que permitirán al operador organizar, relacionar y configurar los recursos del sistema.

Para cumplir con el objetivo se plantearon las siguientes **Tareas de la Investigación**:

1. Realizar un estudio sobre módulos HMI de sistemas SCADA para obtener una mejor comprensión de los mismos.

2. Realizar una descripción de la estructura que soporta la configuración.
3. Realizar la implementación de los elementos que necesarios para la estructura de la configuración y su gestión.
4. Realizar la implementación de las vistas para la interacción del usuario con el sistema.

El presente documento está constituido por tres capítulos, estructurados de la siguiente manera:

En el **Capítulo 1** se hace una descripción de los sistemas SCADAs, se mencionan características principales, ventajas y arquitectura de estos. Se explican las principales tecnologías a utilizar en el desarrollo de la aplicación.

En el **Capítulo 2** se describe la estructura que soportará la configuración del sistema, se enuncian los conceptos de sus componentes y relación entre estos.

En el **Capítulo 3** se expone la solución propuesta. Se describe la arquitectura de la aplicación y las principales clases que están presentes en la misma.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se hace alusión a las características generales de un SCADA, módulos que lo componen y su estructura. Además se describen tecnologías existentes que permiten la construcción de un modelo con el que se pueda configurar el sistema para aplicaciones específicas en la industria.

1.1 Introducción a los sistemas SCADA

Un SCADA controla usualmente un sitio con gran extensión territorial. En realidad la mayor parte del control del sitio se realiza automáticamente por equipos electrónicos especializados para estos fines. Las funciones de control del servidor están casi siempre restringidas a reajustes básicos del sitio o capacidades de nivel de supervisión. Por ejemplo un Controlador Lógico Programable (PLC) puede controlar el flujo de agua fría a través de un proceso, pero un sistema SCADA puede permitirle a un operador cambiar el punto de consigna (*setpoint*) de control para el flujo, y permitirá grabar y mostrar cualquier condición de alarma como podría ser la pérdida de un flujo o el aumento de la temperatura. La realimentación del lazo de control es cerrada a través de la UTR o el PLC; el sistema SCADA monitorea el desempeño general de dicho lazo. (Rodríguez)

El flujo de la información en los sistemas SCADA es como se describe a continuación: el fenómeno físico lo constituye la variable que se desea medir. Dependiendo del proceso, la naturaleza del fenómeno y el modelo seleccionado para su supervisión estás pueden ser muy diversas: presión, temperatura, flujo de potencia, intensidad de corriente, voltaje, etc. Estos parámetros deben traducirse a una variable que sea inteligible para el sistema SCADA, es decir, en una variable eléctrica. Para ello, se utilizan los sensores o transductores. Los sensores o

transductores convierten las variaciones del fenómeno físico en variaciones proporcionales de una variable eléctrica. Las variables eléctricas más utilizadas son: voltaje, corriente, carga, resistencia o capacitancia. Sin embargo, esta variedad de tipos de señales eléctricas debe ser procesada para ser entendida por el computador digital. Para ello se utilizan acondicionadores de señal, cuya función es la de referenciar estos cambios eléctricos a una misma escala de corriente o voltaje. Además, provee aislamiento eléctrico y filtraje de la señal con el objeto de proteger el sistema de transientes y ruidos originados en el campo. Una vez acondicionada la señal, la misma se convierte en un valor digital equivalente en el bloque de conversión de datos. Generalmente, esta función es llevada a cabo por un circuito de conversión analógico/digital. La computadora (PC) almacena esta información, la cual es utilizada para su análisis y para la toma de decisiones. Simultáneamente, se muestra la información al usuario del sistema, en tiempo real. Basado en la información, el operador puede tomar la decisión de realizar una acción de control sobre el proceso. El operador comanda al computador a realizarla, y de nuevo debe convertirse la información digital a una señal eléctrica. Esta señal eléctrica es procesada por una salida de control, el cual funciona como un acondicionador de señal, la cual la transforma de escala para manejar un dispositivo dado: bobina de un relé, *setpoint* de un controlador, etc.

1.1.1 Principales funcionalidades de un SCADA

El nivel de desarrollo que han alcanzado los dispositivos de control, y los sistemas de supervisión en general es muy amplio y diverso. No constituye objeto de este trabajo abordar en profundidad esta temática. No obstante resulta de interés presentar las principales funcionalidades que se pueden observar en un sistema SCADA. Entre las que se pueden encontrar:

- Adquirir información de campo.
- Procesar información de campo
- Generar alarmas
- Almacenar la información adquirida por largos periodos de tiempo
- Generar informes
- Cambiar estados de los dispositivos de campo.

- Permitir la interacción a través de interfaces graficas.

Partiendo de lo anteriormente dicho se pueden definir un conjunto básico de bloques funcionales que aparecen en un SCADA:

- Comunicación con el Campo
- Procesamiento
- Almacenamiento
- Interacción con el Usuario

1.1.2 Ventajas y características de los SCADAs actuales.

Cuando se habla de un sistema SCADA no hay que olvidar que hay algo más que las pantallas que nos informan como van las cosas en nuestra instalación. Tras estas se encuentras multitud de elementos de regulación y control, sistemas de comunicaciones y múltiples utilidades de software que pretenden que el sistema funcione de forma eficiente y segura. Las ventajas más comunes de los sistemas de control automatizado y supervisado se mencionan a continuación:

- El actual nivel de desarrollo de los paquetes de visualización permite creación de aplicaciones funcionales sin necesidad de ser experto en la materia. (Rodríguez)
- La modularidad de los autómatas permite adaptarlos a las necesidades actuales y reconfigurarlos posteriormente si es necesario. (Acevedo)
- Gracias a las herramientas de diagnostico se consigue una localización más rápida de las fallas. Esto permite minimizar los periodos de paro en las instalaciones y repercute en la reducción de costes de mantenimiento. (Rodríguez)
- Un sistema de control remoto (RTU) pueden definirse de manera que puedan funcionar de forma autónoma, aún sin comunicaciones con la estación maestra.
- Los programas de visualización pueden presentar todo tipo de ayuda al usuario, desde la aparición de una alarma hasta la localización de la causa o la parte de sistema eléctrico implicada en la misma. Esto permite reducir los tiempos de localización de averías al proporcionarse información sobre el origen y las causas de los fallos.

- Los protocolos de seguridad permiten una gestión segura y eficiente de los datos, limitando el acceso no autorizado.
- El nivel de descentralización va en aumento, apostando por los sistemas distribuidos. Esto permite una mayor disponibilidad, pues las funciones de control se pueden repartir y/o duplicar. (Roca)
- Monitorear: representando los datos con severas restricciones de tiempo de modo que permita a los operadores de la planta el conocimiento del estado de los procesos.
- Supervisar: el mando y la adquisición de datos de un proceso y herramientas de gestión para la toma de decisiones. Tiene además la capacidad de ejecutar programas que puedan supervisar y modificar el control establecido y, bajo ciertas condiciones, anular o modificar tareas asociadas a los autómatas. Evita una continua supervisión humana.
- Visualizar: mostrando los estados de las señales del sistema (alarmas y eventos) permite el reconocimiento de eventos excepcionales acaecidos en la planta y su inmediata puesta en conocimiento de los operarios para efectuar las acciones correctoras pertinentes.
- Controlar: que el operador pueda cambiar consignas y otros datos claves del proceso directamente posibilitando desde el ordenador. Se escriben datos sobre los elementos de control.
- Asegurar la data: garantizando la integridad de la información tanto en el envió, la recepción o el almacenamiento de los datos.
- Proteger: estableciendo relaciones de usuarios con recursos y restringiendo el acceso a recursos a usuarios no autorizados, registrando todos los accesos y acciones llevadas a cabo por cualquier operador.

1.1.3 Módulos del SCADA HA

- **Manejadores**

  Aseguran mediante una interfaz genérica la comunicación del sistema de supervisión y control con los distintos dispositivos que existen en el campo, ya sean autómatas, PLC, sensores inteligentes etc.

- **Núcleo de Procesamiento de Datos**

  Representa el núcleo principal del procesamiento de los datos, es el encargado del procesamiento y análisis de la información recogida del campo a través de los manejadores. Una vez procesada esta información, es enviada a cualquier módulo que la requiera según las reglas del negocio.

- **Base de Datos Históricos**

  Aquí se implementa el mecanismo encargado del almacenamiento de la información recibida desde el campo, así como la sucesión de alarmas y eventos generados. La información almacenada es utilizada por varias aplicaciones del sistema.

- **Configuración**

  Es el módulo encargado de persistir y suministrar la información base para el funcionamiento de los demás módulos del SCADA HA. Cada módulo (excepto Manejadores) posee una interfaz de comunicación con el servidor de configuración, esta comunicación se establece a través de la capa de conectividad, permitiendo crear, modificar y eliminar los recursos configurables del sistema.

- **Middleware**

  Este módulo representa la capa de software encargada de la comunicación entre los diferentes procesos distribuidos de mediano y alto nivel.

- **Seguridad**

  Provee las interfaces necesarias para que los usuarios puedan autenticarse en el sistema, y de esta forma cada uno solo pueda acceder a los recursos que tiene asignado su rol. Además posee herramientas para la protección ante ataques piratas, fallos eléctricos, problemas de red, servidores, entre otros.

- **Reportes**

En los SCADA existe la necesidad de contar con información sobre el comportamiento de variables en los distintos procesos a través del tiempo, el módulo Reportes es el encargado de brindar este tipo de información al usuario a través de reportes especializados, los cuales se diseñan de acuerdo a la necesidad de los usuarios gracias a un editor que posee el mismo.

- **Ambiente de Configuración**

Esta aplicación permite configurar varios procesos o partes de ellos, aquí se definen y gestionan las variables, la configuración de los manejadores, los comandos, las alarmas y variadas opciones adicionales. Este ambiente funciona como una aplicación de diseño tradicional, con la peculiaridad que los sinópticos se confeccionan a partir de objetos y primitivas básicas predefinidas, que se pueden agrupar, combinar, transformar, importar y exportar entre otras.

- **Ambiente de Ejecución (*Run-time*)**

Aplicación encargada de la visualización, la cual permite representar gráficamente de manera fidedigna los procesos operacionales con los que el usuario interactúa. A través de esta aplicación se pueden visualizar las condiciones operacionales, valores de variables, navegar entre despliegues, visualizar alarmas, enviar comandos, entre otros.

1.1.4 El hardware en los SCADA

A escala conceptual un sistema SCADA está dividido en dos grandes grupos (Rodríguez):

- Captadores de datos: recopilan los datos de los elementos de control del sistema (por ejemplo, autómatas, reguladores, registradores) y los procesan para su utilización. Son los servidores del sistema.
- Utilizadores de datos: los que utilizan la información recogida por los anteriores, como pueden ser las herramientas de análisis de datos de los operadores del sistema. Estos son los clientes.

Entre los componentes *hardware* de un sistema SCADA más destacados podemos encontrar:

- Estación cliente.
- Unidad central.
- Unidad remota.
- Sistema de comunicaciones.

Estación cliente

Generalmente las estaciones cliente son computadoras aunque pueden ser cualquier elemento que pueda visualizar información del proceso. En estos encontramos los sinópticos de control y los sistemas de presentación gráfica.

Unidad central (MTU, *Master Terminal Unit*)

Centraliza el mando del sistema. Se hace uso extensivo de protocolos abiertos, lo cual permite la interoperabilidad de multiplataformas y multisistemas. En el centro de control se realiza, principalmente, la tarea de recopilación y archivado de datos. Toda esta información que se genera en el proceso productivo se pone a disposición de los diversos usuarios que puedan requerirla. Se encarga de: gestionar las comunicaciones, recopilar los datos de todas las estaciones remotas, envío de información, análisis, mando, seguridad entre otros. (Villalobos)

Unidad Remota (RTU, *Remote Terminal Unit*)

Se define como un dispositivo basado en microprocesadores, el cual permite obtener señales independientes de los procesos y enviar la información. Dentro de estos dispositivos podemos encontrar los PLC, que son muy usados en automatización industrial. Hoy en día, los PLC no sólo controlan la lógica de funcionamiento de máquinas, plantas y procesos industriales, sino que también pueden realizar operaciones aritméticas, manejar señales analógicas para realizar estrategias de control. (Villalobos)

Sistema de comunicaciones

Gracias a los controladores suministrados por los diferentes fabricantes y su compatibilidad con la mayoría de los estándares de comunicación existentes, es posible establecer cualquier tipo de comunicación entre un servidor de datos y cualquier elemento de campo.

Un servidor de datos puede gestionar varios protocolos de forma simultánea, estando limitado por su capacidad física de soportar las interfaces de hardware (tarjetas de comunicación). Estas permiten el intercambio de datos bidireccional entre la unidad central y las unidades remotas (RTU) mediante un protocolo de comunicación determinado y un sistema de transporte de la información para mantener el enlace entre los diferentes elementos de la red, como pueden ser líneas telefónicas, cable coaxial, fibra óptica, entre otros. (Rodríguez)

1.2 Tecnologías

1.2.1 Software Libre. GNU/Linux

Software libre es la denominación del software que brinda libertad los usuarios sobre su producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software; de modo más preciso, se refiere a cuatro libertades de los usuarios del software: la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del programa, y adaptarlo a las necesidades; de distribuir copias, con lo que puede ayudar a otros; de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie (para la segunda y última libertad mencionadas, el acceso al código fuente es un requisito previo).

Una **distribución de Linux** es una variante de ese sistema operativo (SO) que incorpora determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones hogareñas, empresariales y para servidores. Pueden ser exclusivamente de software libre, o también incorporar aplicaciones o controladores propietarios.

Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU (acrónimo que significa **GNU No es Unix**); de ahí el nombre: GNU/Linux. (LINUX)

1.2.1.1 Distribución de Linux: Debian GNU/Linux

Debian es un SO libre, utiliza el núcleo Linux (el corazón del sistema operativo), pero la mayor parte de las herramientas básicas vienen del Proyecto GNU. También es una comunidad conformada por desarrolladores y usuarios. Debian nace como una apuesta por separar en sus versiones el software libre del software no libre. El modelo de desarrollo del proyecto es ajeno a motivos empresariales o comerciales, siendo llevado adelante por los propios usuarios, aunque cuenta con el apoyo de varias empresas en forma de infraestructuras. Debian no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuir comercialmente este software mientras se respete su licencia.
Debian es la única distribución importante de GNU/Linux mantenida solamente por voluntarios, lo que permite la actualización de los paquetes de software y la participación abierta de todos aquellos que deseen colaborar. (DEBIAN)

1.2.2 GTK+

GTK+ es un grupo importante de bibliotecas o rutinas para desarrollar interfaces gráficas de usuario para principalmente los entornos gráficos GNOME, XFCE Y ROX de sistemas Linux. GTK+ es la abreviatura de *GIMP toolkit* (conjunto de rutinas para GIMP). Es software libre (bajo la licencia LGPL), multiplataforma y parte importante del proyecto GNU. Actualmente es muy usada por muchos programas en los sistemas GNU/Linux. GTK+ se ha diseñado para permitir programar con lenguajes como C, C++ (gtkmm), C#, Java, Perl, PHP o Python.
GTK+ se basa en varias bibliotecas del equipo de GTK+ y de GNOME (GTK):

- **GTK.** Biblioteca la cual realmente contiene los objetos y funciones para crear la interfaz de usuario. Maneja *widgets* como ventanas, botones, menús, etiquetas, deslizadores, pestañas, etc.

- **GDK.** Biblioteca que actúa como intermediario entre gráficos de bajo nivel y gráficos de alto nivel.
- **ATK**. Biblioteca para crear interfaces con características de una gran accesibilidad muy importante para personas discapacitadas o minusválidos. Pueden usarse utilerías como lupas de aumento, lectores de pantalla, o entradas de datos alternativas al clásico teclado y mouse.
- **Pango**. Biblioteca para el diseño y renderizado de texto, hace hincapié especialmente en la internacionalización. Es el núcleo para manejar las fuentes y el texto de GTK+2.
- **Cairo**. Biblioteca de renderizado avanzado de controles de aplicación.
- **GLib**. Biblioteca de bajo nivel estructura básica de GTK+ y GNOME. Proporciona manejo de estructura de datos para C, portabilidad, interfaces para funcionalidades de tiempo de ejecución como ciclos, hilos, carga dinámica o un sistema de objetos.

1.2.3 IDE Eclipse

Eclipse es un Entorno Integrado de Desarrollo (IDE) de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Con este IDE se han desarrollado importantes aplicaciones como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent Azureus.

Emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente a permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Pyton, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistemas de gestión de base de datos. (ECLIPSE)

CAPÍTULO 2: ESTRUCTURA PARA LA CONFIGURACIÓN DEL SISTEMA.

Introducción

Un paquete SCADA, en su vertiente de herramienta de HMI, comprende todo un conjunto de funciones y utilidades encaminadas a establecer una comunicación lo más clara posible entre el proceso y el operador, lo que permite a un usuario con conocimientos básicos de control configurar al sistema para un determinado proceso o instalación. En este capítulo se describe la estructura que utiliza el sistema SCADA HA, la cual comprende los conceptos más utilizados en el control de procesos.

2.1 Configuración de sistemas SCADA

Generalmente los sistemas SCADA cuentan con funcionalidades para realizar la configuración, de modo que los operadotes puedan definir el entorno de trabajo para adaptarlo a las necesidades de la aplicación. El sistema en que se enmarcará la estructura que se desarrolló cuenta con un módulo de configuración dedicado a estos fines. Entre las prestaciones que se pueden encontrar en dicho módulo se tienen:

- Creación, modificación y organización de los recursos necesarios para la configuración de la aplicación definida.
- Gestionar los módulos del SCADA, permitiendo definir la ubicación de estos en los nodos físicos que se desee.
- Administración de usuarios que accederán al sistema, permitiendo la clasificación de estos según su importancia.
- Creación de grupos con privilegios que permiten o limitan el acceso y la acción de los usuarios sobre los recursos configurados en el sistema.

- Configuración de las comunicaciones, dando la facilidad para especificar los dispositivos que se utilizaran para recolectar los valores a monitorizar, además de los parámetros que estos necesitarán para su correcto funcionamiento.
- Creación de pantallas con imágenes y texto que simulen el proceso a controlar, brindando a los usuarios una mejor comprensión de la aplicación creada.

2.1.1 Árbol de configuración

El árbol de configuración es la estructura en la cual se soporta la configuración del sistema SCADA HA (Ver Figura 1), sus nodos son los recursos y colecciones de estos, los cuales se modificaran y adaptaran según necesidad de la implementación que se desee. A continuación se mencionan los recursos presentes en el sistema:

1. Proyecto.
2. Nodo.
3. Módulo.
4. Punto.
5. Alarma.
6. Canal.
7. Sub-canal.
8. Dispositivo.
9. Despliegue.
10. Menú.
11. Reporte.
12. Teclas Funcionales.
13. Usuario.
14. Grupo Operacional de privilegio.
15. Perfil.
16. Grupo de transferencia a histórico.
17. Zona de alarma.

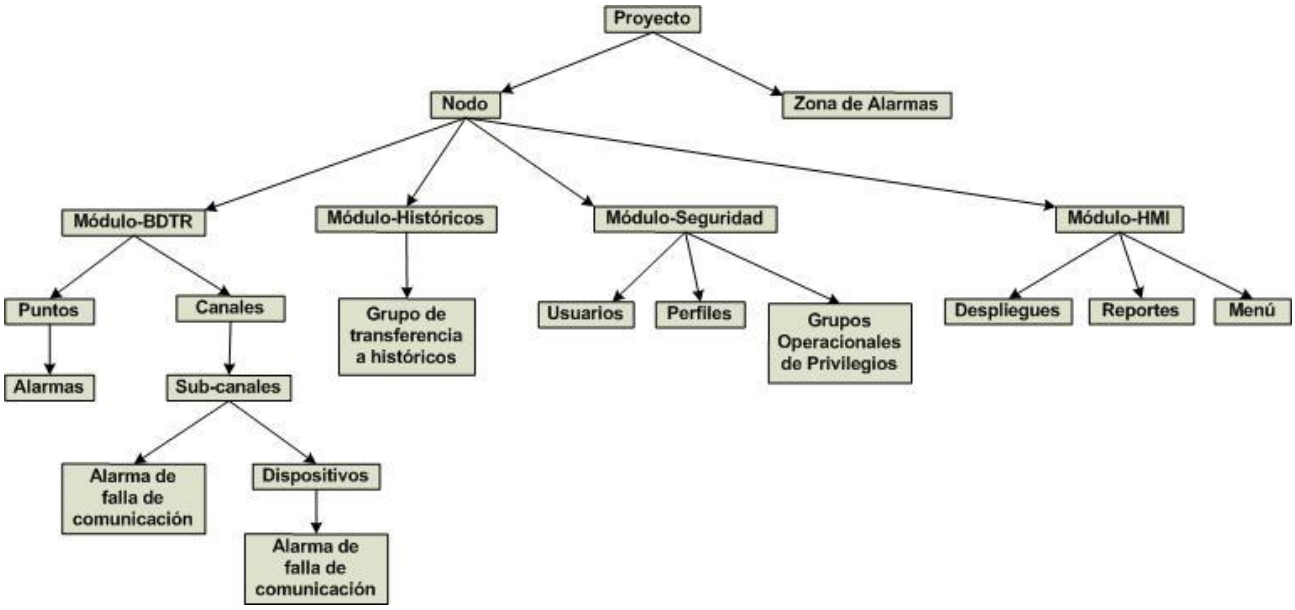


Figura 1 Estructura del árbol de proyecto.

Proyecto.

Es la base de la configuración, representa el sistema o proceso en general a configurar. Se le pueden agregar varios nodos.

Nodo.

Se define como la unidad física donde se instancia uno o varios módulos del sistema. En este sistema se pueden agregar tantos como se desee o también se puede tener solamente uno donde se configuren todos los módulos del sistema. Por ejemplo, si tenemos dos computadoras desde las cuales se deberá monitorizar el proceso, al configurar nuestro sistema, en una de ellas podríamos instanciar el módulo de seguridad y el de bases de datos histórica, y en la otra se pondrían el módulo de base de datos en tiempo real y el módulo de HMI, como se ejemplifica en la siguiente figura:

Figura 2 Ejemplo de nodos en el sistema.

Módulos.

El sistema se configura utilizando cuatro tipos de módulos (módulo HMI, módulo de base de datos en tiempo real (BDTR), módulo de seguridad y módulo de base de datos histórica). En el módulo HMI se configuran los despliegues, menú y reportes, estos serán las principales interfaces de información e interacción de los operadores con el sistema en ejecución. En el módulo BDTR se encuentran los puntos y se define la configuración de las comunicaciones (canales, sub-canales, dispositivos). En el módulo de seguridad se definen los usuarios, perfiles, y grupos operacionales de privilegios. Permitiendo así asignar permisos a los usuarios sobre los diferentes recursos del sistema. En el Módulo de base de datos histórica se configuran los grupos de transferencia a históricos, en está base de datos se guardaran todos los eventos y acciones del sistema para su posterior consulta.

Puntos.

Representan las variables a monitorizar y controlar en el proceso. Los valores de estas se recolectan en el campo y se muestran a los operadores mediante la interfaz a la cual están asociadas. Pueden ser de diferentes tipos, este sistema solo abarcará las analógicas y digitales. Los analógicos se utilizan para medir valores de unidades de ingeniería como pueden ser presión o temperatura, mientras que los digitales representan valores binarios, por ejemplo para monitorizar el estado de un interruptor (abierto o cerrado, encendido o apagado).

Canales.

Representa el medio físico a través del cual se conectan un conjunto de dispositivos, el principal parámetro que se le configura a este recurso es el modo de acceso. La interacción con este medio físico en el sistema es independiente de la interacción con el resto de los dispositivos del SCADA.

Subcanales.

A un canal se le pueden agregar varios sub-canales, estos representan el protocolo de comunicación médiate el cual se conectará uno o varios dispositivos. Nuestro sistema contará en un principio con manejadores para los siguientes protocolos: EthernetIP, ModbusTCP, OPC, ABEthernet, ModbusASCII y ModbusRTU.

Dispositivos.

Los dispositivos son equipos electrónicos que pueden ser autómatas, PLC, reguladores autónomos, sensores inteligentes, controladores etc. Estos son el primer eslabón en la adquisición de los datos. A estos se le asocian variables previamente configuradas, cuyos valores se obtendrán mediante estos equipos.

Alarmas.

Las alarmas se basan en la vigilancia de los parámetros de las variables del sistema. Se evidencian en los sucesos no deseables, porque su aparición puede dar lugar a problemas de funcionamiento. Estas requieren de la atención de un operario para su solución antes de que llegue a una situación crítica que afecte el proceso. A las alarmas en general que se configuran en este sistema como en la mayoría de estos, se les asigna una severidad la cual podrá ser Alta, Media o Baja, este parámetro define la criticidad de la alarma, la de severidad Alta representa una alarma crítica que requiere la acción inmediata del operador. Otro parámetro asignado a las alarmas es la prioridad, de modo que si aparecen varias de forma simultánea, las de mayor prioridad aparecerán primero. Ya que las alarmas están relacionadas con una variable en particular se clasificarán según las características y clasificaciones de estas.

Alarmas digitales.

Dentro de las alarmas digitales las cuales solo se configuran para variables de este tipo encontramos las de estado, a estas se las define el estado de la variable con el cual se desee disparar la alarma, de modo que se tendría notificación cuando la variable pase a este estado.
También encontramos las alarmas de cambio de estado no comandado, las cuales se configuran para monitorizar cuando las variables cambian de estado sin que sea producto de la ejecución de un comando desde el sistema.
En este grupo también se encuentran las alarmas de falla de ejecución de comando El sistema luego del envío de comando para un punto, procede a la espera de un tiempo configurado en el punto para la ejecución del comando, una vez cumplido este tiempo de espera el sistema verifica si el estado de entrada del punto concuerda con el enviado en el comando, de lo contrario procede a la activación de la alarma según los parámetros configurados en el punto.

Alarmas analógicas.

En las alarmas que se crean para verificar el comportamiento de las variables analógicas encontramos a las alarmas de nivel. Por ejemplo, si se tiene una variable que represente un valor de temperatura, esta arrastraría valores que podrían clasificarse en: alto-alto(HH), alto(H), bajo(L) y bajo-bajo(LL), estos son los cuatro niveles que se pueden activar, configurando una alarma para cada uno de ellos y fijando el valor que representan. También se les fija un valor de banda muerta para evitar que pequeñas fluctuaciones entre alguno de los límites generen una gran cantidad de alarmas.

Verificación de los niveles alto-alto, alto, bajo, bajo-bajo:

Para niveles alto o alto-alto se hace una comparación “mayor”.
Para niveles bajo o bajo-bajo se hace una comparación “menor”.

Lógica de activación y desactivación de las alarmas de nivel:

Caso de histéresis negativa alarmas alto y alto-alto:

Si Valor actual > Punto de ajuste entonces Activar Alarma
Sino
Si Valor actual < (Punto de ajuste –Banda muerta) entonces Desactivar alarmas

Caso de histéresis positiva alarmas bajo y bajo-bajo:

Si Valor actual < Punto de ajuste Entonces Activar Alarma
Si Valor actual > (Punto de ajuste + Banda muerta) Entonces Desactivar alarmas

A continuación se muestra el gráfico propuesto para la activación y desactivación de las alarmas de nivel considerando el concepto de histéresis. Positiva para los niveles bajo y bajo- bajo e histéresis negativa para los niveles alto y alto-alto.

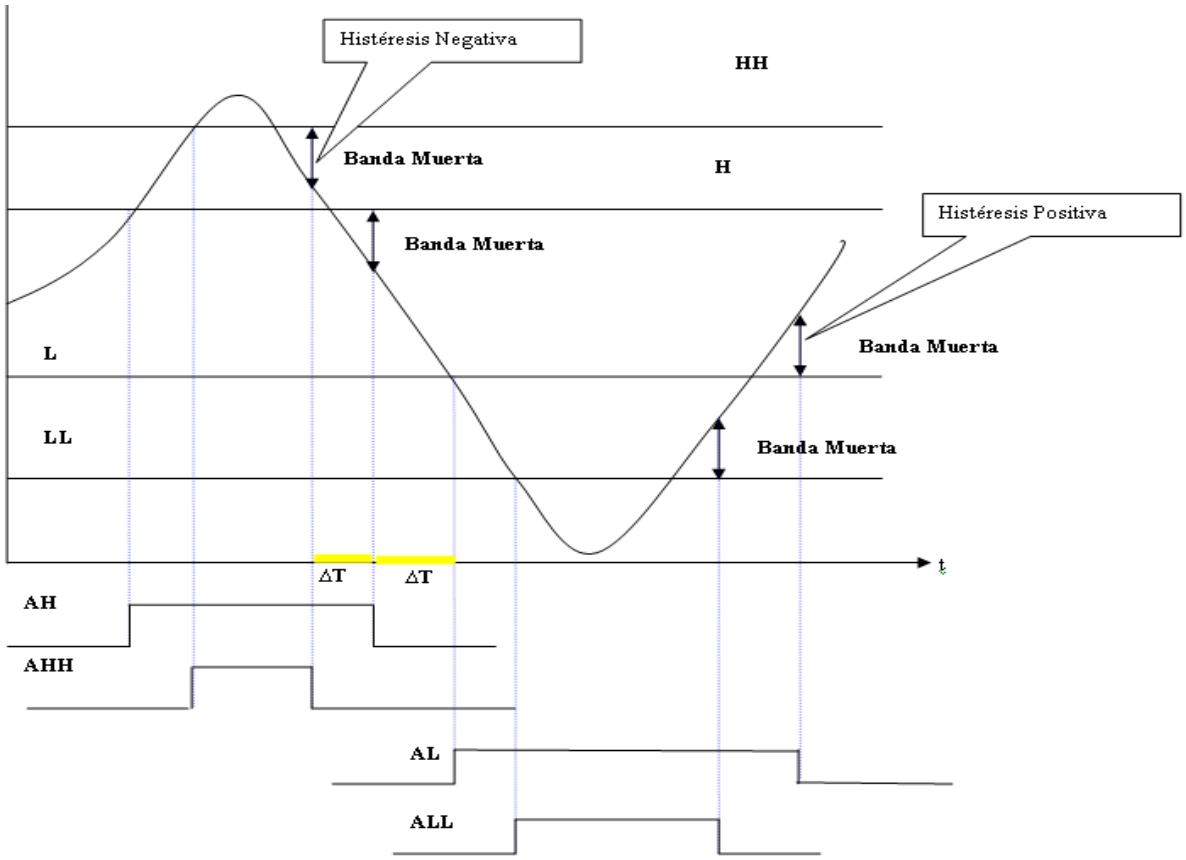


Figura 3 Ejemplo de activación y desactivación de alarmas de nivel.

Las alarmas de tasa de cambio se utilizan para reconocer cuando el valor de una variable cambia de forma brusca. Poniendo el mismo ejemplo de una variable que represente temperatura, si el valor de esta cambia de 0 a 100 ºC en un lapso muy corto de tiempo esto puede ser evidencia de

un mal funcionamiento. A esta se le fija el valor de cambio permitido en las unidades que se está midiendo o también se le puede fijar en un valor porcentual del rango permitido por esta.

Ecuación para la verificación de la tasa de cambio por valor:

$$TC = \frac{|x - x_0|}{\Delta t}$$

X=Valor Anterior, X0=Valor Actual.

Ecuación para la verificación de TC por porcentaje:

$$TC(\%) = \frac{|x - x_0|}{\Delta t} \cdot 100\%$$

En la Figura 4 se muestra como la alarma se activa en caso de que este valor sea mayor que el fijado en la configuración y se desactiva en caso de ser menor.

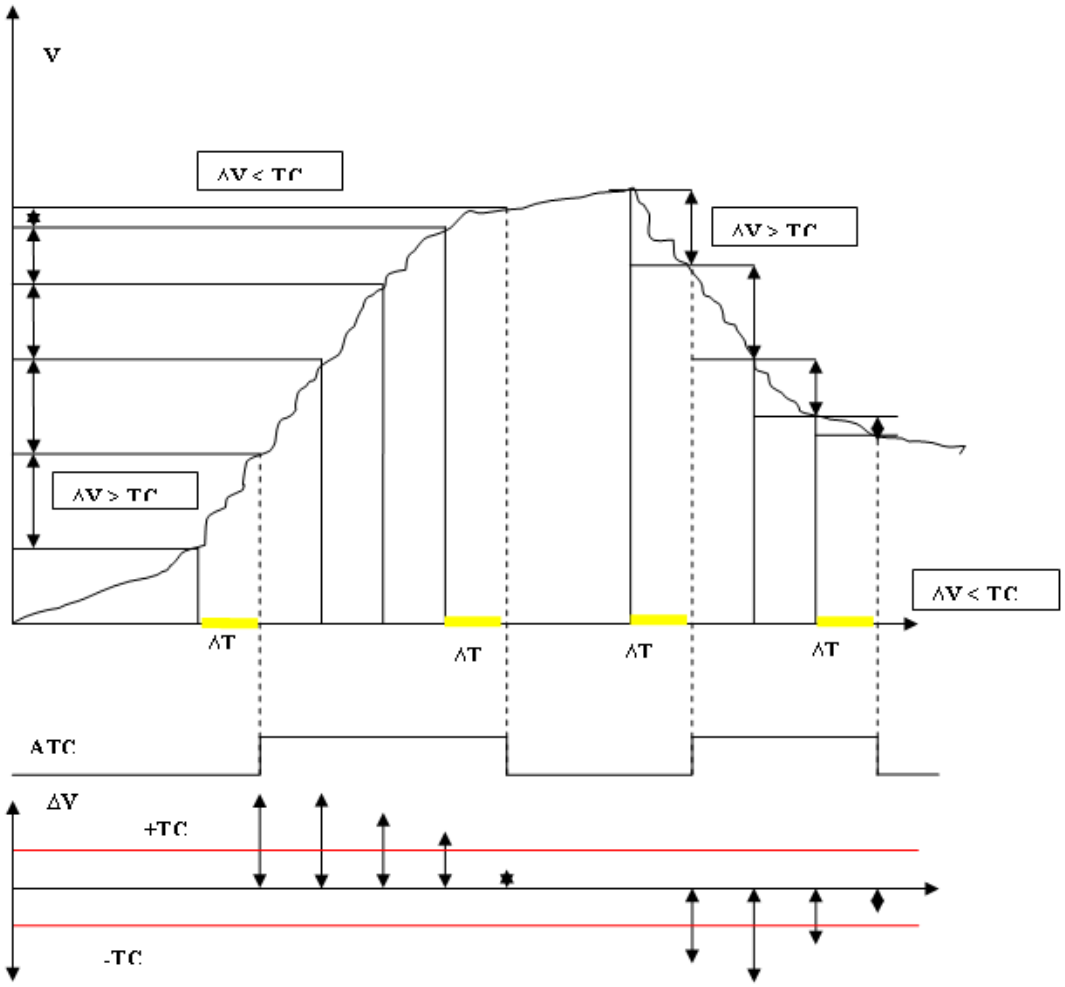


Figura 4 Ejemplo de activación y desactivación de alarmas de tasa de cambio.

A las alarmas de falla de instrumento que se creen solo se le asignan los valores generales, severidad y prioridad, ya que el criterio de activación de esta es la calidad del dato que se recibe, acción que se ejecuta en otro módulo teniendo este las condiciones necesarias para la activación de la misma.

Las señales involucradas en el control del proceso, en muchas necesitan mantenerse dentro de rango de operación que puede depender de otras variables ó funciones lógicas. La alarma de desviación se configura especificándole una expresión de comparación que pede ser operaciones aritméticas donde estén involucradas otras variables, y el valor de umbral (que también puede ser una expresión), esta alarma se activa cuando la señal se compara con el resultado de la expresión y esta supera el umbral de según la siguiente ecuación:

Desviación = |Valor actual - Valor de Expresión|

Si Desviación > Umbral, la Alarma se activa.

La alarma de no variación en el tiempo para alertar en caso de que un valor analógico no ha cambiado después de un número de barridos de lectura que se le especifican en la configuración. Esta alarma indica que existe la posibilidad que el sensor analógico esté “muerto”. Cuando en alguno de las subsecuentes lecturas hay un cambio en el valor, el sistema desactivará la alarma.

Alarma de falla de comunicación.

En el sistema también podemos encontrar alarmas de falla de comunicación. Estas se le configuran a los sub-canales y los dispositivos. Se le especifica un número máximo de reintentos antes de que se active la alarma.

Despliegues.

La herramienta para la configuración del sistema también permite crear pantallas con múltiples combinaciones de imágenes y texto las cuales representarán gráficamente las funcionalidades del proceso a controlar. Los objetos gráficos que se colocaran en estos despliegues brindan la

posibilidad de asociarles las variables del sistema que van a representar. Por ejemplo es posible visualizar una variable analógica mediante un objeto en forma de barra el cual se mostraría según el valor de esta variable.

Menús.

De igual forma se poden definir los menús que se mostrará el sistema en ejecución. Configurando su estructura y definiendo acciones para las etiquetas según se desee. Esto permitirá tener u acceso rápido a acciones tales como navegar entre despliegues, mostrar sumario de alarmas entre otras.

Reportes.

En la monitorización y el control de procesos por lo general se necesitan informes que permitan a los operadores tener estadísticas de los procesos y su variación en un tiempo determinado. Algunos sistemas utilizan generadores de reportes externos a ellos. En este sistema este generador está integrado en su interfaz visual. En la configuración del sistema se puede diseñar las plantillas a partir de las cuales se puedan generar nuevos reportes, estos permite personalizar los diseños incorporando conceptos de formato como los encabezados, pie y número de página. Poseen funcionalidades para la sumarización de los datos que permiten el cálculo de varianzas, medias y totalización así como componentes para la visualización de gráficos de barra, líneas y pastel.

Teclas Funcionales.

La herramienta de configuración también brinda la facilidad de crear accesos directo a acciones mediante combinación de teclas. Estas se permiten crear en configuración como un recurso mas del sistema, pudiéndole agregar una acción previamente definida. Son de gran comodidad al usuario que en ejecución solo debe presionar esta combinación para ejecutar una acción determinada.

Usuarios.

El sistema brinda la facilidad de definir usuarios que harán uso de la aplicación, a estos se le asignará un perfil.

Perfiles.

El perfil definirá las características que pueden tener uno o varios usuarios. Por ejemplo los días que ese usuario trabajará y las horas que tendrá por sesión de trabajo. A los perfiles se le asocia un grupo operacional de privilegios.

Grupos Operacionales de privilegios.

Este recurso permitirá agrupar usuarios y recursos permitiendo el acceso a estos según los permisos que se definen en un perfil al asociarle un grupo, estos permisos pueden ser lectura, escritura y configuración.

Grupos de transferencia a históricos.

Estos grupos definen la forma de almacenamiento de las variables en la base de datos histórica del sistema, se le configura el tipo de almacenamientos y tipo de grupo. Se le asocian las variables que almacenaran según los parámetros configurados en este.

Zona de alarmas.

En las zonas de alarmas de agrupan las variables que presenta una alta probabilidad de comportamiento correlacionado a un proceso, por lo que su captura puede converger a una relación causa efecto. A estas variables se les puede activar la avalancha de alarmas que es un conjunto de eventos que ocurren en modo “Secuencial o simultáneo”.

El objetivo de todo sistema que procure la gestión de avalancha de alarmas es minimizar la cantidad de variables que llamen la atención al operador y ayudarlo a la toma de decisiones oportunas, en resumen la gestión de avalancha de alarma debe procurar focalizar al operador en la atención de la falla procurando brindarle herramientas para que determine la causalidad de la falla.

CAPÍTULO 3: SOLUCIÓN PROPUESTA

Introducción

En este capítulo se introducen las características principales que corresponde a la implementación del módulo Ambiente de Edición del SCADA. En el primer epígrafe se muestra la arquitectura. Además se mostrará la implementación de la estructura para la configuración del sistema y de las vistas que permitirán la interacción del usuario con esta estructura.

3.1 Arquitectura del sub-módulo Ambiente de Edición

La arquitectura que se ha propuesto persigue que el producto pueda dar respuesta a los requisitos de portabilidad, comprensibilidad, extensibilidad y eficiencia de forma satisfactoria. Para estos fines se propone una solución orientada al patrón arquitectónico Modelo-Presentación el cual es una variante del patrón Modelo-Vista-Controlador (MVC).

El patrón Modelo-Presentación se basa en separar las responsabilidades dentro de la aplicación en dos dominios fundamentales. La lógica relacionada a la interacción con el usuario se inscribe en lo que se denomina Presentación, y la lógica perteneciente al negocio en el Modelo.

3.1.1 Dominio Modelo

En la arquitectura del módulo de Ambiente de Edición se identifican dentro del modelo un conjunto de clases que se corresponden con los conceptos presentados en el Capítulo 2. Debido fundamentalmente a que estos conceptos se asocian a estados de la configuración del SCADA y son a su vez independientes de cómo son representados al usuario. También se incluyen otro conjunto de clases las cuales tienen como finalidad la transformación de los estados de las

entidades. Para ayudar a la comprensión del lector se profundiza en más elementos en el epígrafe 3.1.3.

3.1.2 Dominio Presentación

En la capa de Presentación se implementan la interfaz de usuario y las clases que la gestionan. Estas se encargan de enlazar los eventos de la interfaz con manejadores ó *callbacks* que a su vez invocarán las funcionalidades en el modelo para su modificación y encuesta.

3.1.3 Funcionamiento del módulo de Ambiente de Edición. Relación entre el dominio modelo y el dominio presentación

El principal objetivo del Ambiente de Edición es brindar al operador un medio para la creación de forma relativamente sencilla de una aplicación específica para un proceso o una planta que se desee controlar. Para lograr lo anteriormente dicho se han implementado un conjunto de funcionalidades para la correcta realización de las operaciones para la configuración que debe poseer el módulo.

Las clases que se implementaron se agrupan en dos paquetes, el paquete *Base* y el paquete *Views*.

En el paquete *Base* se enmarcarán las clases del Modelo. Entre estas clases podemos encontrar clases de tipo controladoras y de tipo entidad. Entre las clases controladoras se pueden mencionar las siguientes:

- ModelManager
- ProjectManeger
- Factory
- Command
- CreateProject (existe una clase como esta para cada una de los recursos del sistema exceptuando las alarmas).
- ShowProperties

- ShowMultiView
- UpdatePointListView

Como se mencionó anteriormente varias clases del Modelo corresponden con los recursos descritos en el Capítulo 2, estas clases son las clases de tipo entidad. A continuación se mencionan estas clases en relación con el recurso que corresponde (Ver Anexo 1 y Anexo 2):

- Project corresponde a *Proyecto*.
- Node corresponde a *Nodo Físico*.
- Module corresponde a *Módulo*.
- Point corresponde a *Punto*.
- Channel corresponde a *Canal*.
- SubChannel corresponde a *Sub-canal*.
- Device corresponde a *Dispositivo de Control*.
- Alarm corresponde a *Alarma*.
- TreeNodeUser corresponde a *Usuario*.
- TreeNodeProfile corresponde a *Perfil*.
- Group corresponde a *Grupo Operacional de Privilegios*.
- Historical corresponde a *Grupo de Transferencia a Históricos*.
- AlarmZone corresponde a *Zona de Alarmas*.
- TreeNodeScreen corresponde a *Despliegue*.
- TreeNodeReport corresponde a *Reporte*.
- TreeNodeMenu corresponde a *Menú*.
- TreeNode esta clase generaliza a los recursos que se visualizan en el árbol.

En el paquete *Views* se enmarcarán las clases de la Presentación, las cuales representan las vistas de interacción con el usuario. Entre estas clases tenemos las siguientes (Ver Anexo 3):

- ViewManager
- MultiView
- ScreeEditorView
- TreeViewPointList
- TreeView

- Window
- PopupMenu
- PropertiesInspector
- ToolbarView
- ToolboxView

Para que el módulo brinde al usuario las herramientas necesarias para la configuración se ha implementado un conjunto de funcionalidades que permiten al usuario ejecutar las acciones necesarias para la creación y modificación de los recursos del sistema, así como para configurar su organización. Existen un conjunto de operaciones comunes que se pueden ejecutar sobre los recursos, estas son: agregar, modificar y eliminar. Estas operaciones se agrupan en el término gestionar. Entonces las funcionalidades más destacadas serían:

- Gestionar un proyecto.
- Gestionar un nodo.
- Gestionar un módulo.
- Gestionar un punto.
- Gestionar un canal.
- Gestionar un sub-canal.
- Gestionar un dispositivo.
- Gestionar un usuario.
- Gestionar un perfil.
- Gestionar un grupo.
- Gestionar un despliegue.
- Gestionar un reporte.
- Gestionar un menú.
- Gestionar un grupo de transferencia a histórico.
- Gestionar una zona de alarmas.
- Mostrar el árbol de configuración del sistema.
- Mostrar las variables del sistema.

Las operaciones realizadas por la aplicación usan un mecanismo similar para ejecución, a continuación se muestra una figura que ejemplifica la interacción entre la Presentación y el Modelo.

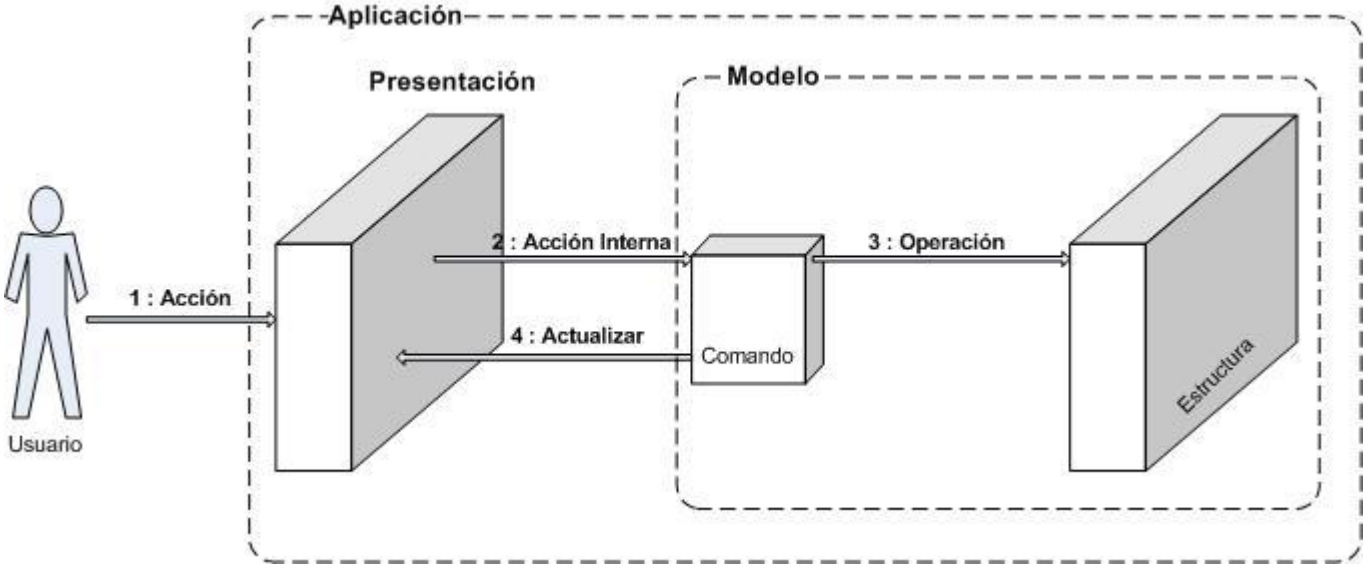


Figura 5 Interacción entre el Modelo y la Presentación.

Seguidamente se exponen algunos ejemplos prácticos que contribuyen a comprender la interacción entre Modelo y la Presentación:

- **Crear un nuevo proyecto**.

Para crear un proyecto la aplicación le brinda al usuario la interfaz para realizar su petición, está interfaz es un menú emergente (Ver Figura 7) que se despliega al hacer clic derecho en el árbol de proyecto. Al hacer clic en el etiqueta “Agregar proyecto” las funcionalidades de la biblioteca gráfica se encargarán de gestionar los eventos sobre ella, eventos que se enlazarán en la presentación con los respectivos manejadores, los manejadores se encargarán de ejecutar una función de una clase controladora del modelo (generalmente una clase de tipo comando) la cual será la encargada de la modificación y encuesta al modelo, en el caso de crear proyecto esta clase controladora sería *CreateProject*, que luego de haber creado el proyecto llamaría a otra controladora (en este caso la clase *ShowProperties*) encargada de mostrar las vista (clase

PropertiesInspector) con los parámetros por defecto del proyecto creado (Ver Figura 8), permitiendo al usuario modificar los que desee. Luego la clase controladora *ShowProperties* se encarga de mandar a actualizar la vista del árbol de configuración (*TreeView*) para mostrar los cambios (Ver Figura 9). A continuación se muestra un diagrama de secuencia del caso correspondiente al ejemplo mencionado.

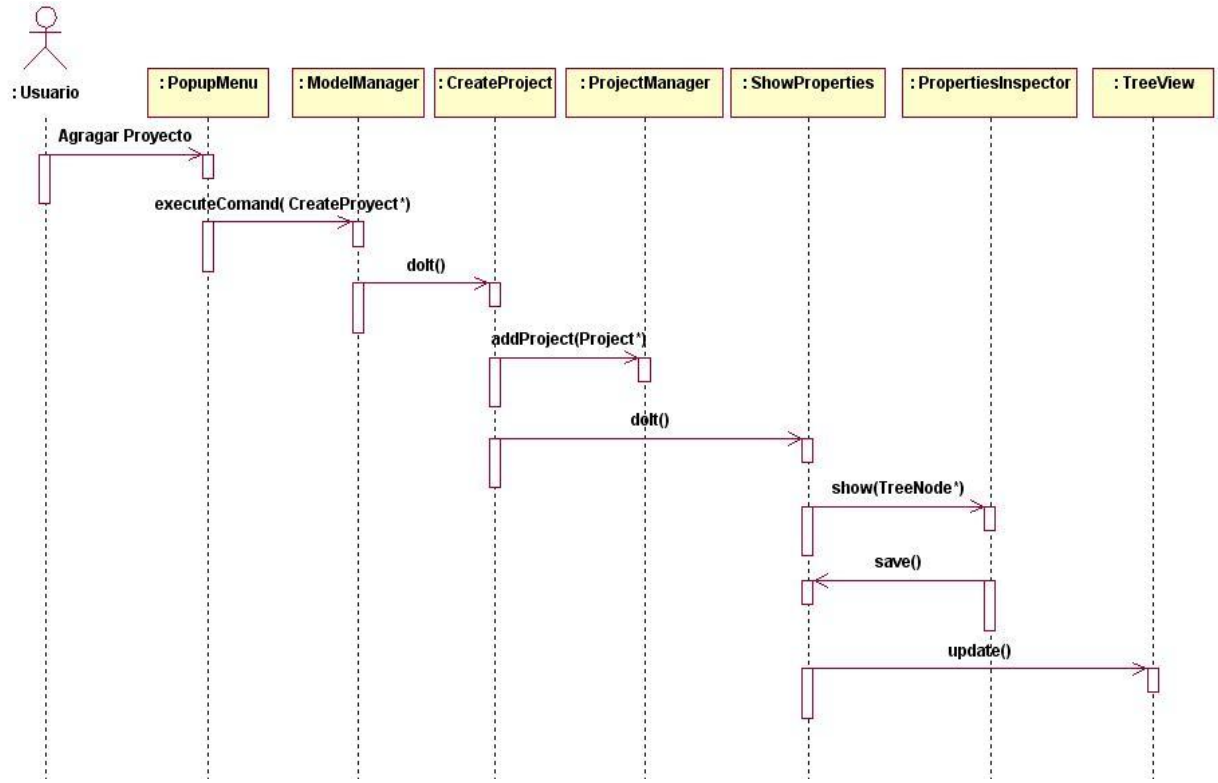


Figura 6 Diagrama de Secuencia “Agregar Proyecto”.

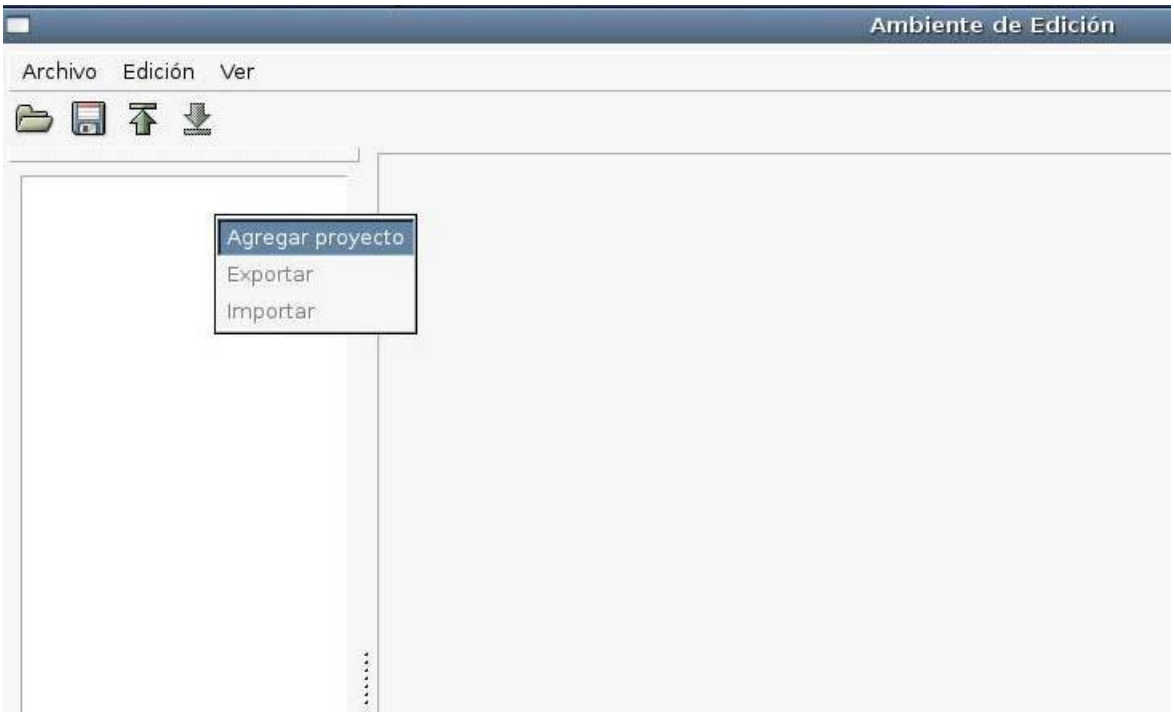


Figura 7 Menú emergente al hacer clic derecho en la vista de árbol de proyecto.

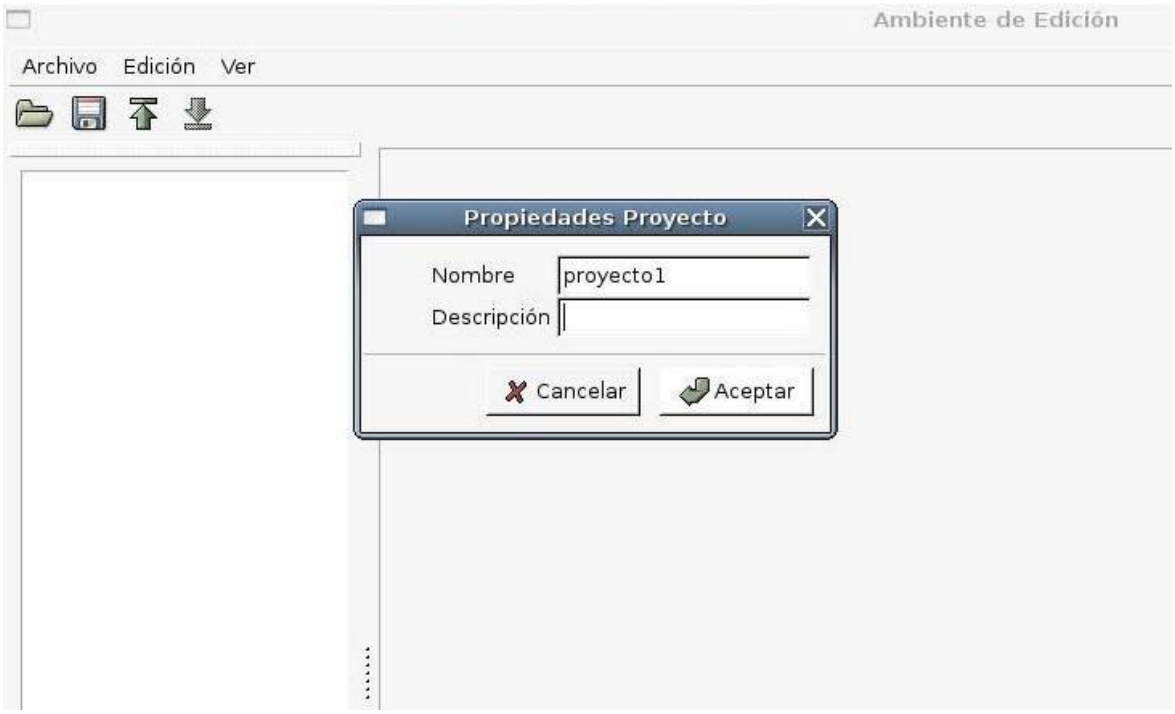


Figura 8 Inspector de propiedades de proyecto.

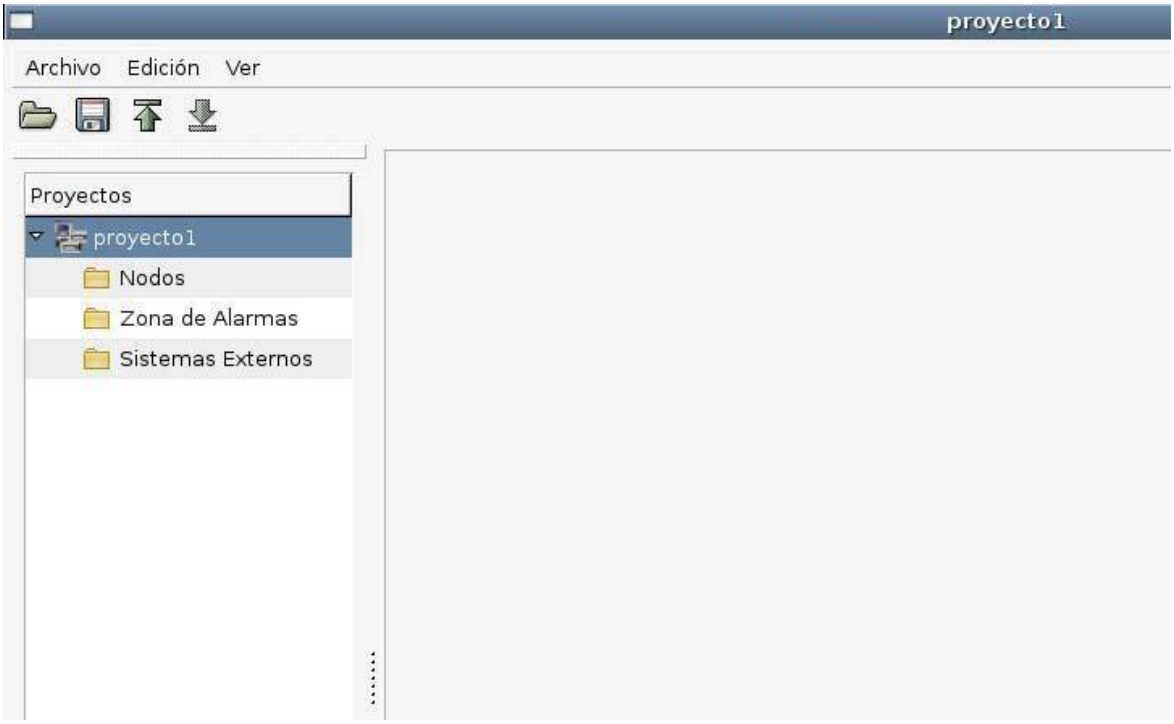


Figura 9 Vista del árbol de proyecto luego de haber agregado un nuevo proyecto y haberse actualizado.

El mecanismo para la creación de todos los recursos es similar al explicado anteriormente, para crear un nodo se siguen pasos similares teniendo otra clase controladora para su creación (CreateNode) y adicionándolo al proyecto, el resto de los pasos se mantiene igual, mostrando sus propiedades por defecto y actualizando la vista con los cambios realizados.

- **Modificar un nodo**

Luego de haber creado un recurso este se mostrará en la vista del árbol del proyecto. Para modificar las propiedades de un nodo físico se hace clic derecho sobre el, se despliega un menú emergente (Ver Figura 11) el cual tiene una etiqueta “Propiedades”, al hacer clic en ella se llama a una clase controladora (ShowProperties) la cual mostrará la vista (PropertiesInspector) de las propiedades del nodo (Ver Figura 12). El usuario modifica los elementos que desee (Ver Figura 13) y la clase controladora manda a actualizar la vista (TreeView) para mostrar los cambios realizados (Ver Figura 14). A continuación se muestra un diagrama de secuencia del caso correspondiente al ejemplo mencionado.

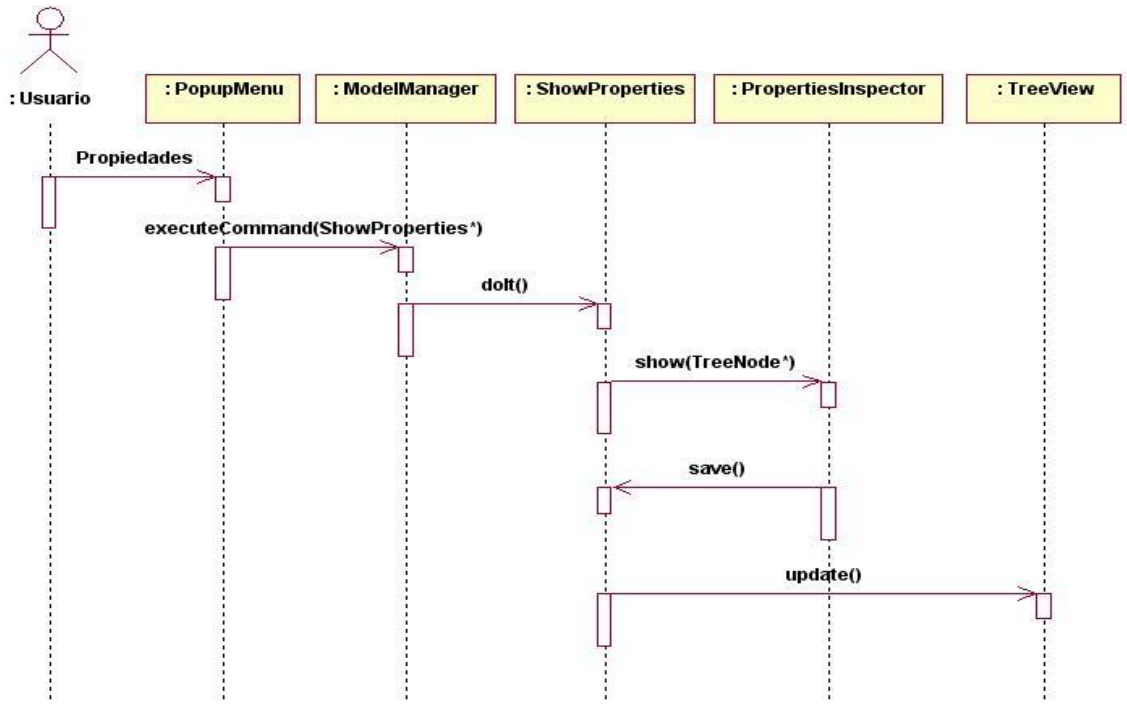


Figura 10 Diagrama de Secuencia “Modificar Nodo”.

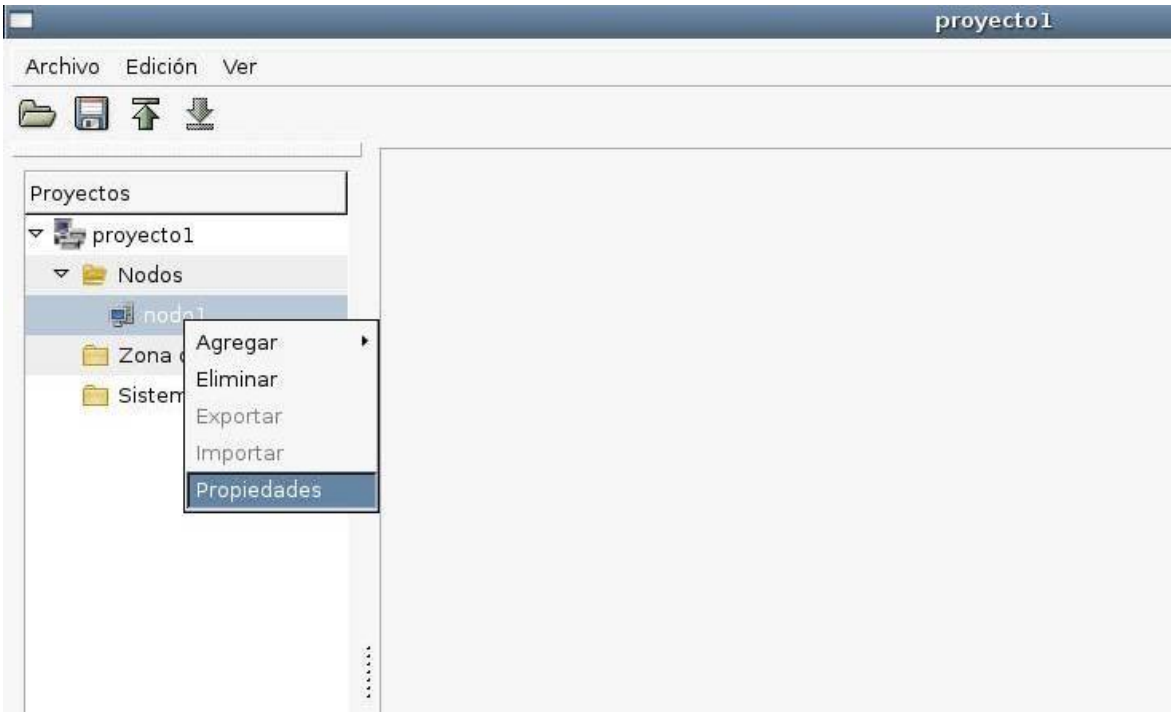


Figura 11 Menú emergente al hacer clic derecho sobre un nodo.

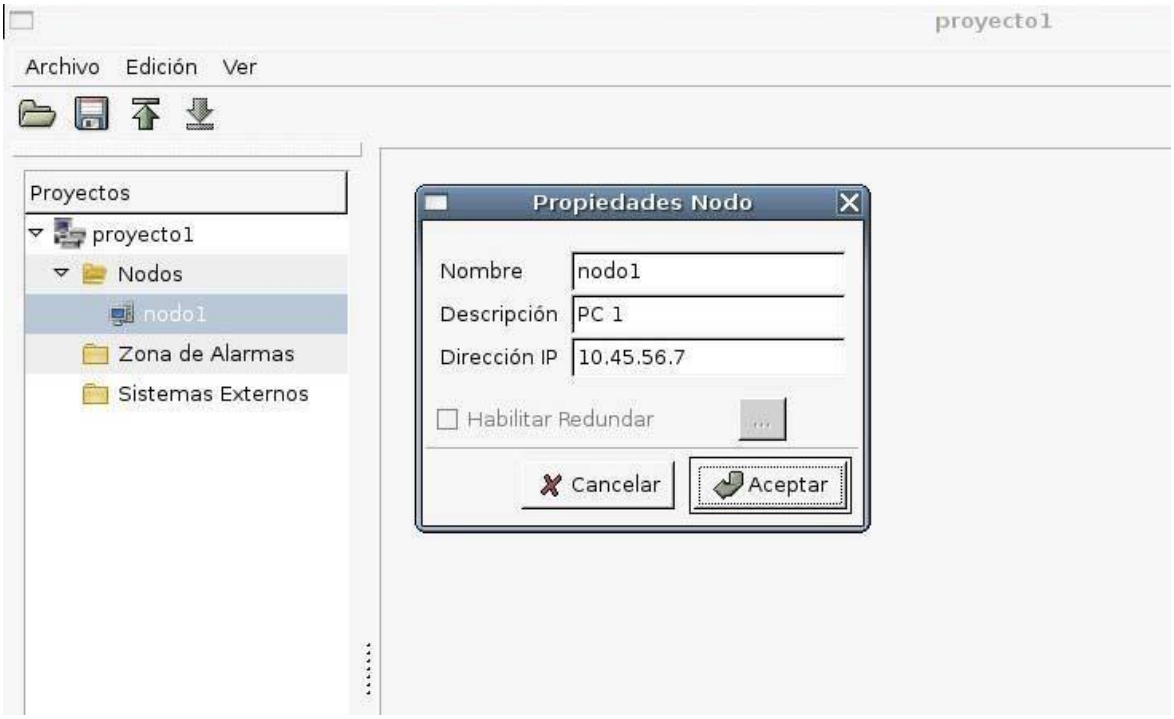


Figura 12 Inspector de propiedades de nodo.

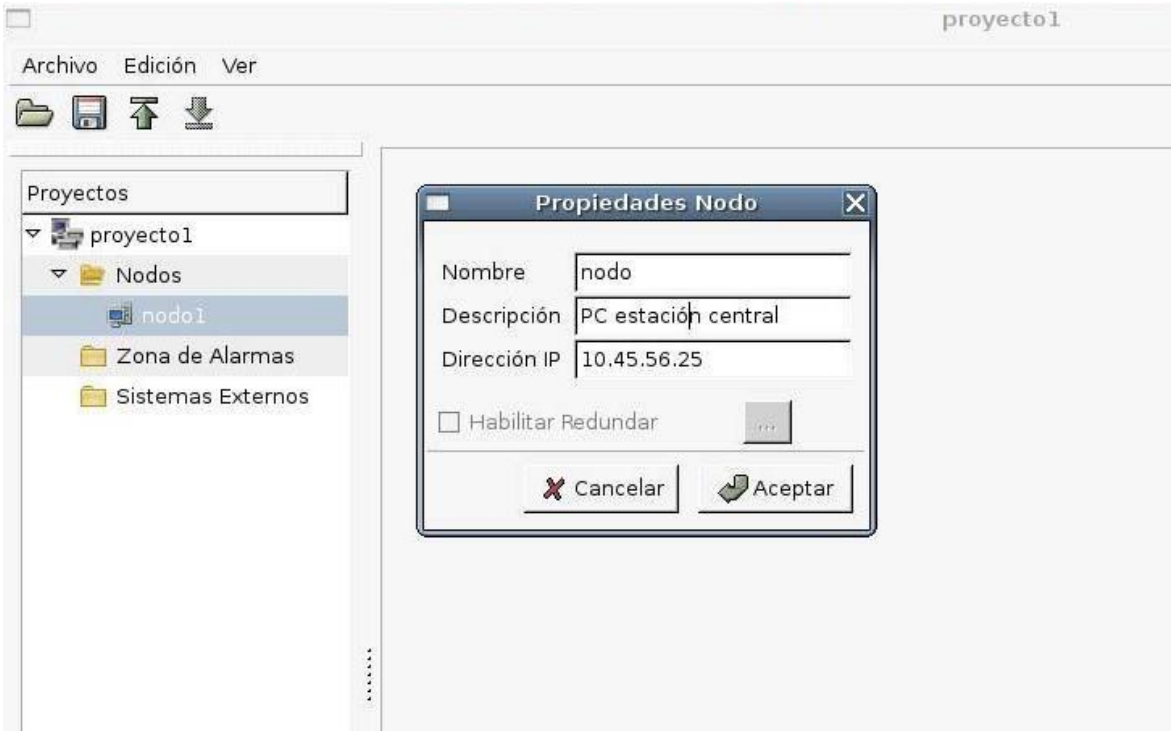


Figura 13 Inspector de propiedades de nodo. Parámetros modificados.

Figura 14 Vista del árbol de proyecto luego de haber modificado las propiedades de un nodo y haberse actualizado.

- **Eliminar un punto**

Para eliminar un punto se hace clic derecho sobre el, seguido se despliega un menú emergente (Ver Figura 16) el cual tiene una etiqueta “Eliminar” al hacer clic en ella se llama a una clase controladora (*DeleteTreeNode*) la cual mostrará la vista (Ver Figura 17) para que el usuario confirme la acción (*MessageWindow*), en caso de aceptar la clase *DeleteTreeNode* manda a eliminar el recurso, operación que realiza la clase controladora *ProjectManger y* luego invoca la actualización de la vista (Ver Figura 17). En caso del usuario no confirmar que desea eliminar el recurso se detiene la operación. A continuación se muestra un diagrama de secuencia del caso correspondiente al ejemplo mencionado.

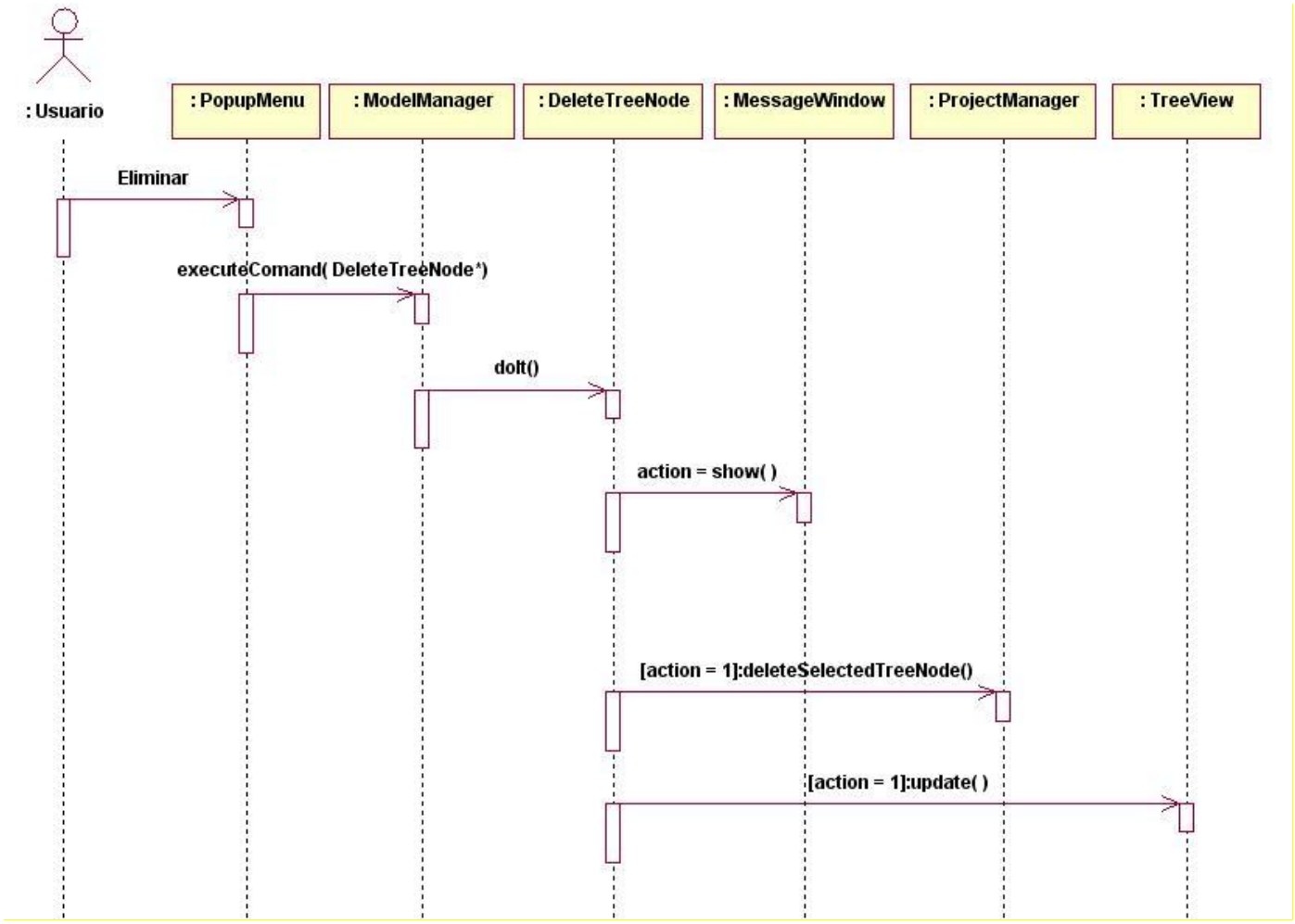


Figura 15 Diagrama de Secuencia “Eliminar Punto”.

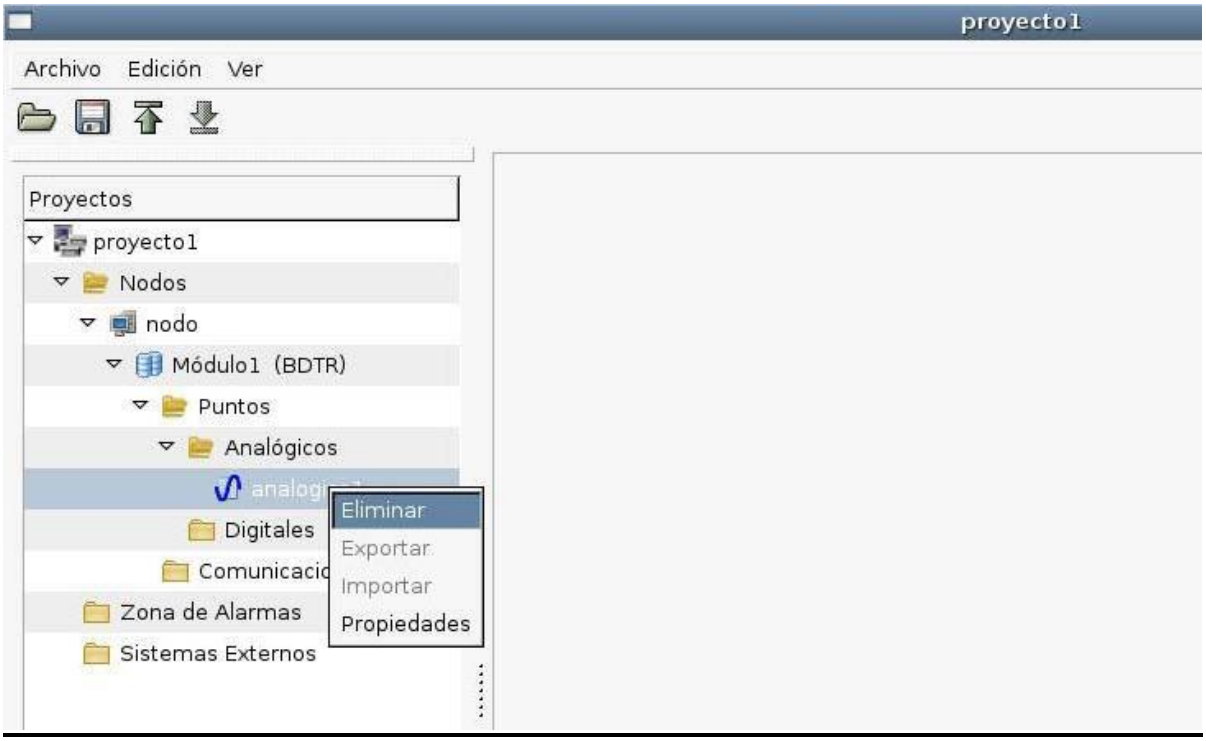


Figura 16 Menú emergente al hacer clic derecho sobre un nodo.

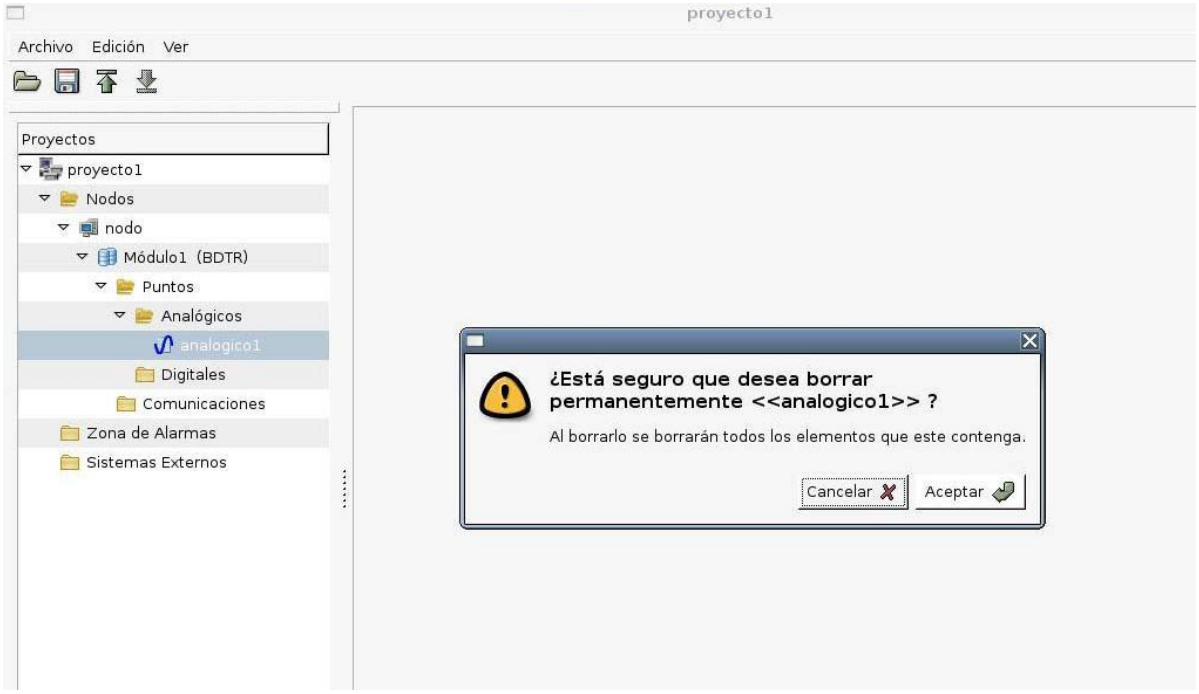


Figura 17 Ventana para la confirmación de la acción “Eliminar”.

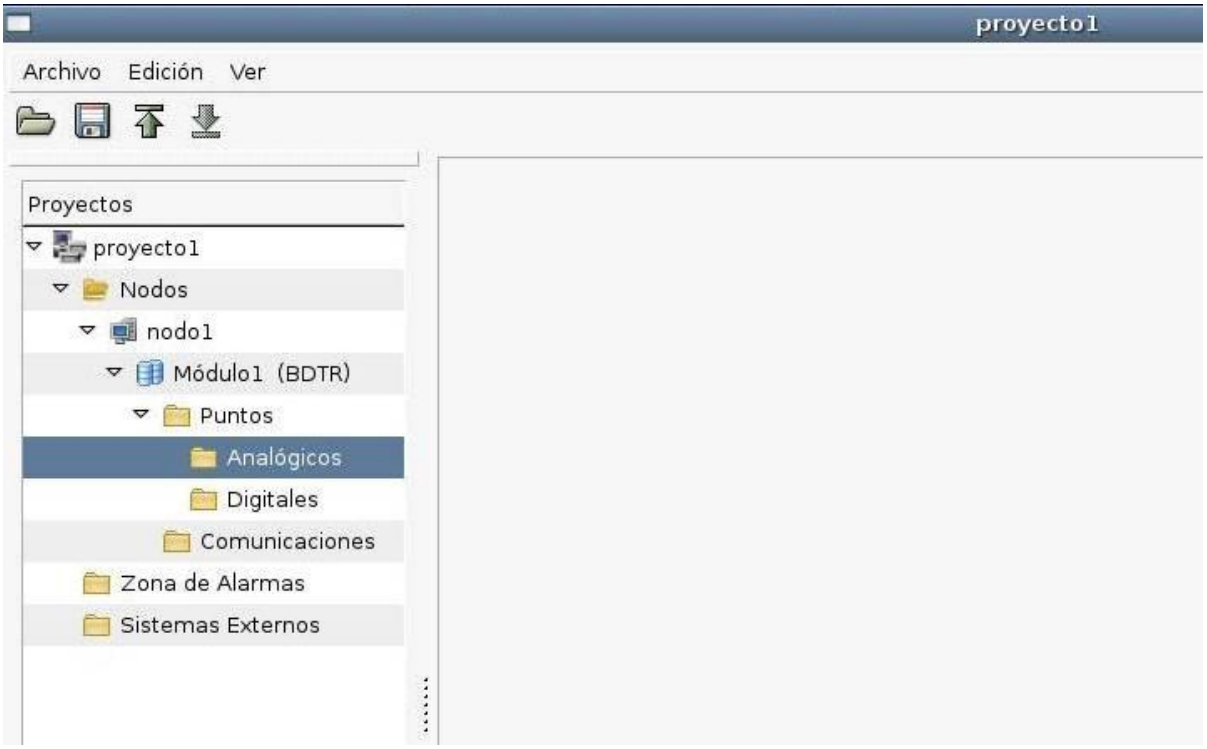


Figura 18 Vista del árbol de proyecto luego de haber eliminado el punto y haberse actualizado.

3.2 Clases de la estructura del sistema (Modelo)

Los elementos que representan el modelo en está aplicación se agrupan en el paquete *Base*. En este paquete se implementarán las clases que darán soporte a estos conceptos y las que se encargarán de manejar su organización y efectuar los cambios que se ejecuten sobre estos recursos.

3.2.1 Descripción de clases significativas

Entre las clases del modelo podemos encontrar la clase *ModelManager*. Que se encarga de administrar el modelo, en esta se incluye el administrador del proyecto y el administrador de las vistas. Está es la clase responsable de ejecutar los comandos que serán el enlace entre las vistas y el modelo para la modificación de este la actualización de las vistas. Ver Anexo 4.

Tabla 1 Clase "ModelManager"

Nombre: ModelManager	
Controladora	
Atributo	**Tipo**
projectManager	ProjectManager
viewsManager	ViewsManager
Para cada responsabilidad	
Nombre:	getProjectManager()
Descripción:	Devuelve el administrador de proyecto
Nombre:	getViewsManager()
Descripción:	Devuelve el administrador de vistas
Nombre:	executeCommand(Command* command)

Descripción:	Ejecuta el comando especificado

La clase *ProjectManager* se encarga de la administración del los recursos del sistema, estos recursos son los nodos del árbol de configuración (treeNode). Esta clase contiene el proyecto que se está configurando y las funciones para la gestión de los recursos en el árbol de configuración como pueden ser: devolver el recurso seleccionado, verificar el identificador de un recurso, verificar el identificador de un nodo físico.

Tabla 2 Clase “ProjectManeger”

Nombre: ModelManager	
Controladora	
Atributo	**Tipo**
projectCounter	int
treeNodeCounter	int
projectCollection	ProjectCollection
activeProject	Project*
selectedTreeNode	TreeNode*
Para cada responsabilidad	
Nombre:	getActiveProject ()
Descripción:	Devuelve el proyecto que se está configurando.
Nombre:	setActiveProject (Project* project)
Descripción:	Se le pasa el proyecto activo.
Nombre:	addProject(TreeNode* project)
Descripción:	Adiciona un proyecto a la conección.
Nombre:	clearSelected()
Descripción:	Deselecciona el node activo.
Nombre:	deleteSelectedTreeNode ()
Descripción:	Elimina el nodo activo.

Nombre:	getProjectCounter ()
Descripción:	Devuelve el contador de proyecto.
Nombre:	getTreeNodeCounter ()
Descripción:	Devuelve el contador de los nodos.
Nombre:	getSelectedTreeNode ()
Descripción:	Devuelve el nodo activo.
Nombre:	setSelectedTreeNode(TreeNode* selectedTreeNode)
Descripción:	Se el pasa el nodo seleccionado.
Nombre:	deleteProjects ()
Descripción:	Eliminar los proyectos creados.
Nombre:	isValidTreeNodeId(unsigned int id)
Descripción:	Verifica el identificador de un recurso
Nombre:	isValidAlarmId(unsigned int id)
Descripción:	Verifica el identificador de una alarma.
Nombre:	checkName(std::string name)
Descripción:	Verifica el nombre de un recurso
Nombre:	checkNodeIP(std::string ip)
Descripción:	Verifica el IP de un nodo físico.

Entre las clases controladoras de tipo comando tenemos a *ShowProperties* que es la encargada de mandar a mostrar la vista con las propiedades del recurso seleccionado.

Tabla 3 Clase “ShowProperties”

Nombre: ShowProperties	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre:	doIt(ModelManager& modelManager)

Descripción:	Muestra la vista con las propiedades del recurso seleccionado.

La clase *Project* representa el recurso proyecto en la estructura del sistema, En esta se encuentra la colección de nodos que podrá tener el sistema a configurar. También contiene algunas funciones de gestión como son: mantener un registro de los recursos que han sido modificados en el proyecto y verificar que se pueda adicionar un módulo a un nodo determinado.

Tabla 4 Clase "Project"

Nombre: Project	
Entidad	
Atributo	**Tipo**
description	string
nodeCollection	NodeCollection
activeNode	Node*
alarmZoneCollection	AlarmZoneCollection
modifyResources	PropertiesCollection
Para cada responsabilidad	
Nombre:	getActiveNode()
Descripción:	Devuelve el nodo físico activo.
Nombre:	setActiveNode(TreeNode* node)
Descripción:	Se le pasa el nodo físico activo.
Nombre:	getNodeCollection ()
Descripción:	Devuelve la colección de nodos físicos en el proyecto.
Nombre:	addNode(TreeNode* node)
Descripción:	Se adiciona un nodo físico a la colección.
Nombre:	addAlarmZone(TreeNode* alarmZone)
Descripción:	Se adiciona un zona de alarmas a la colección.
Nombre:	getAlarmZoneCollection()

Descripción:	Devuelve la colección de alarmas.
Nombre:	setDescription(string description)
Descripción:	Se le pasa la descripción del proyecto.
Nombre:	validateAddModule(string module)
Descripción:	Verifica que se pueda agregar un módulo a un nodo determinado.
Nombre:	addResource(TreeNode *node)
Descripción:	Agrega un recurso a la lista de recursos modificados.
Nombre:	addModifyResource(std::string tag,std::string action)
Descripción:	Se adiciona el nombre de un recurso y la acción que se ejecutó sobre el a la lista de recursos modificados.
Nombre:	deleteModifyResource(std::string tag)
Descripción:	Se elimina un recurso de la lista de modificados.
Nombre:	findModifyResource(std::string tag)
Descripción:	Busca un recurso en la lista de modificados.
Nombre:	getModifyResources()
Descripción:	Devuelve la lista de recursos modificados.

La clase *Node* representa a los nodos físicos del sistema. En esta se pueden tener uno o varios módulos del sistema.

Tabla 5 Clase “Node”

Nombre: Node	
Entidad	
Atributo	**Tipo**
moduleCollection	ModuleCollection
activeModule	Module*
description	string
ipAddress	string

Para cada responsabilidad	
Nombre:	getIPAddress()
Descripción:	Devuelve la dirección IP.
Nombre:	setIPAddress(string newIPAddress)
Descripción:	Se le pasa la dirección IP
Nombre:	getDescription()
Descripción:	Devuelve la descripción.
Nombre:	setDescription(string description)
Descripción:	Se le pasa la descripción.
Nombre:	addModule(Module* module)
Descripción:	Se adiciona un módulo.
Nombre:	getModuleCollection()
Descripción:	Devuelve la lista de módulos.
Nombre:	setActiveModule(Module* module)
Descripción:	Se le pasa el módulo activo.
Nombre:	getActiveModule()
Descripción:	Devuelve le módulo activo.
Nombre:	getSecurityModule()
Descripción:	Devuelve le módulo de seguridad.
Nombre:	getHDBModule()
Descripción:	Devuelve le módulo de base de datos histórica.
Nombre:	getRTDBModule()
Descripción:	Devuelve le módulo de base de datos en tiempo real.
Nombre:	getHMIModule()
Descripción:	Devuelve le módulo de interfaz hombre-máquina.

La clase *Point* representa a una variable del sistema, aquí se encuentran las características generales de estas, ya que otras clases especializan los diferentes tipos de variables existentes

agregándose muchos mas atributos en estos casos, como pueden ser *AnalogicPoint* para variables analógicas y *DigitalPoint* para variables digitales.

Tabla 6 Clase “Point”

Nombre: Point	
Entidad	
Atributo	**Tipo**
alarmZone	AlarmZone*
historical	Historical *
device	Device*
alarmCollection	AlarmCollection
group	Group*
description	string
type	Type
frequency	float
avalanche	bool
Para cada responsabilidad	
Nombre:	addAlarm(Alarm* alarm)
Descripción:	Adiciona una alarma.
Nombre:	getAlarmZone();
Descripción:	Devuelve las zona de alrmas
Nombre:	setAlarmZone(AlarmZone*)
Descripción:	Se le pasa la zona de alarmas.
Nombre:	getHistorical()
Descripción:	Devuelve el grupo de transferencia a histórico.
Nombre:	setHistorical(Historical*)
Descripción:	Se el pasa el grupo de transferencia a histórico.
Nombre:	clearAlarms()
Descripción:	Elimina todas las alarmas

Nombre:	setGroup(Group* group)
Descripción:	Se le pasa el grupo operacional de privilegios.
Nombre:	getGroup()
Descripción:	Devuelve el grupo operacional de privilegios.
Nombre:	getDevice()
Descripción:	Devuelve el dispositivo de control.
Nombre:	setDevice(Editor::Base::Device*)
Descripción:	Se le pasa el dispositivo de control.
Nombre:	getPointType()
Descripción:	Devuelve el tipo del punto.
Nombre:	setPointType(string)
Descripción:	Se le pasa el tipo del punto.
Nombre:	getAlarmCollection()
Descripción:	Devuelve la lista de alarmas.
Nombre:	getLevelAlarmHH()
Descripción:	Devuelve la alarma de nivel alto-alto.
Nombre:	getLevelAlarmH()
Descripción:	Devuelve la alarma de nivel alto.
Nombre:	getLevelAlarmL()
Descripción:	Devuelve la alarma de nivel bajo.
Nombre:	getLevelAlarmLL()
Descripción:	Devuelve la alarma de nivel bajo-bajo.
Nombre:	getChangeRateAlarm()
Descripción:	Devuelve la alarma de tasa de cambio.
Nombre:	getDeviationAlarm()
Descripción:	Devuelve la alarma de desviación.
Nombre:	getInstrumentFaultAlarm()
Descripción:	Devuelve la alarma de falla de instrumento.
Nombre:	getNoTimeVariationAlarm()
Descripción:	Devuelve la alarma de no variación en el tiempo

Nombre:	getExecCommandFailAlarm()
Descripción:	Devuelve la alarma de falla de ejecución de comando.
Nombre:	getNoCommandActionFailAlarm()
Descripción:	Devuelve la alarma de cambio de estado no comandado.
Nombre:	getStateAlarms()
Descripción:	Devuelve la lista de alarmas de estado.
Nombre:	setDescription(string description)
Descripción:	Se le pasa la descripción.
Nombre:	getDescription()
Descripción:	Devuelve la descripción.
Nombre:	setFrequency(float frequency)
Descripción:	Se le pasa la frecuencia.
Nombre:	getFrequency()
Descripción:	Devuelve la frecuencia.
Nombre:	setAvalanche(bool avalanche)
Descripción:	Se le pasa si está activa la avalancha de alarmas.
Nombre:	getAvalanche()
Descripción:	Devuelve si está activa la avalancha de alarmas.

La clase *Alarm* representa una alarma del sistema. En esta estarán las características generales de las alarmas ya que se especializaran en una clase para cada tipo de alarmas existentes.

Tabla 7 Clase “Alarm”

Nombre: Alarm	
Entidad	
Atributo	**Tipo**
priority	AlarmPriority
severity	AlarmSeverity

active	bool
avalanche	bool
Para cada responsabilidad	
Nombre:	setPriority(AlarmPriority priority)
Descripción:	Se le pasa la prioridad.
Nombre:	getPriority()
Descripción:	Devuelve la prioridad
Nombre:	setSeverity(AlarmSeverity severity)
Descripción:	Se le pasa la severidad
Nombre:	getSeverity()
Descripción:	Devuelve la severidad
Nombre:	isAvalanche()
Descripción:	Devuelve si tiene habilitada la avalancha de alarmas.
Nombre:	setAvalanche(bool)
Descripción:	Se le pasa si tiene habilitada la avalancha de alarmas.
Nombre:	setState(bool state)
Descripción:	Se le pasa el estado
Nombre:	getState()
Descripción:	Devuelve el estado.

3.3 Clases de la interfaz para la configuración del sistema (Presentación)

Para la representación de los elementos a configurar es necesario un conjunto de vistas que permitan la interacción del usuario con el sistema, estas se agrupan en el paquete *Views*.

3.3.1 Descripción de clases significativas

La clase *ViewsManager* se encarga administrar de la las vistas. Esta clase incluida en el administrador del modelo permite un rápido acceso a las vistas que mostrarán las características y organización de los recursos del sistema.

Tabla 8 Clase "ViewsManager"

Nombre: ViewsManager	
Controladora	
Atributo	**Tipo**
mainWindow	Window*
treeView	TreeView*
propertiesInspector	PropertiesInspector*
toolboxView	ToolboxView*
toolbarView	ToolbarView*
multiView	MultiView*
messageWindow	MessageWindow*
Para cada responsabilidad	
Nombre:	getTreeView()
Descripción:	Devuelve la vista que muestra el árbol de configuración.
Nombre:	setTreeView(TreeView* treeView)
Descripción:	Se le pasa la vista que muestra el árbol de configuración.
Nombre:	getPropertiesInspector();
Descripción:	Devuelve el inspector de propiedades.
Nombre:	setPropertiesInspector(PropertiesInspector* propertiesInspector)
Descripción:	Se le pasa el inspector de propiedades.
Nombre:	getPopupMenu();
Descripción:	Devuelve el menú emergente.
Nombre:	setPopupMenu(PopupMenu* popupMenu)
Descripción:	Se le pasa el menú emergente.
Nombre:	getToolboxView()

Descripción:	Devuelve el contenedor de herramientas.
Nombre:	setToolboxView(ToolboxView* toolboxView)
Descripción:	Se le pasa el contenedor de herramientas.
Nombre:	getToolbarView()
Descripción:	Devuelve la barra de herramientas.
Nombre:	setToolbarView(ToolbarView* toolbarView)
Descripción:	Se le pasa la barra de herramientas.
Nombre:	getMultiView()
Descripción:	Devuelve el contenedor de múltiples vistas.
Nombre:	setMultiView(MultiView* multiView)
Descripción:	Se le pasa el contenedor de múltiples vistas.
Nombre:	getMessageWindow()
Descripción:	Devuelve la ventana de mensajes.
Nombre:	setMessageWindow(MessageWindow* messageWindow)
Descripción:	Se el pasa la ventana de mensajes.
Nombre:	getWindow()
Descripción:	Devuelve la ventana principal.
Nombre:	setWindow(Window* mainwindow)
Descripción:	Se le pasa la ventana principal.

La clase *TreeView* representa la vista en la cual se visualizará la estructura del sistema en forma de árbol. Permite saber cual de los recursos está seleccionado y deberá ser actualizada ante algún cambio para ser reflejado al usuario. Ver Anexo 5.

Tabla 9 Clase “TreeView”

Nombre: TreeView	
Interfaz	
Atributo	**Tipo**

Para cada responsabilidad	
Nombre:	update()
Descripción:	Actualiza la vista del árbol de configuración.
Nombre:	getSelected()
Descripción:	Devuelve el identificador del elemento seleccionado.
Nombre:	getSelectedItem()
Descripción:	Devuelve el nombre del elemento seleccionado.
Nombre:	getTreeNodeTypeFolder()
Descripción:	Devuelve el tipo de nodo del árbol.

La clase *PropertiesInspector* representa la vista que permitirá representar al usuario las propiedades de un recurso, permitiendo también ser modificadas. Ver Anexo 6.

Tabla 10 Clase “PropertiesInspector”

Nombre: PropertiesInspector	
Interfaz	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre:	setModelManager(ModelManager& modelManager)
Descripción:	Se le pasa el administrador del modelo.
Nombre:	show(TreeNode* treeNode)
Descripción:	Se muestra la vista para configurar las propiedades del objeto que se le pasa por parámetro.
Nombre:	save()
Descripción:	Devuelve una lista con las propiedades del recurso que estaba mostrando

CONCLUSIONES

Para dar cumplimiento a los objetivos trazados en este trabajo se realizó un estudio sobre los sistemas SCADA, específicamente sobre la funcionalidad que permite la configuración de estos mediante una interfaz gráfica. Se realizó una descripción detallada de la estructura que soporta la configuración de los elementos del sistema.

De esta forma se obtuvo un modelo que permite la configuración de este tipo de sistemas para procesos de control industrial. El cual está compuesto por los conceptos más comunes en este tipo de procesos, los que permiten una mejor comprensión a la hora de modelar un proceso en la aplicación.

RECOMENDACIONES

Se hacer las siguientes recomendaciones:

- Agregar nuevos conceptos a la estructura de configuración.
- Implementar funcionalidad para importar nuevos elementos a la estructura de configuración.
- Implementar la funcionalidad para que se pueda encuestar las propiedades de todos los recursos en una sola vista.

REFERENCIAS BIBLIOGRÁFICAS

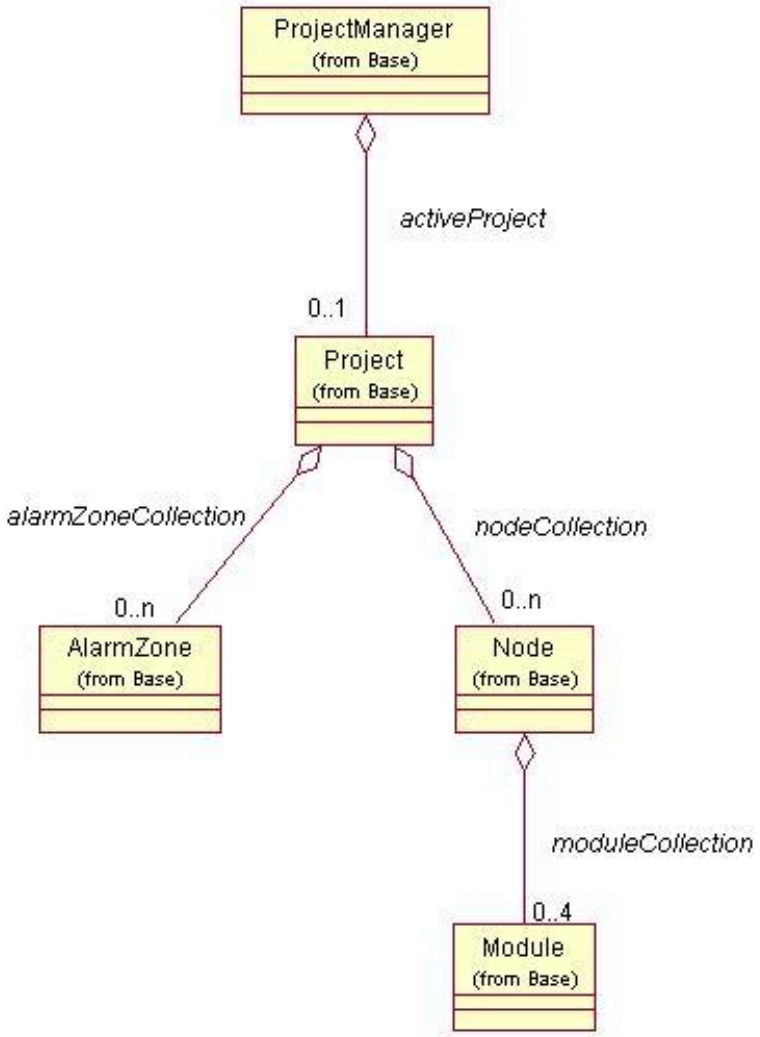
1. **Acevedo Sánchez, Jose. 2002.** *Control Avanzado de Procesos*. s.l. : Diaz de Santos, 2002.
2. **Acevedo Sánchez, Jose. 2006.** *Instrumentación y Control Básico de Procesos*. s.l. : Dias de Santos, 2006.
3. **DEBIAN.** DEBIAN. [En línea] [Citado el: 26 de Junio de 2008.] http://www.es.debian.org.
4. **ECLIPSE.** ECLIPSE. [En línea] [Citado el: 15 de Junio de 2008.] http://www.eclipse.org.
5. **GTK.** GTK. [En línea] [Citado el: 20 de Junio de 2008.] http://www.gtk.org/.
6. **LINUX.** LINUX. [En línea] [Citado el: 25 de Junio de 2008.] http://www.linux-es.org/distribuciones.
7. **Roca Cusido, Alfred. 2000.** *Control de Procesos.*
8. **Rodríguez Penin, Aquilino. 2006.** *Sistemas SCADA.* 2006.
9. **Villalobos Ordaz, Gustavo, y otros. 2006.** *Medición y Control de Procesos Industriales.* 2006.

BIBLIOGRAFÍA

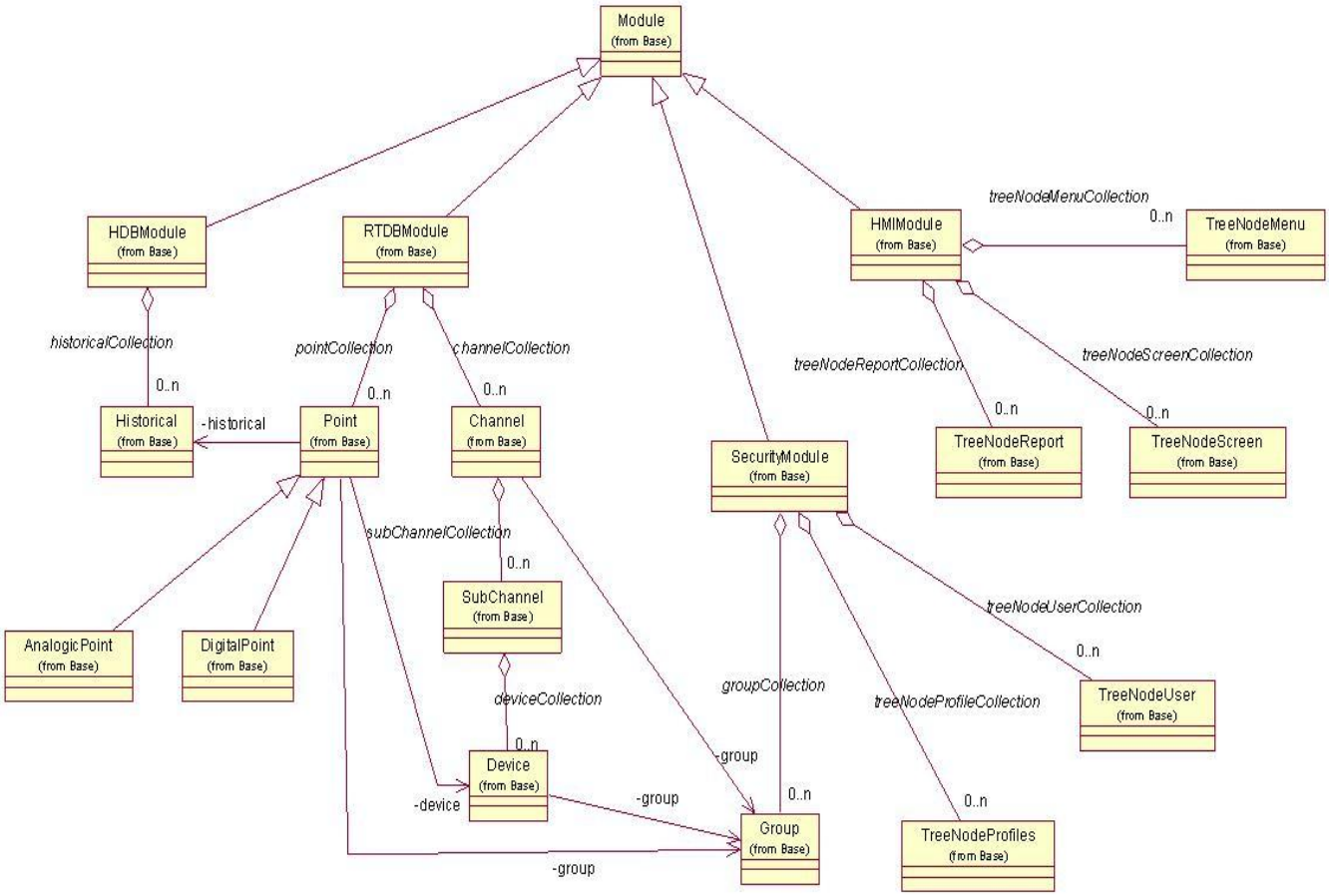
1. **Acevedo Sánchez, Jose. 2002.** *Control Avanzado de Procesos.* s.l. : Diaz de Santos, 2002.
2. **Acevedo Sánchez, Jose. 2006.** *Instrumentación y Control Básico de Procesos.* s.l. : Dias de Santos, 2006.
3. **Horowitz.** *Fundamentals of Programming Languages.* s.l. : Computer Science Press, 1984.
4. **Louden.** *Programming Languages. Principles and Practice.* s.l. : PWS Publishing Company, 1993
5. **Meyer.***Construcción de Software Orientado a Objetos (Segunda Edición).* s.l. : Prentice Hall, Madrid, 1999.
6. **Roca Cusido, Alfred. 2000.** *Control de Procesos.*
7. **Rodríguez Penin, Aquilino. 2006.** *Sistemas SCADA.* 2006.
8. **Sebesta.** *Concepts of Programing Languages. Fourth Edition.* s.l. : Addison-Wesley Longman, 1999.
9. **Stroustrup, Bjarne.** *The C++ Programming Language (Third Edition).* s.l. : Addison-Wesley, 1997.
10. **Tennet.** *Principles of Programming Languages.* s.l. : Prentice Hall International, 1981.
11. **Villalobos Ordaz, Gustavo, y otros. 2006.** *Medición y Control de Procesos Industriales.* 2006.
12. **Watt.** *Programming Language Concepts and Paradigms.* s.l. : Prentice Hall International, 1991.

ANEXOS

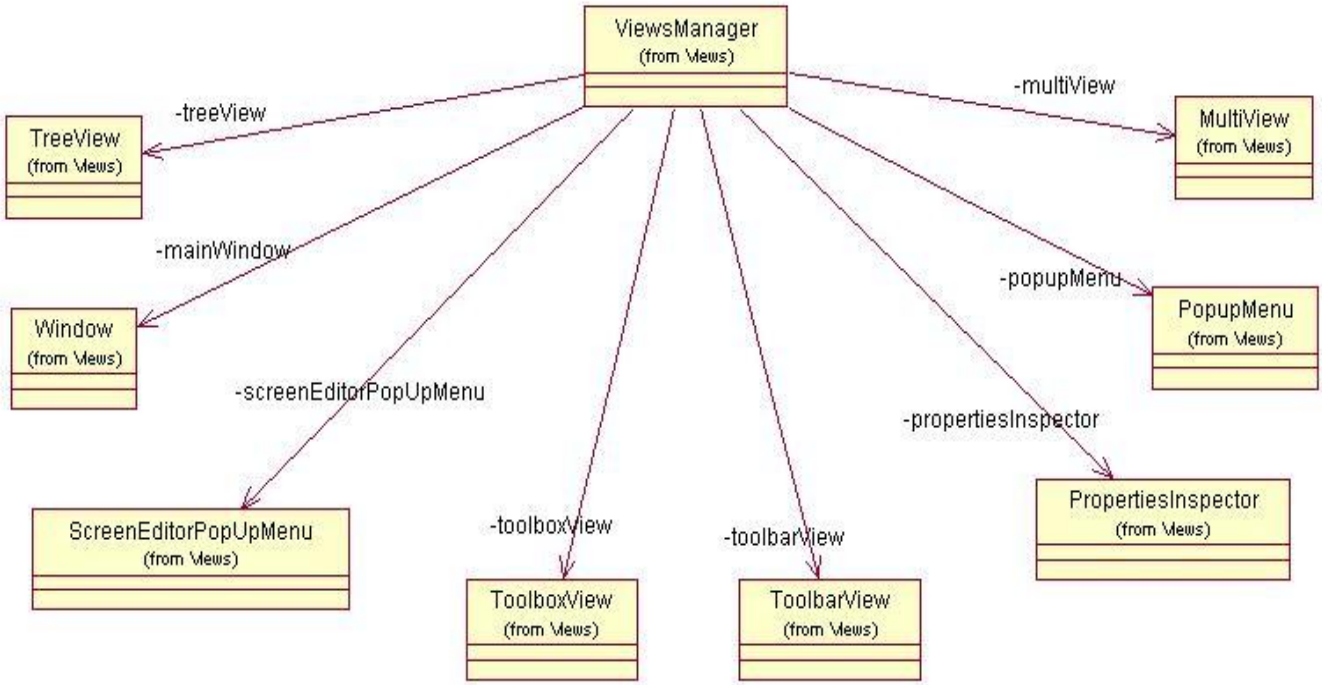
Anexo 1 Diagrama de clases “Estructura de configuración”, imagen 1.



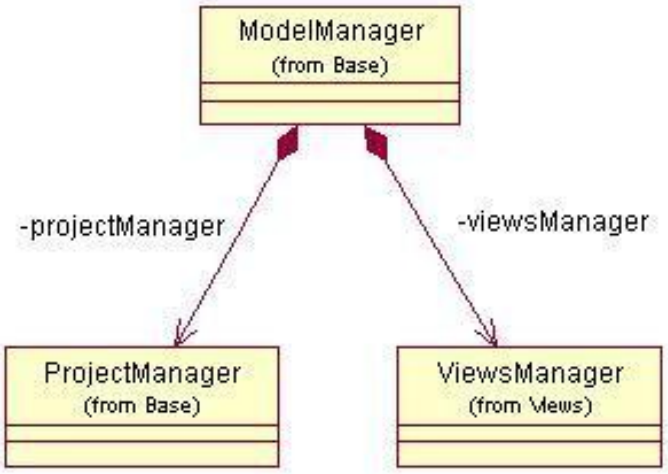
Anexo 2 Diagrama de clases “Estructura de configuración”, imagen 2.



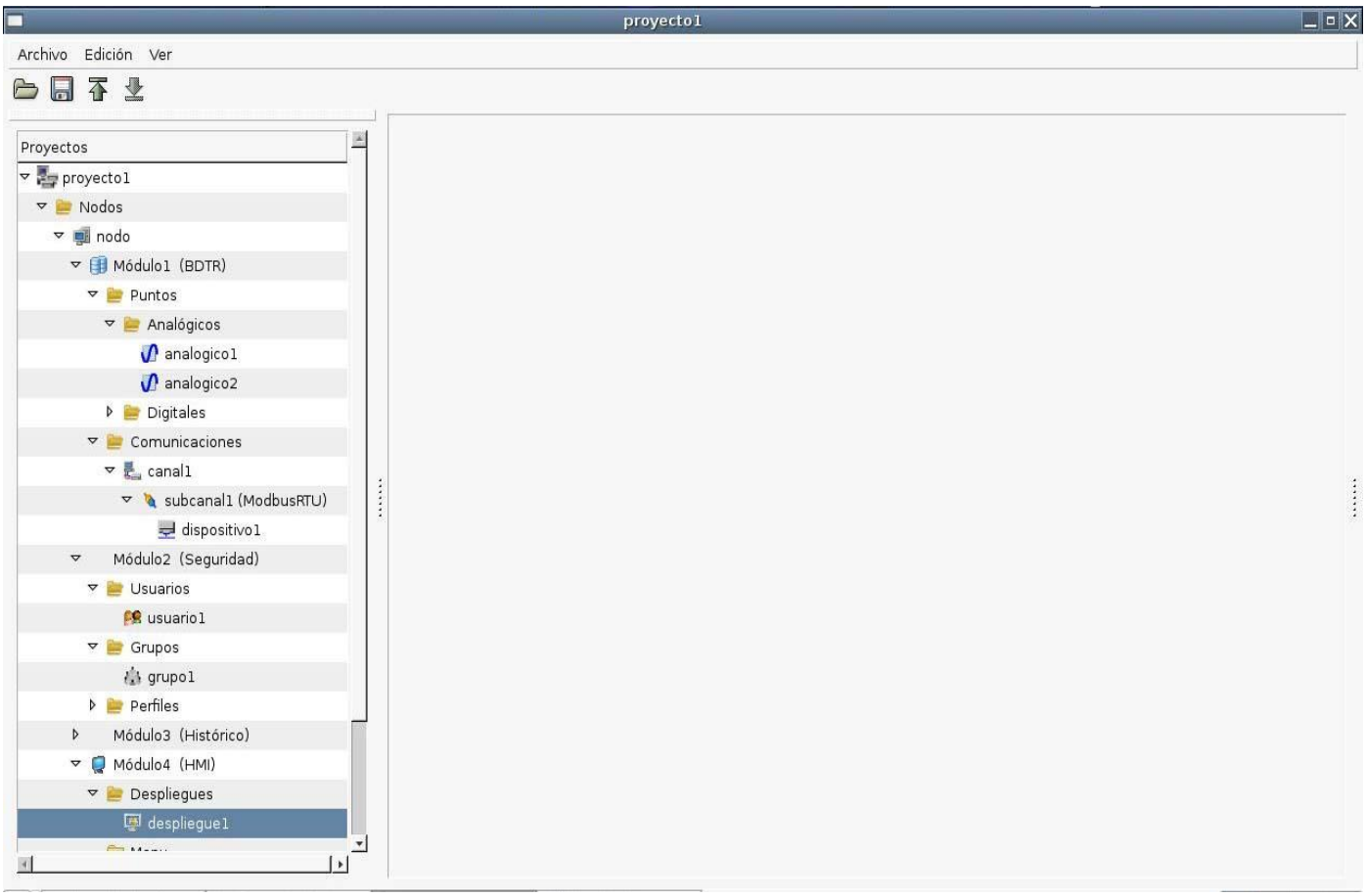
Anexo 3 Diagrama de clases “Vistas del sistema”.



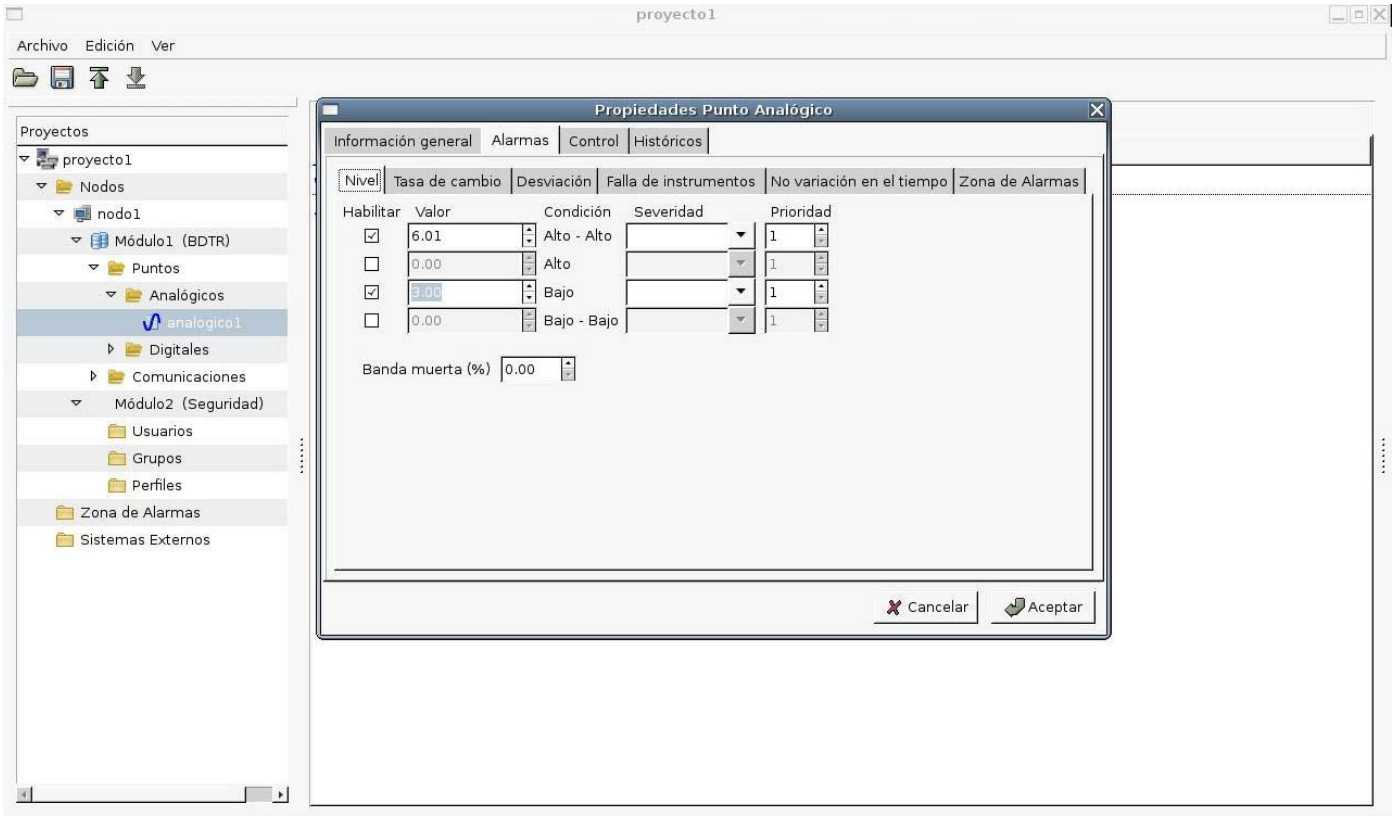
Anexo 4 Diagrama de clases “Administrador del Modelo”.



Anexo 5 Imagen de la vista de árbol de configuración “TreeView”.



Anexo 6 Imagen de la vista de inspector de propiedades "PropertiesInspector". Alarmas de nivel de punto analógico.



GLOSARIO DE TERMINOS

- **Autómatas:** equipo electrónico programable en lenguaje no informático y diseñado para controlar, en tiempo real y en ambiente industrial, procesos secuenciales.
- **Gráficos:** en informática, es el nombre dado a cualquier imagen generada por una computadora.
- **HMI:** *Human Machine Interface* (Interfaz hombre-máquina).
- **Módulo:** es una parte de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará una de dichas tareas (o quizá varias en algún caso).
- **Monitoreo:** observar el curso de uno o varios parámetros para detectar posibles anomalías.
- **Paquetes de software:** es una serie de programas que se distribuyen conjuntamente. Algunas de las razones para ello suelen ser que el funcionamiento de cada uno complementa o requiere a los demás, que sus objetivos están relacionados o como estrategia de mercadotecnia.
- **Proceso:** es un conjunto de actividades o eventos que se realizan o suceden (alternativa o simultáneamente) con un determinado fin.
- **Reguladores:** es un dispositivo electrónico diseñado con el objetivo de proteger aparatos eléctricos y electrónicos delicados de variaciones de diferencia de potencial (tensión/voltaje), descargas eléctricas y "ruido" existente en la corriente alterna de la distribución eléctrica.
- **Relé:** dispositivo electromecánico, que funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.
- **Sensores:** dispositivo capaz de transformar magnitudes físicas o químicas en magnitudes eléctricas.
- **Supervisión:** inspección de un trabajo o actividad por un encargado.
- **Sistema operativo:** es un software de sistema, es decir, un conjunto de programas de computadora destinado a permitir una administración eficaz de sus recursos.
- **Transiente:** incremento de voltaje de muy alta magnitud y muy corta duración.