UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

Creación de sistemas SCADA para el laboratorio de micro-red de la Universidad de Cuenca bajo el enfoque de desarrollo dirigido por modelos

Trabajo de titulación previo a la obtención del título de Ingeniero de Sistemas

Autores:

Diego Ismael Montesdeoca Chuva

CI:1400964001

Correo electrónico: diego.ismael.montesdeoca@gmail.com

David Efrain Buñay Moncayo

CI: 0302362934

Correo electrónico: david.b66@hotmail.com

Director:

Miguel Ángel Zuñiga Prieto

CI: 0102498052

Cuenca - Ecuador

05-noviembre-2021

Resumen:

Los sistemas de Supervisión, Control y Adquisición de Datos mejor conocidos como “SCADA” (del inglés, “Supervisory Control and Data Adquisition”), son aplicaciones que recopilan información acerca de equipos que se encuentran en plantas industriales y están distribuidos en amplias áreas. En donde, las tareas de control y supervisión requieren de sofisticados esquemas de automatización que otorgan acceso a datos de producción y variables distribuidas en el campo a distancia, en varios niveles de automatización de la planta [4], la complejidad de estos sistemas complica sus implementaciones.

Para llevar a cabo la implementación de sistemas SCADA es necesario tener conocimientos de lenguajes de programación específicos de un software de modelado y simulación (e.j., MATLAB) o conocimientos de protocolos para la conexión con dispositivos industriales (ej., MODBUS), por lo que una persona sin esos conocimientos no será capaz de construir un sistema totalmente funcional [1].

Además, el desarrollo de sistemas SCADA lleva un tiempo de implementación alto con lo cual el costo de la construcción del sistema aumenta [1]. Por lo cual, el enfoque de desarrollo dirigido por modelos (MDA) es ampliamente utilizado por la Ingeniería de Software, este enfoque se basa en modelos y aplica transformaciones durante el proceso de desarrollo, de esa manera se pueden desarrollar sistemas de forma automatizada y reutilizando componentes de software. El enfoque MDA propone varios modelos para presentar el sistema, cada modelo describe el sistema desde un nivel de abstracción alto[2].

Para construir sistemas SCADA basados en modelos, se ha llevado a cabo la construcción de una infraestructura de software, con un enfoque de desarrollo dirigido por Modelos para la generación automática del código fuente que implementa el diseño de Sistemas SCADA. La infraestructura provee: i) Un editor gráfico que facilita la creación de modelos que describen el diseño un sistema SCADA, ii) Un motor de transformación que toma como insumo modelos de diseño de sistemas SCADA y genera el código que implementa el sistema SCADA de acuerdo al lenguaje de programación de un software de modelado que permite programar la funcionalidad SCADA (e.j., MATLAB) [3]. MATLAB es un software privativo de MathWorks, además es un sistema de control utilizado por los ingenieros en todas las etapas de desarrollo, desde la modelización de la planta hasta el diseño, ajuste de los algoritmos de control y la lógica de supervisión [3], la efectividad de esta infraestructura tecnológica ha sido puesta a prueba en el laboratorio de micro-red de la Universidad de Cuenca, para diseñar, implementar sistemas SCADA.

Palabras claves: MDA. SCADA. Obeo Designer**.**

Abstract:

Supervisory Control and Data Acquisition (SCADA) systems are applications that collect information about equipment located in industrial plants and distributed over wide areas. Where control and supervision tasks require sophisticated automation schemes that provide access to production data and variables distributed in the field remotely, at various levels of plant automation [4], the complexity of these systems complicates their implementations.

To carry out the implementation of SCADA systems it is necessary to have knowledge of specific programming languages of a modeling and simulation software (e.g., MATLAB) or knowledge of protocols for connection with industrial devices (e.g., MODBUS), so a person without such knowledge will not be able to build a fully functional system [1].

In addition, the development of SCADA systems takes a long implementation time, which increases the cost of building the system [1]. Therefore, the Model Driven Development (MDA) approach is widely used by Software Engineering, this approach is based on models and applies transformations during the development process, so that systems can be developed in an automated way and reusing software components. The MDA approach proposes several models to represent the system, each model describes the system from a high level of abstraction [2].

To build model-based SCADA systems, the construction of a software infrastructure has been carried out, with a Model Driven Development approach for the automatic generation of the source code that implements the design of SCADA Systems. The infrastructure provides: i) A graphical editor that facilitates the creation of models that describe the design of a SCADA system, ii) A transformation engine that takes as input SCADA system design models and generates the code that implements the SCADA system according to the programming language of a modeling software that allows programming the SCADA functionality (e.g., MATLAB) [3]. MATLAB is a proprietary software from MathWorks, and it is a control system used by engineers in all stages of development, from plant modeling to design, adjustment of control algorithms and supervisory logic [3], the effectiveness of this technological infrastructure has been tested in the micro-grid laboratory of the University of Cuenca, to design and implement SCADA systems.

Keywords: MDA. SCADA. Obeo Designer.

Índice del Trabajo

Cláusula de Propiedad Intelectual

Diego Ismael Montesdeoca Chuva, autor/a del trabajo de titulación "Creación de sistemas SCADA para el laboratorio de micro-red de la Universidad de Cuenca bajo el enfoque de desarrollo dirigido por modelos", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

Cuenca-Ecuador, 05-noviembre-2021

Diego Ismael Montesdeoca Chuva

C.I: 1400964001

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Diego Ismael Montesdeoca Chuva en calidad de autor/a y titular de los derechos morales y patrimoniales del trabajo de titulación "Creación de sistemas SCADA para el laboratorio de micro-red de la Universidad de Cuenca bajo el enfoque de desarrollo dirigido por modelos", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca-Ecuador, 05-noviembre-2021

Diego Ismael Montesdeoca Chuva

C.I: 1400964001

Cláusula de Propiedad Intelectual

David Efraín Buñay Moncayo, autor/a del trabajo de titulación "Creación de sistemas SCADA para el laboratorio de micro-red de la Universidad De Cuenca bajo el enfoque de desarrollo dirigido por modelos", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

Cuenca-Ecuador, 05-noviembre-2021

David Efraín Buñay Moncayo

C.I: 0302362934

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

David Efraín Buñay Moncayo en calidad de autor/a y titular de los derechos morales y patrimoniales del trabajo de titulación "Creación de sistemas SCADA para el laboratorio de micro-red de la Universidad De Cuenca bajo el enfoque de desarrollo dirigido por modelos", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca-Ecuador, 05-noviembre-2021

David Efraín Buñay Moncayo

C.I: 0302362934

I. INTRODUCCIÓN

1.1 Creación de Sistemas SCADA y enfoque MDA

Las tareas de control y supervisión de plantas industriales requieren el uso de sofisticados esquemas de automatización que deben poder otorgar acceso a datos de producción y variables distribuidas en el campo a distancia, desde varios niveles de automatización de la planta [4]. Los sistemas de Supervisión, Control y Adquisición de Datos, mejor conocidos como "SCADA" (del inglés, "Supervisory Control and Data Acquisition"), son de los principales sistemas utilizados en las industrias para monitorear y controlar en tiempo real los estados de los equipos que componen la infraestructura de sus plantas [1], facilitando la interacción y retroalimentación con los dispositivos de campo, y controlando el o los procesos automáticamente (Medrano et al. 2018). Sin un sistema SCADA, sería extremadamente difícil, sino imposible, recopilar datos suficientes para que los operadores tomen decisiones consistentemente bien informadas [17].

Los sistemas SCADA comenzaron a popularizarse en la década de los años 60 en las economías industrializadas [19]; siendo el monitoreo y control de las grandes redes eléctricas el impulsor de su desarrollo y evolución [20]. Con el paso del tiempo y la consecuente evolución de la tecnología han pasado de ser sistemas monolíticos a emplear arquitecturas más flexibles que no solo han mejorado su seguridad y precisión, sino también han mejorado sus niveles de customización y funcionalidad [21]. Los sistemas SCADA están conformados por diferentes elementos, en donde los elementos con altos niveles de abstracción facilitan la interacción con el usuario. En el desarrollo de herramientas de software, la reutilización contribuye con su potencial de flexibilidad, permitiendo el uso de componentes en diversos desarrollos, extendiendo las capacidades, mejoras y funcionalidades del software [20]. El enfoque de Desarrollo de Software Dirigido por Modelos (DSDM) está siendo utilizado para soportar arquitecturas más flexibles y reutilización. Este enfoque promueve el uso de modelos a lo largo de todo el proceso de desarrollo de software, elevando el nivel de abstracción y permitiendo que estos modelos puedan ser transformados sucesivamente hasta la obtención del código fuente que implementa una solución. Este enfoque produce un impacto positivo en los tiempos de desarrollo, escalabilidad y la mantenibilidad de los sistemas; además de agilidad en construcción de sistemas para múltiples plataformas ya que sus modelos abstraen al desarrollador de conocimientos relacionados a las plataformas de desarrollo[48]. Con respecto a sistemas SCADA, existen propuestas [4] que aplican este enfoque. Este enfoque ha permitido reducir el coste de construcción en términos de tiempo y además permite la reutilización de componentes desarrollados para aplicaciones similares [6].

1.2 Problemática

A pesar de los beneficios que brindan los sistemas SCADA para las tareas de control y supervisión de plantas industriales, su construcción y mantenimiento representa un reto para los desarrolladores dado que requieren conocimientos técnicos específicos (e.j., lenguaje de programación, protocolos) de la plataforma en la que se construirán. Por lo que una persona sin esos conocimientos no será capaz de construir un sistema totalmente funcional.

Las soluciones de desarrollo de sistemas SCADA bajo el enfoque de desarrollo dirigido por modelos no presentan una interfaz gráfica sencilla e intuitiva para el operador, con lo cual se aumenta la complejidad en el diseño de modelos que representarán el sistema SCADA a generarse (ver sección 2.6).

En el laboratorio de micro red de la universidad de Cuenca se cuenta con sistemas SCADA adquiridos de proveedores externos, en donde al carecer de conocimientos específicos del código fuente de estos, incrementa los tiempos de mantenimiento o limita la incorporación de nuevas funcionalidades. Por lo tanto, se requiere de soluciones que eviten la dependencia de un proveedor específico y permitan a los operadores abstraerse de conocimientos técnicos específicos de la plataforma del proveedor y les permita centrarse en diseñar la solución. Por otro lado, el tiempo de implementación de los sistemas SCADA es alto, por lo que su costo de construcción aumenta [1]; requiriendo de enfoques que reduzcan los tiempos de construcción y mantenimiento.

1.3 Objetivos del proyecto

Construir una infraestructura de software para la generación automática de Sistemas SCADA. Esta infraestructura permitirá crear modelos que describen el diseño de un Sistema SCADA, y generar automáticamente el código fuente que implemente el diseño; facilitando el control y monitorización de dispositivos del laboratorio de micro-red de la Universidad de Cuenca.

Este objetivo se descompone en los siguientes objetivos específicos:

- Definir una arquitectura para la infraestructura tecnológica de generación de Sistemas SCADA.
- Diseñar un Lenguaje Específico de Dominio que facilite el diseño de Sistemas SCADA.

- Crear un editor gráfico que soporte el diseño de Sistemas SCADA conforme al lenguaje de diseño de Sistemas SCADA.
- Diseñar y crear un motor de transformación para la generación de código de Sistemas SCADA.
- Validar la infraestructura de software a través del caso de estudio.

1.4 Metodología

Para llevar a cabo la ejecución del presente trabajo de titulación se utiliza un proceso tradicional de desarrollo que se fundamenta en etapas comunes de ingeniería de software y faces comunes en el desarrollo de proyectos técnicos, las fases que se establecen son las siguientes:

Identificación y planteamiento del problema: Fase en la cual se identifica el problema a ser resuelto y se especifica con un alto grado de precisión.

Revisión bibliográfica: Fase en la cual se recopila información acerca del desarrollo de sistemas SCADA basada en un enfoque de desarrollo dirigido por modelos. Se analizan casos de estudio que implementen sistemas SCADA con este enfoque de desarrollo de software y se identifican los problemas que presentan dichas soluciones.

Investigación de herramientas de desarrollo: Se realiza una comparación entre las herramientas de implementación disponibles para el desarrollo de software dirigido por modelos, con el objetivo de escoger aquellas que puedan ser eficientes para el desarrollo de la solución tecnológica.

Definición de la arquitectura de la infraestructura: Se define el ciclo de vida y la arquitectura de la infraestructura de diseño y construcción de sistemas SCADA.

Requerimientos de la solución tecnológica: Se realizan entrevistas con los involucrados en el proyecto para definir requerimientos del sistema y refinar los mismos.

Implementación e integración: Se implementan los componentes de la infraestructura de software y se evalúa su integración con el resto del sistema.

Evaluación del sistema: Se pone a prueba la solución desarrollada mediante el diseño, implementación y ejecución de sistemas SCADA que estarán operando en una planta de producción de energía.

1.5 Estructura de la tesis

En este primer capítulo se han presentado los antecedentes, la problemática, el objetivo principal y específicos del trabajo de titulación planteado.

Capítulo 2 Marco teórico y trabajos relacionados

En este capítulo se presentan y analizan todos los conceptos necesarios para un mejor entendimiento del lector de los temas a tratar, como son: Sistemas SCADA en automatización industrial,elementos de un sistema SCADA, Human Machine Interface (HMI), MTU (Master Terminal Unit), API (Automatic Programable Industrial),PLC (Control Lógico Programable), pirámide de automatización, arquitectura dirigida por modelo, lenguajes específicos de dominio (DSL), metamodelo, modelado de editores gráficos, transformaciones modelo a modelo (M2M), transformaciones de modelo a texto, UML (Unified Modeling Language), SPEM (System and Software Process Engineering Metamodel), espacios Tecnológicos, Eclipse Modeling Framework, Obeo Designer, el lenguaje de transformación ACCELEO, herramienta de prototipado Figma, lenguajes de implementación de sistema SCADA y trabajos relacionados.

Capítulo 3 Aproximación tecnológica para la generación de sistemas SCADA

Dentro de este capítulo se presentan la arquitectura, se describen cada una de las fases del ciclo de vida de sistemas SCADA llevada a cabo para la creación de un infraestructura de software que permita crear sistemas SCADA a partir de modelos; también se explica el desarrollo de los componentes principales de la arquitectura como : DSL, metamodelo, motor de transformación de código.

Capítulo 4 Casos de estudio

En este capítulo se presenta el resultado de utilizar la Infraestructura de Software para la generación de sistemas SCADA en el desarrollo de dos casos de estudio. Se crearon dos sistemas SCADA para controlar, monitorear y almacenar datos de equipos del laboratorio de micro red de la Universidad de Cuenca.

Capítulo 5 Conclusiones y trabajo futuro

En este capítulo se presentan los resultados obtenidos al finalizar el presente trabajo de titulación, así como el trabajo futuro a realizar y las limitaciones existentes de la infraestructura desarrollada.

MARCO TEÓRICO Y TRABAJOS RELACIONADOS

En este capítulo se presentan los fundamentos teóricos para llevar a cabo el presente trabajo de titulación: desarrollo de sistemas SCADA y enfoque de desarrollo dirigido por modelos. En el capítulo también se analizan propuestas existentes para desarrollar sistemas SCADA mediante el uso del enfoque de desarrollo dirigido por modelos.

2.1 Sistemas SCADA en automatización industrial

El continuo aumento de la industria, y el fuerte empuje de las Tecnologías de la Información y la Comunicación (TIC), ha propiciado en los últimos años la aparición de soluciones tecnológicas que permiten la optimización de procesos industriales, [19]. Los procesos industriales cobran una gran importancia a la hora de conseguir una reducción de costes, optimización de la operatividad, gracias a sistemas cada vez más sofisticados y de bajo coste de implementación [38]. Para llevar a cabo un sistema de monitorización remota, distintas tecnologías se integran en el diseño e implementación de los distintos elementos, tanto hardware como software, así como el funcionamiento del sistema completo [19], ese tipo de sistemas se conocen como SCADA, que es una aplicación o conjunto de aplicaciones de software especialmente diseñada para funcionar sobre ordenadores de control mediante la comunicación con los equipos y actuadores a través de interfaz gráfica de alto nivel. El usuario final puede comunicarse con los equipos de campo mediante una interfaz gráfica que proporciona funciones de control, monitorización, configuración y almacenamiento de información.

2.2 Elementos de un sistema SCADA

Para que un sistema SCADA pueda realizar las tareas de control y monitoreo asignadas, requiere de varios elementos, los cuales tienen funciones específicas; como son la MTU, la HMI, redes de comunicación e instrumentos de campo (PLC, sensores, actuadores , alarmas, etc). [48]. En la figura 2-2 se ilustran los elementos que componen un sistema SCADA:

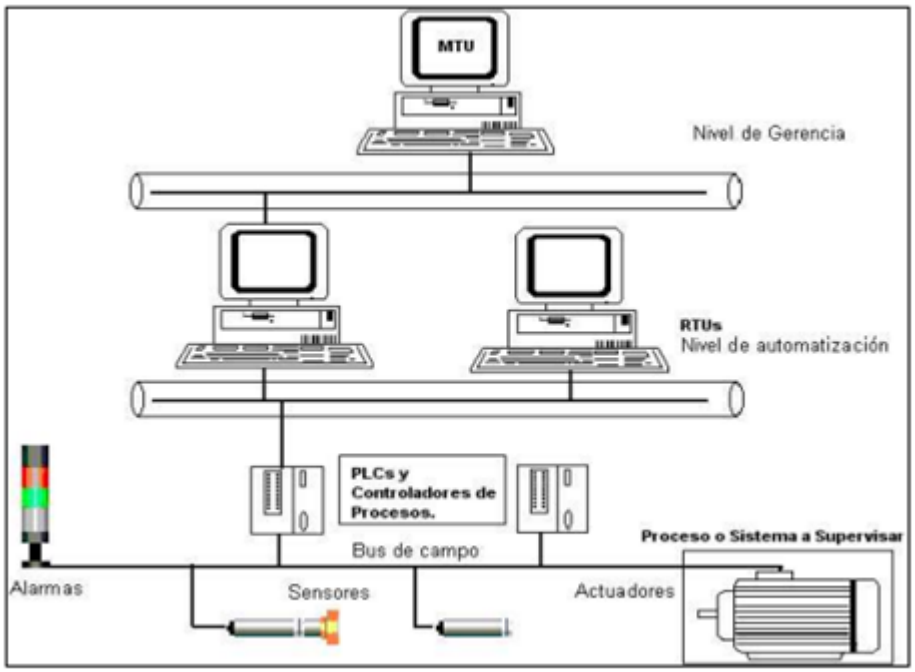


Figura 2-2: Elementos de un sistema SCADA [48]

2.2.1 Human Machine Interface (HMI)

HMI es la abreviación en inglés de Interfaz Hombre Máquina (IHM), previamente denominada "MMI" (Man Machine Interface). HIM es el dispositivo que permite la interfaz entre la persona y el proceso [36]. Los sistemas HMI son la "ventana" del proceso; esta ventana puede estar en dispositivos especiales como paneles del operador o en una computadora compuestos por indicadores y comandos, tales como luces pilotos, indicadores digitales y análogos, registradores, pulsadores, selectores y otros que se interconectan con la máquina [30]. De esta manera se establece el operador puede interactuar con el sistema SCADA.

2.2.2 MTU (Master Terminal Unit)

Se refiere a una parte del sistema SCADA encargada de centralizar todas las Unidades Terminales Remotas, es decir la información proporcionada por los equipos de instrumentación de campo; todos esos datos son almacenados en la base de datos, así como la configuración de la misma [35]. Para que el componente MTU funcione correctamente debe trabajar con APIs y PLCs.

2.2.3 RTU (Remote Terminal Unit)

Son ordenadores situados en nodos estratégicos del sistema, sus funciones son gestionar y controlar las subestaciones del sistema al recibir las señales de los sensores de campo; también se encarga de la comunicación con los elementos finales de control ejecutando el software del sistema SCADA [48]. Se encuentran en el nivel intermedio o de automatización, a un nivel superior está el MTU y a un nivel inferior los distintos instrumentos de campo que son los que ejercen la automatización física del sistema, control y adquisición de datos.

Este tipo de ordenadores no necesariamente tienen que ser PCs, ya que la necesidad de soportar un HMI no es tan grande a este nivel, por lo tanto suelen ser ordenadores industriales tipo armarios de control, aunque en sistemas muy complejos pueden haber subestaciones intermedias en formato HMI [48].Ciertos PLCs cuentan con la capacidad de funcionar como RTUs, debido a un nivel de integración mayor y recursos de procesamiento con mayor potencia de cálculo.

2.2.4 PLC (Control Lógico Programable)

Un PLC es un dispositivo electrónico que cuenta con una memoria programable para el almacenamiento de instrucciones, permitiendo la implementación de funciones específicas como: lógicas, secuenciales, temporizadas, de conteo y aritméticas; con el objeto de controlar máquinas , procesos en tiempo real y en ambientes industriales [41]; con la sincronización de estos dispositivos, los sistemas SCADA pueden ofrecer un funcionamiento correcto de acuerdo al objetivo para el que se haya diseñado.

Una característica importante al momento de trabajar con sistemas SCADA es el control de un proceso , éste actúa sobre sensores y actuadores con una combinación de posibilidades de regulación PID (regulación de acción proporcional, integral y derivada) y de control secuencial; el autómata programable satisface las exigencias tanto de procesos continuos como discontinuos [35].

2.3 Pirámide de automatización

La automatización es la facultad que poseen algunos procesos físicos para desarrollar las actividades de operación y funcionamiento en forma autónoma [37], es decir por cuenta propia. La pirámide de automatización utiliza tecnologías asociadas con la aplicación de sistemas de tipo mecánicos, electrónicos y computarizados, para la operación y control de la producción, en algunos casos con cierto grado de participación físico-humana y de

inteligencia artificial [41]. Los diferentes tipos de automatización hacen uso de redes industriales las cuales se agrupan jerárquicamente para establecer conexiones lo más adecuadas a cada área. Tradicionalmente se definen cuatro niveles (gestión, control, campo y entradas/salidas) dentro de una red industrial [39] .Los sistemas SCADA se encuentran en el nivel de control dentro de la pirámide de automatización, en la figura 2-1 se ilustran los niveles de automatización.

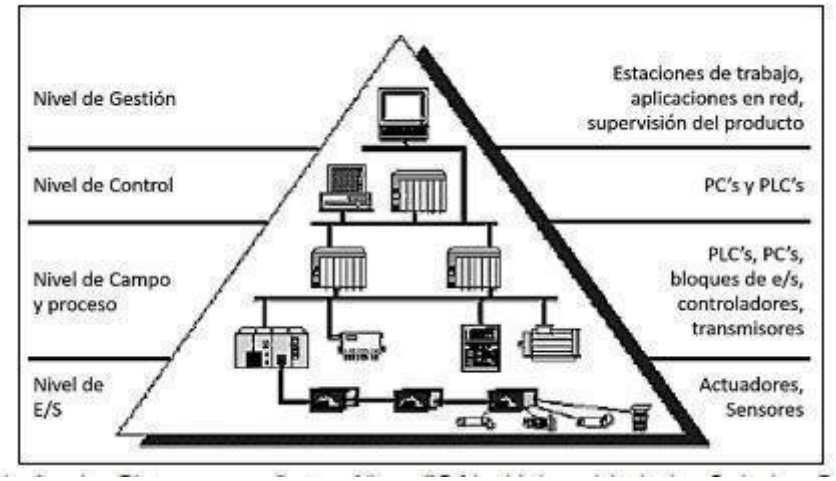


Figura 2-1 Pirámide de automatización industrial [39

]

2.4 Arquitectura dirigida por modelos

En las últimas décadas se ha trabajado en el área de desarrollo de sistemas para encontrar técnicas que permitan incrementar la productividad y el control de calidad en cualquier proceso de elaboración de software [31]. Dentro de esta área, la Ingeniería de software es una de las temáticas que más desarrollo demanda en los momentos actuales [26] Las investigaciones relacionadas se han dirigido hacia el desarrollo de métodos y algoritmos apropiados que conlleven a la creación y explotación de herramientas que asistan por medios computacionales al desarrollo de otros sistemas [25]. Entre las alternativas que han surgido figura el marco de trabajo MDA (Arquitectura dirigida por modelos), que constituye un intento de proporcionar una solución al problema de las plataformas cambiantes y la portabilidad de los sistemas de información [26]. MDA como arquitectura está destinada a aumentar la productividad en el desarrollo de software, reduciendo el coste del mismo en términos de tiempo y de reutilización de

componentes desarrollados para aplicaciones similares dentro de las diferentes etapas del ciclo de desarrollo de software [32].

MDA es un concepto promovido por el OMG (acrónimo inglés de Object Management Group) a partir de 2000, con el objetivo de afrontar los desafíos de integración de las aplicaciones y los continuos cambios tecnológicos [48]. MDA proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos [32].

En MDA, la funcionalidad del sistema es definida en primer lugar como un modelo independiente de la plataforma (Platform-Independent Model o PIM) a través de un lenguaje específico para el dominio del que se trate [32]. En este punto aparece además un tipo de modelo existente en MDA, el modelo de definición de la plataforma (Platform Definition Model o PDM). Entonces, dado un PDM correspondiente a CORBA,.NET, Web, etc., el modelo PIM puede traducirse a uno o más modelos específicos de la plataforma (Platform-Specific Models o PSMs) para la implementación correspondiente, usando diferentes lenguajes específicos del dominio, o lenguajes de propósito general como Java, C#, Python, etc [26].

2.4.1. Lenguajes Específicos de Dominio (DSL)

El Diseño de un nuevo lenguaje que permita modelar nuevas propiedades técnicas de una manera más simple y fácil, describir o implementar soluciones hace referencia a un lenguaje específico de dominio [23] La creación de un nuevo lenguaje es una tarea que requiere mucho tiempo, necesita experiencia y, por lo tanto, generalmente la realizan ingenieros de software.

Hoy en día, la necesidad de nuevos lenguajes para varios dominios está aumentando considerablemente [48]. Afortunadamente, también existen herramientas más sofisticadas que permiten a los ingenieros de software definir un nuevo lenguaje con un esfuerzo razonable y mejorar la productividad de los desarrolladores dentro de dominios específicos [22].

2.4.2 Metamodelo

La razón al crear un metamodelo para un proceso de software que describe la naturaleza del problema a abstraer, se da a través de flujos de trabajo que pueden ser interrumpidos por eventos de cambio [21]. El metamodelo mejora la comprensión del dominio porque generaliza los conceptos y actividades esenciales en un estándar y simplifica la

comunicación, además permiten la creación de instancias de múltiples modelos de proceso y facilita la adaptación y personalización [21]. Es importante una buena abstracción porque los procesos pueden ser modelados de acuerdo a la estructura del metamodelo [41].

MDA se usa como pautas para definir los ecosistemas de aprendizaje utilizando modelos conceptuales de alto nivel, modelos independientes de plataforma [21]. Los PIM pueden más tarde traducirse a especificaciones ejecutables concretas o código usando mapeos estándar [30]. Esto se realiza de acuerdo con la arquitectura de metamodelo de cuatro capas de OMG, las cuatro capas son la capa metamodelo (M3), la capa metamodelo (M2), el usuario capa de modelo (M1) y la capa de objeto de usuario (M0) [44].

Con la madurez adquirida MDA aporta a la creación de estándares para la implementación de modelos independientes de la plataforma [9] De esa forma MDA separa la descripción funcional de un sistema de los detalles específicos de su plataforma o dependientes de la tecnología [10].

2.4.3 Modelado de Editores gráficos

De acuerdo a [38] los editores gráficos son una parte importante en el trabajo de modelado de sistemas, reemplazando la generación de código tradicional que es la programación y dando un papel fundamental a los modelos, que se definen a través de especificaciones de lenguaje de dominio. Las propiedades más sobresalientes de los editores gráficos comúnmente se localizan en un archivo propio de cada framework, como el MoDiGen Core [30]. Por lo general estos son archivos de texto que sirven como la raíz para la generación de código haciendo referencia al metamodelo y su diagrama asignado.

Los diagramas asignados son la representación visual de la abstracción del sistema que está asignado a un DSL [38]. El mapeo de los diagramas y la posterior instancia del metamodelo contienen la lógica que representa la sintaxis correcta de los editores gráficos. Luego de entender la arquitectura y crear la instancia se especifica las opciones básicas y relaciones de la representación visual del editor, con estas opciones se puede conocer el funcionamiento que tiene el mismo. Después de la instancia se realiza el modelo navegacional que tendrá el editor, este muestra detalladamente la interacción entre cada elemento perteneciente a la interfaz del editor en base a la arquitectura se puede conocer los módulos directamente relacionados con el mismo [28].

2.4.4 Transformaciones modelo a modelo (M2M)

Las transformaciones modelo a modelo convierten un modelo de entrada (o múltiples) en un modelo de salida (o múltiples). Los modelos tanto de entrada como de salidas pueden ser instancias del mismo o de diferente metamodelo. Este tipo de transformaciones son necesarias en ciertos casos donde es necesario contar con un modelo intermedio para disminuir la complejidad de una transformación directa entre PIMs y PSMs cuando los niveles de abstracción son muy distintos. Por ejemplo, cuando se intenta transformar desde un diagrama de clases a una implementación de EJB (Enterprise Java Beans) algunas herramientas, como OptimalJ, generan un modelo de componentes EJB intermedio que contiene toda la información necesaria para producir el código Java [15].

2.4.5 Transformaciones de modelo a texto

Las transformaciones de Modelo a Texto (M2T, Model-To-Text) tienen por objetivo la generación de artefactos de tipo texto a partir de un modelo. Estas transformaciones son utilizadas en el manejo de operaciones de modelos, para la generación de código, para la documentación y serialización. También son conocidas como transformaciones de modelo a código y pueden ser vistas como un caso especial de las transformaciones modelo a modelo en donde lo único que debemos proveer es el metamodelo de lenguaje de salida [27].

2.4.6 UML (Unified Modeling Language)

UML, el lenguaje de modelado unificado de software es un estándar que tiene una amplia aceptación, su propósito era el diseño a mayor detalle. Su capacidad para describir elementos y las relaciones entre ellos lo hacen potencialmente aplicable a muchos ámbitos de la ingeniería de software.[6]

UML se ha convertido en el lenguaje de modelado de facto para el desarrollo de software. Debido a sus variadas características ha aumentado su popularidad ya que tiene una notación estandarizada, fuerte en expresividad. UML proporciona 13 tipos de diagramas que permiten modelar varias vistas y niveles de abstracción diferentes [7] Además, UML admite extensiones específicas de dominio utilizando estereotipos y valores etiquetados. Existen muchas herramientas de casos que integran el modelado UML con otras tareas como generar código y modelos de ingeniería inversa a partir del código [7].

2.4.7 SPEM (System and Software Process Engineering Metamodel)

SPEM (Software Process Engineering Metamodel), una especificación de OMG (Object Management Group) que está basado en MOF (MetaObject Facility) y es un metamodelo UML (Uniform Model Language [53].

SPEM permite representar una amplia gama de procesos de desarrollo de software, así como sus componentes. Provee un conjunto de elementos de modelado de procesos por lo cual constituye un exhaustivo proceso de desarrollo de software [54].

SPEM proporciona una sintaxis y estructura para cada aspecto de los procesos desarrollo tales como:

- Roles
- Tarea.
- Artefactos
- Lista de verificación
- Productos de trabajo.
- Técnicas y herramientas.
- Estructuras de trabajo.
- Capacidad de rastreo y refinamiento.
- Ayuda sensible al contexto, guía y lineamientos.
- Descripción textual de elementos

2.5 Espacios Tecnológicos

MDA, es un enfoque de desarrollo de software que permite automatizar la generación de código para el desarrollo de aplicaciones de diversos dominios. Los modelos son el principal componente en el proceso de desarrollo de software basado en MDA, ya que tienen un nivel de abstracción alto y facilitan la automatización en la creación de aplicaciones.

Entonces para llevar a cabo este proceso de desarrollo de software es necesario contar con herramientas que permiten realizar las siguientes acciones tanto en los metamodelos y modelos:

- Diseño
- Interpretación
- Comparación
- Validación
- Transformación

2.5.1 Eclipse Modeling Framework

EMF (Eclipse Modeling Framework), es un entorno de modelado y generación de código que permite construir aplicaciones basadas en un modelo de datos estructurado. Un modelo puede ser especificado en un estándar XMI para el cual EMF proporciona herramientas que permitan producir clases JAVA basadas en dicho modelo, además EMF permite la visualización y edición tanto del modelo y del editor gráfico basado en comandos.[7]

Dentro de EMF se admiten tres niveles de generación de código:

- Modelo: Proporciona interfaces y clases Java para todas las clases del modelo, además de una clase de implementación de tipo factory y package (metadatos).
- Adaptadores: Genera clases denominadas Item Providers que permiten adaptar las clases del modelo para realizar su respectiva edición y visualización.
- Editor: Produce un editor que se ajusta a los estilos de modelos EMF y es el punto de inicio para personalizar el editor.

2.5.2 Obeo Designer

Obeo Designer, es una solución de código abierto que permite la implementación de entornos de trabajo para modelados gráficos personalizados que se ajustan a dominios específicos [8] como: aplicaciones de software, sistemas industriales, o la organización de empresas.

Los entornos de trabajo realizados con Obeo Designer están formados por diferentes tipos de editores: diagramas, tablas o árboles [9]. Los editores se ajustan a las necesidades del dominio; la herramienta obeo designer permite a los desarrolladores crear los diagramas de forma rápida.

2.5.3 El lenguaje de transformación ACCELEO

Acceleo, es una tecnología basada en plantillas que permite crear generadores de códigos personalizados. Acceleo recibe una fuente de datos en formato EMF y los transforma en cualquier tipo de código fuente [46], el cual va ser utilizado en un dominio específico como es el caso de código matlab para el funcionamiento de sistemas SCADA.

Acceleo ofrece ventajas como:

- alta capacidad de personalización
- interoperabilidad

- inicio fácil

2.5.4 Herramienta de prototipado Figma

Figma es una herramienta de prototipado basado en la web, su principal ventaja es que está disponible en Internet y se puede trabajar de forma colaborativa en un mismo proyecto. Usando Figma, se puede diseñar cualquier cosa como una interfaz de usuario para páginas web y diseño, imágenes de gráficos vectoriales, etc. Figma es muy flexible y su curva de aprendizaje es baja [8].

2.5.5 Lenguajes de implementación de sistemas SCADA

Los sistemas SCADA permiten controlar la configuración y parámetros del hardware de un sistema automatizado. Dichos sistemas están desarrollados en lenguajes de programación que manejan datos en tiempo real. Además las formas de comunicación entre las puertas de enlace de las terminales RTU y los dispositivos IOT son complejos, requieren de lenguajes y librerías que soportan el manejo de los enlaces. Existen varios lenguajes de programación para desarrollar los sistemas SCADA, a continuación, se describen algunos de ellos.

Gráfico Ladder Logic (KOP) es un lenguaje de programación que está basado en la representación de diagramas de circuitos para programar. Se utiliza para programar un PLC (controlador lógico programable). Es un lenguaje gráfico de programación de PLC que expresa operaciones lógicas con notación simbólica usando diagramas de escalera, muy parecido a los rieles de un circuito lógico de relé tradicional [50].

Para la implementación del control de transmisión automático y el desarrollo de controles secuenciales de equipos electrónicos se utiliza el lenguaje de programación S7Graph, es un lenguaje gráfico de alto nivel para especificar el comportamiento secuencial [52]. El gráfico de función secuencial se origina a partir de Petri-net, una herramienta de modelado para describir sistemas distribuidos [51]. El gráfico de funciones secuenciales tiene muchas similitudes con “IEC 848”, un estándar para preparar gráficos de funciones para el proceso de control [51].

Un lenguaje de programación de código abierto para conectar dispositivos de hardware es NODE-RED [11], es una herramienta desarrollada por IBM. Este lenguaje trabaja con APIS y servicios online que permiten manejar de forma precisa datos de sensores desde Arduinos y pasar esa información a servidores para su posterior análisis.

Matlab es un lenguaje de programación ampliamente utilizado para desarrollar sistemas de control que trabajan con el protocolo de comunicación modbus. Este protocolo trabaja con una arquitectura maestro - esclavo para la sincronización de la comunicación entre el emisor y receptor. Además, Matlab permite la interacción con la herramienta OPC cliente, la cual permite la comunicación con un servidor del mismo estándar [12].

2.6 Trabajos relacionados

En esta sección se analizan diferentes soluciones, para implementar sistemas SCADA, mediante el enfoque de desarrollo dirigido por modelos, de los cuales se analiza su proceso de desarrollo, así como sus fortalezas y debilidades.

En [12] se desarrolla una solución tecnológica que permite construir sistemas PLC de forma automatizada mediante una arquitectura de software basada en modelos; dicha solución ayuda a la mejora de la producción, la optimización de procesos y la reducción de tiempos y costos que se utilizan en los sistemas de medición y control de procesos industriales (IPMCS). Este trabajo propone un modelo basado en componentes de la implementación de sistemas de control para el diseño de arquitecturas tanto de hardware y software. A partir del modelo de aplicación, descrito en el lenguaje de marcado (XML), se genera automáticamente el proyecto de automatización para cada PLC presente en la aplicación. El modelo propuesto posteriormente se transforma como un lenguaje de marcado que permite describir los sistemas de control industrial (ICSML). La implementación hace uso de esquemas XML y tecnologías de reglas de esquemas XML para implementar las reglas de composición del estilo arquitectónico. Los modelos de la solución propuesta en este trabajo arrojan como resultado archivos .xml, los cuales contienen la lógica de implementación de sistemas de control y monitoreo, para poder interpretar y transformar esos archivos generados se hace usos de herramientas con interfaz gráfica, que realicen esas tareas como la herramienta ISAGRAF que se muestra en la figura 2-4.

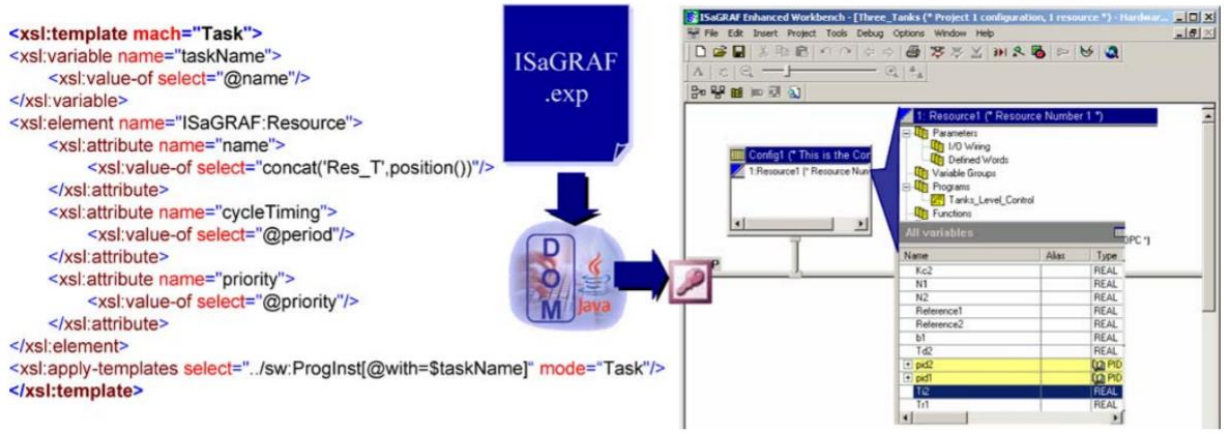


Figura 2-4 Generación del proyecto de automatización con el uso de la herramienta ISaGRAF [12]

La solución del trabajo [12] está limitada a que los modelos generan un código en archivos .xml y para poder transformarlos en un proyecto automatizado de sistemas de control y monitoreo, necesitan de herramientas externas con la capacidad de generar esos archivos en un archivo ejecutable.

En [13] se presenta la implementación de un entorno para generar sistemas embebidos con características de seguridad y protección, aplicando el lenguaje de modelado SysML-sec[1], la cual es una extensión del lenguaje de modelado SysML, como el medio para el diseño de dichos sistemas. El autor se basa en la metodología la cual está conformada por las siguientes etapas:

- Supuestos
- Requerimientos
- Ataques
- particionamiento
- Diseño de software
- Implementación

[1] SysML-Sec: Es un entorno de diseño seguro para sistemas embebidos basado en el lenguaje SysML. Con este entorno se pueden diseñar componentes tanto de hardware como de software.

Para llevar a cabo la implementación se hace uso de la herramienta TTool, la cual permite la edición de diagramas SysML[2] y también permite la generación de modelos de sistemas embebidos y además permite generar código de forma automática para probar la solución de los distintos problemas de seguridad que pueden presentar sistemas SCADA.

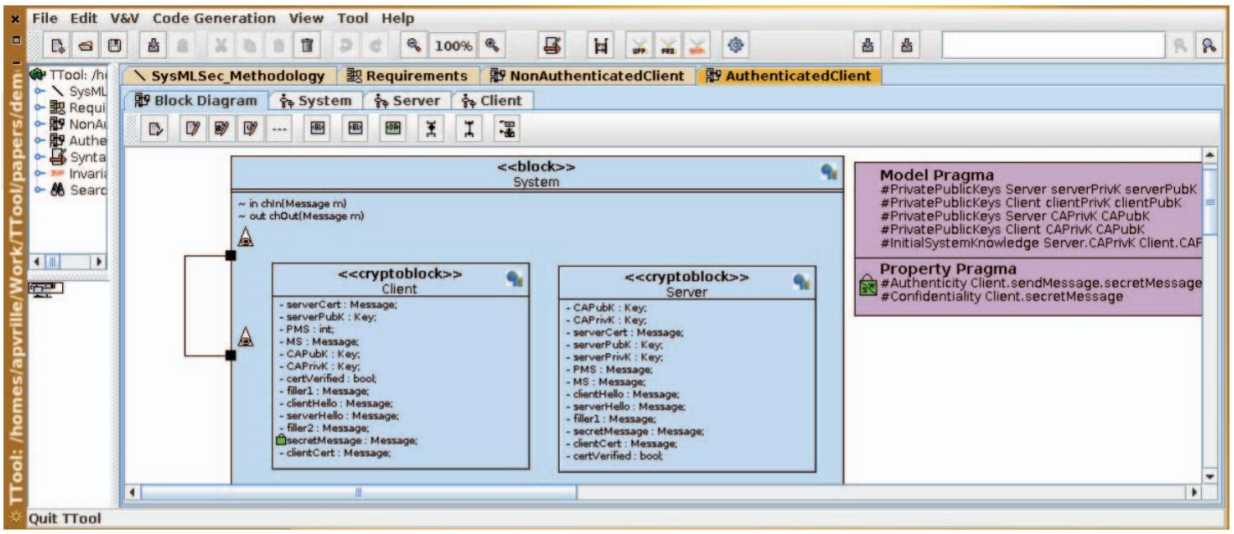


Figura 2-5 Modelo SysML en la herramienta Ttool de un modelo de verificación de seguridad de un sistema SCADA embebido [13]

El autor concluye que las metodologías de desarrollo de software se centran primero muy rápidamente en el tiempo de comercialización de estos sistemas, dejando las consideraciones de seguridad a segundo plano. Es por eso por lo que con la implementación del enfoque de desarrollo dirigido por modelos se pueden generar simulaciones de sistemas embebidos en los cuales se puede medir su seguridad antes de su lanzamiento comercial y por ende reducir la vulnerabilidad ante a diversos ataques de seguridad. El sistema desarrollado en el artículo [13] a través de los modelos que representan los sistemas de seguridad, mediante el uso de la herramienta Ttool (figura 2-5), permite generar un código de forma automatizado para

probar la seguridad de los sistemas SCADA. Sin embargo, no permiten generar un Sistema SCADA funcional en el cual se puede representar el monitoreo y control de datos en tiempo real.

[2] SysML: Es un lenguaje de modelado desarrollado por OMG (Object Management Group) como perfil de UML 2.0, el cual permite analizar, desarrollar, especificar , validar y probar sistemas tanto de software como hardware.

Mientras tanto en [14] se describe una implementación dirigida por modelos para un sistema de gestión de energía del cliente (CEM), el cuál actúa como un punto de conexión entre una smart grid y una smart home habilitando el intercambio de señales con el intercambio de energía en tiempo real. En la figura 2-6 se presenta el esquema entre el punto de conexión entre un smart grid y los servicios que pueden ser consumidos.

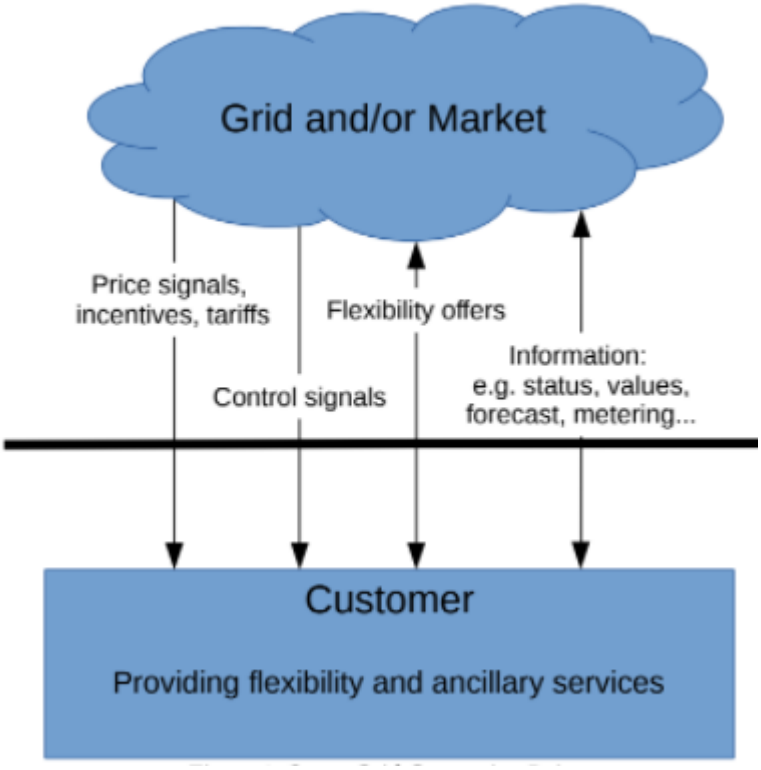


Figura 2-6 Punto de conexión de un Smart Grid [14]

El autor sostiene que uno de los principales requerimientos para la implementación de sistemas de gestión de energía es la interoperabilidad en dos niveles, semántico y sintáctico. Es por ello que un modelo común podría cubrir esos requerimientos. El software permite la implementación de una máquina de estado, recibiendo como entrada instancias de modelos y presenta salidas de esos modelos usando una base de datos en tiempo real. Los modelos son posibles editarlos de acuerdo con el lenguaje específico de dominio (DSL).

La ventaja de este enfoque de desarrollo de software está en la posibilidad de desarrollar productos estandarizados integrando el conocimiento que se encuentra en los modelos, los cuales son definidos por un estándar, con lenguaje de dominio específico generado

automáticamente y la debilidad que presenta es que no puede aplicarse a dominios diferentes en sistemas de gestión de energía, además no ofrece un editor gráfico que soporte el diseño del sistema SCADA y que facilite el trabajo al desarrollador de sistemas SCADA.

En cuanto a la arquitectura descrita en el artículo [14] se basa en una transformación bidireccional que engloba dos metamodelos de datos. Instancias de esos dos modelos son tomadas como entradas de la aplicación y producirán nuevas salidas de modelos de datos.

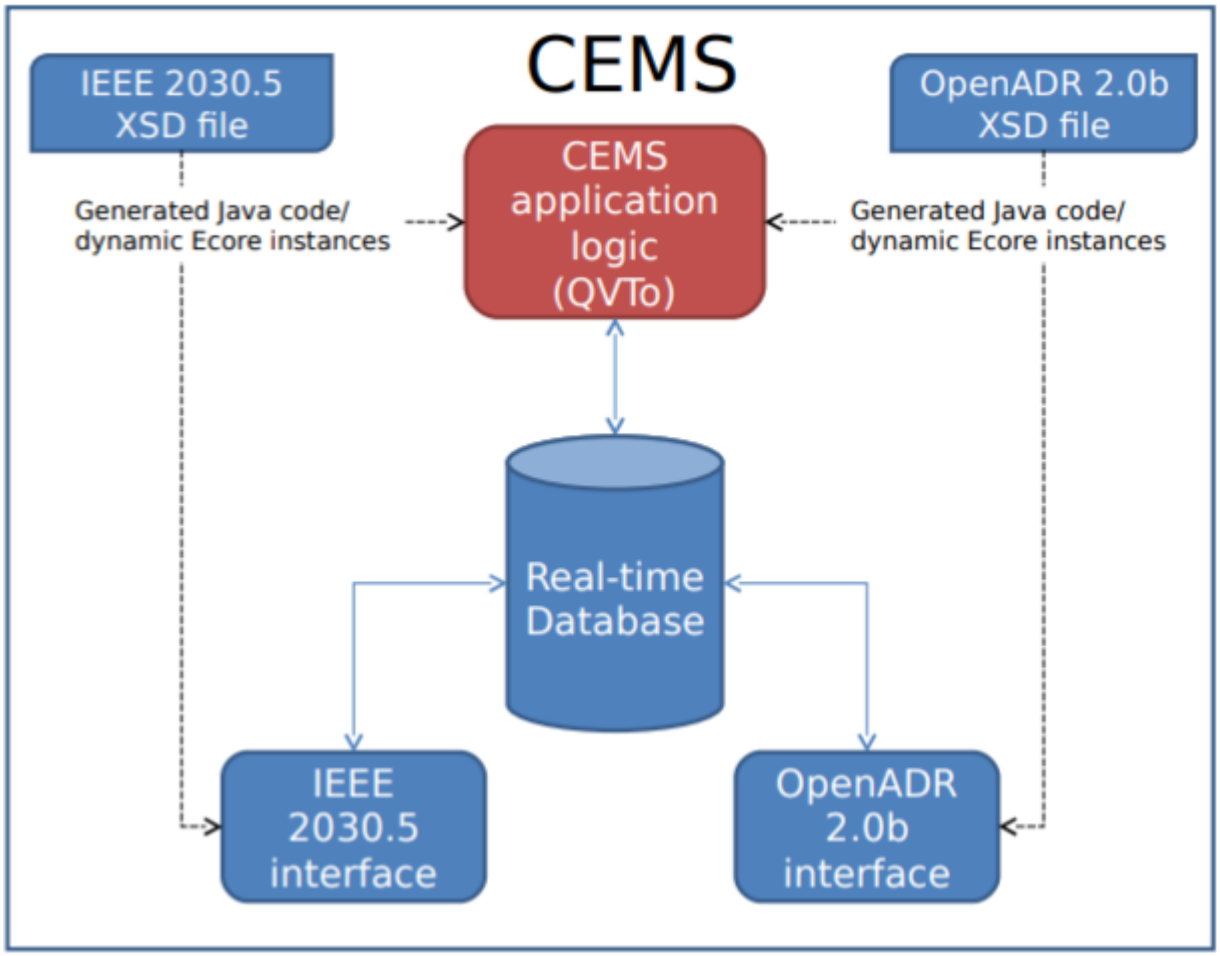


Figura 2-7 Implementación dirigida por modelos de CEMS [14]

El autor concluye que las principales ventajas de este enfoque de desarrollo de software se dan es la capacidad para especificar la lógica de la aplicación de una forma abstracta, usando lenguajes que son automáticamente derivador de los modelos de datos iniciales o metamodelos. Pero la solución desarrollada no genera scripts de implementación que puedan ser directamente ejecutados y que puedan cumplir con las necesidades de un

sistema de control y monitoreo. Si no que necesitan de una transformación adicional para generar código fuente que esté listo para desplegarlo en un entorno de ejecución.

El trabajo presentado en [15] describe un método de desarrollo de software para la generación automática de controladores PLC. El método parte de modelar el comportamiento deseado de un sistema de una máquina de estados, la cual posee la capacidad para medir el tiempo y termina con un programa completo escrito en un lenguaje de diagrama llamado ladder[3]. El modelo es formal, pero legible y se puede verificar con los requisitos de comportamiento y seguridad. La conversión del modelo en un programa se realiza automáticamente, lo que reduce la necesidad de desarrollar cada modelo. El proceso de convertir una especificación en un código de programa se define formalmente, utilizando un modelo de máquina del tiempo de estados finitos. El método tiene las siguientes ventajas:

- Especificación de requisitos gráficos, basada en un subconjunto de diagramas de estado UML.
- Modelo formal de la especificación con significado y comportamiento formalmente definidos.
- Definición formal del proceso de conversión.
- El potencial para el análisis formal utilizando una lógica temporal del cálculo de duración.

El programa generado consta de una secuencia de expresiones booleanas, generada por una herramienta automática desde un estado, diagrama, o un conjunto de diagramas de estado, que definen con mayor detalle un programa para un PLC.

[3] Ladder: lenguaje de programación gráfico muy popular dentro de los autómatas programables debido a que está basado en los esquemas eléctricos de control clásicos, se emplea para los controladores lógicos programables (PLCs) estandarizados con IEC 61131-3.

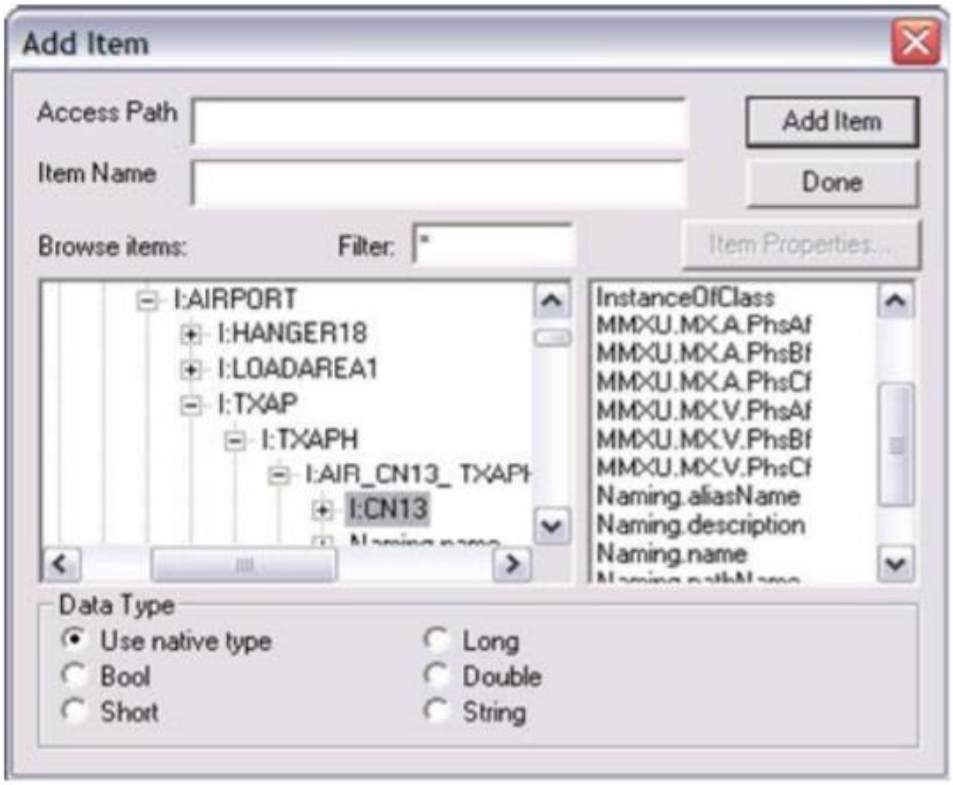


figura 2-9 GUI De la navegación del modelo de datos [15]

Finalmente, [16] se centra en el campo de la calidad de la energía y su monitoreo, y propone un modelo de desarrollo del sistema mediante la arquitectura MDA. La arquitectura del sistema parte del metamodelo de MDA y técnicas de mapeo. El metamodelo es expresado preferiblemente con tecnologías OMG, como Meta Object Facility (MOF), Common Metamodelo de almacén (CWM) o lenguaje de modelado unificado (UML). La figura 2-10 ilustra cómo encajan los estándares OMG en MDA y presenta los diferentes escenarios en los que se puede aplicar.

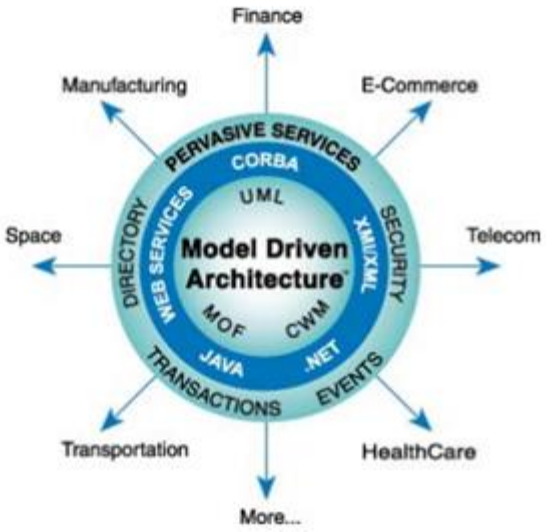


Figura 2-10 Arquitectura dirigida por modelos [6]

Mediante el enfoque MDA se puede observar una clara separación de PIM, PSM y la implementación específica de la plataforma. En la figura 2-11 se detalla dicha separación para el desarrollo de PQMS (Power Quality Management System). En la capa PIM (Platform Independent Model) se encuentran los componentes lógicos basados en estándares generales como IEC 61850, un estándar para la automatización de estaciones en el cual se definen un conjunto de protocolos para la comunicación de dispositivos pertenecientes a una gama de subestaciones.

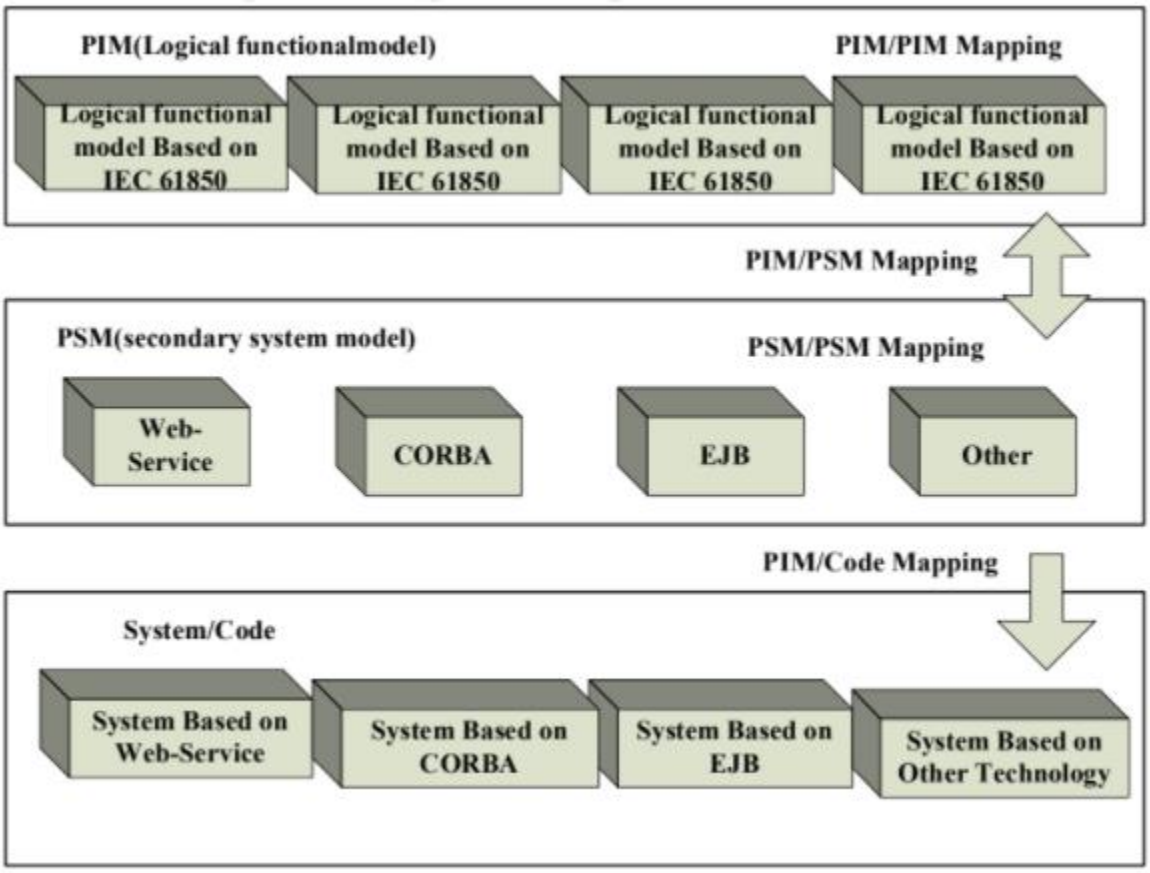


Figura 2-11 Enfoque MDA para el proceso de desarrollo de PQMS [16]

En la capa PSM (Platform Specific Model) ya se encuentran componentes a un nivel más específico del tipo de tecnologías, como servicios web, CORBA , etc.

La última capa de la arquitectura de este sistema es la de codificación, en la cual se implementa el código fuente de acuerdo con los modelos de la capa superior como por ejemplo el código fuente de un servicio web.

La arquitectura, ilustrada en la Figura 2-12. consta de una parte GUI para el modelado de PQMS (los PIM se guardan en el repositorio de modelos en IEC 61850), un conjunto de módulos convertidores de modelos y un conjunto de módulos que utilizan PSM, que son probados y finalmente generan el código fuente automáticamente.

La solución de este artículo se basa en capas en las cuales se recibe un artefacto de software como entrada, pasa por uno o varios componentes de software y retorna otro artefacto de software como salida.

En la primera capa se recibe como entrada un archivo .xml el cual representa las configuraciones basadas en el estándar IEC 61850, en la segunda capa (Capa de conversión

de modelos) el componente Model Converter se encarga de convertir el archivo XML en un modelo de texto , para realizar la conversión se utiliza un editor basado en texto. En la tercera capa (Capa Plataforma Independiente de modelos) se transforma el modelo, resultado de la segunda capa, en un código fuente específico de la tecnología.

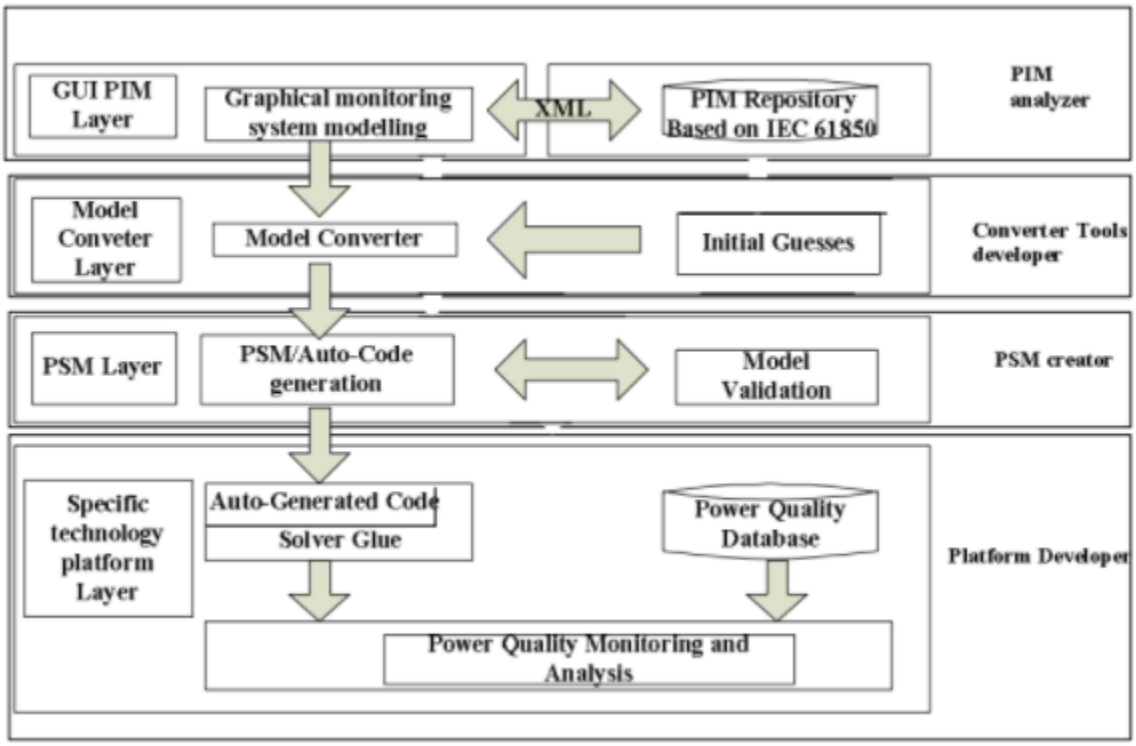


Figura 2-12 Estructura de PQMS con MDA [16]

Este método se ha aplicado al desarrollo de un sistema de control de la calidad de energía para ZheGan Electric Railway.

Una debilidad de esta propuesta tecnológica es que la fase de diseño de los modelos está basado en una estructura de árbol, lo cual dificulta la creación de la solución porque no se puede crear modelos de forma sencilla e intuitiva.

APROXIMACIÓN TECNOLÓGICA PARA LA GENERACIÓN DE SISTEMAS SCADA

En esta sección se presenta la aproximación tecnológica que da soporte al diseño e implementación de sistemas SCADA. Esta aproximación consiste en un prototipo completamente funcional de una infraestructura de software construida durante el desarrollo de este trabajo de titulación.

3.1 Arquitectura de la aproximación tecnológica

La infraestructura software ha sido construida para soportar el diseño e implementación de sistemas SCADA siguiendo un enfoque de desarrollo dirigido por modelos.

En la figura 3.1, se presenta la relación entre el ciclo de vida de un sistema SCADA, descrito utilizando SPEM; y la arquitectura de la infraestructura software construida para soportar las actividades del ciclo de vida, descrita utilizando UML. Cabe aclarar que la definición del ciclo de vida de sistemas SCADA está fuera del alcance de este trabajo de titulación. Sin embargo, se han identificado las actividades genéricas de este ciclo de vida (e.j., Diseño de Sistemas SCADA, construcción y despliegue de Sistemas SCADA, Uso de Sistemas SCADA) [47] para ilustrar cómo los componentes de la infraestructura software creada soportan un proceso genérico de creación de sistemas SCADA.

La infraestructura de software incluye: i) un *DSL de Sistemas SCADA* que, durante la actividad de *Diseño de Sistemas SCADA*, facilita la creación de *Modelos de Sistemas SCADA*, abstrayendo al operador de conocimientos específicos de protocolos o plataformas de modelado y simulación ii) Un *Motor de Transformación* que, durante la actividad de *construcción y despliegue de Sistemas SCADA*, genera *Scripts de Implementación* que implementan el diseño de acuerdo a un lenguaje de programación que soporte la plataforma de modelado y simulación (e.j., MATLAB), o de acuerdo a un protocolo específico (e.j., MODBUS).

La figura 3-1 ilustra los componentes involucrados en la arquitectura de la infraestructura software y las relaciones entre ellos. La infraestructura software provee:

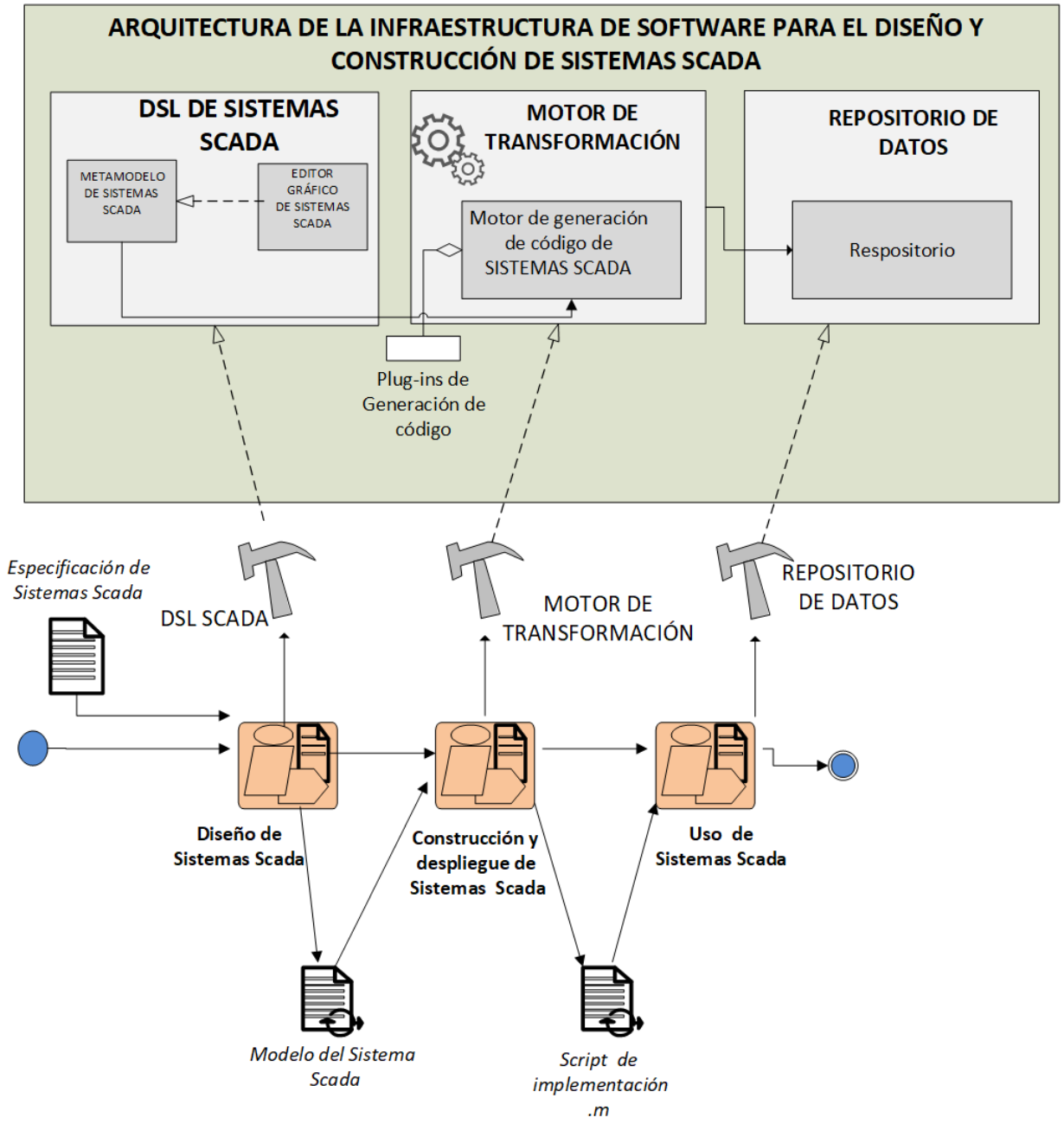


Figura 3-1 Arquitectura de la Infraestructura software para la generación de sistemas SCADA

- Un DSL de Sistemas SCADA, el cual permite definir los conceptos y reglas para el diseño de sistemas SCADA.
    - Metamodelo de sistemas SCADA: Permite definir la sintaxis abstracta, conceptos y relaciones entre conceptos durante el diseño de sistemas SCADA.

- Editor gráfico de sistemas SCADA: Define las instancias del metamodelo, es decir la sintaxis concreta del DSL.

- Motor de transformación de código: En este componente se convierten los modelos independientes de la plataforma a texto (M2T). Es decir, los *Modelos del Sistema SCADA* a *Scripts de Implementación.* Esta versión de la infraestructura incluye un plug-in que genera un archivo con el código fuente de una aplicación en matlab, la extensión del archivo es ".m". Dicho código fuente representa un sistema SCADA funcional.

La infraestructura software ha sido implementada como un conjunto de plug-ins de Eclipse Modelling Framework. Eclipse define Ecore, una implementación del estándar Essential MOF (EMOF) [11]. Esta solución emplea los niveles M1 y M2 previstos en el capítulo 2, que corresponden a los niveles de modelado de software.

3.2 Aproximación tecnológica de soporte al diseño de Sistemas SCADA

La infraestructura software planteada en el presente trabajo de titulación da soporte a la etapa de diseño de sistemas SCADA (figura 3-1) mediante el DSL. Los componentes principales del DSL son: metamodelo y editor gráfico.

3.2.1 Lenguaje para la descripción de Sistemas SCADA (metamodelo).

El metamodelo contiene todos los conceptos y las relaciones entre los elementos que conforman un sistema SCADA, es decir refleja la abstracción de los componentes necesarios para diseñar un sistema SCADA. De esta manera el metamodelo será el componente base para el diseño de sistemas SCADA, ya que contiene todos los conceptos necesarios para describir el diseño del sistema.

El metamodelo se describe utilizando MOF que es un lenguaje de meta-modelado que es parte de la OMG. Para la construcción del metamodelo se utilizó el IDE Obeo designer.

El Metamodelo de sistemas SCADA (Figura 3-2) facilita la descripción de los elementos que implementan un sistema SCADA. Como se puede observar en la Figura 3-2, el Metamodelo de sistemas SCADA agrupa los elementos de un sistema SCADA en dos tipos, *API* y *Panel*, los cuales abarcan subelementos que permiten describir tanto la parte visual (interfaz gráfica) y la parte lógica.

La metaclase API del metamodelo representa un equipo del laboratorio de micro red el cual es una especie de middleware, ya que la comunicación entre un equipo electrónico y un sistema SCADA tiene que pasar por esta API para luego dirigirse hacia el equipo electrónico correspondiente. La *API* contiene una dirección ip para permitir la comunicación entre los equipos electrónicos y el o los sistemas SCADA. De esa forma cualquier comunicación hacia los equipos tiene que pasar primero por la API.

La metaclase *API* del metamodelo está conformado por las siguientes metaclases:

- *Equipo:* Representa un equipo electrónico del laboratorio y contiene atributos necesarios para que un equipo pueda recibir comandos para su control y para el monitoreo de datos. Los atributos *direccionIP* y *puerto* permiten la comunicación del equipo con el sistema SCADA.
  Un equipo está conformado por dos subelementos: *Tabla de operaciones y Variable*.

- *Tabla de operaciones:* Representa el conjunto de operaciones que un equipo electrónico puede realizar. Una tabla de operaciones tiene un nombre y una observación.
- *Variable: Address* representa un valor de un registro del equipo electrónico. Mediante el atributo se pueden escribir (write) o leer datos (read) del equipo.
- *Operación:* Representa las acciones que puede realizar un equipo electrónico como, por ejemplo: marcha, arranque, reinicio etc.

La metaclase *Panel* del metamodelo representa la interfaz gráfica del sistema SCADA, la cual contiene elementos visuales y de interacción tales como:

- *Navegación*: Es una metaclase visual que representa la navegación entre paneles, es decir permite cambiar de un panel hacia otro.
- *Elemento*: Es una metaclase visual que representa un elemento dentro del *Panel*. Un elemento visual se divide en tres tipos: elementos de decoración, visualización y control, los cuales a nivel de metamodelo, entendidas como clases son interfaces, que serán implementados por subelementos visuales.
- *Visualización*: Entendida como una metaclase, es una interfaz que representa elementos dinámicos que obtienen datos de los equipo del sistema SCADA tal como: un tanque, un label o texto, una imagen de un equipo.
- *Decoración*: Entendida como una metaclase es una interfaz que representa elementos estáticos de un sistema SCADA como: texto, líneas, imágenes.
- *Control*: Entendida como una metaclase es una interfaz que representa elementos que permiten enviar comandos para controlar los equipos del sistema SCADA, dichos elementos de control pueden ser accionadas por la interacción del usuario o

pueden ser activados en un tiempo programado, tales elementos son: entradas de texto (*inputText*),*Button*,*Trigger.*

- *Button*: Entendida como una metaclase es una interfaz que representa botones que permiten enviar acciones hacia los equipos del sistema SCADA. Estas acciones pueden ser envío de valores constantes o envío de valores específicos. Los botones son de dos tipos: *BottonValorConstante* y *BottonAValorActual*.

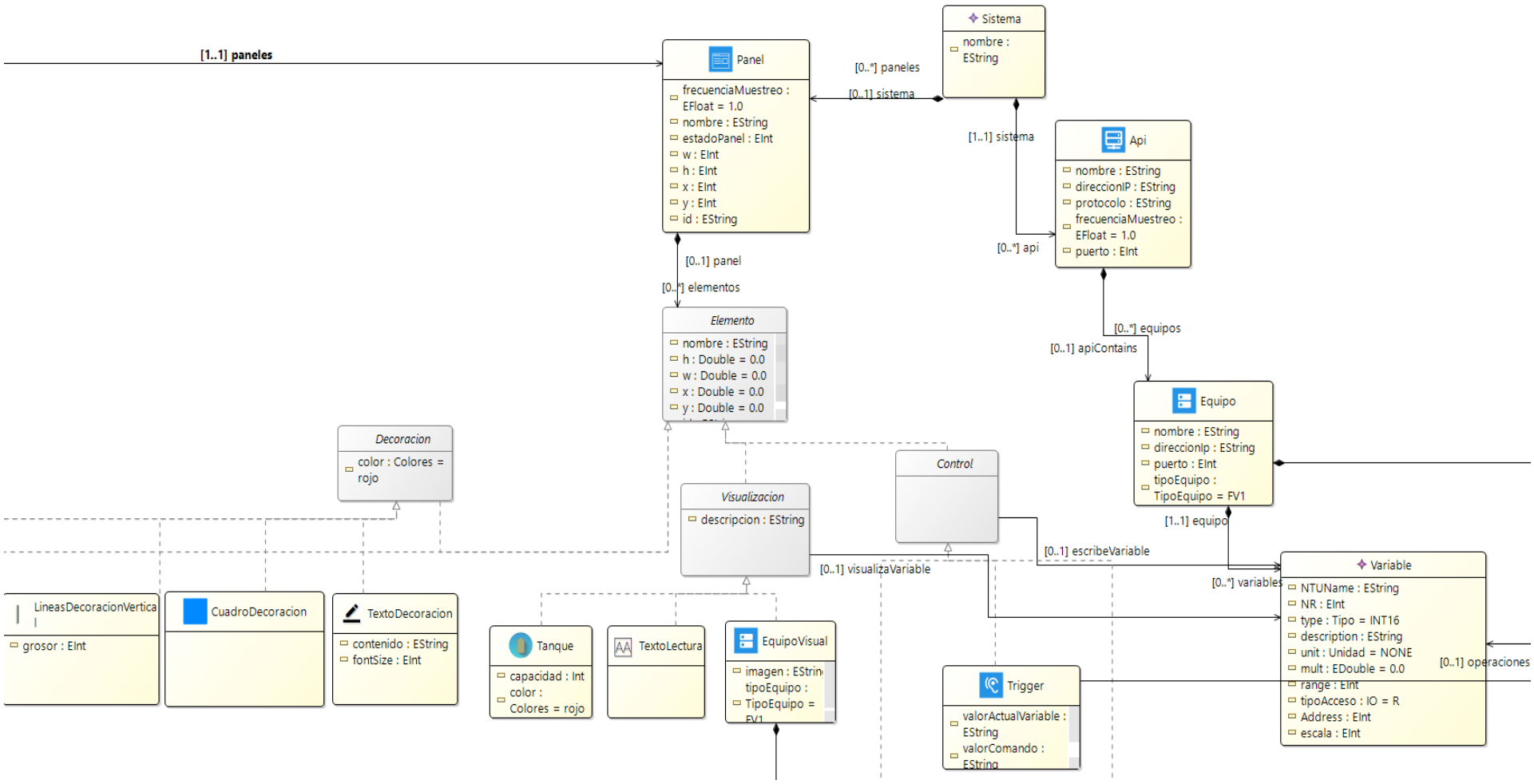


Figura 3-2 Extracto 1 de Metamodelo de sistemas SCADA

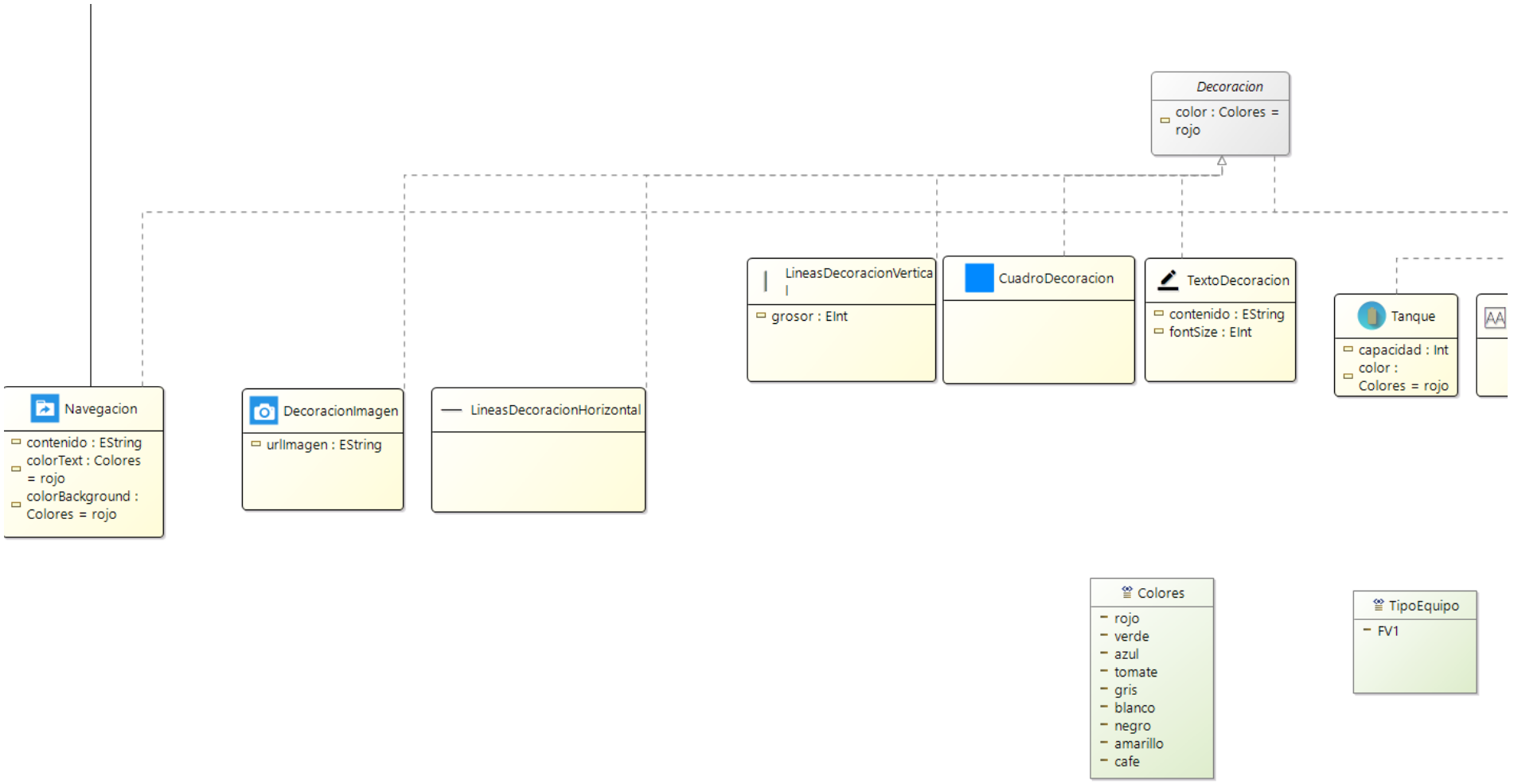


Figura 3-3 Extracto 2 de Metamodelo de sistemas SCADA

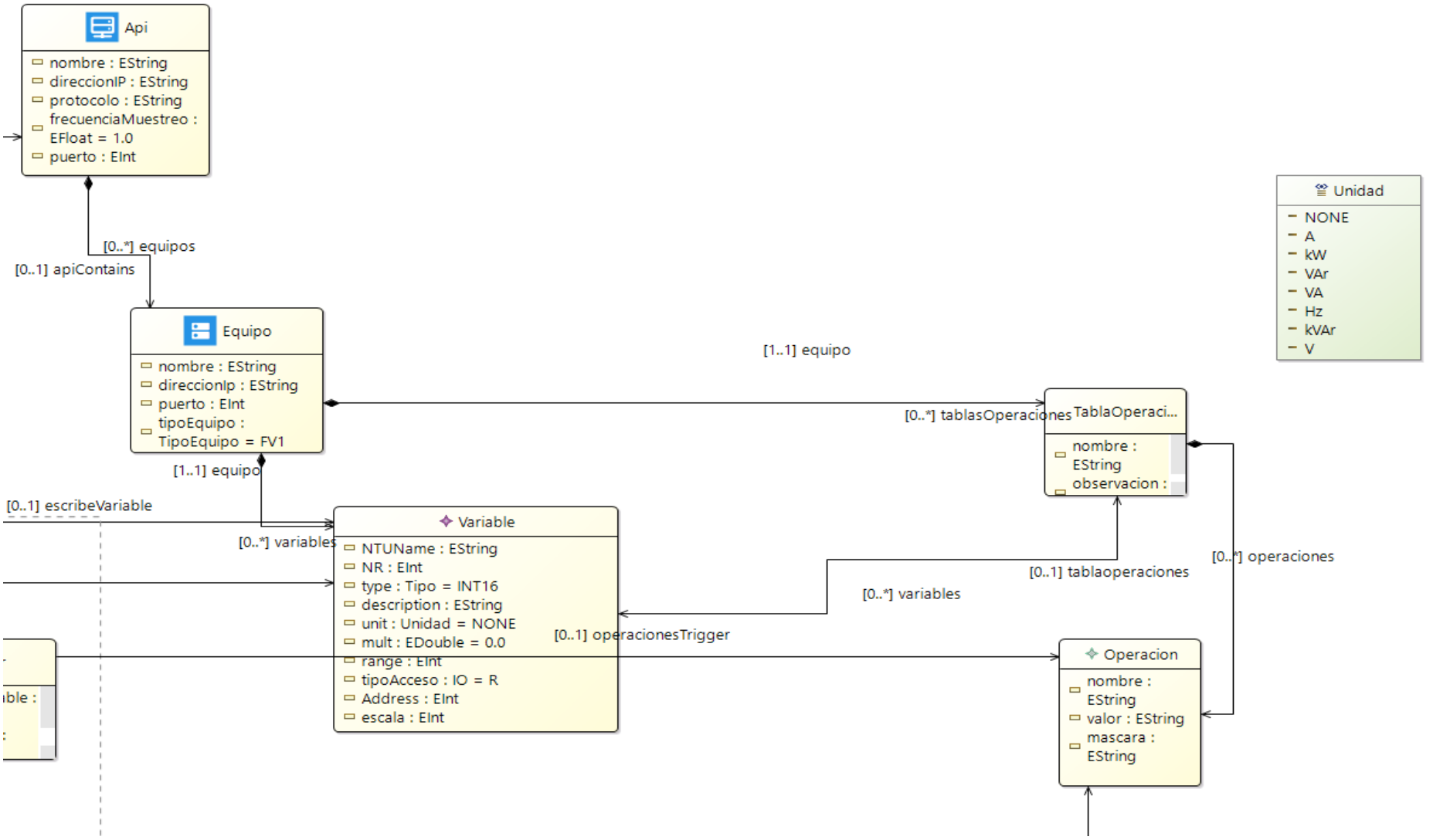


Figura 3-4 Extracto 3 de Metamodelo de sistemas SCADA

3.2.2 Editor gráfico para la descripción de Sistemas SCADA

El editor gráfico para la descripción de Sistemas SCADA desarrollado se implementó en varias etapas. En donde, en una primera etapa se diseñaron prototipos (Anexos 1 y 2) de baja fidelidad de las interfaces gráficas de usuario correspondientes a las funcionalidades del editor menos entendidas durante el diseño de la infraestructura software. En una segunda etapa, una vez entendidas claramente las funcionalidades que debía proveer el editor gráfico se procedió a la construcción de un prototipo funcional del editor, haciendo uso para ello del IDE Obeo Designer.

3.2.2.1 Prototipado de baja fidelidad

En las etapas tempranas del diseño del editor gráfico existieron funcionalidades que no estaban claramente definidas o entendidas (e.j., descripciones de interfaces de programación de aplicaciones, equipos, enlaces, interacciones) por lo que se crearon prototipos de baja fidelidad haciendo uso de la herramienta de generación de prototipos Figma[4]. Se consideraron dos alternativas de interfaces gráficas de usuario para describir elementos y configuraciones de sistemas SCADA. Ver Anexos 1 y 2.

3.2.2.2 Prototipo funcional del Editor de Sistemas SCADA

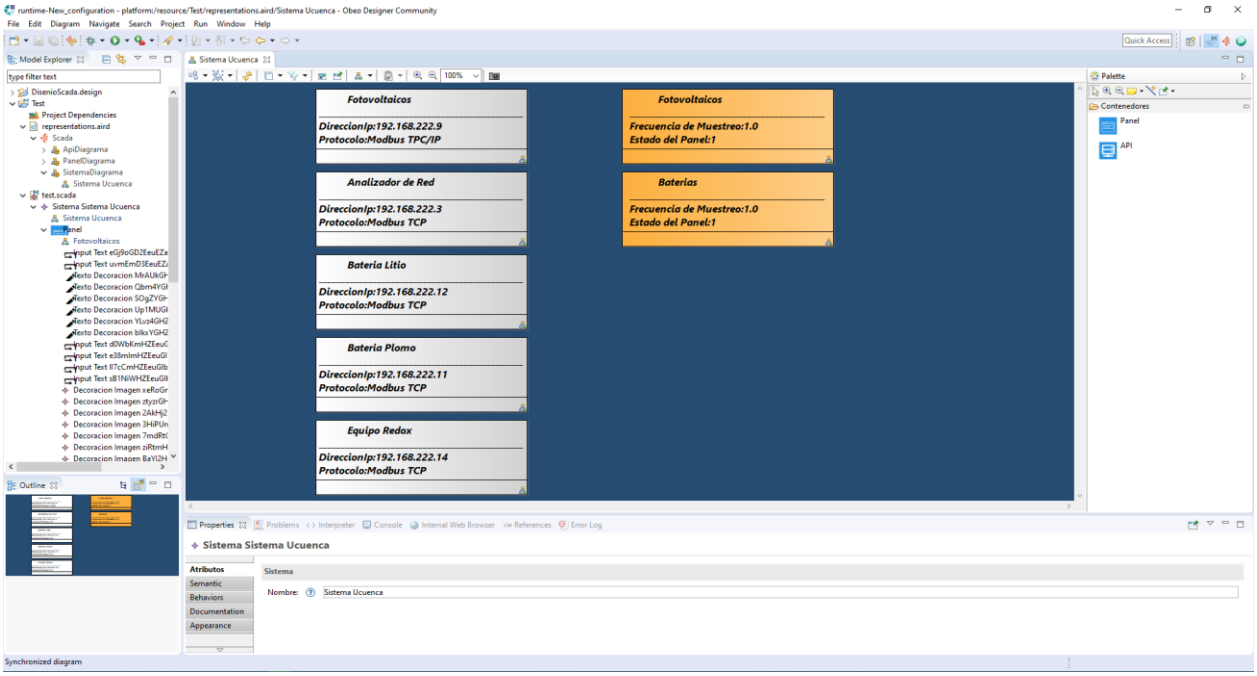


Figura 3-5 Editor de sistemas SCADA(Interfaz de Sistema SCADA)

[4] Figma es un editor de gráficos vectorial y una herramienta de generación de prototipos, principalmente basada en la web, con características off-line adicionales habilitadas por aplicaciones de escritorio en macOS y Windows.

El editor de sistemas SCADA provee a los técnicos en operaciones especificar la interacción de los elementos del lenguaje específico de dominio para sistemas SCADA a través de una interfaz intuitiva que le permite describir los entornos SCADA mediante los elementos del DSL los cuales son etiquetados con diferentes iconos. Esto ayuda a los técnicos en operaciones en su tarea de especificación, evitando que deba navegar entre los atributos de los elementos del DSL para conocer su tipo.

En la tarea de especificación, el técnico en operaciones utiliza el editor para crear APIs arrastrando y soltando el contenedor de API, El API guarda la información de conexión con los PLCs y es contenedor de los Equipos los cuales contienen los atributos de monitoreo que se almacenan en la base de datos. Una vez que el técnico de operaciones establece las APIs necesarias para el monitoreo, procede a definir la interfaces gráficas que en este caso son representados como paneles. Para definir los contenedores de paneles, el técnico crea un elemento Panel arrastrando y soltando en la vista de Sistema, mostrada en la Figura 3-5, la cual es una de la vista del editor gráfico para sistemas SCADA, continuación en la construcción del Panel incluye las características de su configuración, y posteriormente agrega los elementos visuales que se presentan en un menú y están disponibles en interfaz de Panel.

3.3 Aproximación tecnológica de soporte a la construcción y despliegue de Sistemas SCADA

La infraestructura software planteada en el presente trabajo de titulación da soporte a la etapa de construcción y despliegue de sistemas SCADA (figura 3-1) mediante el Motor de Transformación. Sus componentes principales son: Motor de transformación de código de sistemas SCADA.

Por otro lado, la tarea de generación de código de implementación es soportada por transformaciones M2T que son parte del componente de la infraestructura software Motor de Transformación.

3.3.1 Motor de transformación

Este motor de transformación compuesto de plug-ins de transformaciones M2T creados en Acceleo y que tienen la responsabilidad de transformar un *Modelo del Sistema SCADA* producido durante el Diseño del Sistema SCADA en *Scripts de Implementación* de acuerdo a una plataforma de modelado y simulación específica o a un protocolo específico. Las

transformaciones M2T hacen uso de OCL para definir queries y consultas para la extracción de la información de un *Modelo del Sistema SCADA.* La Figura 3-6 muestra un extracto de la transformación M2T que ha sido implementada, el cual inicializa las variables globales de conexión modbus y variables usadas para el estado de la interfaz, de acuerdo con la plataforma de modelado y simulación MATLAB versión 17, toda la transformación se adjunta en el Anexo 3.

```
[file (deleteSpace(aSistema.nombre)+'.m', false, 'UTF-8')]
    function SistemaUcuenca
        [for (api : Api | aSistema.api)]
            global mb[deleteSpace(api.nombre)/] ;
            global timerAPI[deleteSpace(api.nombre)/] ;
        [/for]
            hs = addcomponents; % Make figure visible after adding comp
            hs.fig.Visible = 'on';
            asignarTimers(hs);
            conectarApis();
            iniciarTimers();
            function hs = addcomponents   % add components, save handle
                posicion= ['['/] 0 -1000 50 30 [']'/]; % posicion y siz
                [for (p : Panel | aSistema.paneles)]
                    [desplazarDerecha(p)/]
                [/for]
                [for (p : Panel | aSistema.paneles)]
                    [rotarEjes(p)/]
                [/for]
                hs.fig = figure('Name','Sistema Balzay' ,'NumberTitle',
            [for (p : Panel | aSistema.paneles)]
                hs.panel[deleteSpace(p.nombre)/]=uipanel('Parent',hs.fi
            [/for]
            [for (p : Panel | aSistema.paneles)]
                [for (elem : Elemento | p.elementos)]
                [if (isEquipoVisual(elem))]
                    [let var : EquipoVisual =getEquipoVisual(elem) ]
```

Figura 3-6 Código OCL (Define la creación de archivo.m)

La estructura del Script de Implementación está conformada por:

3.3.1.1 Función principal

La función principal cumple con el rol de control o la orquestación de las funcionalidades (e.j., *asignarTimer*, *conectarApis*, *iniciarTimers*) de la interfaz del sistema SCADA, permitiendo al usuario que amplíe la funcionalidad del sistema SCADA, además de ejecutar la función *addcomponents*, que inserta el código que posiciona los objetos gráficos obtenidos de los diferentes elementos del modelo y los asocia con eventos en la interfaz.

También realiza las instancias de las variables globales del mapa modbus (e.j., mbFotovoltaicos, etc) y timers de las APIs insertadas dentro de un panel, con el editor Gráfico, Figura 3-7 muestra la definición de la variables globales de conexión modbus de Fotovoltaicos, Bateria Litio, Batería Plomo que son usadas para el monitoreo de los valores de los equipos correspondientes.

```
1     function SistemaUcuenca
2         global mbFotovoltaicos ;
3         global timerAPIFotovoltaicos ;
4         global mbAnalizadordeRed ;
5         global timerAPIAnalizadordeRed ;
6         global mbBateriaLitio ;
7         global timerAPIBateriaLitio ;
8         global mbBateriaPlomo ;
9         global timerAPIBateriaPlomo ;
10        global mbSuperCondensadores ;
11        global timerAPISuperCondensadores ;
12        hs = addcomponents; % Make figure visible after adding components
13        hs.fig.Visible = 'on';
14        asignarTimers(hs);
15        conectarApis();
16        iniciarTimers();
17        function hs = addcomponents    % add components, save handles in a struct
349   end
```

Figura 3-7 Función Principal (SistemaUcuenca)

3.3.1.2 Funciones Secundarias

Este trabajo de tesis incluye diferentes funcionalidades que son implementadas dentro del sistema mediante funciones secundarias encargadas de modificar el estado del Sistema SCADA, y que se puede ubicar en cualquier parte dentro del script de implementación, encargándose principalmente de funciones específicas en el sistema SCADA, las cuales son:

A. **Función asignarTimers**: encargada de construir las estructuras de datos (e.j., Diccionarios) que almacena las variables a monitorear, asignando un timer(funciones de monitoreo), podemos visualizar parte del código en la Figura 3-8 de la función.

```
function asignarTimers(interfaz)
        global timerAPIFotovoltaicos;
        global mbFotovoltaicos;
        global timerAPIAnalizadordeRed;
        global mbAnalizadordeRed;
        global timerAPIBateriaLitio;
        global mbBateriaLitio;
        global timerAPIBateriaPlomo;
        global mbBateriaPlomo;
        global timerAPISuperCondensadores;
        global mbSuperCondensadores;
        diccionarioEquipoFV1Fotovoltaicos=conta
        diccionarioEquipoFV2Fotovoltaicos=conta
        diccionarioEquipoFV3Fotovoltaicos=conta
        diccionarioEquipoAR1AnalizadordeRed=con
        diccionarioEquipoBateriaLitioBateriaLit
        diccionarioEquipoBateriaPlomoBateriaPlo
```

Figura 3-8 Función asginarTimers

B. **Función Leer_Datos_Api:** Realiza la función de extraer los datos de la APIs pertenecientes a la infraestructura de planta, mediante el uso de la conexión modbus, y asignarle los valores extraídos a un objeto de la interfaz. En la Figura 3-9 se puede ver el código de una de las funciones de lectura de las APIs pertenecientes a la infraestructura de planta.

```
function leer_Datos_Fotovoltaicos(interfaz,lista_Datos)
    nombreApi=lista_Datos{1};
    conexionModbus=lista_Datos{2};
    timer_Api=lista_Datos{3};
    for k=1:1:length(lista_Datos{4})
       direcciones_Memoria=values(lista_Datos{4}{k});
       for m=1:1:length(direcciones_Memoria)
           direccion_Mem=direcciones_Memoria{m};
           nombreVar = direccion_Mem{3};
           tipoVar = direccion_Mem{4};
           nombreEquipo = direccion_Mem{5};
           posicion_registro= direccion_Mem{1}-400000;
           registro = read(conexionModbus,'holdingreg',posici
           hijos= allchild(interfaz);
           for i=1:1:size(hijos)
                 if class(hijos(i))=="matlab.ui.control.UICont
                      tag = strsplit(hijos(i).Tag,'-'); % div
                      if length(tag)>1
                          dir_mem=tag{1};
                          api_nom=tag{2};
                          if dir_mem==string(int2str(direccio
                             set(hijos(i),'string',registro/d
                             conn = database('Tesis','','');
                             query = strcat('INSERT INTO regi
                             disp(query);
                             execQuery = exec(conn,query);
                             execQuery = fetch(execQuery);
                          end
                      end
                 end
           end
       end
    end
end
```

Figura 3-9 Función Leer_Datos_Api

C. **Función conectarModbus:** Encargada de establecer las conexiones de comunicaciones (e.j.,Modbus TCP) con los equipos proveedores de datos(e.j. PLC), los cuales se encuentran la infraestructura de planta, La comunicación se establece mediante una interfaz de programación de aplicaciones la cual se encuentra en el entorno de implementación del sistema SCADA (la micro red del laboratorio de Balzay) en el Figura 3-10 podemos ver su código.

```
function m = conectarModbus(ip, port)
    try
        m = modbus('tcpip',ip,port,'Timeout',1);
    catch
        % En caso de no lograr conectarse devuelve 0
        m = 0;
    end
end

function callBackBottonNavegacion(hObject,event,fig,panel)
    if panel == "Monitor de Variables"
        MonitordeVariables();
    else

    hijos= allchild(fig);
    for i=1:1:size(hijos)
         if class(hijos(i))=="matlab.ui.container.Panel" && panel~=hijos(i)
            hijos(i).Visible='Off';
         end
         if class(hijos(i))=="matlab.ui.container.Panel" && panel==hijos(i)
             hijos(i).Visible='On';
         end
    end

    end

end
```

Figura 3-10 Función Leer_Datos_Api

D. Función conectarApis: Realiza la instancia de la conexión con las APIs donde los parámetros necesarios para realizar la conexión son el puerto y la dirección ip de la Api a la cual queremos conectar. En la Figura 3-11 se muestra el código de la función conectarApis que inicializa las conexiones modbus (Fotovoltaicos, Bateria Litio, Batería Plomo) con sus respectivas direcciones ips y puertos .

```
function conectarApis(interfaz)
        global mbFotovoltaicos;
        global mbAnalizadordeRed;
        global mbBateriaLitio;
        global mbBateriaPlomo;
        global mbSuperCondensadores;
        mbFotovoltaicos = conectarModbus('192.168.222.9', 502);
        mbAnalizadordeRed = conectarModbus('192.168.222.3', 502);
        mbBateriaLitio = conectarModbus('192.168.222.12', 502);
        mbBateriaPlomo = conectarModbus('192.168.222.11', 502);
        mbSuperCondensadores = conectarModbus('192.168.222.14', 502);

end
```

Figura 3-11 Función ConectarApis

E. **Función iniciarTimers:** Encargada de verificar la conexión y ejecutar el monitoreo de los datos disponibles en los PLC, en un frecuencia determinada por los objectos de actualización del sistema SCADA(e.j., timers),parte del código de la función se puede ver en la Figura 3.13 y se encarga de verificar la conexión de fotovoltaicos, Batería de litio y Analizador de Red.

```
function iniciarTimers()
        global mbFotovoltaicos;
        global  timerAPIFotovoltaicos;
        if (mbFotovoltaicos~=0)
            start(timerAPIFotovoltaicos);
            msgbox('Conexion exitosa','Fotovoltaicos','warn');
        else
            msgbox('Error al conectar','Fotovoltaicos','error');
        end
        global mbAnalizadordeRed;
        global  timerAPIAnalizadordeRed;
        if (mbAnalizadordeRed~=0)
            start(timerAPIAnalizadordeRed);
            msgbox('Conexion exitosa','Analizador de Red','warn');
        else
            msgbox('Error al conectar','Analizador de Red','error');
        end
        global mbBateriaLitio;
        global  timerAPIBateriaLitio;
        if (mbBateriaLitio~=0)
            start(timerAPIBateriaLitio);
            msgbox('Conexion exitosa','Bateria Litio ','warn');
        else
```

Figura 3-12 Función iniciarTimers

3.3.1.3 Almacenamiento de información

El almacenamiento de la información es una función subyacente en los sistemas SCADA, es la encargada, durante el monitoreo y control del sistema almacenar en la de base datos (e.j., MySQL) del sistema, los valores leídos desde las variables definidas para los diferentes equipos del sistema SCADA . En la siguiente Figura 3-13 se muestra el código encargado de insertar la información mediante sql en la base de datos del sistema.

```
direccion_Mem=direcciones_Memoria{m};
nombreVar = direccion_Mem{3};
tipoVar = direccion_Mem{4};
nombreEquipo = direccion_Mem{5};
posicion_registro= direccion_Mem{1}-400000;
registro = read(conexionModbus,'holdingreg',posicion_registro,1,direccion_Mem{2});
hijos= allchild(interfaz);
for i=1:1:size(hijos)
      if class(hijos(i))=="matlab.ui.control.UIControl" && isprop(hijos(i),'tag')==true
           tag = strsplit(hijos(i).Tag,'-'); % dividimos el tag  (direccion-nombreApi)
           if length(tag)>1
               dir_mem=tag{1};
               api_nom=tag{2};
               if dir_mem==string(int2str(direccion_Mem{1})) && api_nom==string(nombreApi)
                  set(hijos(i),'string',registro/direccion_Mem{6});
                  conn = database('registosVariables','','');
                  query = strcat('INSERT INTO registro(api,nombre,valor,fecha,tipo,equipo)
                  disp(query);
                  execQuery = exec(conn,query);
                  execQuery = fetch(execQuery);
               end
           end
      end
end
```

Figura 3-13 Almacenamiento de información en la base de datos

3.3.1.4 Presentación de información

La presentación de información se realiza mediante una ventana de monitoreo, la cual está integrada a la interfaz de sistema SCADA y se encarga de manera jerárquica dar a conocer al usuario sobre el estado de los equipos (e.j., Fotovoltaicos, Batería de litio, Variable VDC) dentro de una infraestructura industrial (e.j., Micro Red de laboratorio de Balzay), los cuales se abstrajeron en los modelos del editor gráfico. Esta ventana extrae la información de un base de datos diseñada en MySQL mediante consultas SQL, las cuales llevan un rango de tiempo para la extracción de la información.

La ventana de monitoreo se invoca en el sistema SCADA mediante un nombre predefinido “Monitor de Variables” el cual se inserta en el elemento Navegación del editor gráfico. En la Figura 3-14 se muestra un ejemplo de tres botones de navegación el tercero pertenece al monitor de variables,

Figura 3-14 Botón de Navegación(Monitor de Variables)

En la siguiente Figura 3-15 se muestra la ventana del Monitor de variables la cual presenta una gráfica de datos de la variable voltaje de corriente continua (VDC) que pertenece al equipo fotovoltaico 1 y a la API fotovoltaico. La consulta de datos está establecida entre un rango de tiempo.

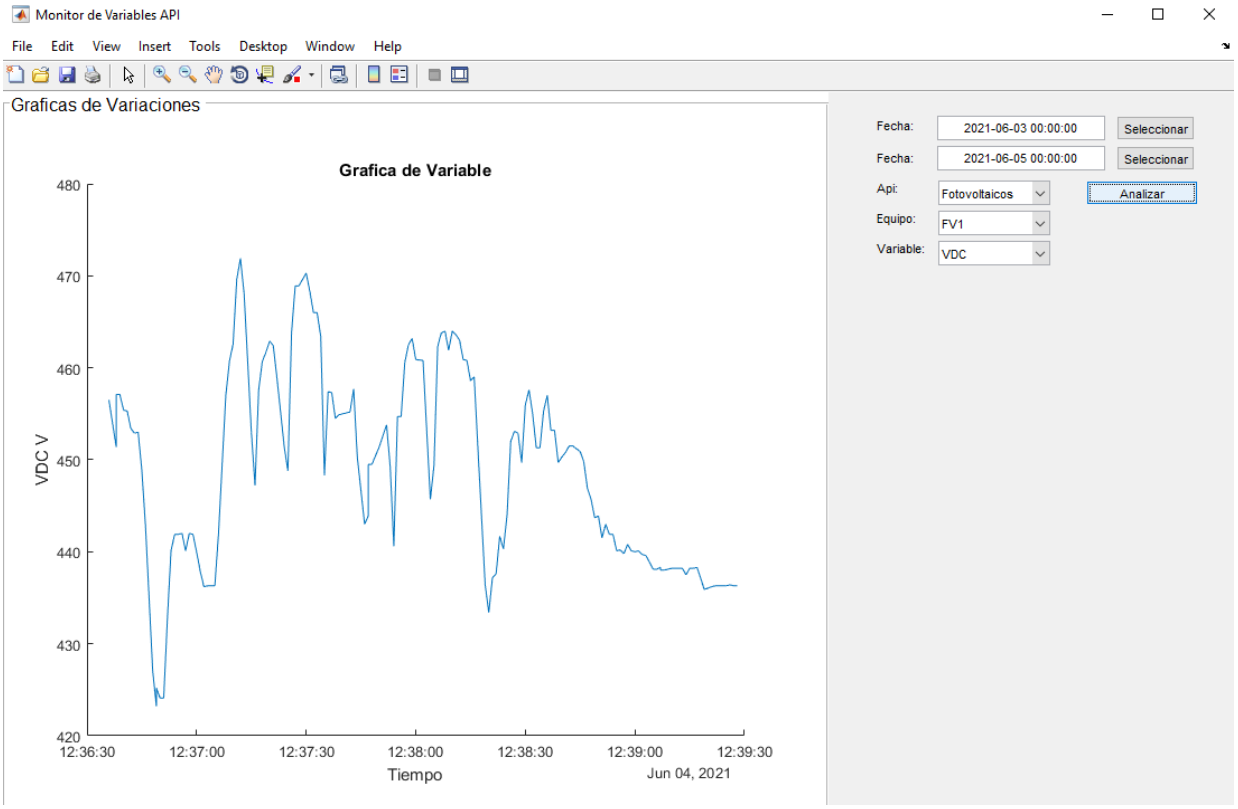


Figura 3-15 Monitor de Variables

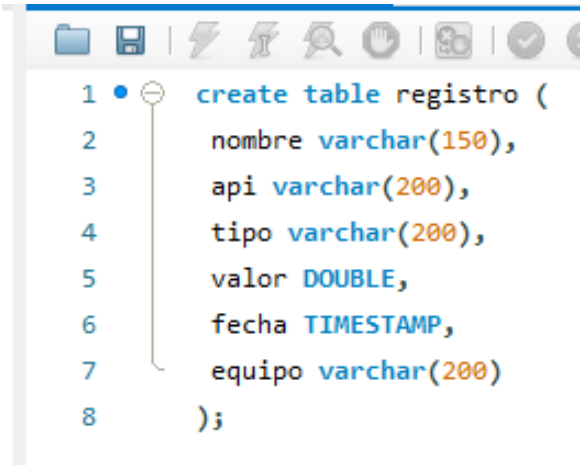
La gráfica de variable, mostrada en la Figura 3-15, presenta una secuencia de mediciones, que son los valores de las variables almacenados en base de datos del sistema, estas mediciones están unidos mediante líneas en lapsos de tiempo de un segundo. El eje Y

corresponde al voltaje y el eje X al tiempo. El eje X no tiene unidades definidas debido a que el lapso puede variar de días a años.

3.4 Aproximación tecnológica de soporte al uso de Sistemas SCADA

La tarea de ejecución del script de implementación tiene como soporte un gestor de base de datos (MySQL), el cual está incluido en el componente de repositorio de datos, el componente alberga una instancia de la base de datos “registrosVariables” con una tabla “registros” para el almacenamiento de los datos producto de monitoreo en tiempo real del sistema SCADA. El objetivo de la base de datos dentro del sistema es almacenar información y proveer la información de los valores de las variables relacionadas con los equipos que indican su estado, para su posterior análisis a través de gráficas que muestran el histórico de los valores en lapsos de tiempo definidos en la interfaz de monitoreo perteneciente al sistema SCADA.

A continuación, se presenta la tabla generada con sus atributos (nombre, api, tipo, valor, fecha, equipo). El código SQL necesario para instanciar tabla se muestra en la Figura 3-16.



```sql
create table registro (
 nombre varchar(150),
 api varchar(200),
 tipo varchar(200),
 valor DOUBLE,
 fecha TIMESTAMP,
 equipo varchar(200)
);
```

Figura 3-16.SQL para crear tabla de registro

En la Figura 3-17 se presenta la instancia de la base de datos creada en el gestor de datos MySQL

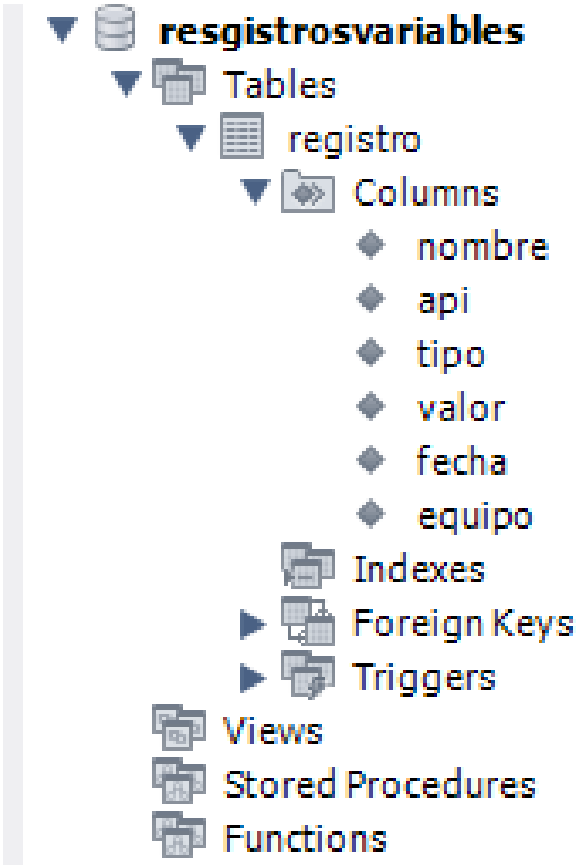


Figura 3-17 Base de datos registrosVariables.

4. CASOS DE ESTUDIO

En este capítulo se presentan dos casos de estudio de sistemas SCADA, para el laboratorio de micro red de la universidad de Cuenca, realizados con la infraestructura de software desarrollada en el presente trabajo de titulación.

4.1. Sistema SCADA Fotovoltaicos.

En esta sección se presenta la implementación de un sistema SCADA para el monitoreo de los equipos del sistema de fotovoltaicos para el Laboratorio de Micro red de la Universidad de Cuenca, generado mediante la infraestructura de software desarrollada. El sistema SCADA Fotovoltaicos está conformado por equipos electrónicos(Fotovoltaico1, Fotovoltaco2, Fotovoltaico3) de los cuales se monitorea y almacena información de valores de variables(VDC,PDC, PAC) cada cierto periodo de tiempo ,con el objetivo de monitorear las tensiones y corrientes de fase y las potencias totales que entregan a la micro red. .

La aplicación la infraestructura de software sigue un ciclo de vida de sistemas SCADA en cual está organizado en fases:

- **Diseño:** Fase de proceso encargada de representar los equipos (Fotovoltaico1, Fotovoltaico2, Fotovoltaico3) físicos del sistema de fotovoltaicos en el editor gráfico y agregar elementos para la visualización de las variables(VDC,PDC, PAC), obteniendo como resultado el diseño del panel de fotovoltaicos.
- **Construcción:** Fase de generación del script de implementación, en base al modelo del sistema Scada Fotovoltaicos creado en el editor de Scada y con la ayuda de herramienta del motor de transformación.
- **Uso**: Fase encargada de ejecutar el script de implementación del sistema de fotovoltaicos en un gestor de base de datos(MySQL).

Para ejecutar los sistemas Scada se requiere la instalación de un compilador de código matlab en el computador desde donde se va a realizar el monitoreo y control.

4.1.1 Diseño

Dentro de la fase de diseño, el técnico de operaciones hace uso de los manuales de los fabricantes de los equipos (Fotovoltaico1, Fotovoltaico2, Fotovoltaico3) para obtener sus

atributos y de los diagrama de red del sistema de fotovoltaicos para obtener la comunicación de los equipos. El técnico de operaciones para realizar el modelo elige los elementos gráficos haciendo uso del menú lateral del editor gráfico y arrastrando con el ratón cada elemento e incrustando dentro del diseño para representar los equipos físicos.

A continuación se presentan los distintos pasos para realizar el proceso de diseño.

El técnico de operaciones inicialmente hace uso de la interfaz de Sistema(Figura 3-5) presente en el editor gráfico el cual cuenta con sus 2 secciones de apis, paneles y a la derecha el menú agregar apis(Fotovoltaicos) y paneles(Fotovoltaicos) .Los usuarios pueden agregar APIs y Paneles arrastrando y soltando elementos dentro del contenedor de APIs y Paneles, y puede modificar los valores de sus atributos haciendo clic sobre el elemento a modificar como se muestra en la Figura 4-1.

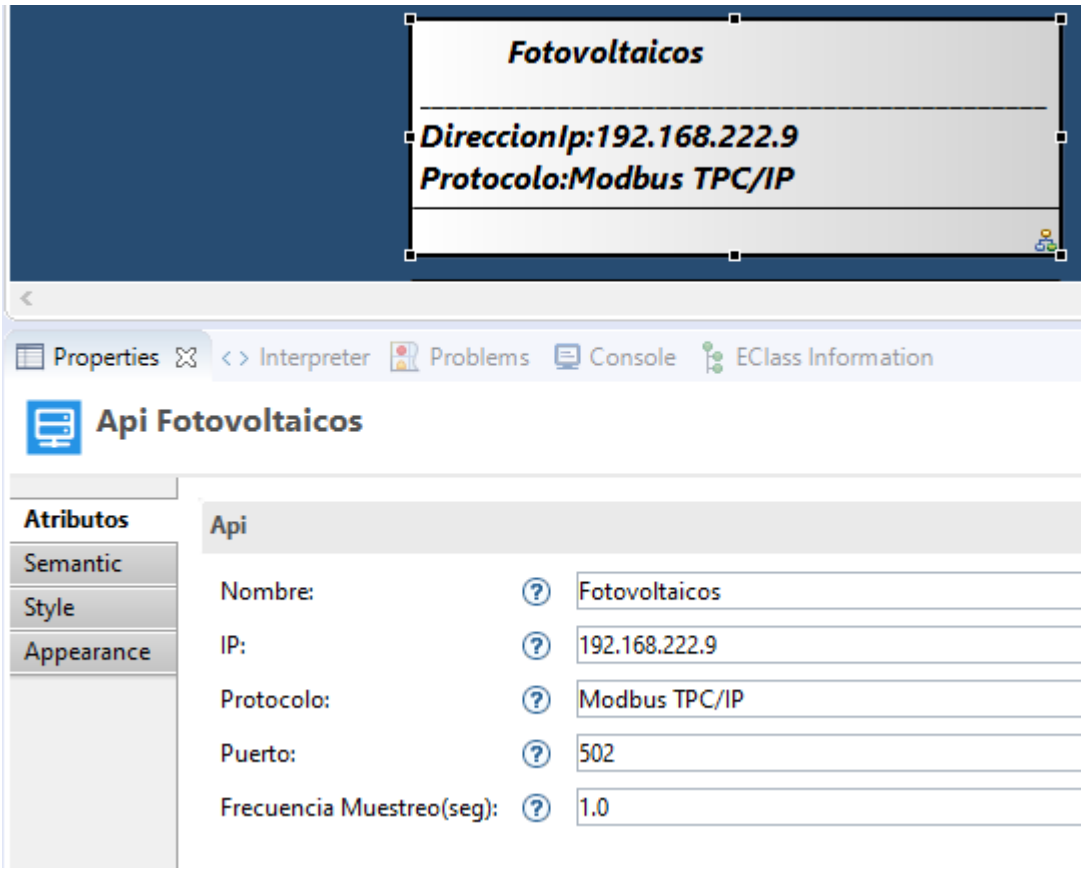


Figura 4-1. Instancia de Api Fotovoltaicos.

Como se puede ver en la Figura 4-1 el usuario puede modificar los valores de los atributos de un api con la sección llamada Atributos, en el caso de Panel es el mismo proceso.

En segundo lugar el técnico de operaciones accede a la interfaz de Api(Api Fotovoltaicos) haciendo uso del elemento gráfico creado mostrado en la Figura 4-1 y haciendo doble clic sobre él o directamente sobre la sección ApiDiagrama. La interfaz de una API(Fotovoltaicos) seleccionada contiene un menú con un elemento que es el de equipo,se crea la instancia de equipo(Fotoltaicos1, Fotovoltaicos2,Fotovoltaivos3) arrastrando y soltando el elemento dentro del interfaz de Api como se muestra en la Figura 4-2.

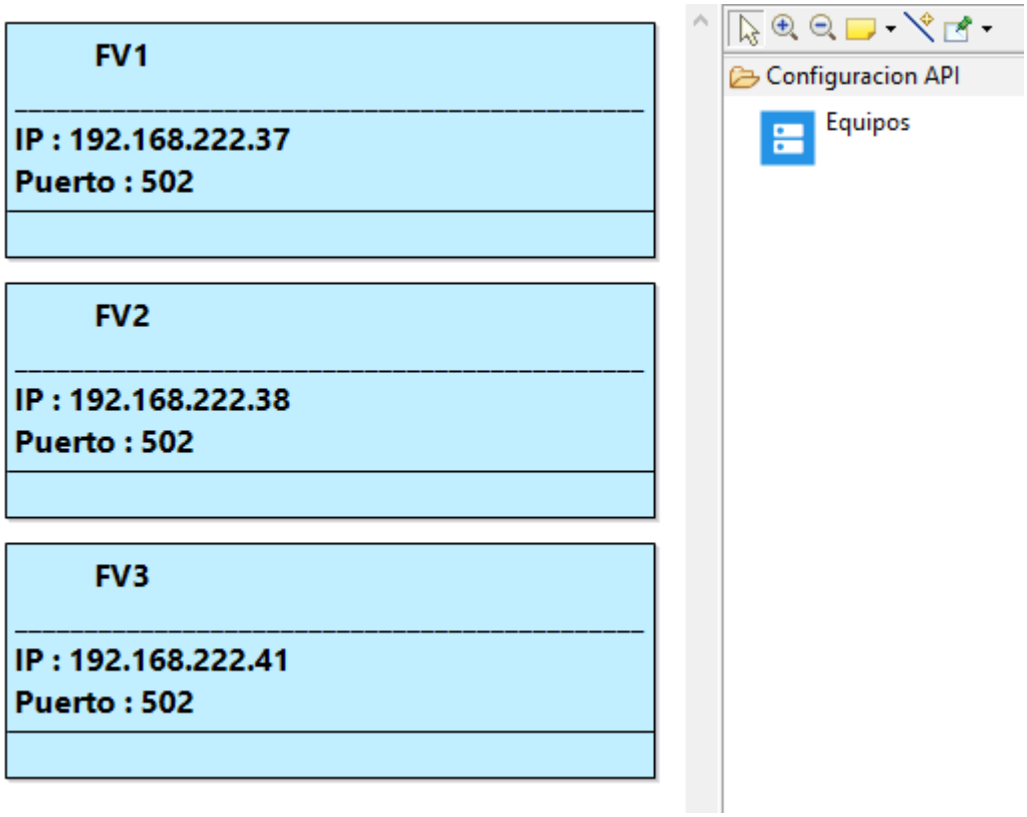


Figura 4-2.Interfaz Api Fotovoltaicos.

El técnico de operaciones cuenta con ventanas emergentes para la modificación y agregación de los valores de los atributos de los equipos, como se demuestra en Figura 4-3, el usuario debe seleccionar un elemento de la tabla y hacer clic sobre los botones modificar y eliminar respectivamente.

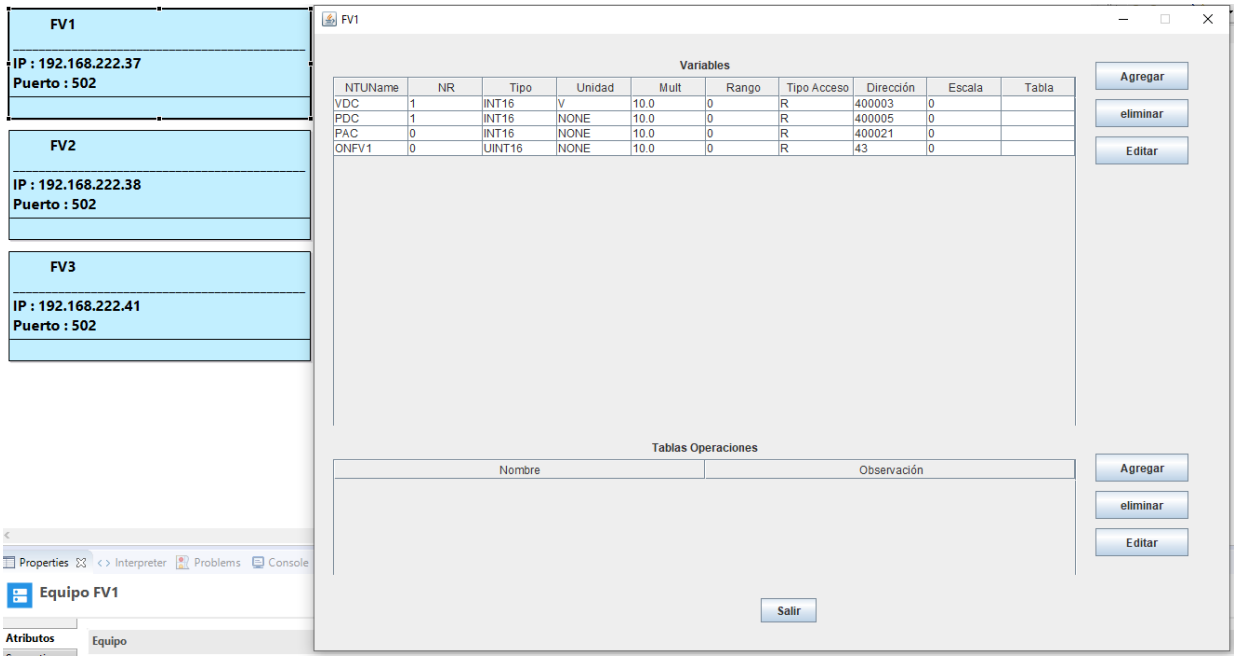


Figura 4-3 Ventana Modificar Equipo FV1(Fotovoltaico1)

Finalmente el técnico en operaciones agrega Paneles(Fotovoltaicos) a la interfaz de sistema , el cual se almacena en ApiDiagrama. Para acceder a la interfaz de panel Fotovoltaicos se debe hacer doble clic sobre el panel de la misma forma que las APIs.

En la interfaz de Panel el técnico de operaciones puede generar el diseño del sistema SCADA Fotovoltaicos, arrastrando y soltando lo elementos, el modelo(Figura 4-4) está conformado por tres equipos: Fotovoltaico1 (Figura 4-5) , Fotovoltaico2 (Figura 4-6) , Fotovoltaico3 (Figura 4-7), y los cuales están conectados al PLC(APIS1_FV) que se muestra en la Figura 4-8. el técnico puede agregar elementos gráficos arrastrando los elementos del menú lateral de la interfaz para realizar su diseño

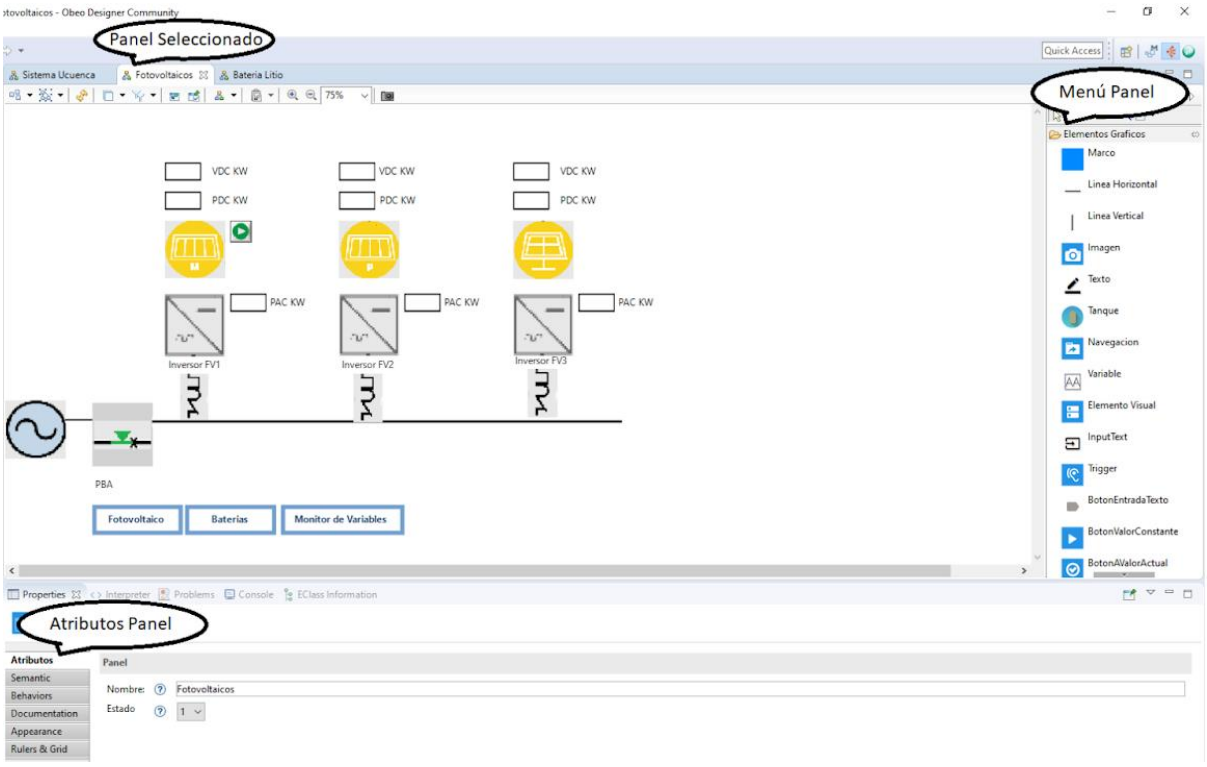


Figura 4-4 Diseño.Panel Fotovoltaicos

Figura 4-5 Api Monocristalinos

Figura 4-6 Api Policristalinos

Figura 4-7 Api Paneles Solares

Figura 4-8. API_1

Terminada esta fase de diseño, se hace uso de la transformación de código, que usa las representaciones y valores de los atributos almacenados para generar un interfaz en código Matlab que monitoree las variables de los equipos establecidos dentro de una API. Previo a la ejecución del motor de transformación, se ejecuta la función ”Generar Posición ” que se muestra en cualquier interfaz del editor haciendo clic izquierdo sobre la interfaz.

4.1.2 Construcción

En la fase de construcción en técnico de operaciones hace uso del motor de mediante una interfaz de Acceleo y ejecutando el plugin transformación, el cual utiliza el modelo de sistema SCADA fotovoltaicos generado en editor de SCADA, este motor de transformación usa el modelo como entra y devuelve un archivo .m estructurado de la siguiente manera:

Función principal:

La función principal (SistemaUcuenca) es la encargada de invocar a la conexión modbus (mbFotovoltaicos) y al timer(timerAPIFotovoltaicos) las cuales se encarga de establecer un puente de comunicación constante para la recepción de datos del PLC(API_1).

Además la función posiciona los equipos(Fotovoltaico1, Fotovoltaico2, Fotovoltaico3) dentro de la interfaz de sistema SCADA Fotovoltaicos e invoca la funciones secundarias, parte del código de la función principal de muestra en la Figura 4-9, el resto del código se puede encontrar en el Anexo 4.

```
function SistemaUcuenca
        global mbFotovoltaicos ;
        global timerAPIFotovoltaicos ;
        hs = addcomponents; % Make figure visible after adding components
        hs.fig.Visible = 'on';
        asignarTimers(hs);
        conectarApis();
        iniciarTimers();
        function hs = addcomponents   % add components, save handles in a
            posicion= [ 0 -1000 50 30 ]; % posicion y size
            hs.fig = figure('Name','Sistema Balzay' ,'NumberTitle','off','
            hs.panelFotovoltaicos=uipanel('Parent',hs.fig,'Title','Fotovol
                    %Input Text
                    posicion=[ 264.0 632.0 60.0 30.0]; % posicion y size
                    hs.InputeTexteGj9oGD2EeuEZaODg90yXA=uicontrol('Parent'

                    %Input Text
                    posicion=[ 264.0 584.0 60.0 30.0]; % posicion y size
                    hs.InputeTextuvmEmD3EeuEZaODg90yXA=uicontrol('Parent',

                    %Texto Decoracion
                     posicion=[ 336.0 632.0 73.0 30.0]; % posicion y size
                     hs.TextoDecoracionMrAUkGHZEeuGIbEIiLYpQ=uicontrol('Pa

                    %Texto Decoracion
                     posicion=[ 612.0 632.0 74.0 30.0]; % posicion y size
                     hs.TextoDecoracionQbm4YGHZEeuGIbEIiLYpQ=uicontrol('Pa
```

Figura 4-9 Función Principal de sIstema SCADA Fotovoltaicos

Funciones secundarias:

Las funciones secundarias que complementan a la función principal, presentes dentro del script de implementación son las funcionalidades (Asignar timer, conectar apis , iniciar timers , leer datos fotovoltaicos) , la cuales ayudan para obtener la funcionalidad de presentación de los datos en la interfaz del sistema de fotovoltaicos. Parte del código implementado se muestra en la siguiente Figura 4-10.

```
function leer_Datos_Fotovoltaicos(interfaz,lista_Datos)
    nombreApi=lista_Datos{1};
    conexionModbus=lista_Datos{2};
    timer_Api=lista_Datos{3};
    for k=1:1:length(lista_Datos{4})
       direcciones_Memoria=values(lista_Datos{4}{k});
       for m=1:1:length(direcciones_Memoria)
           direccion_Mem=direcciones_Memoria{m};
           nombreVar = direccion_Mem{3};
           tipoVar = direccion_Mem{4};
           nombreEquipo = direccion_Mem{5};
           posicion_registro= direccion_Mem{1}-400000;
```

Figura 4-10 Función secundaria Leer Datos Fotovoltaicos

4.1.3 Uso

En la fase de uso el técnico de operaciones deberá hacer uso de un compilador de código matlab para ejecutar el script de implementación. En la Figura 4-11 se presenta el sistema SCADAA Fotovoltaicos ejecutado en el compilador, el cual monitorea y almacena tres variables: VDC,PDC, PAC, de los equipos. La variable VDC hace referencia al voltaje de corriente continua, PDC indica el valor de la potencia de corriente continua y la variable PAC muestra en valor de potencia de corriente alterna. Los valores de las variables se presentan en el panel mediante un labels y están disponibles en un PLC(APIS1_FV), el cual recibe constantemente de los equipos físicos fotovoltaicos.

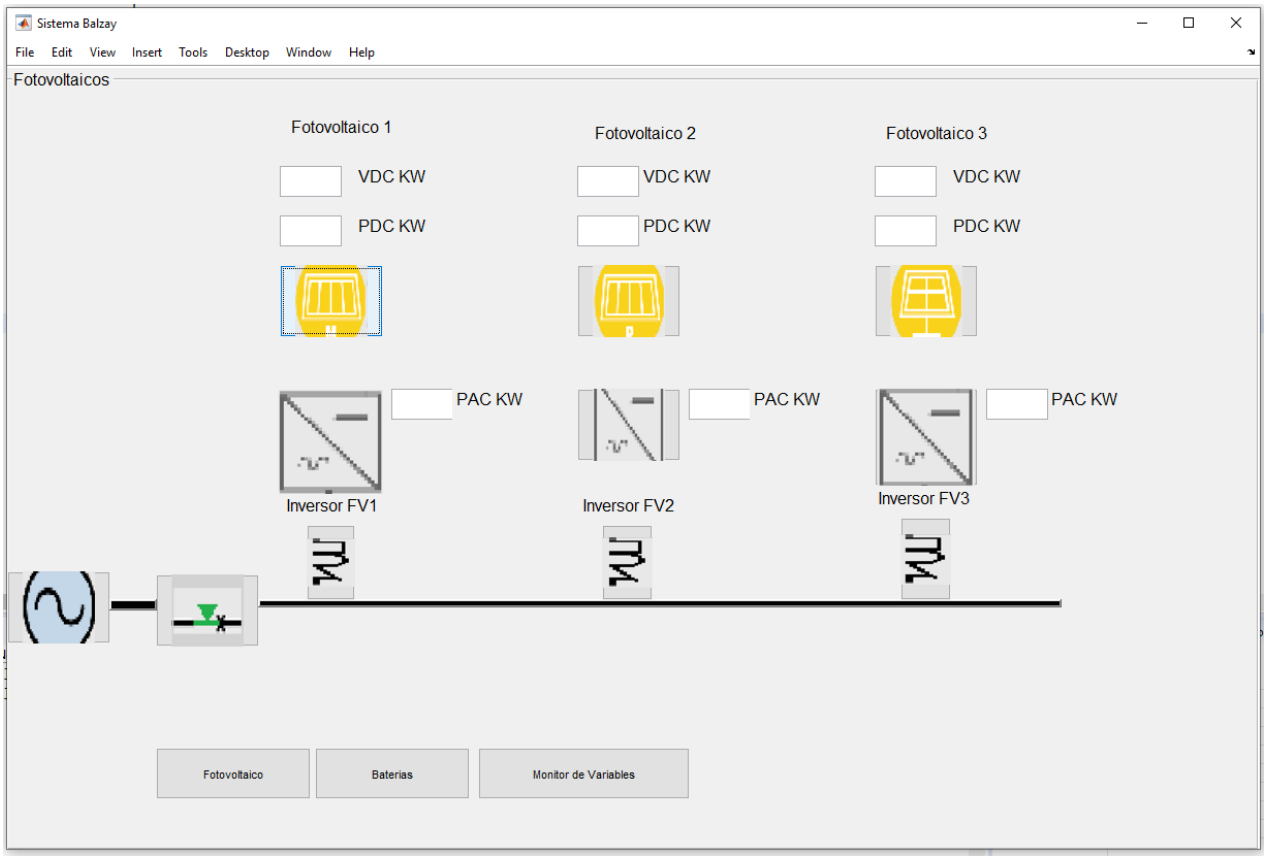


Figura 4-11 Panel Fotovoltaicos

4.2. Sistema SCADA de Baterías de Carga.

En esta sección se presenta la implementación de un sistema SCADA para el monitoreo de los equipos del sistema de Baterías de Carga, con el fin de de monitorear las tensiones y corrientes de fase y las potencias totales que entregan a la micro red la batería de plomo, batería de litio, y equipo redox La aplicación la infraestructura de software sigue un proceso semejante al caso 1 con fases:

- **Diseño:** Fase de proceso encargada de representar los equipos (Batería de Litio, Batería de Plomo, Equipo Redox) físicos del sistema de baterías de Carga en el editor gráfico y agregar elementos para la visualización de las variables (PDC, IDC Y P), obteniendo como resultado el diseño del panel de fotovoltaicos.
- **Construcción:** Fase de generación del script de implementación, en base al modelo del sistema SCADA Batería de Carga es creado en el editor de SCADA y con la ayuda herramienta del motor de transformación.

- **Uso**: Fase encargada de ejecutar el script de implementación del sistema de baterías de Carga en un gestor de base de datos(MySQL).

4.2.1 Diseño

La operación de diseño del caso 2 es similar al caso 1, en primer lugar el técnico de operaciones hace uso del interfaz de la interfaz de Sistema(Figura 4-12) presente en el editor gráfico el cual cuenta con sus 2 secciones de APIs, paneles y a la derecha el menú agregar APIs (Batería de Litio, Batería de Plomo, Equipo Redox) y paneles(Baterías) .El técnico agrega los elementos con el mismo proceso de arrastrar y soltando del caso 1, y puede modificar los valores de sus atributos haciendo clic sobre el elemento a modificar como se muestra en la Figura 4-13.

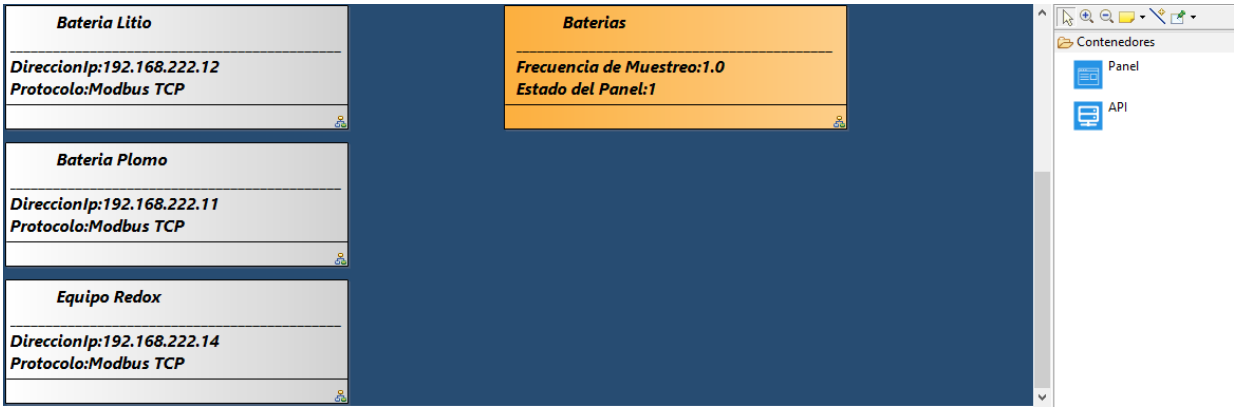


Figura 4-12 Interfaz Sistema (Contenedores de sistema de Baterías de Carga)

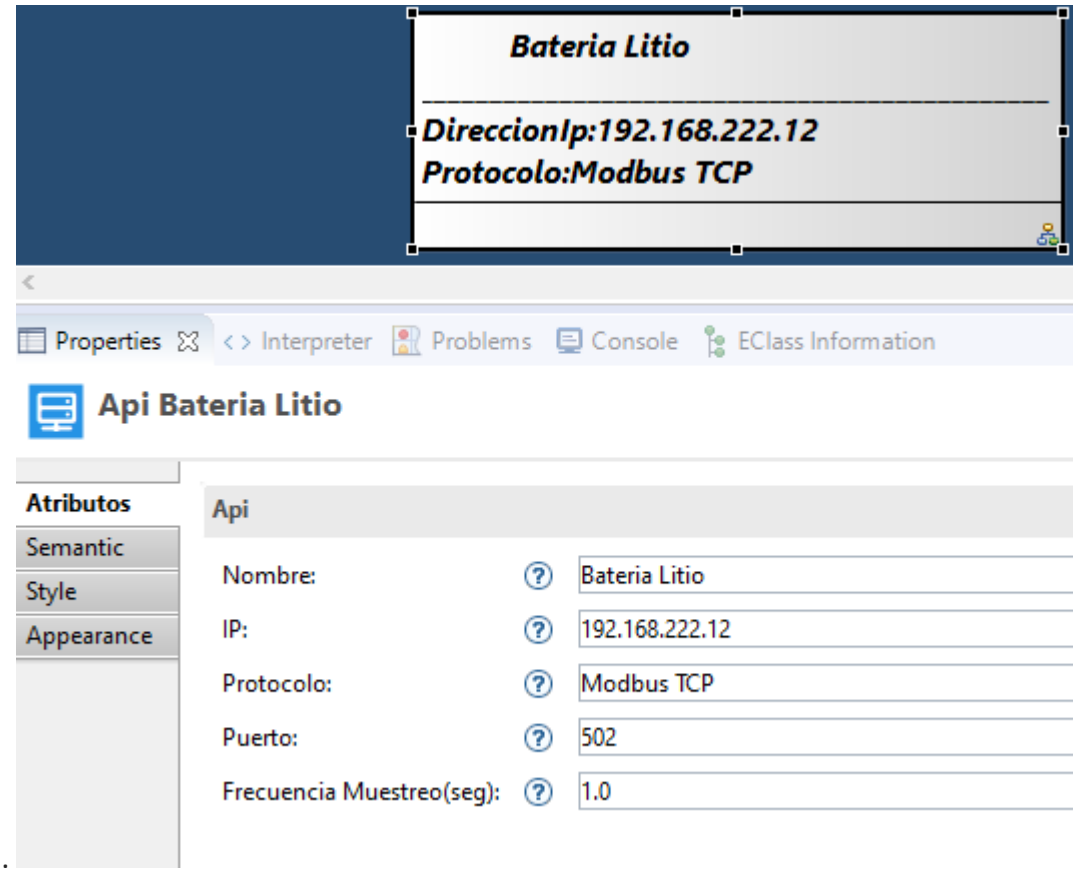


Figura 4-13 Instancia de Api Bateria Litio

En segundo lugar el técnico de operaciones accede a la interfaz de Api(Api Batería litio, Api Batería Plomo, Api Equipo Redox) haciendo uso del elemento gráfico creado para cada api y haciendo doble clic sobre él o directamente sobre la sección ApiDiagrama. La interfaz de una API(Api Batería litio, Api Batería Plomo, Api Equipo Redox) seleccionada contiene un menú con un elemento que es el de equipo, el cual se crea la instancia de equipo(Bateria litio, Batería Plomo, Equipo Redox), arrastrando y soltando el elemento dentro del interfaz de Api como se muestra en la Figura 4-14.

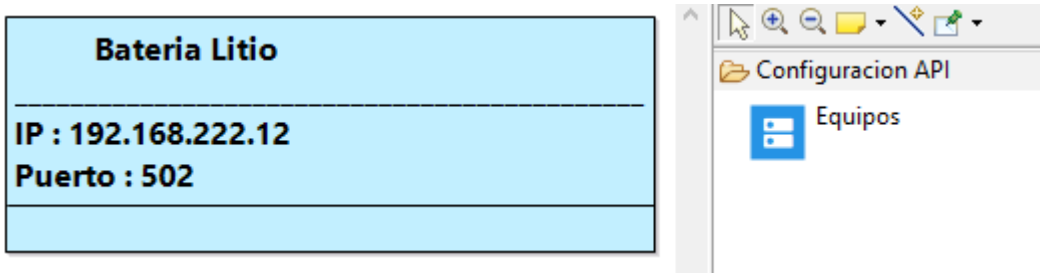


Figura 4-14 Interfaz Api Batería Litio.

Finalmente el técnico en operaciones agrega paneles(Baterías) a la interfaz de sistema de la misma forma que el caso 1.

En interfaz de Panel el técnico de operaciones puede generar el diseño del sistema SCADA Baterías, arrastrando y soltando los elementos, el modelo (Figura 4-15) está conformado por tres equipos: Batería litio, Batería Plomo, Equipo Redox, y los cuales están conectados al PLC(*APIS2_ALM*) que se muestra en la Figura 4-16.

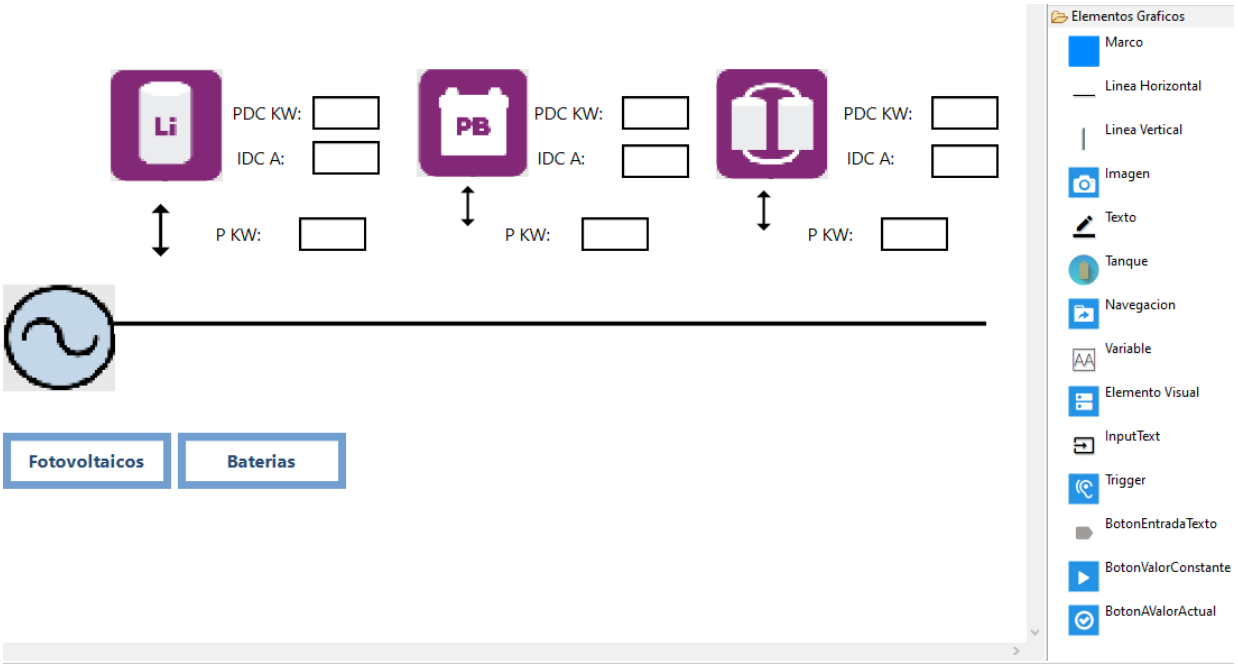


Figura 4-15 Diseño de Panel Baterías de Carga

Figura 4-16 API_2

4.2.2 Construcción

Esta sección es similar al caso 1, donde el técnico de operaciones utiliza el modelo de sistema SCADA Baterías de Carga generado en editor de SACA para obtener el script de implementación, mediante el motor de transformación el cual usa el modelo como entra y devuelve un archivo .m estructurado de la siguiente manera:

Función principal:

La función principal (SistemaUcuenca) es la encargada de invocar a la conexión modbus (mbBateriaLitio, mbBateria) y al timer(timerAPIBateriaPlomo, timerAPIBateriaLitio) las

cuales se encarga de establecer un puente de comunicación constante para la recepción de datos del PLC(API_2)

Además la función posiciona los equipos(Batería litio, Batería Plomo, Equipo Redox) dentro de la interfaz de sistema SCADA Baterías e invoca la funciones secundarias, parte del código de la función principal de muestra en la Figura 4-17.

```
function SistemaUcuenca
        global mbBateriaLitio ;
        global timerAPIBateriaLitio ;
        global mbBateriaPlomo ;
        global timerAPIBateriaPlomo ;
        global mbSuperCondensadores ;
        global timerAPISuperCondensadores ;
        hs = addcomponents; % Make figure visible after adding components
        hs.fig.Visible = 'on';
        asignarTimers(hs);
        conectarApis();
        iniciarTimers();
        function hs = addcomponents   % add components, save handles in a stru
            posicion= [ 0 -1000 50 30 ]; % posicion y size

            hs.fig = figure('Name','Sistema Balzay' ,'NumberTitle','off','Vis
            hs.panelBaterias=uipanel('Parent',hs.fig,'Title','Baterias','Font
                    %Input Text
```

Figura 4-17 Función Principal de SIstema Scada Baterías de Carga

Funciones secundarias:

Las funcionalidades que complementan a la función principal son similares al caso 1(Asignar timer, conectar apis, iniciar timers, leer datos), estas funcionalidades son la que ayudan a la presentación de los datos en la interfaz del sistema SCADA de Baterías. Parte del código implementado se muestra en la siguiente Figura 4-18.

```
function leer_Datos_BateriaLitio(interfaz,lista_Datos)
    nombreApi=lista_Datos{1};
    conexionModbus=lista_Datos{2};
    timer_Api=lista_Datos{3};
    for k=1:1:length(lista_Datos{4})
       direcciones_Memoria=values(lista_Datos{4}{k});
       for m=1:1:length(direcciones_Memoria)
           direccion_Mem=direcciones_Memoria{m};
           nombreVar = direccion_Mem{3};
           tipoVar = direccion_Mem{4};
           nombreEquipo = direccion_Mem{5};
           posicion_registro= direccion_Mem{1}-400000;
           registro = read(conexionModbus,'holdingreg',posicion_registro
           hijos= allchild(interfaz);
           for i=1:1:size(hijos)
                 if class(hijos(i))=="matlab.ui.control.UIControl" && is
                      tag = strsplit(hijos(i).Tag,'-'); % dividimos el t
                      if length(tag)>1
                          dir_mem=tag{1};
                          api_nom=tag{2};
                          if dir_mem==string(int2str(direccion_Mem{1}))
                             set(hijos(i),'string',registro/direccion_Me
                             conn = database('registosVariables','','');
                             query = strcat('INSERT INTO registro(api,no
                             disp(query);
                             execQuery = exec(conn,query);
                             execQuery = fetch(execQuery);
                          end
                      end
                 end
           end
       end
    end
end
```

Figura 4-18 Función secundaria Leer Datos (Batería Litio)

4.2.3 Uso

De la misma forma que en el caso 1, el técnico de operaciones debe ejecutar el script de implementación en un compilador (Matlab) de código, acorde al tipo(.m) de archivo creado. En la Figura 4-19 presenta la interfaz del sistema SCADA para baterías de carga, la cual monitorea las variables de *PDC, IDC y P*.

La variable *PDC* hace referencia a la potencia de corriente directa, la variable *IDC* indica la intensidad de corriente directa y *P* la potencia total. Estas variables hacen referencia a los atributos que se almacenan dentro de una batería(equipo) en un *PLC*(*APIS2_ALM*).

Los valores de las variables obtenidas de un PLC mediante una API y mostradas en el panel sirven para obtener el estado de las baterías de carga siendo la potencia la que nos indica si la batería está en carga o descarga.

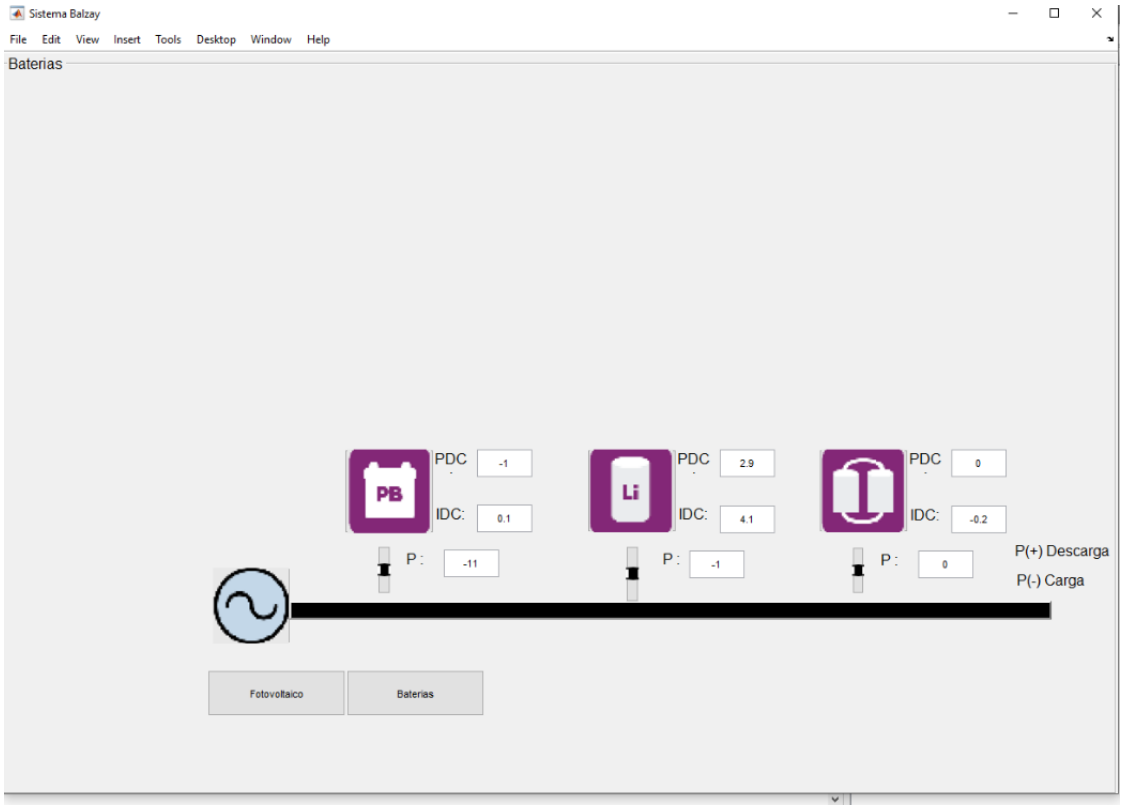


Figura 4-19 Panel Baterías de carga

5. CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se presentan las principales conclusiones resultado del desarrollo de este trabajo de titulación y el examen de en qué medida se lograron alcanzar los objetivos de la misma.

5.1 Conclusiones

La implementación de sistemas SCADA conlleva unas etapas de desarrollo de software, en las cuales se tiene que cumplir con sus diferentes procesos en cada una de ellas, con lo cual se requiere de conocimientos y expertos en software para poder llevar a cabo esas etapas. Con el objetivo de simplificar el proceso de desarrollo de sistemas SCADA, se ha propuesto y validado una infraestructura de software basada en un enfoque de desarrollo dirigido por modelos. A continuación, se analiza el grado de cumplimiento de cada uno de los objetivos planteados en este trabajo de titulación en la sección 1.3.

5.1.1 Definición de una arquitectura para la infraestructura tecnológica de generación de Sistemas SCADA

Se identificaron los componentes de software necesarios para dar soporte al diseño e implementación de un sistema SCADA. De igual manera se identificó las relaciones y dependencias entre estos componentes. La definición de la arquitectura obedece a un enfoque de desarrollo dirigido por modelos y sus componentes facilitan la generación de scripts de implementación de sistemas SCADA a partir de diseños a un nivel de abstracción que no tienen en consideración aspectos de las plataformas de simulación o protocolos a emplear. Esta plataforma ha sido diseñada de manera modular con el propósito de dotar de flexibilidad. En este sentido el diseño de un del motor de transformación basado en plugins (adaptadores) tiene los siguientes beneficios:

- Capacidad de expansión. En caso de requerir que un sistema SCADA se ejecute en una plataforma de modelado y simulación, o que satisfaga un protocolo no considerado en la infraestructura software creada en este trabajo de titulación, lo único que se requiere es crear un adaptador de software e incorporarlo al componente motor de transformación. Este adaptador será una transformación M2T que interpreta Modelos de Diseño de Sistemas SCADA y genera el script de acuerdo a la nueva plataforma o protocolo.
- Reutilización de Modelos de Diseño: En caso de requerir que un sistema SCADA se ejecute en diferentes plataformas de modelado y simulación, o en diferentes plataformas no será necesario modificar el diseño del sistema SCADA. Los adaptadores incorporados al motor de transformación reutilizan los modelos de diseños de sistemas SCADA creados y se encargan de generar los scripts de implementación correspondientes. Los aspectos específicos de la plataforma o estándar para la ejecución están embebidos en los adaptadores por lo que no se requiere modificar los modelos de diseño

5.1.2 Diseño de un Lenguaje Específico de Dominio (DSL) que facilite el diseño de Sistemas SCADA

Para cumplir con este objetivo se ha definido un conjunto de conceptos y relaciones entre conceptos aplicables durante el diseño de sistemas SCADA. En donde, estos conceptos son genéricos e independientes de la plataforma de modelado y simulación (o protocolo) en el cual se ejecutará el sistema SCADA. Este lenguaje permite elevar el nivel de abstracción durante el diseño de sistemas SCADA. Evitando que los OPERADORES de campo requieran de conocimientos específicos de la plataforma o protocolo de ejecución; enfocando sus esfuerzos al diseño de las actividades de monitoreo y control de los elementos que conforman el sistema.

5.1.3 Creación de un editor gráfico que soporte el diseño de Sistemas SCADA conforme al lenguaje de diseño de Sistemas SCADA

Se ha desarrollado un editor gráfico para facilitar la la creación de diseños de sistemas SCADA que posteriormente son transformados a un archivo de texto con una extensión específica para un lenguaje de programación que implemente sistemas SCADA como por ejemplo .m (Matlab).

El editor gráfico permite crear sistemas SCADA arrastrando elementos visuales mediante una interfaz gráfica, también permite configurar los parámetros de esos elementos y además permite decorar los diseños mediante elementos de decoración como, por ejemplo: líneas, cuadros de texto, colores, imágenes. Con el editor gráfico la complejidad en el desarrollo y diseño de un sistema SCADA se disminuye, ya que la creación de los componentes tanto lógicos como visuales son parametrizables, como por ejemplo en la creación de un equipo fotovoltaico se arrastra el elemento visual o lógico correspondiente y se ingresa sus atributos como la ip, puerto.

El editor gráfico permite adaptar el contenido a las necesidades de los requerimientos de sistema, sin necesidad de crear varias veces el sistema generado, así el operador podrá reducir sus esfuerzos y tiempo en programación y enfocarse más en la calidad de los diseños de las interfaces y la visibilidad de la información.

5.1.4 Diseño y creación de un motor de transformación para la generación de código de Sistemas SCADA

Para la generación automatizada de código a partir de los diseños de sistemas SCADA, se ha implementado un motor de transformación de código, el cual aporta con la generación de cualquier tipo de fichero que soporte un entorno de desarrollo de sistemas SCADA (como

por ejemplo matlab, labview), bastaría con iterar todos los elementos del modelo y generar el tipo de fichero que se acople a la tecnología que interese. Al trabajar con modelos que referencian componentes como cajas de texto, botones, imágenes, se generan los ficheros que se ejecuten en su entorno específico para cada plataforma de forma automatizada.

Este proceso de generar código específico de plataforma a partir de modelos podría alargarse tanto como se requiera y tanto como se refine el modelo de diseño. Cuanto más refinado esté y cuanto más completo sea el modelo se reduce la carga de trabajo de los desarrolladores de software.

De esta manera con el motor de transformación de código, se reduce la creación de objetos, clases y artefactos de software, ya que cada vez que se requiera añadir nuevos atributos o funcionalidades al sistema a implementar, se evita que un desarrollador tenga que analizar y cambiar varios ficheros de código, y comprobar que el resto del sistema funciona correctamente. Con el motor de transformación de código, cada una de las nuevas funcionalidades requiere únicamente una cierta modificación del modelo de diseño y el motor se encarga de generar el código a partir de ese modelo.

5.1.5 Validación de la infraestructura de software a través de casos de estudio

Mediante la infraestructura de software desarrollada se implementaron dos casos de uso de sistemas SCADA para el laboratorio de micro-red de la Universidad de Cuenca.

Con el uso de la infraestructura de software se evidenció que el desarrollo de sistemas SCADA es una tarea mucho más sencilla que implementarlos desde cero, además con la solución propuesta, se evita que los técnicos deban conocer aspectos tecnológicos de cómo implementar sistema SCADA (ej., los técnicos no requieren conocer cómo se crean las gráficas, ni tampoco conocer acerca del lenguaje de programación en el que se implementan los sistemas SCADA) de esta manera se pueden desarrollar sistemas SCADA a través del diseño de modelos mediante el uso de una interfaz gráfica (editor gráfico), lo cual se traduce en una reducción en el tiempo de desarrollo.

5.2 Limitaciones

La herramienta Sirius disponible en Obeo Designer aunque útil para realizar lo deseado en el trabajo de titulación, presenta ciertos inconvenientes y limitaciones en su uso. El inconveniente radica en que no existe una buena documentación en línea, y la escasa documentación que existe es vaga y poco clara. En cuanto a limitaciones, la herramienta en

aspectos de implementación es bastante rígida, en cuanto a tamaños de los objetos gráficos que se crean, funcionalidades, tipo de imágenes que admite, manipulación de posicionamiento de objetos gráficos etc. Para solucionar ciertos inconvenientes se recurrió a invocar funciones en java para hacer ciertas tareas y extender las funcionalidades de la herramienta en funciones propias como “Posicionar” que se encarga de setear los valores de los atributos de posiciones de los objetos gráficos.

Todos los paneles que se implementan en el editor implementa una comunicación MODBUS TCP, por este motivo los paneles "Panel Fotovoltaicos y Panel Baterías de carga ", utilizan recursos externos para esta comunicación, que están disponibles en ciertos software de simulación y diseño como Matlab, aunque se recalca si se logra adaptar las funcionalidades de comunicación en otro software puede soportar el despliegue del sistema implementando una transformación de código acorde al software utilizado.

Los sistemas SCADA creados, aunque estén basados en las recomendaciones de un experto en área de desarrollo de esos sistemas, no se pudo realizar las pruebas pertinentes y validaciones para comprobar si se cumple con todas las necesidades de los técnicos en diferentes arquitecturas industriales, ya que no se contó con un grupo de diseños o paneles con las diferentes necesidades que describen las diferentes industrias.

5.3 Trabajos Futuros

En cuanto al aumento de características del editor gráfico, se plantea integrar la obtención en tiempo real de los valores de los atributos de cada equipo, ya que en este trabajo solo se puede ver los datos de los atributos una vez ejecutada la transformación a código y obtenido el archivo .m. De esta forma el técnico puede obtener retroalimentación de los tipos de datos mostrados y obtener un mejor diseño.

Para mejorar la variabilidad de los SCADA que genera el editor presentado en este trabajo se plantea realizar pruebas con grupo mayor de diseños y observar los elementos gráficos necesarios para su desarrollo y de esta forma retroalimentar el set de elementos gráficos del menú en el editor gráfico, logrando diseñar mejores sistemas SCADA.

Como se mencionó anteriormente con los objetos gráficos existentes existió problemas para integrar aspectos de posicionamiento y rotación, por eso se plantea crear nuevas librerías que desde un comienzo sean diseñados para soportar los diferentes cambios de posicionamiento de los objetos gráficos. Así se podrá mostrar el contenido de una mejor manera en el archivo .m y extender la creatividad del técnico para crear su diseño.

Anexos

A. Prototipo de baja fiabilidad (Alternativa 1)

La primera alternativa presenta los equipos como una lista, los elementos a ser ingresados hacen uso de tablas, en cada elemento se agrega botones de edición y eliminado para que el usuario pueda manipularlos. En la Figura A-1 se muestra una captura del diseño de la interfaz de Equipo A perteneciente a API 1, la cual contiene las variables en tablas y la tabla de operaciones como listas:

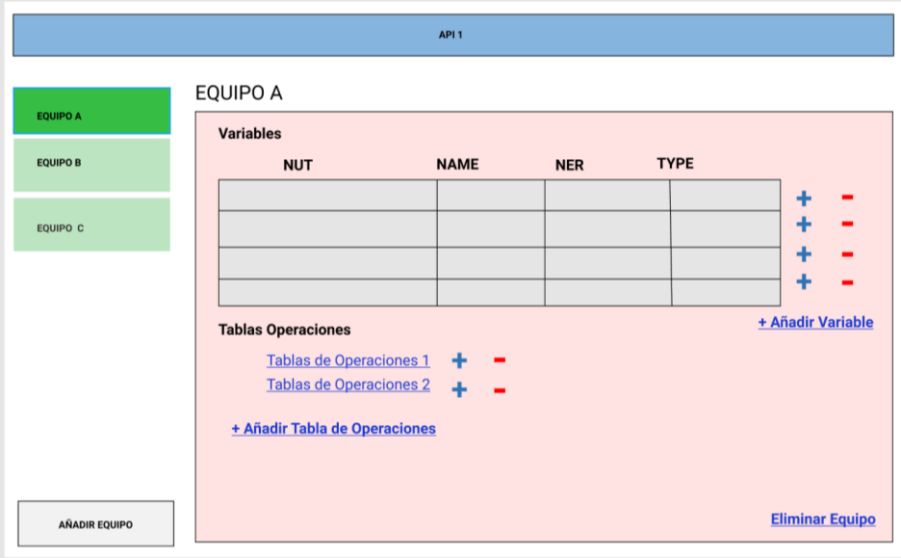


Figura A-1 Diseño Interfaz de Api

A continuación la Figura A-2 se muestra una captura del diseño de la ventana emergente la cual sirve para agregar una tabla de operaciones al Equipo A cuyos atributos son nombre tabla y observaciones, y donde cada operación perteneciente a la tabla de operaciones se encuentra dentro de una fila y sus atributos son: nombre,valor, mascara y se encuentra ubicadas en las columnas de la tabla:

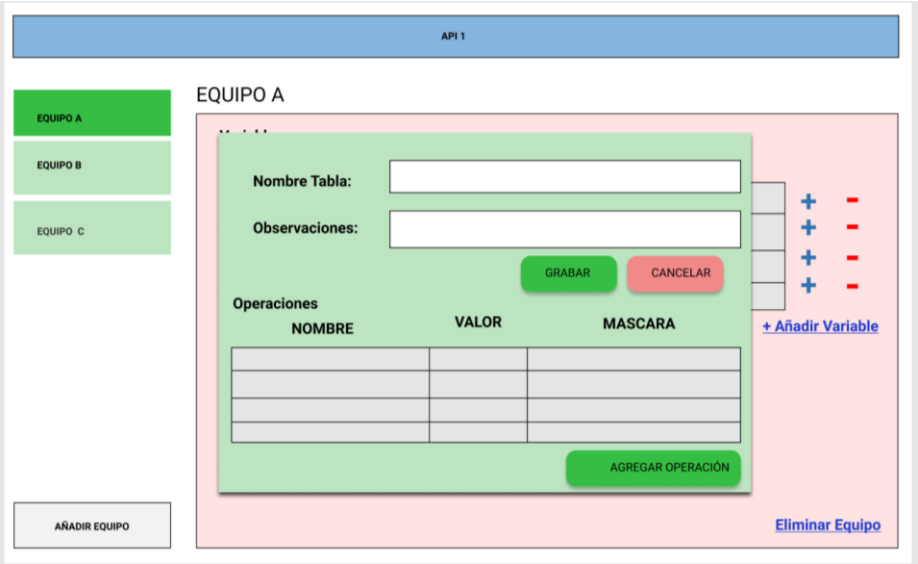


Figura A-2 Diseño ventana emergente para agregar tabla de operaciones

La ventana emergente para crear un equipo con sus atributos de nombre, dirección y puerto se activa al presionar el botón de añadir equipo.La Figura A-3 presenta el diseño de la venta para agregar un equipo.Además en la Figura A-3 podemos ver el botón eliminar equipo el cual elimina el Equipo A perteneciente a la API 1:

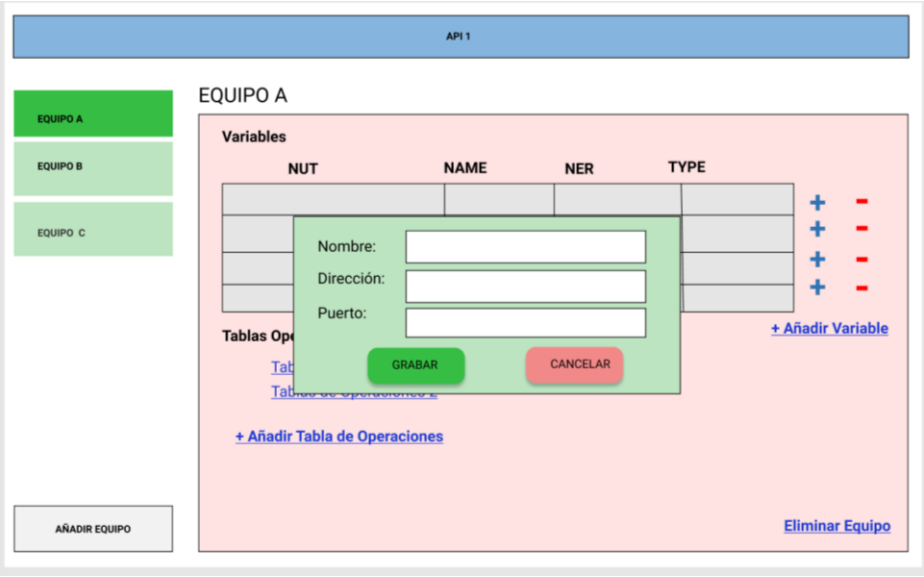


Figura A-3 Diseño ventana emergente para asignar valores a un equipo

En la Figura A-4 se ve el diseño de la interfaz de apis y paneles,cada grupo de elementos ocupa la mitad de la interfaz y se organiza de manera que cada elemento agregado se posicione en la parte inferior de los elementos existentes de la sección perteneciente. Los atributos de los elementos se presentan como listas dentro de cada api la cual posee

atributos que son: nombre, dirección ip, protocolo o dentro de cada panel cuyos atributos son: nombre, frecuencia de muestreo y estado.

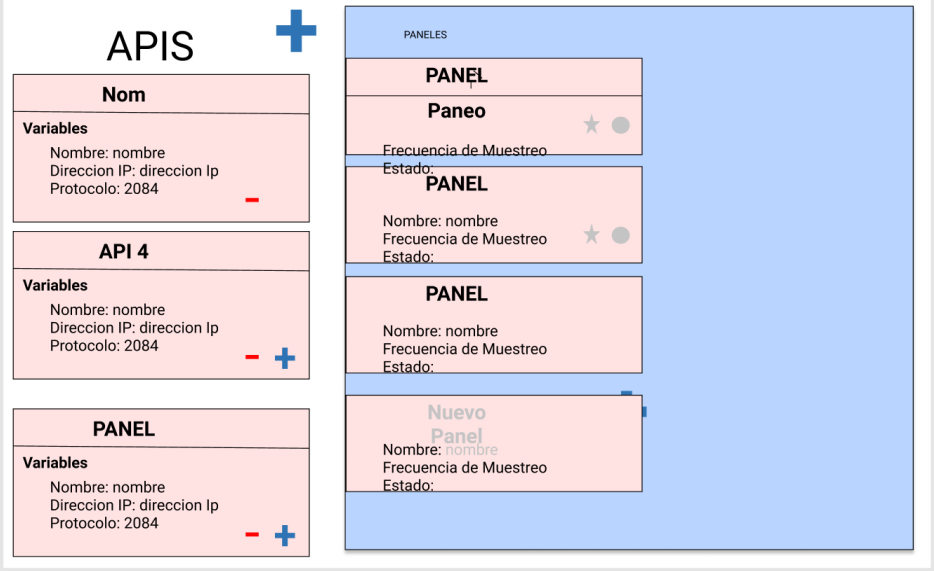


Figura A-4 Diseño Interfaz para apis y paneles(Sistema).

El prototipo presentado nos presenta el ingreso de las variables, operaciones mediante tablas y la navegación entre las api y paneles mediante contenedores que se muestra en la Figura A-4, estas características son las usadas para la implementación del editor gráfico en la interfaz de sistema y api , debido a la facilidad que Obeo Designer ofrece con la implementación de estas características

B. Prototipo de baja fiabilidad (Alternativa 2)

A continuación, se da a conocer la segunda alternativa, para con el diseño del sistema a desarrollar de generación de sistemas SCADA. Este diseño a diferencia con el anterior hace uso de contenedores y listas para agregar elementos dentro de una api y demás elementos.

En la Figura B-1 se muestra una captura del diseño de la interfaz de API 1, donde los equipos se muestran cómo contenedores a diferencia de la primera alternativa, además de que posee un menú lateral para agregar los elemento gráficos(Equipos) haciendo clic y arrastrando el elemento. En este caso cada elemento que conforma cada equipo se agrega en una lista de variables y tablas de operaciones, cada elemento que conforman los equipos presenta botones para eliminar y agregar datos.

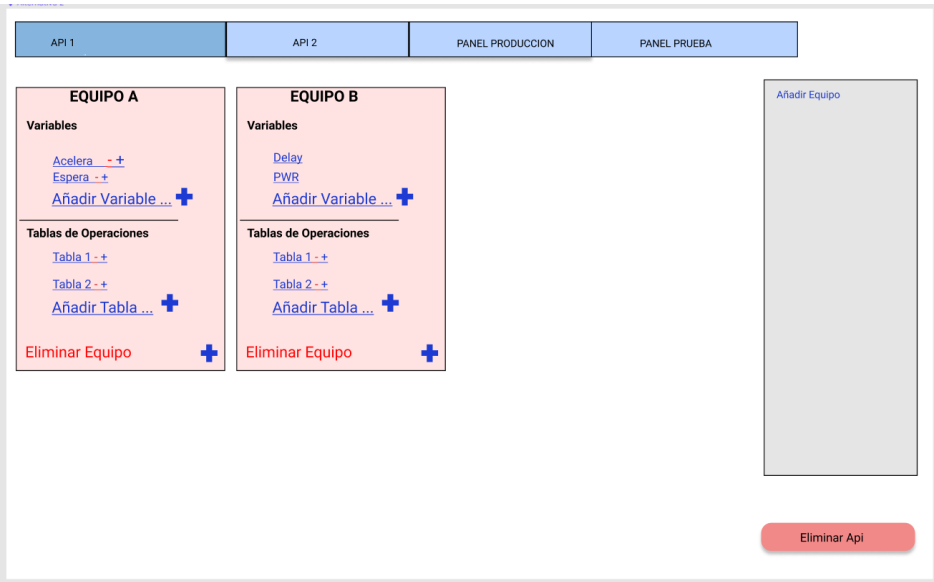


Figura B-1 Diseño Interfaz para Api

La siguiente Figura B-2 da a conocer una captura del diseño de la ventana emergente, para agregar una tabla de operaciones y sus atributos, el cambio del diseño de la primera alternativa radica que las operaciones ya no se agregan como tabla si no como una lista de elementos.

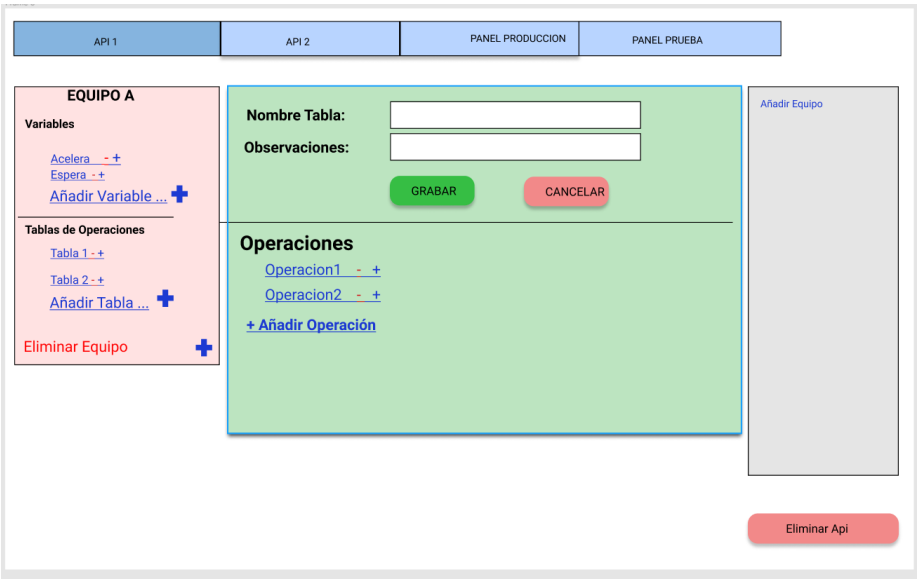


Figura B-2 Diseño ventana emergente para agregar tabla de operaciones

A continuación, en la Figura B-3 se muestra una captura del diseño de la ventana emergente, para agregar una variable con label y entries a un equipo determinado, las

variables se muestran como una lista dentro del contenedor de equipo. El cual tiene un divisor entre variables y tablas de operaciones:

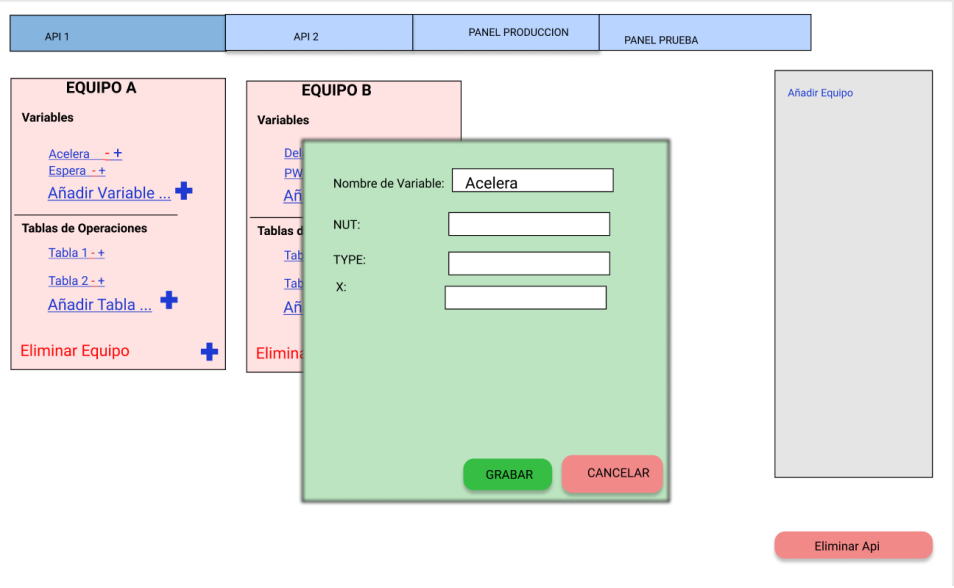


Figura B-3 Diseño ventana emergente para agregar variable

En la Figura B-4 se muestra una captura de la ventana emergente para crear un equipo con sus atributos de nombre , dirección y puerto, el diseño no presenta cambios con la primera alternativa respecto a la ventana emergente.

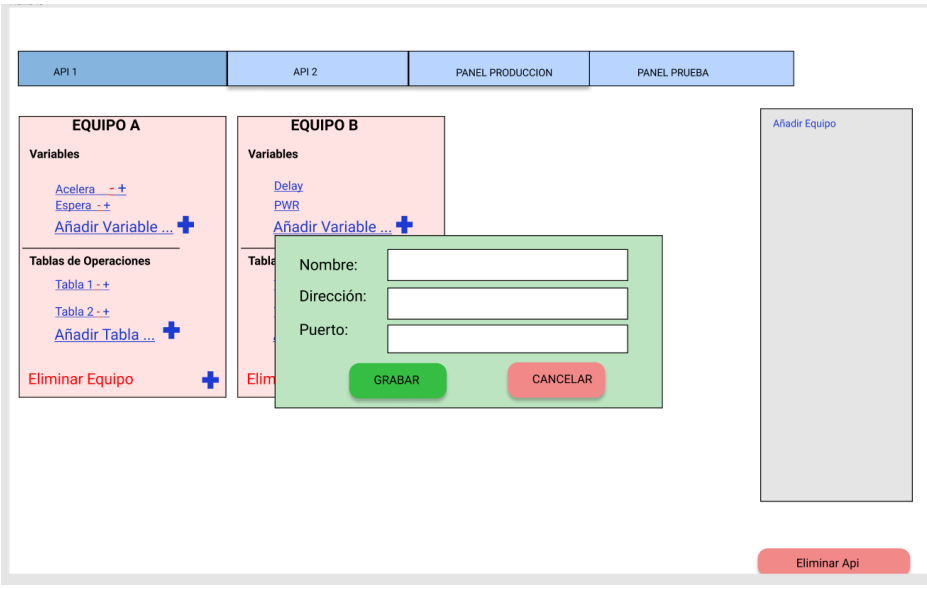


Figura B-4 Diseño ventana emergente para agregar equipo

En cambio el contenedor de equipo tiene dos secciones separadas una de ellas pertenece a las tablas de operaciones creadas como se muestran en la Figura B-4. En esta sección existe un botón para agregar tablas de operaciones que abre la misma ventana que en la primera alternativa con el cambio de que los datos de la operación se agregar mediante una ventana emergente, En la Figura B-5 se muestra una captura del diseño de la ventana emergente que es la diferencia con la primera alternativa que agrega las operaciones con una tabla:

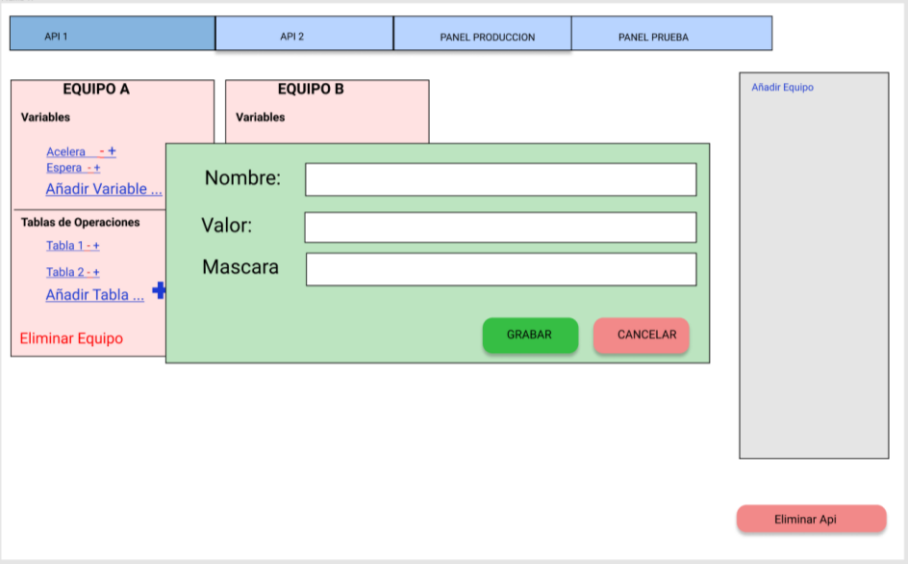


Figura B-5 Diseño ventana emergente para agregar una operación en la tabla de operaciones

En el caso de la asociación de variables y tablas de operaciones en las 2 alternativas existentes es igual, ya que las 2 presentan botones que abren una ventana para la asociación en la Figura B-6 se muestra una captura del diseño de la ventana emergente que asocia una variable con una tabla de operaciones existente.

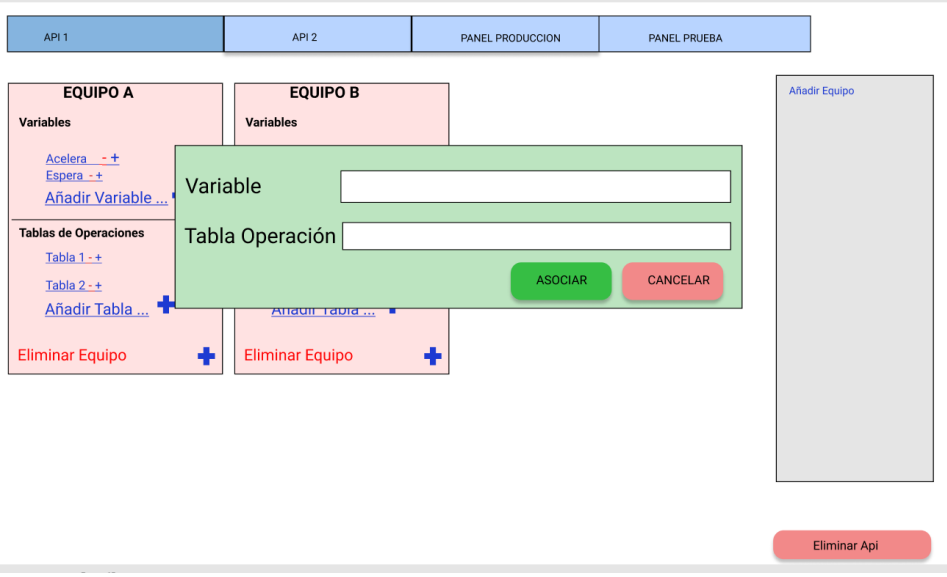


Figura B-6 Diseño ventana emergente para asociar variables y tabla de operaciones.

Este prototipo presentado sirve para obtener los elementos gráficos necesarios para la implementación del editor, ya que la alternativa nos presentan un menú lateral para el ingreso de elementos gráficos a cada api o panel creado estas características son las usadas para la implementación del editor gráfico en la ventanas emergentes para la modificación de los atributos de los equipos y la agregación de equipos, debido a la facilidad que Obeo Designer ofrece con la implementación de estas características .

C. Archivo generate.mtl del motor de transformación

```
[comment encoding = UTF-8 /]

[module generate('http://www.example.org/scada')]

[import TransformacionScada::ConectarApis /]

[import TransformacionScada::IniciarTimer /]

[import TransformacionScada::AsignarTimers /]

[import TransformacionScada::CrearFuncionesAValorActual /]

[import TransformacionScada::crearFuncionesLecturaApi /]
```

```
[template public generateElement(aSistema : Sistema)]



[comment @main /]


[file (deleteSpace(aSistema.nombre)+'.m', false, 'UTF-8')]
     function SistemaUcuenca
          [for (api : Api | aSistema.api)]
               global mb[deleteSpace(api.nombre)/] ;
          global timerAPI[deleteSpace(api.nombre)/] ;
          [/for]
               hs = addcomponents; % Make figure visible after
adding components
          hs.fig.Visible = 'on';
               asignarTimers(hs);
               conectarApis();
          iniciarTimers();
               function hs = addcomponents  % add components,
save handles in a struct
                    posicion= ['['/] 0 -1000 50 30 [']'/]; %
posicion y size
                    [for (p : Panel | aSistema.paneles)]
                         [desplazarDerecha(p)/]
                    [/for]
                    [for (p : Panel | aSistema.paneles)]
                         [rotarEjes(p)/]
                    [/for]
```

```
                    hs.fig = figure('Name','Sistema Balzay'
,'NumberTitle','off','Visible','on','Resize','on','Tag','fig','Col
or','white','Toolbar','none','Position',[getCorcheteDerecho()/]600
,100,[setSizeWWindows(aSistema)/],[setSizeHWindows(aSistema)/][get
CorcheteIzquierdo()/]);

                [for (p : Panel | aSistema.paneles)]


     hs.panel[deleteSpace(p.nombre)/]=uipanel('Parent',hs.fig,'Tit
le','[p.nombre/]','FontSize',12,'units','pixels','Position',[getCo
rcheteDerecho()/]0 0 [setSizeWWindows(aSistema)/]
[setSizeHWindows(aSistema)/][getCorcheteIzquierdo()/]);

                [/for]

                [for (p : Panel | aSistema.paneles)]

                     [for (elem : Elemento | p.elementos)]

                     [if (isEquipoVisual(elem))]

                          [let var : EquipoVisual
=getEquipoVisual(elem) ]

                               %Equipo Visual


     [getCorcheteDerecho()/]x,map[getCorcheteIzquierdo()/]=imread(
'imagenes/[var.imagen.substring(31) /]');

                    E[var.id/]=imresize(x,
[getCorcheteDerecho()/][var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]);


     posicion=[getCorcheteDerecho()/][var.eGet('x')/]
[var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size


     hs.EquipoVisual[var.id/]=uicontrol('Parent',hs.panel[deleteSp
ace(p.nombre)/],'style','pushbutton','units','pixels','position',p
osicion,'cdata',E[var.id/],'enable','inactive');

                          [/let]

                    [elseif (isTexto(elem))]
```

```
                              [let var : TextoDecoracion
=getTexto(elem) ]

                                   %Texto Decoracion

                                    posicion=[getCorcheteDerecho()/]
[var.eGet('x')/] [var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size


hs.TextoDecoracion[var.id/]=uicontrol('Parent',hs.panel[deleteSpac
e(p.nombre)/],'style','text','units','pixels','position',posicion,
'String','[var.contenido/]','FontSize',[var.fontSize/]);

                              [/let]

                         [elseif (isInputText(elem))]

                              [let var : InputText =getInputText(elem)
]

                                   %Input Text

                                   posicion=[getCorcheteDerecho()/]
[var.eGet('x')/] [var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size


     hs.InputeText[var.id/]=uicontrol('Parent',hs.panel[deleteSpac
e(p.nombre)/],'style','edit','units','pixels','position',posicion,
'enable','inactive','tag','[var.escribeVariable.Address/]-
[deleteSpace(var.escribeVariable.equipo.apiContains.nombre)/]');

                              [/let]

                         [elseif (isVarTexto(elem))]

                              [let var : TextoLectura
=getVarTexto(elem) ]

                                   %Texto lectura

                                   posicion=[getCorcheteDerecho()/]
[var.eGet('x')/] [var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size


     hs.InputeText[var.id/]=uicontrol('Parent',hs.panel[deleteSpac
e(p.nombre)/],'style','edit','units','pixels','position',posicion,
```

```
'enable','inactive','tag','[var.visualizaVariable.Address/]-
[deleteSpace(var.visualizaVariable.equipo.apiContains.nombre)/]');

                              [/let]

                         [elseif (isLineasV(elem))]

                              [let var : LineasDecoracionVertical =
getLineasV(elem) ]

                                   %Linea Vertical

                                   posicion=[getCorcheteDerecho()/]
[var.eGet('x')/] [var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size


     hs.lineaVertical[var.id/]=uicontrol('Parent',hs.panel[deleteS
pace(p.nombre)/],'style','pushbutton','units','pixels','position',
posicion,'enable','inactive','background','[getColor(var.color.toS
tring())/]');

                              [/let]

                         [elseif (isLineasH(elem))]

                              [let var : LineasDecoracionHorizontal =
getLineasH(elem) ]

                                   %Linea Vertical

                                   posicion=[getCorcheteDerecho()/]
[var.eGet('x')/] [var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size


     hs.LineaVertical[var.id/]=uicontrol('Parent',hs.panel[deleteS
pace(p.nombre)/],'style','pushbutton','units','pixels','position',
posicion,'enable','inactive','background','[getColor(var.color.toS
tring())/]');

                              [/let]

                         [elseif (isBottomValorConstante(elem))]

                              [let var : BottonValorConstante =
getBottomValorConstante(elem) ]

                                   %Bottom valor constante
```

```
     [getCorcheteDerecho()/]x,map[getCorcheteIzquierdo()/]=imread(
'imagenes/[var.imagen.substring(31) /]');

                                B[var.id/]=imresize(x,
[getCorcheteDerecho()/][var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]);

                                posicion=[getCorcheteDerecho()/]
[var.eGet('x')/] [var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size


     hs.BottomValorConstante[var.id/]=uicontrol('Parent',hs.panel[
deleteSpace(p.nombre)/],'style','pushbutton','units','pixels','pos
ition',posicion,'cdata',B[var.id/],'enable','on','CallBack',{@call
BackBotton[var.id/]});

                           [/let]

                      [elseif (isImagen(elem))]

                           [let var : DecoracionImagen =
getImagen(elem) ]

                                % Imagen


     [getCorcheteDerecho()/]x,map[getCorcheteIzquierdo()/]=imread(
'imagenes/[var.urlImagen.substring(31) /]');

                                I[var.id/]=imresize(x,
[getCorcheteDerecho()/][var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]);

                                posicion=[getCorcheteDerecho()/]
[var.eGet('x')/] [var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size


     hs.Imagen[var.id/]=uicontrol('Parent',hs.panel[deleteSpace(p.
nombre)/],'style','pushbutton','units','pixels','position',posicio
n,'cdata',I[var.id/],'enable','inactive');

                           [/let]

                      [elseif (isNavegacion(elem))]
```

```
                        [let var : Navegacion =
getNavegacion(elem) ]


                        [if panelExiste(var.paneles)]
                              % Navegacion
                              posicion=[getCorcheteDerecho()/]
[var.eGet('x')/] [var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size

     hs.botonNavegacion[var.id/]=uicontrol('Parent',hs.panel[delet
eSpace(p.nombre)/],'style','pushbutton','units','pixels','position
',posicion,'enable','on','String','[var.contenido/]','CallBack',{@
callBackBottonNavegacion,hs.fig,hs.panel[deleteSpace(var.paneles.n
ombre)/]});
                              [else]
                              % Navegacion
                              [deleteSpace(var.contenido)/] =
"[var.contenido/]"
                              posicion=[getCorcheteDerecho()/]
[var.eGet('x')/] [var.eGet('y')/] [var.eGet('w')/]
[var.eGet('h')/][getCorcheteIzquierdo()/]; % posicion y size

     hs.botonNavegacion[var.id/]=uicontrol('Parent',hs.panel[delet
eSpace(p.nombre)/],'style','pushbutton','units','pixels','position
',posicion,'enable','on','String','[var.contenido/]','CallBack',{@
callBackBottonNavegacion,hs.fig,[deleteSpace(var.contenido)/]});


                        [/if]


                        [/let]
                  [else]
                  [/if]
```

```
                    [/for]
                [/for]
                end
            end
            % Funcion de conexion mediante Modbus
            function m = conectarModbus(ip, port)
              try
                m = modbus('tcpip',ip,port,'Timeout',1);
              catch
                % En caso de no lograr conectarse devuelve 0
                m = 0;
              end
            end

            function
callBackBottonNavegacion(hObject,event,fig,panel)
                if panel == "Monitor de Variables"
                    MonitordeVariables();
                else

                hijos= allchild(fig);
              for i=1:1:size(hijos)
                if class(hijos(i))=="matlab.ui.container.Panel" &&
panel~=hijos(i)
                 hijos(i).Visible='Off';
                end
                if class(hijos(i))=="matlab.ui.container.Panel" &&
panel==hijos(i)
```

```
                hijos(i).Visible='On';
            end
        end

            end

    end

    function MonitordeVariables

    fig = figure('Position',['['/]600 200 1100
640[']'/],'Name','Monitor de Variables API','NumberTitle','off');

    p = uipanel('Title','Graficas de
Variaciones','FontSize',12,...
            'BackgroundColor','white',...
            'Position',['['/].0 .0 .67 1.0[']'/]);

    ax = axes(p,'Position',['['/]0.1 0.1 0.8 0.8[']'/]);
    title(ax,'Grafica de Variable');
    ylabel(ax,'Variable');
    xlabel(ax,'Tiempo(seg)');

    dateEditInicioBoxHandle = uicontrol(fig, 'Style',
'Edit', ...
    'Position', ['['/]835 598 150 22[']'/], ...
    'BackgroundColor', 'w');
```

```
            dateInicioTextHandle = uicontrol(fig, 'Style', 'Text',
...
            'String', 'Fecha:', ...
            'Horizontalalignment', 'left', ...
            'Position', ['['/']780 596 50 22[']'/']);

            calendarButtonHandle = uicontrol(fig, 'Style',
'PushButton', ...
            'String', 'Seleccionar', ...
            'Position', ['['/']995 598 70 22[']'/], ...
            'callback',
{@buttonSelectFecha,dateEditInicioBoxHandle});

            dateFinEditBoxHandle = uicontrol(fig, 'Style', 'Edit',
...
            'Position', ['['/']835 571 150 22[']'/], ...
            'BackgroundColor', 'w');

            dateFinTextHandle = uicontrol(fig, 'Style', 'Text', ...
            'String', 'Fecha:', ...
            'Horizontalalignment', 'left', ...
            'Position', ['['/']780 567 50 22[']'/]);

            calendarButtonHandle2 = uicontrol(fig, 'Style',
'PushButton', ...
            'String', 'Seleccionar', ...
            'Position', ['['/']995 571 70 22[']'/], ...
            'callback', {@buttonSelectFecha,dateFinEditBoxHandle});
```

```
            start =
uicontrol(fig,'Style','pushbutton','Position',['['/]968 540 100
22[']'/],'String', 'Analizar','Callback',{@startMonitoreo});



            connApi = database('registosVariables','','');

            queryApi = 'Select DISTINCT api From registro';

            execQueryApi = exec(connApi,queryApi);

            execQueryApi = fetch(execQueryApi);

            %dataApi = execQueryApi.Data(['['/]2:3, 3:end[']'/]);

            dataApi = execQueryApi.Data();

            close(connApi);


            uicontrol(fig, 'Style', 'Text','String',
'Api:','Horizontalalignment', 'left','Position', ['['/]780 540 50
22[']'/]);

            yourcellVariables= ['['/]' '[']'/];

            yourcellEquipos = ['['/]' '[']'/];

            yourcellTipo= ['['/]' '[']'/];

            apiSelect = uicontrol('Style',
'popupmenu','Position',['['/]836 540 100 22[']'/],...

            'string',dataApi,'Callback',{@selectionApi,dataApi});



            uicontrol(fig, 'Style', 'Text', 'String',
'Equipo:','Horizontalalignment', 'left','Position', ['['/]780 513
50 22[']'/]);

            equipoSelect= uicontrol('Style',
'popupmenu','Position',['['/]836 513 100 22[']'/],...
```

```
     'string',yourcellEquipos,'Callback',{@selectionEquipo,yourcel
lEquipos,apiSelect});


          uicontrol(fig, 'Style', 'Text', 'String',
'Variable:','Horizontalalignment', 'left','Position', ['['/]780
486 50 22[']'/]);

          variableSelect = uicontrol('Style',
'popupmenu','Position',['['/]836 486 100 22[']'/],...

          'string',yourcellVariables);

          function selectionApi(hObj,event,yourcellApi)

            v=get(hObj,'value')

            apiName = yourcellApi(v);

            if apiName ~= "null"

              conn = database('registosVariables','','');

              queryEquipo = strcat("Select DISTINCT equipo From
registro where (registro.api = '",apiName,"');")

              ['['/]data[']'/] = select(conn,queryEquipo)

              yourcellEquipos = data.equipo;

              equipoSelect.String = yourcellEquipos;

              close(conn);

            end

          end

          function
selectionEquipo(hObj,event,yourcellEquipo,selectApi)

            v=get(hObj,'value')

            apiName = dataApi(selectApi.Value);

            disp(yourcellEquipo)

            equipoName =
yourcellEquipos(get(equipoSelect,'value'));
```

```
            if apiName ~= "null" && equipoName ~= "null"

              conn = database('registosVariables','','');

              queryVariable = strcat("Select DISTINCT nombre ,
tipo From registro where (registro.api = '",apiName,"' AND
registro.equipo = '",equipoName,"');");

              disp(queryVariable)

              ['['/]data[']'/] = select(conn,queryVariable)

              yourcellVariables = data.nombre;

              yourcellTipo = data.tipo;

              variableSelect.String = yourcellVariables;

              close(conn);

            end

          end

          function startMonitoreo(hObj,event)

            apiNombre = dataApi{get(apiSelect,'value')};

            varNombre =
yourcellVariables(get(variableSelect,'value'));

            tipoNombre =
yourcellTipo(get(variableSelect,'value'));

            equipoName =
yourcellEquipos(get(equipoSelect,'value'));

            if (dateEditInicioBoxHandle.String ~="") &&
(dateFinEditBoxHandle.String ~= "") && (apiNombre ~= "") &&
(varNombre ~= " ") && (equipoName ~= " ")



              conn = database('registosVariables','','');

              queryVariable = strcat("Select valor , fecha From
registro where registro.api = '",apiNombre,"'", " and
registro.nombre = '",varNombre, "' and registro.equipo =
'",equipoName,"' and registro.fecha
```

```
>='",dateEditInicioBoxHandle.String,"' and registro.fecha <=
'",dateFinEditBoxHandle.String,"'");
                ['['/]data[']'/] = select(conn,queryVariable)
                close(conn);
                if ~isempty(data)
                  x = ['['/][']'/];
                  for v = 1:length(data.valor)
                    auxvarV= data.valor(v);
                    x(length(x)+1) = auxvarV;
                  end
                  hold on;
                  title(ax,'Grafica de Variable');
                  ylabel(ax,strcat(varNombre," ", tipoNombre));
                  xlabel(ax,'Tiempo');
                  plot(ax,datetime(data.fecha),x);
                end


              end
            end
            end


            function buttonSelectFecha(hcbo,
eventStruct,dateEditBoxHandle)
              uicalendar('Weekend', ['['/]1 0 0 0 0 0 1[']'/], ...
              'SelectionType', 1, ...
              'DestinationUI', dateEditBoxHandle, ...
              'OutputDateFormat','yyyy-mm-dd HH:MM:SS');
```

```
            end

      [asignarTimer(aSistema)/]

      [conectarApis(aSistema)/]

      [iniciarTimer(aSistema)/]

      [crearFuncionesBVA(aSistema)/]

      [leerDatosApi(aSistema)/]

[/file]

[/template]

[query public getH(eClass : Panel) : Integer =
invoke('TransformacionScada.Services', 'getH(scada.Panel)',
Sequence{eClass})/]


[query public panelExiste(eClass : Panel) : Boolean =
invoke('TransformacionScada.Services', 'panelExiste(scada.Panel)',
Sequence{eClass})/]


[query public setSizeWWindows(eClass : Sistema) : Integer =
invoke('TransformacionScada.Services',
'setSizeWWindows(scada.Sistema)', Sequence{eClass})/]

[query public setSizeHWindows(eClass : Sistema) : Integer =
invoke('TransformacionScada.Services',
'setSizeHWindows(scada.Sistema)', Sequence{eClass})/]


[query public getW(eClass : Panel) : Integer =
invoke('TransformacionScada.Services', 'getW(scada.Panel)',
Sequence{eClass})/]

[query public getCorcheteDerecho() : String =
invoke('TransformacionScada.Services', 'getCorcheteDerecho()',
Sequence{})/]

[query public getCorcheteIzquierdo() : String =
invoke('TransformacionScada.Services', 'getCorcheteIzquierdo()',
Sequence{})/]
```

```
[query public isEquipoVisual(eClass : Elemento) : Boolean =
invoke('TransformacionScada.Services',
'isEquipoVisual(scada.Elemento)', Sequence{eClass})/]

[query public getEquipoVisual(eClass : Elemento) : EquipoVisual =
invoke('TransformacionScada.Services',
'getEquipoVisual(scada.Elemento)', Sequence{eClass})/]

[query public isTexto(eClass : Elemento) : Boolean =
invoke('TransformacionScada.Services', 'isTexto(scada.Elemento)',
Sequence{eClass})/]

[query public getTexto(eClass : Elemento) : TextoDecoracion =
invoke('TransformacionScada.Services', 'getTexto(scada.Elemento)',
Sequence{eClass})/]

[query public isInputText(eClass : Elemento) : Boolean =
invoke('TransformacionScada.Services',
'isInputText(scada.Elemento)', Sequence{eClass})/]

[query public getInputText(eClass : Elemento) : InputText =
invoke('TransformacionScada.Services',
'getInputText(scada.Elemento)', Sequence{eClass})/]

[query public isLineasV(eClass : Elemento) : Boolean =
invoke('TransformacionScada.Services',
'isLineasV(scada.Elemento)', Sequence{eClass})/]

[query public getLineasV(eClass : Elemento) :
LineasDecoracionVertical = invoke('TransformacionScada.Services',
'getLineasV(scada.Elemento)', Sequence{eClass})/]

[query public isLineasH(eClass : Elemento) : Boolean =
invoke('TransformacionScada.Services',
'isLineasH(scada.Elemento)', Sequence{eClass})/]

[query public getLineasH(eClass : Elemento) :
LineasDecoracionHorizontal =
invoke('TransformacionScada.Services',
'getLineasH(scada.Elemento)', Sequence{eClass})/]

[query public isBottomValorConstante(eClass : Elemento) : Boolean
= invoke('TransformacionScada.Services',
'isBottomValorConstante(scada.Elemento)', Sequence{eClass})/]

[query public getBottomValorConstante(eClass : Elemento) :
BottonValorConstante = invoke('TransformacionScada.Services',
'getBottomValorConstante(scada.Elemento)', Sequence{eClass})/]
```

```
[query public isImagen(eClass : Elemento) : Boolean =
invoke('TransformacionScada.Services', 'isImagen(scada.Elemento)',
Sequence{eClass})/]

[query public getImagen(eClass : Elemento) : DecoracionImagen =
invoke('TransformacionScada.Services',
'getImagen(scada.Elemento)', Sequence{eClass})/]

[query public isVarTexto(eClass : Elemento) : Boolean =
invoke('TransformacionScada.Services',
'isVarTexto(scada.Elemento)', Sequence{eClass})/]

[query public getVarTexto(eClass : Elemento) : TextoLectura =
invoke('TransformacionScada.Services',
'getVarTexto(scada.Elemento)', Sequence{eClass})/]

[query public isNavegacion(eClass : Elemento) : Boolean =
invoke('TransformacionScada.Services',
'isNavegacion(scada.Elemento)', Sequence{eClass})/]

[query public getNavegacion(eClass : Elemento) : Navegacion =
invoke('TransformacionScada.Services',
'getNavegacion(scada.Elemento)', Sequence{eClass})/]


[query public deleteSpace(eClass : String) : String =
invoke('TransformacionScada.Services',
'deleteSpace(java.lang.String)', Sequence{eClass})/]

[query public getEstadoPanel(eClass : Integer) : String =
invoke('TransformacionScada.Services',
'getEstadoPanel(java.lang.Integer)', Sequence{eClass})/]

[query public getColor(eClass : String) : String =
invoke('TransformacionScada.Services',
'getColor(java.lang.String)', Sequence{eClass})/]

[query public rotarEjes(eClass : Panel) : OclVoid =
invoke('TransformacionScada.Services', 'rotarEjes(scada.Panel)',
Sequence{eClass})/]

[query public desplazarDerecha(eClass : Panel) : OclVoid =
invoke('TransformacionScada.Services',
'desplazarDerecha(scada.Panel)', Sequence{eClass})/]
```

D. Código fuente del sistema Scada fotovoltaicos

```
function SistemaUcuenca
        global mbFotovoltaicos ;
    global timerAPIFotovoltaicos ;
        global mbAnalizadordeRed ;
    global timerAPIAnalizadordeRed ;
        global mbBateriaLitio ;
    global timerAPIBateriaLitio ;
        global mbBateriaPlomo ;
    global timerAPIBateriaPlomo ;
        global mbSuperCondensadores ;
    global timerAPISuperCondensadores ;
        hs = addcomponents; % Make figure visible after
adding components
    hs.fig.Visible = 'on';
        asignarTimers(hs);
        conectarApis();
    iniciarTimers();
        function hs = addcomponents  % add components,
save handles in a struct
            posicion= [ 0 -1000 50 30 ]; % posicion y
size
            hs.fig = figure('Name','Sistema Balzay'
,'NumberTitle','off','Visible','on','Resize','on','Tag','fig','Col
or','white','Toolbar','none','Position',[600,100,1140.0,758.0]);

    hs.panelFotovoltaicos=uipanel('Parent',hs.fig,'Title','Fotovo
ltaicos','FontSize',12,'units','pixels','Position',[0 0 1140.0
758.0]);
```

```
     hs.panelBaterias=uipanel('Parent',hs.fig,'Title','Baterias','
FontSize',12,'units','pixels','Position',[0 0 1140.0 758.0]);

                                %Input Text

                                posicion=[ 264.0 632.0 60.0 30.0];
% posicion y size

     hs.InputeTexteGj9oGD2EeuEZaODg90yXA=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','edit','units','pixels','position',posicio
n,'enable','inactive','tag','400003-Fotovoltaicos');


                                %Input Text

                                posicion=[ 264.0 584.0 60.0 30.0];
% posicion y size

     hs.InputeTextuvmEmD3EeuEZaODg90yXA=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','edit','units','pixels','position',posicion
,'enable','inactive','tag','400005-Fotovoltaicos');


                                %Texto Decoracion

                                 posicion=[ 336.0 632.0 73.0
30.0]; % posicion y size


hs.TextoDecoracionMrAUkGHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','text','units','pixels','position',posicion
,'String','VDC V','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 612.0 632.0 74.0
30.0]; % posicion y size


hs.TextoDecoracionQbm4YGHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','text','units','pixels','position',posicion
,'String','VDC V','FontSize',12);
```

```
                                %Texto Decoracion

                                 posicion=[ 912.0 632.0 73.0
30.0]; % posicion y size

hs.TextoDecoracionSOgZYGHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','text','units','pixels','position',posicion
,'String','VDC V','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 912.0 584.0 73.0
30.0]; % posicion y size

hs.TextoDecoracionUp1MUGHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','text','units','pixels','position',posicion
,'String','PDC KW','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 336.0 584.0 73.0
30.0]; % posicion y size

hs.TextoDecoracionYLvz4GHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','text','units','pixels','position',posicion
,'String','PDC KW','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 612.0 584.0 74.0
30.0]; % posicion y size

hs.TextoDecoracionbIkxYGHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','text','units','pixels','position',posicion
,'String','PDC KW','FontSize',12);
```

```
                                %Input Text

                                posicion=[ 552.0 632.0 60.0 30.0];
% posicion y size

     hs.InputeTextd0WbKmHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','edit','units','pixels','position',posicion
,'enable','inactive','tag','400103-Fotovoltaicos');


                                %Input Text

                                posicion=[ 552.0 584.0 60.0 30.0];
% posicion y size

     hs.InputeTexte38mImHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','edit','units','pixels','position',posicion
,'enable','inactive','tag','400105-Fotovoltaicos');


                                %Input Text

                                posicion=[ 840.0 632.0 60.0 30.0];
% posicion y size

     hs.InputeTextlI7cCmHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','edit','units','pixels','position',posicion
,'enable','inactive','tag','400302-Fotovoltaicos');


                                %Input Text

                                posicion=[ 840.0 584.0 60.0 30.0];
% posicion y size

     hs.InputeTextsB1NiWHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.pane
lFotovoltaicos,'style','edit','units','pixels','position',posicion
,'enable','inactive','tag','400304-Fotovoltaicos');


                                % Imagen
```

```
[x,map]=imread('imagenes/FV1.PNG');

                                IxeRoGmHZEeuGIbEIiLYpQ=imresize(x,
[100.0 70.0]);

                                posicion=[ 264.0 496.0 100.0
70.0]; % posicion y size

hs.ImagenxeRoGmHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.panelFot
ovoltaicos,'style','pushbutton','units','pixels','position',posici
on,'cdata',IxeRoGmHZEeuGIbEIiLYpQ,'enable','inactive');


                                % Imagen

[x,map]=imread('imagenes/FV2.PNG');

                                IztyzrGHZEeuGIbEIiLYpQ=imresize(x,
[100.0 70.0]);

                                posicion=[ 552.0 496.0 100.0
70.0]; % posicion y size

hs.ImagenztyzrGHZEeuGIbEIiLYpQ=uicontrol('Parent',hs.panelFot
ovoltaicos,'style','pushbutton','units','pixels','position',posici
on,'cdata',IztyzrGHZEeuGIbEIiLYpQ,'enable','inactive');


                                % Imagen

[x,map]=imread('imagenes/FV3.PNG');

                                I2AkHj2HZEeuGIbEIiLYpQ=imresize(x,
[100.0 70.0]);

                                posicion=[ 840.0 496.0 100.0
70.0]; % posicion y size

hs.Imagen2AkHj2HZEeuGIbEIiLYpQ=uicontrol('Parent',hs.panelFot
ovoltaicos,'style','pushbutton','units','pixels','position',posici
on,'cdata',I2AkHj2HZEeuGIbEIiLYpQ,'enable','inactive');
```

```
                                % Imagen
                                [x,map]=imread('imagenes/RED-
ELECTRICA.PNG');
                                I3HiPUmHcEeuGIbEIiLYpQ=imresize(x,
[100.0 70.0]);
                                posicion=[ 0.0 199.0 100.0 70.0];
% posicion y size

     hs.Imagen3HiPUmHcEeuGIbEIiLYpQ=uicontrol('Parent',hs.panelFot
ovoltaicos,'style','pushbutton','units','pixels','position',posici
on,'cdata',I3HiPUmHcEeuGIbEIiLYpQ,'enable','inactive');


                                % Imagen
                                [x,map]=imread('imagenes/INVERSOR-
FV-TOP.PNG');
                                I7mdRtGHcEeuGIbEIiLYpQ=imresize(x,
[101.0 103.0]);
                                posicion=[ 263.0 343.0 101.0
103.0]; % posicion y size

     hs.Imagen7mdRtGHcEeuGIbEIiLYpQ=uicontrol('Parent',hs.panelFot
ovoltaicos,'style','pushbutton','units','pixels','position',posici
on,'cdata',I7mdRtGHcEeuGIbEIiLYpQ,'enable','inactive');


                                % Imagen
                                [x,map]=imread('imagenes/INVERSOR-
FV-TOP.PNG');
                                IziRtmHcEeuGIbEIiLYpQ=imresize(x,
[100.0 70.0]);
                                posicion=[ 552.0 376.0 100.0
70.0]; % posicion y size
```

```
    hs.ImagenziRtmHcEeuGIbEIiLYpQ=uicontrol('Parent',hs.panelFoto
voltaicos,'style','pushbutton','units','pixels','position',posicio
n,'cdata',IziRtmHcEeuGIbEIiLYpQ,'enable','inactive');


                                % Imagen

                                [x,map]=imread('imagenes/INVERSOR-
FV-TOP.PNG');

                                IBaYI2HdEeuGIbEIiLYpQ=imresize(x,
[100.0 94.0]);

                                posicion=[ 840.0 352.0 100.0
94.0]; % posicion y size


    hs.ImagenBaYI2HdEeuGIbEIiLYpQ=uicontrol('Parent',hs.panelFoto
voltaicos,'style','pushbutton','units','pixels','position',posicio
n,'cdata',IBaYI2HdEeuGIbEIiLYpQ,'enable','inactive');


                                %Equipo Visual

                      [x,map]=imread('imagenes/CONEXCION-RED-
ELECTRICA-ON.PNG');

                      EfpU0GHeEeuGIbEIiLYpQ=imresize(x, [100.0
70.0]);

                      posicion=[144.0 196.0 100.0 70.0]; % posicion
y size


    hs.EquipoVisualfpU0GHeEeuGIbEIiLYpQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','pushbutton','units','pixels','position',p
osicion,'cdata',EfpU0GHeEeuGIbEIiLYpQ,'enable','inactive');


                                %Linea Vertical

                                posicion=[ 243.0 234.0 778.0 8.0];
% posicion y size


    hs.LineaVerticalkwjxUGJzEeuVZoCkRgAswQ=uicontrol('Parent',hs.
```

```
panelFotovoltaicos,'style','pushbutton','units','pixels','position
',posicion,'enable','inactive','background','black');


                                %Linea Vertical

                                posicion=[ 99.0 231.0 46.0 11.0];
% posicion y size

     hs.LineaVerticalqfCIWJzEeuVZoCkRgAswQ=uicontrol('Parent',hs.p
anelFotovoltaicos,'style','pushbutton','units','pixels','position'
,posicion,'enable','inactive','background','black');


                                %Texto Decoracion

                                 posicion=[ 267.0 313.0 97.0
30.0]; % posicion y size

hs.TextoDecoracionDAHgMGJ0EeuVZoCkRgAswQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','text','units','pixels','position',posicio
n,'String','Inversor FV1','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 555.0 313.0 94.0
30.0]; % posicion y size

hs.TextoDecoracionJ5ucUGJ0EeuVZoCkRgAswQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','text','units','pixels','position',posicio
n,'String','Inversor FV2','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 840.0 320.0 97.0
30.0]; % posicion y size

hs.TextoDecoracionPixIcGJ0EeuVZoCkRgAswQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','text','units','pixels','position',posicio
n,'String','Inversor FV3','FontSize',12);
```

```
                                % Imagen
                                [x,map]=imread('imagenes/CONEXION-
INVERSON-TOP.PNG');

     IZAiqd2J0EeuVZoCkRgAswQ=imresize(x, [47.0 65.0]);
                                posicion=[ 290.0 249.0 47.0 65.0];
% posicion y size

     hs.ImagenZAiqd2J0EeuVZoCkRgAswQ=uicontrol('Parent',hs.panelFo
tovoltaicos,'style','pushbutton','units','pixels','position',posic
ion,'cdata',IZAiqd2J0EeuVZoCkRgAswQ,'enable','inactive');


                                % Imagen
                                [x,map]=imread('imagenes/CONEXION-
INVERSON-TOP.PNG');

     IfAsPBGJ0EeuVZoCkRgAswQ=imresize(x, [49.0 72.0]);
                                posicion=[ 576.0 242.0 49.0 72.0];
% posicion y size

     hs.ImagenfAsPBGJ0EeuVZoCkRgAswQ=uicontrol('Parent',hs.panelFo
tovoltaicos,'style','pushbutton','units','pixels','position',posic
ion,'cdata',IfAsPBGJ0EeuVZoCkRgAswQ,'enable','inactive');


                                % Imagen
                                [x,map]=imread('imagenes/CONEXION-
INVERSON-TOP.PNG');

     IjhGfIWJ0EeuVZoCkRgAswQ=imresize(x, [49.0 80.0]);
                                posicion=[ 865.0 241.0 49.0 80.0];
% posicion y size
```

```
hs.ImagenjhGfIWJ0EeuVZoCkRgAswQ=uicontrol('Parent',hs.panelFo
tovoltaicos,'style','pushbutton','units','pixels','position',posic
ion,'cdata',IjhGfIWJ0EeuVZoCkRgAswQ,'enable','inactive');


                                %Input Text

                                posicion=[ 372.0 416.0 60.0 30.0];
% posicion y size

hs.InputeText0wxQKGJ0EeuVZoCkRgAswQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','edit','units','pixels','position',posicio
n,'enable','inactive','tag','400021-Fotovoltaicos');


                                %Input Text

                                posicion=[ 660.0 416.0 60.0 30.0];
% posicion y size

hs.InputeText3JjFY2J0EeuVZoCkRgAswQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','edit','units','pixels','position',posicio
n,'enable','inactive','tag','400121-Fotovoltaicos');


                                %Input Text

                                posicion=[ 948.0 416.0 60.0 30.0];
% posicion y size

hs.InputeText4hRGT2J0EeuVZoCkRgAswQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','edit','units','pixels','position',posicio
n,'enable','inactive','tag','400320-Fotovoltaicos');


                                %Texto Decoracion

                                 posicion=[ 431.0 416.0 73.0
30.0]; % posicion y size

hs.TextoDecoracionAUGJYGJ1EeuVZoCkRgAswQ=uicontrol('Parent',hs.pan
```

```
elFotovoltaicos,'style','text','units','pixels','position',posicio
n,'String','PAC KW','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 719.0 416.0 73.0
30.0]; % posicion y size

hs.TextoDecoracionGij2wGJ1EeuVZoCkRgAswQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','text','units','pixels','position',posicio
n,'String','PAC KW','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 1008.0 416.0 71.0
30.0]; % posicion y size

hs.TextoDecoracionIXTK0GJ1EeuVZoCkRgAswQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','text','units','pixels','position',posicio
n,'String','PAC KW','FontSize',12);



                                % Navegacion

                                posicion=[ 144.0 48.0 150.0 50.0];
% posicion y size

     hs.botonNavegacionaXbhK6F7EeuV7OIePQZv2w=uicontrol('Parent',h
s.panelFotovoltaicos,'style','pushbutton','units','pixels','positi
on',posicion,'enable','on','String','Fotovoltaico','CallBack',{@ca
llBackBottonNavegacion,hs.fig,hs.panelFotovoltaicos});



                                % Navegacion
```

```
                                posicion=[ 298.0 48.0 150.0 50.0];
% posicion y size

     hs.botonNavegaciongVk6qKF7EeuV7OIePQZv2w=uicontrol('Parent',h
s.panelFotovoltaicos,'style','pushbutton','units','pixels','positi
on',posicion,'enable','on','String','Baterias','CallBack',{@callBa
ckBottonNavegacion,hs.fig,hs.panelBaterias});




                                % Navegacion

                                MonitordeVariables = "Monitor de
Variables"

                                posicion=[ 456.0 48.0 205.0 50.0];
% posicion y size

     hs.botonNavegacionA9ju8LN7EeudZd0VOcZLA=uicontrol('Parent',hs
.panelFotovoltaicos,'style','pushbutton','units','pixels','positio
n',posicion,'enable','on','String','Monitor de
Variables','CallBack',{@callBackBottonNavegacion,hs.fig,MonitordeV
ariables});




                                %Texto Decoracion

                                 posicion=[ 264.0 674.0 120.0
36.0]; % posicion y size

hs.TextoDecoracionkO1GQMsLEeuVufZgMdbmrQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','text','units','pixels','position',posicio
n,'String','Fotovoltaico 1','FontSize',12);


                                %Texto Decoracion
```

```
                                    posicion=[ 552.0 674.0 132.0
30.0]; % posicion y size

hs.TextoDecoraciontDfZ0MsLEeuVufZgMdbmrQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','text','units','pixels','position',posicio
n,'String','Fotovoltaico 2','FontSize',12);


                                   %Texto Decoracion

                                    posicion=[ 828.0 674.0 144.0
30.0]; % posicion y size

hs.TextoDecoracionvqqQYMsLEeuVufZgMdbmrQ=uicontrol('Parent',hs.pan
elFotovoltaicos,'style','text','units','pixels','position',posicio
n,'String','Fotovoltaico 3','FontSize',12);


                                   % Imagen

                                   [x,map]=imread('imagenes/RED-
ELECTRICA.PNG');

     IEsNQ0M0rEeuUrNkL2944cA=imresize(x, [100.0 70.0]);

                                   posicion=[ 252.0 196.0 100.0
70.0]; % posicion y size

     hs.ImagenEsNQ0M0rEeuUrNkL2944cA=uicontrol('Parent',hs.panelBa
terias,'style','pushbutton','units','pixels','position',posicion,'
cdata',IEsNQ0M0rEeuUrNkL2944cA,'enable','inactive');


                                   %Linea Vertical

                                   posicion=[ 351.0 225.0 778.0
13.0]; % posicion y size

     hs.LineaVerticalKnrsM0rEeuUrNkL2944cA=uicontrol('Parent',hs.p
anelBaterias,'style','pushbutton','units','pixels','position',posi
cion,'enable','inactive','background','black');
```

```
                                % Imagen

     [x,map]=imread('imagenes/BateriaLitio.PNG');

     IPtTmws0rEeuUrNkL2944cA=imresize(x, [100.0 94.0]);

                                posicion=[ 348.0 364.0 100.0
94.0]; % posicion y size

     hs.ImagenPtTmws0rEeuUrNkL2944cA=uicontrol('Parent',hs.panelBa
terias,'style','pushbutton','units','pixels','position',posicion,'
cdata',IPtTmws0rEeuUrNkL2944cA,'enable','inactive');


                                % Imagen

     [x,map]=imread('imagenes/BateriaPlomo.PNG');

     IRi2zJM0rEeuUrNkL2944cA=imresize(x, [100.0 94.0]);

                                posicion=[ 621.0 364.0 100.0
94.0]; % posicion y size

     hs.ImagenRi2zJM0rEeuUrNkL2944cA=uicontrol('Parent',hs.panelBa
terias,'style','pushbutton','units','pixels','position',posicion,'
cdata',IRi2zJM0rEeuUrNkL2944cA,'enable','inactive');


                                % Imagen

     [x,map]=imread('imagenes/SuperCondensadores.PNG');

     IThTc1c0rEeuUrNkL2944cA=imresize(x, [100.0 94.0]);

                                posicion=[ 888.0 364.0 100.0
94.0]; % posicion y size
```

```
     hs.ImagenThTc1c0rEeuUrNkL2944cA=uicontrol('Parent',hs.panelBa
terias,'style','pushbutton','units','pixels','position',posicion,'
cdata',IThTc1c0rEeuUrNkL2944cA,'enable','inactive');



                                % Navegacion

                                posicion=[ 252.0 84.0 150.0 50.0];
% posicion y size

     hs.botonNavegacionYCfJAM0rEeuUrNkL2944cA=uicontrol('Parent',h
s.panelBaterias,'style','pushbutton','units','pixels','position',p
osicion,'enable','on','String','Fotovoltaicos','CallBack',{@callBa
ckBottonNavegacion,hs.fig,hs.panelFotovoltaicos});



                                % Navegacion

                                posicion=[ 408.0 84.0 150.0 50.0];
% posicion y size

     hs.botonNavegacionahWYI80rEeuUrNkL2944cA=uicontrol('Parent',h
s.panelBaterias,'style','pushbutton','units','pixels','position',p
osicion,'enable','on','String','Baterias','CallBack',{@callBackBot
tonNavegacion,hs.fig,hs.panelBaterias});



                                %Texto Decoracion

                                 posicion=[ 447.0 404.0 82.0
30.0]; % posicion y size

hs.TextoDecoraciondaJKMM0rEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','text','units','pixels','position',posicion,'St
ring','PDC KW:','FontSize',12);
```

```
                                %Texto Decoracion

                                 posicion=[ 996.0 404.0 73.0
30.0]; % posicion y size

hs.TextoDecoracionfLZscM0rEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','text','units','pixels','position',posicion,'St
ring','PDC KW:','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 720.0 404.0 73.0
30.0]; % posicion y size

hs.TextoDecoraciongObPsM0rEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','text','units','pixels','position',posicion,'St
ring','PDC KW:','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 447.0 364.0 70.0
30.0]; % posicion y size

hs.TextoDecoracionnej1cM0rEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','text','units','pixels','position',posicion,'St
ring','IDC A:','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 720.0 364.0 61.0
30.0]; % posicion y size

hs.TextoDecoracionqFZU0M0rEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','text','units','pixels','position',posicion,'St
ring','IDC A:','FontSize',12);
```

```
                                %Texto Decoracion

                                 posicion=[ 996.0 364.0 61.0
30.0]; % posicion y size

hs.TextoDecoracion1NrQ0M0rEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','text','units','pixels','position',posicion,'St
ring','IDC A:','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 696.0 296.0 49.0
30.0]; % posicion y size

hs.TextoDecoracion23nqIM0rEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','text','units','pixels','position',posicion,'St
ring','P KW:','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 960.0 296.0 61.0
30.0]; % posicion y size

hs.TextoDecoracion4WmJkM0rEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','text','units','pixels','position',posicion,'St
ring','P KW:','FontSize',12);


                                %Texto Decoracion

                                 posicion=[ 432.0 296.0 61.0
30.0]; % posicion y size

hs.TextoDecoracion5LrgM0rEeuUrNkL2944cA=uicontrol('Parent',hs.panel
Baterias,'style','text','units','pixels','position',posicion,'Stri
ng','P KW:','FontSize',12);


                                % Imagen
```

```
[x,map]=imread('imagenes/flechadobledireccionVertical.PNG');

                                IAUqM0sEeuUrNkL2944cA=imresize(x,
[17.0 93.0]);

                                posicion=[ 384.0 245.0 17.0 93.0];
% posicion y size

hs.ImagenAUqM0sEeuUrNkL2944cA=uicontrol('Parent',hs.panelBate
rias,'style','pushbutton','units','pixels','position',posicion,'cd
ata',IAUqM0sEeuUrNkL2944cA,'enable','inactive');


                                % Imagen

[x,map]=imread('imagenes/flechadobledireccionVertical.PNG');

IGKwdGc0sEeuUrNkL2944cA=imresize(x, [13.0 112.0]);

                                posicion=[ 660.0 242.0 13.0
112.0]; % posicion y size

hs.ImagenGKwdGc0sEeuUrNkL2944cA=uicontrol('Parent',hs.panelBa
terias,'style','pushbutton','units','pixels','position',posicion,'
cdata',IGKwdGc0sEeuUrNkL2944cA,'enable','inactive');


                                % Imagen

[x,map]=imread('imagenes/flechadobledireccionVertical.PNG');

IIqEYEs0sEeuUrNkL2944cA=imresize(x, [13.0 107.0]);

                                posicion=[ 924.0 243.0 13.0
107.0]; % posicion y size

hs.ImagenIqEYEs0sEeuUrNkL2944cA=uicontrol('Parent',hs.panelBa
terias,'style','pushbutton','units','pixels','position',posicion,'
cdata',IIqEYEs0sEeuUrNkL2944cA,'enable','inactive');
```

```
                                %Input Text

                                posicion=[ 528.0 364.0 60.0 30.0];
% posicion y size

     hs.InputeTextet9tFM0sEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','edit','units','pixels','position',posicion,'en
able','inactive','tag','400027-BateriaLitio');


                                %Input Text

                                posicion=[ 528.0 404.0 60.0 30.0];
% posicion y size

     hs.InputeTextiejp1s0sEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','edit','units','pixels','position',posicion,'en
able','inactive','tag','400028-BateriaLitio');


                                %Input Text

                                posicion=[ 516.0 296.0 60.0 30.0];
% posicion y size

     hs.InputeTextm1tQQc0sEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','edit','units','pixels','position',posicion,'en
able','inactive','tag','400044-BateriaLitio');


                                %Input Text

                                posicion=[ 804.0 361.0 60.0 30.0];
% posicion y size

     hs.InputeTextrKuIss0sEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','edit','units','pixels','position',posicion,'en
able','inactive','tag','400004-BateriaPlomo');
```

```
                                %Input Text

                                posicion=[ 768.0 296.0 60.0 30.0];
% posicion y size

     hs.InputeTextuNzfOM0sEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','edit','units','pixels','position',posicion,'en
able','inactive','tag','400021-BateriaPlomo');


                                %Input Text

                                posicion=[ 804.0 404.0 60.0 30.0];
% posicion y size

     hs.InputeTextxXKc8c0sEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','edit','units','pixels','position',posicion,'en
able','inactive','tag','400005-BateriaPlomo');


                                %Input Text

                                posicion=[ 1080.0 361.0 60.0
30.0]; % posicion y size

     hs.InputeText1kmtoc0sEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','edit','units','pixels','position',posicion,'en
able','inactive','tag','400004-SuperCondensadores');


                                %Input Text

                                posicion=[ 1080.0 404.0 60.0
30.0]; % posicion y size

     hs.InputeText68h0oM0sEeuUrNkL2944cA=uicontrol('Parent',hs.pan
elBaterias,'style','edit','units','pixels','position',posicion,'en
able','inactive','tag','400005-SuperCondensadores');


                                %Input Text
```

```
                                    posicion=[ 1035.0 296.0 60.0
30.0]; % posicion y size

     hs.InputeTextySlO80sEeuUrNkL2944cA=uicontrol('Parent',hs.pane
lBaterias,'style','edit','units','pixels','position',posicion,'ena
ble','inactive','tag','400021-SuperCondensadores');


                end

            end

            % Funcion de conexion mediante Modbus

            function m = conectarModbus(ip, port)

              try

                m = modbus('tcpip',ip,port,'Timeout',1);

              catch

                % En caso de no lograr conectarse devuelve 0

                m = 0;

              end

            end


            function
callBackBottonNavegacion(hObject,event,fig,panel)

                if panel == "Monitor de Variables"

                     MonitordeVariables();

                else


                hijos= allchild(fig);

              for i=1:1:size(hijos)

                 if class(hijos(i))=="matlab.ui.container.Panel" &&
panel~=hijos(i)
```

```
                hijos(i).Visible='Off';
                end
                if class(hijos(i))=="matlab.ui.container.Panel" &&
panel==hijos(i)
                  hijos(i).Visible='On';
                end
              end


                end


            end


            function MonitordeVariables


            fig = figure('Position',[600 200 1100
640],'Name','Monitor de Variables API','NumberTitle','off');


            p = uipanel('Title','Graficas de
Variaciones','FontSize',12,...
                  'BackgroundColor','white',...
                  'Position',[.0 .0 .67 1.0]);


            ax = axes(p,'Position',[0.1 0.1 0.8 0.8]);
            title(ax,'Grafica de Variable');
            ylabel(ax,'Variable');
            xlabel(ax,'Tiempo(seg)');
```

```
            dateEditInicioBoxHandle = uicontrol(fig, 'Style',
'Edit', ...
            'Position', [835 598 150 22], ...
            'BackgroundColor', 'w');


            dateInicioTextHandle = uicontrol(fig, 'Style', 'Text',
...
            'String', 'Fecha:', ...
            'Horizontalalignment', 'left', ...
            'Position', [780 596 50 22]);


            calendarButtonHandle = uicontrol(fig, 'Style',
'PushButton', ...
            'String', 'Seleccionar', ...
            'Position', [995 598 70 22], ...
            'callback',
{@buttonSelectFecha,dateEditInicioBoxHandle});


            dateFinEditBoxHandle = uicontrol(fig, 'Style', 'Edit',
...
            'Position', [835 571 150 22], ...
            'BackgroundColor', 'w');


            dateFinTextHandle = uicontrol(fig, 'Style', 'Text', ...
            'String', 'Fecha:', ...
            'Horizontalalignment', 'left', ...
            'Position', [780 567 50 22]);
```

```
            calendarButtonHandle2 = uicontrol(fig, 'Style',
'PushButton', ...
            'String', 'Seleccionar', ...
            'Position', [995 571 70 22], ...
            'callback', {@buttonSelectFecha,dateFinEditBoxHandle});


            start =
uicontrol(fig,'Style','pushbutton','Position',[968 540 100
22],'String', 'Analizar','Callback',{@startMonitoreo});



            connApi = database('registosVariables','','');
            queryApi = 'Select DISTINCT api From registro';
            execQueryApi = exec(connApi,queryApi);
            execQueryApi = fetch(execQueryApi);
            %dataApi = execQueryApi.Data([2:3, 3:end]);
            dataApi = execQueryApi.Data();
            close(connApi);


            uicontrol(fig, 'Style', 'Text','String',
'Api:','Horizontalalignment', 'left','Position', [780 540 50 22]);
            yourcellVariables= [' '];
            yourcellEquipos = [' '];
            yourcellTipo= [' '];
            apiSelect = uicontrol('Style',
'popupmenu','Position',[836 540 100 22],...
            'string',dataApi,'Callback',{@selectionApi,dataApi});
```

```
          uicontrol(fig, 'Style', 'Text', 'String',
'Equipo:','Horizontalalignment', 'left','Position', [780 513 50
22]);

          equipoSelect= uicontrol('Style',
'popupmenu','Position',[836 513 100 22],...


     'string',yourcellEquipos,'Callback',{@selectionEquipo,yourcel
lEquipos,apiSelect});



          uicontrol(fig, 'Style', 'Text', 'String',
'Variable:','Horizontalalignment', 'left','Position', [780 486 50
22]);

          variableSelect = uicontrol('Style',
'popupmenu','Position',[836 486 100 22],...

          'string',yourcellVariables);




          function selectionApi(hObj,event,yourcellApi)


            v=get(hObj,'value')

            apiName = yourcellApi(v);

            if apiName ~= "null"

              conn = database('registosVariables','','');

              queryEquipo = strcat("Select DISTINCT equipo From
registro where (registro.api = '",apiName,"');")

              [data] = select(conn,queryEquipo)

              yourcellEquipos = data.equipo;

              equipoSelect.String = yourcellEquipos;

              close(conn);

            end
```

```
            end


            function
selectionEquipo(hObj,event,yourcellEquipo,selectApi)


              v=get(hObj,'value')

              apiName = dataApi(selectApi.Value);

              disp(yourcellEquipo)

              equipoName =
yourcellEquipos(get(equipoSelect,'value'));

              if apiName ~= "null" && equipoName ~= "null"

                conn = database('registosVariables','','');

                queryVariable = strcat("Select DISTINCT nombre ,
tipo From registro where (registro.api = '",apiName,"' AND
registro.equipo = '",equipoName,"');");

                disp(queryVariable)


                [data] = select(conn,queryVariable)

                yourcellVariables = data.nombre;

                yourcellTipo = data.tipo;

                variableSelect.String = yourcellVariables;

                close(conn);

              end

            end


            function startMonitoreo(hObj,event)


              apiNombre = dataApi{get(apiSelect,'value')};
```

```
            varNombre =
yourcellVariables(get(variableSelect,'value'));

            tipoNombre =
yourcellTipo(get(variableSelect,'value'));

            equipoName =
yourcellEquipos(get(equipoSelect,'value'));

            if (dateEditInicioBoxHandle.String ~="") &&
(dateFinEditBoxHandle.String ~= "") && (apiNombre ~= "") &&
(varNombre ~= " ") && (equipoName ~= " ")


              conn = database('registosVariables','','');

              queryVariable = strcat("Select valor , fecha From
registro where registro.api = '",apiNombre,"'", " and
registro.nombre = '",varNombre, "' and registro.equipo =
'",equipoName,"' and registro.fecha
>='",dateEditInicioBoxHandle.String,"' and registro.fecha <=
'",dateFinEditBoxHandle.String,"'");

              [data] = select(conn,queryVariable)

              close(conn);

              if ~isempty(data)

                x = [];

                for v = 1:length(data.valor)

                  auxvarV= data.valor(v);

                  x(length(x)+1) = auxvarV;

                end

                hold on;

                title(ax,'Grafica de Variable');

                ylabel(ax,strcat(varNombre," ", tipoNombre));

                xlabel(ax,'Tiempo');

                plot(ax,datetime(data.fecha),x);

              end
```

```
            end
        end
        end

        function buttonSelectFecha(hcbo,
eventStruct,dateEditBoxHandle)
            uicalendar('Weekend', [1 0 0 0 0 0 1], ...
            'SelectionType', 1, ...
            'DestinationUI', dateEditBoxHandle, ...
            'OutputDateFormat','yyyy-mm-dd HH:MM:SS');
        end



        function asignarTimers(interfaz)
                    global timerAPIFotovoltaicos;
        global mbFotovoltaicos;
                    global timerAPIAnalizadordeRed;
        global mbAnalizadordeRed;
                    global timerAPIBateriaLitio;
        global mbBateriaLitio;
                    global timerAPIBateriaPlomo;
        global mbBateriaPlomo;
                    global timerAPISuperCondensadores;
        global mbSuperCondensadores;
```

```
    diccionarioEquipoFV1Fotovoltaicos=containers.Map;

    diccionarioEquipoFV2Fotovoltaicos=containers.Map;

    diccionarioEquipoFV3Fotovoltaicos=containers.Map;

    diccionarioEquipoAR1AnalizadordeRed=containers.Map;

    diccionarioEquipoBateriaLitioBateriaLitio=containers.Map;

    diccionarioEquipoBateriaPlomoBateriaPlomo=containers.Map;

    diccionarioEquipoSuperCondensadoresSuperCondensadores=contain
ers.Map;

    diccionarioEquipoFV1Fotovoltaicos('VDC')={400003,'INT16','VDC
','V','FV1',10.0};

    diccionarioEquipoFV1Fotovoltaicos('PDC')={400005,'INT16','PDC
','NONE','FV1',10.0};

    diccionarioEquipoFV1Fotovoltaicos('PAC')={400021,'INT16','PAC
','NONE','FV1',10.0};

    diccionarioEquipoFV2Fotovoltaicos('VDC')={400103,'INT16','VDC
','V','FV2',10.0};

    diccionarioEquipoFV2Fotovoltaicos('PDC')={400105,'INT16','PDC
','NONE','FV2',10.0};

    diccionarioEquipoFV2Fotovoltaicos('PAC')={400121,'INT16','PAC
','NONE','FV2',10.0};

    diccionarioEquipoFV3Fotovoltaicos('VDC')={400302,'UINT16','VD
C','V','FV3',100.0};
```

```
    diccionarioEquipoFV3Fotovoltaicos('PDC')={400304,'INT16','PDC
','V','FV3',1000.0};


    diccionarioEquipoFV3Fotovoltaicos('PAC')={400320,'INT16','PAC
','V','FV3',1000.0};


    diccionarioEquipoAR1AnalizadordeRed('PLA')={4000003,'INT16','
PLA','V','AR1',10.0};


    diccionarioEquipoBateriaLitioBateriaLitio('IDC')={400027,'INT
16','IDC','A','Bateria Litio',10.0};


    diccionarioEquipoBateriaLitioBateriaLitio('PDC')={400028,'INT
16','PDC','V','Bateria Litio',10.0};


    diccionarioEquipoBateriaLitioBateriaLitio('P')={400044,'INT16
','P','V','Bateria Litio',10.0};


    diccionarioEquipoBateriaPlomoBateriaPlomo('IDC')={400004,'INT
16','IDC','A','Bateria Plomo',10.0};


    diccionarioEquipoBateriaPlomoBateriaPlomo('PDC')={400005,'INT
16','PDC','V','Bateria Plomo',10.0};


    diccionarioEquipoBateriaPlomoBateriaPlomo('P')={400021,'INT16
','P','V','Bateria Plomo',10.0};


    diccionarioEquipoSuperCondensadoresSuperCondensadores('IDC')=
{400004,'INT16','IDC','A','Super Condensadores',10.0};


    diccionarioEquipoSuperCondensadoresSuperCondensadores('PDC')=
{400005,'INT16','PDC','V','Super Condensadores',10.0};


    diccionarioEquipoSuperCondensadoresSuperCondensadores('P')={4
00021,'INT16','P','V','Super Condensadores',1.0};
```

```
lista_Datos_Fotovoltaicos={'Fotovoltaicos',mbFotovoltaicos,timerAPIFotovoltaicos,{diccionarioEquipoFV1Fotovoltaicos,diccionarioEquipoFV2Fotovoltaicos,diccionarioEquipoFV3Fotovoltaicos,}};

lista_Datos_AnalizadordeRed={'AnalizadordeRed',mbAnalizadordeRed,timerAPIAnalizadordeRed,{diccionarioEquipoAR1AnalizadordeRed,}};

lista_Datos_BateriaLitio={'BateriaLitio',mbBateriaLitio,timerAPIBateriaLitio,{diccionarioEquipoBateriaLitioBateriaLitio,}};

lista_Datos_BateriaPlomo={'BateriaPlomo',mbBateriaPlomo,timerAPIBateriaPlomo,{diccionarioEquipoBateriaPlomoBateriaPlomo,}};

lista_Datos_SuperCondensadores={'SuperCondensadores',mbSuperCondensadores,timerAPISuperCondensadores,{diccionarioEquipoSuperCondensadoresSuperCondensadores,}};

                        timerAPIFotovoltaicos = timer('executionMode','fixedRate','Period',uint8(1.0),'TimerFcn',@(~,~)leer_Datos_Fotovoltaicos(interfaz.panelFotovoltaicos,lista_Datos_Fotovoltaicos));

                        timerAPIAnalizadordeRed = timer('executionMode','fixedRate','Period',uint8(1.0),'TimerFcn',@(~,~)leer_Datos_AnalizadordeRed(interfaz.panelFotovoltaicos,lista_Datos_AnalizadordeRed));

                        timerAPIBateriaLitio = timer('executionMode','fixedRate','Period',uint8(1.0),'TimerFcn',@(~,~)leer_Datos_BateriaLitio(interfaz.panelBaterias,lista_Datos_BateriaLitio));

                        timerAPIBateriaPlomo = timer('executionMode','fixedRate','Period',uint8(1.0),'TimerFcn',@(~,~)leer_Datos_BateriaPlomo(interfaz.panelBaterias,lista_Datos_BateriaPlomo));

                        timerAPISuperCondensadores = timer('executionMode','fixedRate','Period',uint8(1.0),'TimerFcn',@(~,~)leer_Datos_SuperCondensadores(interfaz.panelBaterias,lista_Datos_SuperCondensadores));
```

```
            end

            function conectarApis(interfaz)

                        global mbFotovoltaicos;

                        global mbAnalizadordeRed;

                        global mbBateriaLitio;

                        global mbBateriaPlomo;

                        global mbSuperCondensadores;

                        mbFotovoltaicos =
conectarModbus('192.168.222.9', 502);

                        mbAnalizadordeRed =
conectarModbus('192.168.222.3', 502);

                        mbBateriaLitio =
conectarModbus('192.168.222.12', 502);

                        mbBateriaPlomo =
conectarModbus('192.168.222.11', 502);

                        mbSuperCondensadores =
conectarModbus('192.168.222.14', 502);


            end

            function iniciarTimers()

                        global mbFotovoltaicos;

                        global timerAPIFotovoltaicos;

                    if (mbFotovoltaicos~=0)

                      start(timerAPIFotovoltaicos);

                      msgbox('Conexion
exitosa','Fotovoltaicos','warn');

                    else

                      msgbox('Error al
conectar','Fotovoltaicos','error');
```

```
                    end
                        global mbAnalizadordeRed;
                        global timerAPIAnalizadordeRed;
                    if (mbAnalizadordeRed~=0)
                      start(timerAPIAnalizadordeRed);
                      msgbox('Conexion exitosa','Analizador de
Red','warn');
                    else
                      msgbox('Error al conectar','Analizador de
Red','error');
                    end
                        global mbBateriaLitio;
                        global timerAPIBateriaLitio;
                    if (mbBateriaLitio~=0)
                      start(timerAPIBateriaLitio);
                      msgbox('Conexion exitosa','Bateria
Litio','warn');
                    else
                      msgbox('Error al conectar','Bateria
Litio','error');
                    end
                        global mbBateriaPlomo;
                        global timerAPIBateriaPlomo;
                    if (mbBateriaPlomo~=0)
                      start(timerAPIBateriaPlomo);
                      msgbox('Conexion exitosa','Bateria
Plomo','warn');
                    else
```

```
                    msgbox('Error al conectar','Bateria
Plomo','error');
                end
                    global mbSuperCondensadores;
                    global timerAPISuperCondensadores;
                if (mbSuperCondensadores~=0)
                  start(timerAPISuperCondensadores);
                  msgbox('Conexion exitosa','Super
Condensadores','warn');
                else
                  msgbox('Error al conectar','Super
Condensadores','error');
                end
        end
              function
leer_Datos_Fotovoltaicos(interfaz,lista_Datos)
                nombreApi=lista_Datos{1};
                conexionModbus=lista_Datos{2};
                timer_Api=lista_Datos{3};
                for k=1:1:length(lista_Datos{4})
                  direcciones_Memoria=values(lista_Datos{4}{k});
                  for m=1:1:length(direcciones_Memoria)
                    direccion_Mem=direcciones_Memoria{m};
                          nombreVar = direccion_Mem{3};
                          tipoVar = direccion_Mem{4};
                          nombreEquipo = direccion_Mem{5};
                     posicion_registro= direccion_Mem{1}-400000;
```

```
                        registro =
read(conexionModbus,'holdingreg',posicion_registro,1,direccion_Mem
{2});

                    hijos= allchild(interfaz);

                    for i=1:1:size(hijos)

                       if
class(hijos(i))=="matlab.ui.control.UIControl" &&
isprop(hijos(i),'tag')==true

                         tag = strsplit(hijos(i).Tag,'-'); %
dividimos el tag (direccion-nombreApi)

                         if length(tag)>1

                           dir_mem=tag{1};

                           api_nom=tag{2};

                           if
dir_mem==string(int2str(direccion_Mem{1})) &&
api_nom==string(nombreApi) %buscar por la direccion de memoria


set(hijos(i),'string',registro/direccion_Mem{6});

                                                          conn =
database('registosVariables','','');

                         query = strcat('INSERT INTO
registro(api,nombre,valor,fecha,tipo,equipo) VALUES
(''',api_nom,''',''',nombreVar,''',',num2str(registro/10),',''',da
testr(now,'yyyy-mm-dd
HH:MM:SS'),''',''',tipoVar,''',''',nombreEquipo,''');');

                         disp(query);

                         execQuery = exec(conn,query);

                         execQuery = fetch(execQuery);

                             end

                        end

                       end

                    end
```

```
                    end
                  end
                end
                function
leer_Datos_AnalizadordeRed(interfaz,lista_Datos)
                  nombreApi=lista_Datos{1};
                  conexionModbus=lista_Datos{2};
                  timer_Api=lista_Datos{3};
                  for k=1:1:length(lista_Datos{4})
                    direcciones_Memoria=values(lista_Datos{4}{k});
                    for m=1:1:length(direcciones_Memoria)
                      direccion_Mem=direcciones_Memoria{m};
                            nombreVar = direccion_Mem{3};
                            tipoVar = direccion_Mem{4};
                            nombreEquipo = direccion_Mem{5};
                       posicion_registro= direccion_Mem{1}-400000;
                        registro =
read(conexionModbus,'holdingreg',posicion_registro,1,direccion_Mem
{2});
                      hijos= allchild(interfaz);
                      for i=1:1:size(hijos)
                        if
class(hijos(i))=="matlab.ui.control.UIControl" &&
isprop(hijos(i),'tag')==true
                          tag = strsplit(hijos(i).Tag,'-'); %
dividimos el tag (direccion-nombreApi)
                          if length(tag)>1
                            dir_mem=tag{1};
                            api_nom=tag{2};
```

```
                                if
dir_mem==string(int2str(direccion_Mem{1})) &&
api_nom==string(nombreApi) %buscar por la direccion de memoria


set(hijos(i),'string',registro/direccion_Mem{6});

                                                        conn =
database('registosVariables','','');

                          query = strcat('INSERT INTO
registro(api,nombre,valor,fecha,tipo,equipo) VALUES
(''',api_nom,''',''',nombreVar,''',',num2str(registro/10),',''',da
testr(now,'yyyy-mm-dd
HH:MM:SS'),''',''',tipoVar,''',''',nombreEquipo,''');');

                          disp(query);

                          execQuery = exec(conn,query);

                          execQuery = fetch(execQuery);

                               end

                             end

                           end

                         end

                       end

                     end

                   function
leer_Datos_BateriaLitio(interfaz,lista_Datos)

                     nombreApi=lista_Datos{1};

                     conexionModbus=lista_Datos{2};

                     timer_Api=lista_Datos{3};

                     for k=1:1:length(lista_Datos{4})

                       direcciones_Memoria=values(lista_Datos{4}{k});

                       for m=1:1:length(direcciones_Memoria)
```

```
                        direccion_Mem=direcciones_Memoria{m};
                              nombreVar = direccion_Mem{3};
                              tipoVar = direccion_Mem{4};
                              nombreEquipo = direccion_Mem{5};
                         posicion_registro= direccion_Mem{1}-400000;
                         registro =
read(conexionModbus,'holdingreg',posicion_registro,1,direccion_Mem
{2});
                        hijos= allchild(interfaz);
                        for i=1:1:size(hijos)
                           if
class(hijos(i))=="matlab.ui.control.UIControl" &&
isprop(hijos(i),'tag')==true
                             tag = strsplit(hijos(i).Tag,'-'); %
dividimos el tag (direccion-nombreApi)
                             if length(tag)>1
                               dir_mem=tag{1};
                               api_nom=tag{2};
                               if
dir_mem==string(int2str(direccion_Mem{1})) &&
api_nom==string(nombreApi) %buscar por la direccion de memoria

set(hijos(i),'string',registro/direccion_Mem{6});
                                                        conn =
database('registosVariables','','');
                        query = strcat('INSERT INTO
registro(api,nombre,valor,fecha,tipo,equipo) VALUES
('''',api_nom,''','''',nombreVar,''','',num2str(registro/10),','''',da
testr(now,'yyyy-mm-dd
HH:MM:SS'),''','''',tipoVar,''','''',nombreEquipo,''');');
                        disp(query);
                        execQuery = exec(conn,query);
```

```
                        execQuery = fetch(execQuery);
                            end
                          end
                        end
                      end
                    end
                  end
                  function
leer_Datos_BateriaPlomo(interfaz,lista_Datos)
                    nombreApi=lista_Datos{1};
                    conexionModbus=lista_Datos{2};
                    timer_Api=lista_Datos{3};
                    for k=1:1:length(lista_Datos{4})
                      direcciones_Memoria=values(lista_Datos{4}{k});
                      for m=1:1:length(direcciones_Memoria)
                        direccion_Mem=direcciones_Memoria{m};
                              nombreVar = direccion_Mem{3};
                              tipoVar = direccion_Mem{4};
                              nombreEquipo = direccion_Mem{5};
                         posicion_registro= direccion_Mem{1}-400000;
                         registro =
read(conexionModbus,'holdingreg',posicion_registro,1,direccion_Mem
{2});
                        hijos= allchild(interfaz);
                        for i=1:1:size(hijos)
                          if
class(hijos(i))=="matlab.ui.control.UIControl" &&
isprop(hijos(i),'tag')==true
```

```
                            tag = strsplit(hijos(i).Tag,'-'); %
dividimos el tag (direccion-nombreApi)
                            if length(tag)>1
                              dir_mem=tag{1};
                              api_nom=tag{2};
                              if
dir_mem==string(int2str(direccion_Mem{1})) &&
api_nom==string(nombreApi) %buscar por la direccion de memoria


set(hijos(i),'string',registro/direccion_Mem{6});
                                                          conn =
database('registosVariables','','');
                           query = strcat('INSERT INTO
registro(api,nombre,valor,fecha,tipo,equipo) VALUES
(''',api_nom,''',''',nombreVar,''',',num2str(registro/10),',''',da
testr(now,'yyyy-mm-dd
HH:MM:SS'),''',''',tipoVar,''',''',nombreEquipo,''');');
                           disp(query);
                           execQuery = exec(conn,query);
                           execQuery = fetch(execQuery);
                                end
                              end
                             end
                          end
                        end
                      end
                    end
                    function
leer_Datos_SuperCondensadores(interfaz,lista_Datos)
                      nombreApi=lista_Datos{1};
                      conexionModbus=lista_Datos{2};
```

```
                    timer_Api=lista_Datos{3};
                    for k=1:1:length(lista_Datos{4})
                      direcciones_Memoria=values(lista_Datos{4}{k});
                      for m=1:1:length(direcciones_Memoria)
                        direccion_Mem=direcciones_Memoria{m};
                              nombreVar = direccion_Mem{3};
                              tipoVar = direccion_Mem{4};
                              nombreEquipo = direccion_Mem{5};
                         posicion_registro= direccion_Mem{1}-400000;
                          registro =
read(conexionModbus,'holdingreg',posicion_registro,1,direccion_Mem
{2});
                        hijos= allchild(interfaz);
                        for i=1:1:size(hijos)
                           if
class(hijos(i))=="matlab.ui.control.UIControl" &&
isprop(hijos(i),'tag')==true
                             tag = strsplit(hijos(i).Tag,'-'); %
dividimos el tag (direccion-nombreApi)
                             if length(tag)>1
                               dir_mem=tag{1};
                               api_nom=tag{2};
                               if
dir_mem==string(int2str(direccion_Mem{1})) &&
api_nom==string(nombreApi) %buscar por la direccion de memoria

set(hijos(i),'string',registro/direccion_Mem{6});
                                                    conn =
database('registosVariables','','');
                          query = strcat('INSERT INTO
registro(api,nombre,valor,fecha,tipo,equipo) VALUES
```

```
(''',api_nom,''',''',nombreVar,''',',num2str(registro/10),',''',da
testr(now,'yyyy-mm-dd
HH:MM:SS'),''',''',tipoVar,''',''',nombreEquipo,''');');
                        disp(query);
                        execQuery = exec(conn,query);
                        execQuery = fetch(execQuery);
                             end
                           end
                         end
                       end
                     end
                   end
                 end
```

Referencias

[1] Lisbel Bárzaga Martell, Roberto C. Mompie Paneque, Bárbaro Valdés Cuesta. Ingeniería Electrónica, Automática y Comunicaciones.La Habana, Cuba. 2016.Sistemas SCADA para la automatización de los procesos productivos del CIGB.

[2] Oscar Pastor, Ignacio Panach, Nathalie Aquino. Centro de Investigación en Métodos de Producción de Software (ProS) Universidad Politécnica de Valencia. 2008.Model-Driven Development. DOI:10.1007/s00287-008-0275-8

[3] Beatriz Mora, Francisco Ruiz, Félix García, Mario Piattini Grupo Alarcos. Universidad de Castilla-La Mancha, España.,2006,Definición de Lenguajes de Modelos MDA vs DSL.

[4] Bejan, C. A., Iacob, M., & Andreescu, G.-D. (2009). SCADA automation system laboratory, elements and applications. 2009 7th International Symposium on Intelligent Systems and Informatics.

[5] Menéndez Domínguez, V. H. y Castellanos Bolaños, M. E. (2015). SPEM: Software Process Engineering Metamodel. Revista Latinoamericana de Ingeniería de Software, 3(2), 92.

[6] Describing Software Architecture with UML C. Hofmeister, R. L. Nord, D. Soni Siemens Corporate Research, Princeton, New Jersey, USA {chofmeister, mord, dsonij@scr.siemens.com

[7] Lange, C. F. J., Chaudron, M. R. V., & Muskens, J. (2006). In practice: UML software architecture and design description. IEEE Software, 23(2), 40–46.

[8] Sharma, V., & Tiwari, A. K. A Study on User Interface and User Experience Designs and its Tools.

[9] Mellor, S. J., Scott, K., Uhl, A., & Weise, D. (2002). Model-Driven Architecture. Lecture Notes in Computer Science, 290–297.

[10] Grzegorz Łabiak, and Grzegorz Bazydło, Model driven architecture approach to logic controller design,AIP Conference Proceedings 2040, 080003 (2018).

[11] Object Management Group Management Meta-Object Facility . URL http://www.omg.org/

[12] Estévez, E., Marcos, M., & Orive, D. (2007). Automatic generation of PLC automation projects from component-based models. The International Journal of Advanced Manufacturing Technology, 35(5-6), 527–540.

[13] Apvrille, L., Li, L., & Roudier, Y. (2016). Model-Driven Engineering for Designing Safe and Secure Embedded Systems. 2016 Architecture-Centric Virtual Integration (ACVI)

[14] Model-driven development of a standard-compliant Customer Energy Manager

[15] Lopez, R., Moore, A., & Gillerman, J. (2010). A model-driven approach to Smart Substation automation and integration for Comision Federal de Electricidad. IEEE PES T&D 2010.

[16] Xin, J. B. (2007). Building Power Quality Monitoring System Using Model Driven Architecture and IEC 61850. 2007 2nd IEEE Conference on Industrial Electronics and Applications

[17] R. I. Rajkumar, T. J. Alexander and P. Devi, 2016."ZigBee based design of low cost SCADA system for industrial process applications," 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, pp. 1-4.

[18] Medrano, K., Altuve, D., Belloso, K., & Bran, C, 2018. Development of SCADA using a RTU based on IoT controller. 2018 IEEE International Conference on Automation/XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA).

[19] V. Hassanpour, S. Rajabi, Z. Shayan, Z. Hafezi and M. M. Arefi, 2017."Low-cost home automation using Arduino and Modbus protocol," 2017 5th International Conference on Control, Instrumentation, and Automation (ICCIA), Shiraz, pp. 284-289.

[20] García Holgado, A., & García Peñalvo, F. J. (2017). A Metamodel Proposal for Developing Learning Ecosystems. Learning and Collaboration Technologies. Novel Learning Ecosystems, 100–109.

[21] Krusche, S., & Bruegge, B. (2017). CSEPM - A Continuous Software Engineering Process Metamodel. 2017 IEEE/ACM 3rd International Workshop on Rapid Continuous Software Engineering (RCoSE

[22] Puneet Patwari, Amar Banerjee, Subhrojyoti Roy Chaudhuri,Swaminathan Natarajan. (2016). Learning's from Developing a Domain Specific Engineering Environment for Control Systems. In Proceedings of the 9th India Software Engineering Conference (ISEC '16). Association for Computing Machinery, New York, NY, USA, 177–183

[23] Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schindler, M., & Völkel, S. (2014). Design Guidelines for Domain Specific Languages. ArXiv.

[24] Juan Cadavid, Benoit Combemale, Benoit Baudry.(2012) Ten years of Meta-Object Facility: an Analysis of Metamodeling Practices. [Research Report] RR-7882, INRIA.

[25] Giulianelli Daniel Alberto , Pons Claudia, Rodríguez Rocío Andrea, Vera Pablo Martín, Fernández, Victor. (2010). Integrando UML y DSL en el enfoque MDA. XVI Congreso Argentino de Ciencias de la Computación. Argentina, 514-523.

[26] Jacho, N., Soler Martínez, S., Lamoth Borrero, L., & Moreno Rodríguez, R. (2015). Análisis de la Transformación de Modelo CIM a PIM en el Marco de Desarrollo de la Arquitectura Dirigida por Modelos (MDA). Revista Politécnica, 36(3), 63.

[27] López, D. C., & Ibargüengoytia, M. A. (2017). *Reglas de traducción de restricciones entre OCL y LN* (Doctoral dissertation, Universidad Nacional de La Plata)

[28] Vásquez, L. M. G. (2016). *Editor gráfico de ontologías para los lenguajes semánticos RDF, RDF (S) y OWL, como extensión del framework Ontoconcept* (Doctoral dissertation, Universidad Tecnológica de Pereira. Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación. Ingeniería de Sistemas y Computación).

[29] Ortiz Suárez, D. F., Castillo Fonnegra, C. A. (2017). *Editor gráfico para modelos de característica con A y C* (Bachelor's thesis, Universidad Piloto de Colombia).

[30] Gerhart, M., & Boger, M. (2016). Modigen: model-driven generation of graphical editors in Eclipse. *International Journal of Computer Science and Information Technology*, *8*(5), 73-91.

[31] OMG'S METAOBJECT FACILITY™.:the mof and omg's model driven architecture . https://www.omg.org/mof/

[32] OMG'S ™.:THE ARCHITECTURE OF CHOICE FOR A CHANGING WORLD . https://www.omg.org/mda/

[33] Paéz Heyder et. al Programación de Controladores Lógicos (PLC) mediante Ladder y Lenguaje de Control Estructurado (SCL) en MATLAB

[34] Departamento de Ingeniería de Sistemas y Automática.: Automatización industrial: Introducción a los Autómatas Programables. 06 de Enero de 2016. http://eueti.uvigo.es/files/material_docente/1705/tema1introduccionalosautomatasprograma.pdf: [Consulta : 28 de Abril del 2020]

[35] BOYER, Stuart A.: Collecting Data from Distant Facilities. https://www.isa.org/standards-and-publications/isa-publications/intechmagazine/2007/october/automa tion-basics-collecting-data-from-distant-facilities/]

[36]-Rosado Alfredo: Diseño de interfaces Hombre-Máquina (HMI)

[37]SALAH, Mohammad.: SCADA Systems.http://www.msalah.com/A/SCADA.pdf [Consulta : 28 de Abril del 2020].

[38] Lozano, Carlo. Romero, Cristóbal.: Introducción a SCADA. http://www.uco.es/investiga/grupos/eatco/automatica/ihm/descargar/scada.pdf [Consulta : 28 de Abril del 2020].

[39] Universidad de Huelva, Dpto. de Ingeniería Electrónica, de Sistemas, informática y Automática.: Introducción a la Automatización. http://www.uhu.es/diego.lopez/AI/auto_trans-tema1.pdf [Consulta :28 de Abril del 2020].

[40] GÓMEZ, Fabio.: Automatización de Sistemas de Producción Transparencias de la Asignatura. http://www.esi2.us.es/~fabio/TransASP.pdf [Consulta : 28 de Abril del 2020].

[41] O. R. Kulkarni and R. A. Metri, "Automatic Toll Monitoring System using PLC-SCADA programming,"2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2019, pp. 126-129.

[42] Gutiérrez Guerrero, J. M. (2018). Desarrollo de sistemas software industriales dirigido por modelos: aplicación a OPC UA y IEC 61131-3.

[43] López, R., Moore, A., & Gillerman, J. (2010). *A model-driven approach to Smart Substation automation and integration for Comision Federal de Electricidad. IEEE PES*

[44] M. K. Shiferaw and A. K. Jena, "Code Generator for Model- Driven Software Development Using UML Models," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, 2018, pp. 1671-1678, doi: 10.1109/ICECA.2018.8474690.

[45] Ruiz, Francisco & Roda-García, José-Luis & García, Antonio & Avila-García, Orlando & Rebull, Eduardo. (2006). *Generación* de Editores Gráficos de Modelos .

[46] Management Group Inc., 2021. Meta-Object Facility. URL https://www.eclipse.org/acceleo/overview.html

[47] Krajewski, J. (s. f.). *The Future of HMI/SCADA - ppt download*. SlidePlayer - Upload and Share your PowerPoint presentations. https://slideplayer.com/slide/15717977/

[48] Pons Claudia & Silvia Giandini Roxana & Pérez Gabriela. Desarrollo de software dirigido por modelos: Conceptos teóricos y su aplicación práctica.

[49] Romagosa Jaume & Gallego David & Pacheco Porras. MINIPROYECTO AUTOMATIZACIÓN INDUSTRIAL (AUTI). Universidad Politécnica de Catalunya, 2004.

[50] Cousineau, D., Mentré, D. y Inoue, H. (2019). Automated deductive verification for ladder programming. Electronic Proceedings in Theoretical Computer Science, 310, 7–12.

[51] R. Ramanathan, "The IEC 61131-3 programming languages features for industrial control systems," 2014 World Automation Congress (WAC), 2014, pp. 598-603.

[52] Wareham, R., "Ladder diagram and sequential function chart languages in programmable controllers," Conf. Proc. on Programmable Control and Automation Technology Conference and Exhibition, 1988.

[53] Object Management Group (OMG), “Software Process Engineering Metamodel (SPEM) Specification” , 2021. Recuperado de http://www.omg.org

[54] Menéndez Domínguez, V. H. y Castellanos Bolaños, M. E. (2015). SPEM: Software Process Engineering Metamodel. Revista Latinoamericana de Ingeniería de Software, 3(2), 92.