Universidad de las Ciencias Informáticas

Facultad 4

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Desarrollo de paleta de componentes gráficos de generadores eléctricos para el *SCADA* SAINUX 2.0

Autor: Gabriel Alejandro Mahy Martínez

Tutores: Ing. Adolfo Y. Santana Rojas

Ing. Tahumara González Galbán

La Habana, junio de 2018

"Año 60 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2018.

_____

Firma del Autor

Gabriel A. Mahy Martínez

_____

Firma del Tutor

Ing. Adolfo Y. Santana Rojas

_____

Firma del Tutor

Ing. Tahumara González Galbán

Dedicatoria

Por todo lo que has hecho por mí, incluso cuando menos me lo he merecido, por poner siempre mis intereses por encima de cualquier cosa y por dedicar prácticamente toda tu vida a mi bienestar, el presente trabajo de diploma no puede ser dedicado a otra persona que no seas tú MAMÁ…te amo

Agradecimientos

Mamá, como agradecimiento te digo que a partir de ahora me toca a mí luchar por los dos y que me voy a encargar de que tengas la mejor vejez que puedas desear.

Papá: te agradezco porque a pesar de no convivir juntos siempre has estado presente y tus consejos me han ayudado en la vida.

Quiero agradecer a mis abuelos: Moraima y Papito por malcriarme y como dice mi mamá hoy soy como soy por ustedes y por eso les estaré eternamente agradecido. A mi abuela Mirian por siempre estar atenta a mis estudios, nunca voy a olvidar sus repasos de español y matemática, todas las tardes después de llegar de la escuela. A mi tía Mildred por quererme como un hijo y exigirme en todo porque siempre espera lo mejor de mí. A Gladis por todo el cariño y la atención que me ha brindado. A mi novia por todos los momentos que hemos vividos juntos.

A mis tutores por su gran paciencia conmigo, por su apoyo y por confiar en mí. A dos grandes amigos: Luis y Rosmery que fueron prácticamente cotutores de mi tesis, y nunca me dijeron que no cuando necesité su ayuda. A Dayani por ser la compañera del aula que más molesté para las pruebas y trabajos de la escuela y nunca me falló.

A todas las personas que conocí en estos 5 años que ahora son mis amistades, a todos los que formaron parte del grupo 5103, a los que convivieron conmigo en el apartamento (Pabel, Eric, Amado, Nadir, Dayan, Michel y Eduardo). A esos amigos que se convirtieron en hermanos (Frank, Mario, Kevin, Almirola, Yander y Keiger).

Y a la UCI porque aquí he pasado los mejores años de mi vida.

Resumen

La Informática Industrial es la rama de la Ciencia de la Computación encargada del tratamiento automático de la información proveniente de los procesos industriales. Los sistemas *SCADA* (*Supervisory Control and Data Adquisition*, por sus siglas en inglés) son un ejemplo de automatización, diseñados con la finalidad de supervisar y controlar procesos a distancia, basándose en la adquisición de datos generados por dispositivos de campo. La Universidad de las Ciencias Informáticas cuenta con el Sistema de Automatización Industrial UX 2.0 (SAINUX 2.0), un *SCADA* que permite el monitoreo y control de un área de trabajo que puede extenderse sobre una gran distancia. Entre los módulos con los que cuenta este sistema se encuentra el *HMI*, el cual se encarga de representar los procesos que ocurren en el campo dándole al operador total control sobre estos. Para la representación de los procesos a supervisar en el *HMI* se utilizan sinópticos gráficos, pero en el caso de los circuitos eléctricos, el sistema no contaba con los componentes para representar los generadores eléctricos, disminuyendo así su capacidad para adaptarse al sector eléctrico. Por lo anteriormente planteado, se propuso como objetivo de la investigación desarrollar una solución integrada al sistema SAINUX 2.0 que permitiera contribuir con la visualización y representación de generadores en circuitos eléctricos en el módulo *HMI*. Obteniéndose como resultado la paleta de los componentes gráficos para la representación de generadores en circuitos eléctricos.

Palabras clave: generadores eléctricos, *HMI*, *SCADA*.

Índice

Introducción....1
Capítulo 1: Fundamentación teórica....6
Introducción....6
1.1 Sistemas SCADA....6
1.1.1 Funciones y módulos de un sistema *SCADA*....7
1.1.2 Módulo Interfaz Hombre Máquina (*HMI*)....8
1.2 Componentes gráficos para generadores eléctricos....9
1.2.1 Componentes definidos....10
1.2.2 Soluciones similares....11
1.3 Métodos, herramientas y tecnologías a utilizar....14
1.3.1 Gráficos Vectoriales Escalables (SVG)....14
1.3.2 Inkscape....14
1.3.3 Lenguaje de programación....15
1.3.4 Marco de trabajo....15
1.3.5 Entorno Integrado de Desarrollo....16
1.3.6 Herramienta *CASE*....16
1.4 Metodología de desarrollo....17
1.4.1 Metodología AUP - UCI....18
Conclusiones parciales....19
Capítulo 2: Análisis y diseño....20
Introducción....20
2.1 Modelo de dominio....20
2.2 Especificación de requisitos....23
2.2.1 Requisitos funcionales....24
2.2.2 Requisitos no funcionales....25
2.3 Historias de Usuario....27
2.4 Planificación del desarrollo....37
2.4.1 Estimación de esfuerzos e iteraciones por Historia de Usuario....37
2.4.2 Plan de iteraciones....38
2.5 Diagrama de paquetes....39

2.6 Diseño de clases 40
2.7 Patrones de arquitectura 41
2.8 Patrones de diseño 43
Conclusiones parciales 45
Capítulo 3: Implementación y prueba 46
Introducción 46
3.1 Diagrama de componentes 46
3.2 Diagrama de despliegue 47
3.2.1 Descripción del diagrama de despliegue 48
3.3 Estándar de codificación 48
3.4 Solución del problema 49
3.4.1 Despliegues del sistema 49
3.5 Pruebas del sistema 51
3.5.1 Prueba de Caja Negra 51
3.5.2 Casos de prueba 52
Conclusiones parciales 54
Conclusiones generales 55
Recomendaciones 56
Referencias bibliográficas 57
Anexos 60
Anexo 1: Pruebas Funcionales 60
Anexo 2: Generadores Eléctricos (31) 65

Índice de Figuras

Fig. 1 Generador de corriente continua con excitación compuesta corta *10*

Fig. 2 Magneto Generador Manual *10*

Fig. 3 Generador fotovoltaico *11*

Fig. 4 Generador no rotativo. Símbolo general *11*

Fig. 5 Paleta de los componentes gráficos en el Citect. *12*

Fig. 6 Representación de las propiedades en el Citect. *13*

Fig. 7 Área de edición del Citect, visualizando los generadores eléctricos. *13*

Fig. 8 Modelo Dominio *21*

Fig. 9 Diagrama de Paquetes *39*

Fig. 10 Diagrama de clases de Generadores Eléctricos *41*

Fig. 11 Patrón de Arquitectura Modelo/Vista *42*

Fig. 12 Diagrama de Componentes *46*

Fig. 13 Diagrama de Despliegue *48*

Fig. 14 Componentes Básicos *50*

Fig. 15 Paleta de Componentes de Generadores Eléctricos *50*

Fig. 16 Inspector de Propiedades *50*

Fig. 17 Componentes en el área de edición. *51*

Fig. 18 Plan de Iteraciones *53*

Fig. 19 Generador Eléctrico Símbolo Genérico *65*

Fig. 20 Generador de CA Frecuencias Bajas *65*

Fig. 21 Generador de CA Frecuencias Medias *65*

Fig. 22 Generador de CA Frecuencias Medias *65*

Fig. 23 Célula Fotovoltaica 65

Fig. 24 Dinamo Generador de CC 65

Fig. 25 Generador de CC 65

Fig. 26 Generador de Corriente Controlado 65

Fig. 27 Generador de Corriente Ideal 66

Fig. 28 Generador de Impulsos 66

Fig. 29 Generador de Onda Triangular 66

Fig. 30 Generador de Resonancia 66

Fig. 31 Generador de Tensión 66

Fig. 32 Generador de Tensión Controlado 66

Fig. 33 Generador de Tensión Ideal 66

Fig. 34 Generador o Motor Eléctrico 66

Fig. 35 Fuente de CA 66

Fig. 36 Generador Sinusoidal CA 67

Fig. 37 Generador de CA no Rotatorio 67

Índice de Tablas

Tabla 1: Fases de AUP - UCI (15) 18

Tabla 2: Requisitos funcionales 24

Tabla 3 Historia de Usuario #1 Seleccionar componente gráfico 27

Tabla 4 Historia de Usuario #2 Eliminar el componente 28

Tabla 5 Historia de Usuario #3 Definir nombre del componente 29

Tabla 6 Historia de Usuario #4 Definir descripción del componente 30

Tabla 7 Historia de Usuario #5 Configurar la dimensión del ancho del componente 31

Tabla 8 Historia de Usuario #6 Configurar la dimensión de la altura del componente ... 32

Tabla 9 Historia de Usuario #7 Configurar la posición (X,Y) del componente 33

Tabla 10 Historia de Usuario #8 Configurar la opacidad del componente 34

Tabla 11 Historia de Usuario #9 Permitir la rotación del componente 35

Tabla 12 Historia de Usuario #10 Definir color de base para el componente 36

Tabla 13: Estimación de esfuerzo 37

Tabla 14 Caso de Prueba Funcional #1 52

Tabla 15 Diseño de Caso Prueba 53

Tabla 16 Caso de Prueba Funcional #2 60

Tabla 17 Caso de Prueba Funcional #3 60

Tabla 18 Caso de Prueba Funcional #4 61

Tabla 19 Caso de Prueba Funcional #5 61

Tabla 20 Caso de Prueba Funcional #6 62

Tabla 21 Caso de Prueba Funcional #7 62

Tabla 22 Caso de Prueba Funcional #8 63

Tabla 23 Caso de Prueba Funcional #9 63

Tabla 24 Caso de Prueba Funcional #10 64

Introducción

A partir del avance de las Tecnologías de Información y Comunicaciones (TIC), la sociedad ha visto nacer una era marcada por nuevas tendencias y mecanismos en el desarrollo de sus procesos. A nivel global, el manejo de la información se ha convertido en el centro de la toma de decisiones, y su correcta utilización es capaz de definir en la actualidad una posición importante en el ámbito competitivo de la industria, el mercado y los servicios.

La informática industrial es la rama de la ingeniería que integra y unifica el campo industrial a la informática. Esta se ha convertido en los últimos tiempos en compañera inseparable de la automatización industrial, que a su vez se encarga del tratamiento automático de la información proveniente de los procesos industriales, utilizando para ello computadoras (1). Los datos son introducidos a un ordenador, adquiriendo significado cuando se hace una interpretación de los mismos.

En Cuba, la búsqueda de nuevas alternativas que posibiliten una gestión eficiente de los grandes volúmenes de información que se generan, a partir de las ventajas que ofrecen las nuevas tecnologías, también ha sido una marcada tendencia, basada en el proceso de informatización de la sociedad.

La Universidad de las Ciencias Informáticas (UCI), cuenta con el Centro de Informática Industrial (CEDIN) perteneciente a la Facultad 4, este centro tiene como misión desarrollar productos y servicios informáticos de automatización industrial con un alto valor agregado y que cumplan las necesidades y expectativas de los clientes, potenciando la formación especializada y la investigación. Entre los proyectos con los que cuenta el centro se encuentra Continuidad del Sistema de Automatización Industrial UX (SAINUX 2.0).

Este proyecto tiene como salida un producto para la Supervisión, Control y Adquisición de Datos (del inglés, *SCADA*[1]), permitiendo el monitoreo y control de un área de trabajo que puede extenderse sobre una gran distancia (kilómetros/millas). La instalación de un sistema *SCADA* necesita un *hardware* de señal de entrada y salida, sensores y

[1] SCADA (del inglés, *Supervisory, Control and Data Acquisition)*

actuadores, controladores, *HMI*[2], redes, comunicaciones y base de datos. Además, está compuesto por módulos de software como son Configuración, Interfaz gráfica del operador, Módulo de proceso, Gestión y archivo de datos y Comunicaciones (2).

El módulo *HMI* en el *SCADA* se encarga de representar, en un ordenador, los procesos que ocurren en el campo; muestran los componentes implicados, los sensores, las estaciones remotas. Por su parte, permite al operador estar en contacto directo con el sistema, realizar la supervisión y el control del proceso en general. Trabaja sobre dos entornos: el entorno de configuración (EC), donde los mantenedores realizan la configuración de la información específica del área que se desea supervisar y diseñan los despliegues, los cuales hacen uso de los componentes gráficos que permiten simular los procesos de campo; y por otro lado, el entorno de visualización (EV), donde el operador puede supervisar y controlar la configuración realizada en el EC, interactuando con los componentes gráficos para emitir control sobre el sistema, monitorear los cambios de estado de las variables, gestionar alarmas y generar reportes (3) (4).

Tanto para el EC como para el EV, el *HMI* provee una biblioteca de componentes gráficos que permiten recrear de una manera lo más real posible los procesos de campo. Estos gráficos pueden ser simples figuras geométricas como: línea, rectángulo, elipse, texto, polígono, polilínea, curvas; pero también muy complejas como los componentes de *hardware* utilizados en la industria que se desean supervisar como son: válvulas, generadores, motores de combustión (3) (4).

Entre los sinópticos gráficos usados en la representación de circuitos eléctricos existen los generadores eléctricos, los cuales no se encuentran en la biblioteca de componentes gráficos del *HMI* SAINUX 2.0. Para solventar dicha situación el mantenedor utiliza figuras básicas o imágenes para la representación de éstos. Esto ocasiona varias desventajas como son:

- El proceso de configuración del circuito eléctrico se hace más lento y complicado mediante el uso de figuras básicas para la representación gráfica, afectando en gran medida el desarrollo de las representaciones de circuitos eléctricos.

[2] *HMI* (del inglés, *Human Machine Interface*) Interfaz Hombre Máquina

- Mediante el empleo y agrupación de figuras básicas para la representación de los generadores eléctricos, el componente gráfico resultante no puede ser guardado como un nuevo componente, afectándose su reutilización en otros despliegues.
- La agrupación de figuras básicas no puede ser tratada como un solo componente afectando la personalización de este.
- Mediante el uso del componente imagen, se debería importar una imagen previamente diseñada en un formato vectorial evitándose así una pérdida en cuanto la calidad de la representación gráfica, la cual llevaría un previo trabajo de diseño.

Para dar solución a esta problemática se propone como **problema de la investigación:** ¿Cómo contribuir a la representación de componentes gráficos de generadores eléctricos en el módulo *HMI* del *SCADA* SAINUX 2.0? En correspondencia con el problema planteado, el **objeto de estudio** de la presente investigación es: El proceso de representación y configuración de componentes gráficos para procesos industriales, delimitando como **campo de acción**: Componentes gráficos para la representación de generadores en circuitos eléctricos en el módulo Interfaz Hombre Máquina del *SCADA* SAINUX 2.0

Con el propósito de solucionar el problema planteado se define como **objetivo general** de la investigación: Desarrollar una paleta de componentes gráficos al *SCADA* SAINUX 2.0 que permita la representación de generadores eléctricos en el módulo de *HMI*.

Como **posibles resultados** se espera:

- Actualización del estado del arte de la utilización de generadores en circuitos eléctricos.
- Especificación de requerimientos del sistema.
- Componentes gráficos para la representación de generadores en circuitos eléctricos.

A partir de lo anteriormente planteado se desglosan las siguientes **tareas a cumplir:**

- Elaboración del marco teórico de la investigación a través de la actualización del estado del arte que existe actualmente sobre el tema.

- Identificación y caracterización de los diferentes tipos de generadores eléctricos existentes.
- Realización del levantamiento de requisitos funcionales y no funcionales.
- Implementación de componentes gráficos que brinden la solución al problema planteado.
- Realización de pruebas para validar el cumplimiento de los requerimientos.

Para el desarrollo de la investigación se utilizaron diferentes métodos científicos agrupados en **métodos teóricos y empíricos**, los cuales tienen su sustento en la concepción materialista dialéctica y permiten una mejor recopilación de la información para el modelado de análisis y diseño de los componentes a realizar. De los métodos teóricos se utiliza el analítico-sintético y el histórico-lógico. De los métodos empíricos se hizo uso del análisis de documentos.

El método **analítico-sintético** permitió el análisis de la bibliografía utilizada, además de garantizar la fiabilidad de la información empleada. Permitió obtener, describir y resumir los elementos más importantes que intervienen en los componentes gráficos para procesos industriales para construir el modelo de negocio y realizar el diseño del sistema adecuadamente. Mientras que el **histórico-lógico** permitió un estudio relacionado con los sistemas *SCADA*, para así obtener un conocimiento histórico de su desarrollo y comportamiento tanto a nivel nacional como internacional.

De los métodos empíricos, se hizo uso del **análisis de documentos** para definir los diferentes requerimientos del sistema y recopilar información referente a las características y funcionamiento de los componentes gráficos para visualización de procesos industriales y representación de generadores eléctricos en *SCADA*.

El presente documento está estructurado de la siguiente manera:

Capítulo 1. Fundamentación teórica: Se realiza un esbozo teórico centrado en los conceptos y características fundamentales de los componentes gráficos para el proyecto Continuidad del Sistema de Automatización Industrial UX (SAINUX 2.0) para la representación y visualización de generadores en circuitos eléctricos.

Capítulo 2. Análisis y diseño: Se analiza y se diseña la solución propuesta para el sistema, se realiza el levantamiento de requisitos funcionales y no funcionales a partir de la metodología de desarrollo utilizada.

Capítulo 3. Implementación y prueba: En este capítulo se describe la fase de implementación del sistema, según la metodología propuesta. Se realiza la fase de pruebas, a la cual se somete el producto para validar su correcto funcionamiento, permitiendo comprobar que el mismo cumple con todos los requerimientos.

Capítulo 1: Fundamentación teórica

Introducción

En este capítulo se realiza un estudio del estado del arte, a partir de los conceptos asociados a un sistema *SCADA*. Se explican los módulos y el funcionamiento de un sistema *SCADA* para la gestión y control de datos, destacándose el módulo *HMI*. Se describen también las tecnologías que se utilizaron para construir la paleta de componentes de generadores eléctricos, lenguaje de programación y herramientas de desarrollo para darle solución al problema planteado.

1.1 Sistemas *SCADA*

El *SCADA* es un sistema de tiempo real, distribuido en módulos que trabajan de manera conjunta posibilitando el funcionamiento del sistema como un todo. Estos módulos se encuentran interconectados a través de un *software* para la distribución de los servicios en la red, conocido como *software* de comunicación entre aplicaciones. Estos pueden tener múltiples aplicaciones como por ejemplo, control de bombas de una planta industrial, visualización y control de diferentes líneas de montaje de una fábrica, control de temperaturas, presiones, válvulas de un proceso determinado para realizar un producto entre otras aplicaciones (2).

Un sistema *SCADA* se caracteriza por:

- Adquisición y almacenamiento de datos para recoger, procesar y almacenar la información recibida en forma continua y confiable.
- Representación gráfica y animada de variables de proceso y su monitorización por medio de alarmas.
- Arquitectura abierta y flexible con capacidad de ampliación y adaptación.
- Supervisión para observar desde un monitor la evolución de las variables de control.
- Transmisión de información con dispositivos de campo y otras PC.
- Base de datos, gestión de datos con bajos tiempos de acceso.
- Presentación, representación gráfica de los datos. Interfaz del Operador o *HMI*.

- Explotación de los datos adquiridos para gestión de la calidad, control estadístico, gestión de la producción y gestión administrativa y financiera.
- Alertar al operador sobre cambios detectados en la planta, tanto aquellos que no se consideren normales (alarmas) como los que se produzcan en su operación diaria (eventos). Estos cambios son almacenados en el sistema para su posterior análisis (2).

1.1.1 Funciones y módulos de un sistema *SCADA*

Un *SCADA* debe cumplir cuatro funciones principales:

Monitoreo: Es la habilidad de obtener y mostrar datos en tiempo real. Estos datos se pueden mostrar como números, texto o gráficos que permitan una lectura más fácil de interpretar.

Supervisión: Esta función permite junto con el monitoreo la posibilidad de ajustar las condiciones de trabajo del proceso directamente desde la computadora. Posee alarmas para la capacidad de reconocer eventos excepcionales dentro del proceso y reportar estos eventos. Las alarmas son reportadas basadas en límites de control preestablecidos.

Control: Es la capacidad de aplicar algoritmos que ajustan los valores del proceso y así mantener estos valores dentro de ciertos límites. Este va más allá del control de supervisión, removiendo la necesidad de la interacción humana. Sin embargo, la aplicación de esta función desde un *software* corriendo en una PC puede quedar limitada por la confiabilidad que quiera obtenerse del sistema.

Históricos: Es la capacidad de muestrear y almacenar en archivos, datos del proceso a una determinada frecuencia. Este almacenamiento de datos es una poderosa herramienta para la optimización y corrección de procesos (5).

Los módulos o bloques *software* que permiten las actividades de adquisición, supervisión y control son los siguientes:

- **Configuración:** permite al usuario definir el entorno de trabajo de su *SCADA*, adaptándolo a la aplicación particular que se desea desarrollar (2).

- **Interfaz Hombre Máquina:** proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos gráficos almacenados en el ordenador de proceso y generados desde el editor incorporado en el *SCADA* o importados desde otra aplicación durante la configuración del paquete.
- **Módulo de proceso:** ejecuta las acciones de mando pre programado a partir de los valores actuales de variables leídas.
- **Gestión y archivo de datos:** se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.
- **Comunicaciones:** se encarga de la transferencia de información entre la planta y la arquitectura *hardware* que soporta el *SCADA*, y entre esta y el resto de elementos informáticos de gestión (5).

1.1.2 Módulo Interfaz Hombre Máquina (*HMI*)

El módulo interfaz hombre – máquina (*HMI*) es el sistema que presenta los datos a un operador (humano) y a través de la cual este controla el proceso. Son interfaces gráficas que muestran información del proceso en tiempo real, utilizando diagramas esquemáticos, algunos contornos y hasta animaciones en pantalla o paneles.

Los sistemas *HMI* facilitan las tareas de diseño debido a que incorporan protocolos para comunicarse con los dispositivos de campo más conocidos, tienen herramientas para crear bases de datos dinámicas, permiten crear y animar pantallas de forma sencilla y además, incluyen gran cantidad de bibliotecas de objetos para representar los diferentes dispositivos de la industria como motores, tanques, indicadores, interruptores, válvulas y bomba, signos de advertencias, señales de dirección, cámara de seguridad (6).

En el *SCADA* SAINUX 2.0, el módulo *HMI* presenta dos ambientes, el ambiente de edición que permite configurar los procesos, definir y gestionar las variables, editar los controladores, los comandos, las alarmas y opciones adicionales. En este entorno las pantallas o procesos definen sus relaciones entre ellas, determinando el orden de aparición, los drivers, los comandos, las alarmas y las variables a visualizar que después se procesarán y controlarán en forma de listas o tablas para una mejor gestión.

Por otra parte, el visualizador es el que permite a un operador definir el ambiente de trabajo del *SCADA*, adaptándolo mejor a la aplicación particular que se desea desarrollar. Este entorno visualiza los datos históricos en forma de gráficos de tendencia y hace manejo de alarmas y eventos. Es el encargado de supervisar de manera directa el proceso puesto que interactúa con los operadores (7).

1.2 Componentes gráficos para generadores eléctricos

Un componente gráfico es un elemento que presenta como objetivo la comunicación visual a partir de una aplicación (8). El *SCADA* SAINUX 2.0 cuenta con una paleta de componentes gráficos necesarios para el diseño de los reportes en el proceso de representación y configuración de los procesos industriales a supervisar. Para el caso de los generadores eléctricos, no existe en la paleta un componente con el que se puedan representar.

Un generador es una máquina eléctrica rotativa que transforma energía mecánica en energía eléctrica. Lo consigue gracias a la interacción de los dos elementos principales que lo componen: la parte móvil llamada rotor, y la parte estática que se denomina estator.

Cuando un generador eléctrico está en funcionamiento, una de las dos partes genera un flujo magnético (actúa como inductor) para que el otro lo transforme en electricidad (actúa como inducido).

Los generadores eléctricos se diferencian según el tipo de corriente que producen. Se dividen en dos grandes grupos de máquinas eléctricas rotativas: los alternadores y las dinamos.

Los alternadores generan electricidad en corriente alterna. El elemento inductor es el rotor y el inducido el estator. Un ejemplo son los generadores de las centrales eléctricas, las cuales transforman la energía mecánica en eléctrica alterna (9).

Generador de Corriente Alterna (CA): El funcionamiento del generador de corriente alterna, se basa en el principio general de inducción de voltaje en un conductor en movimiento cuando atraviesa un campo magnético. Este generador consta de dos partes

fundamentales, el inductor, que es el que crea el campo magnético y el inducido que es el conductor el cual es atravesado por las líneas de fuerza de dicho campo (10).

Generador de Corriente Continua (CC): Los generadores de corriente continua son máquinas que producen tensión su funcionamiento se reduce siempre al principio de la bobina giratorio dentro de un campo magnético. Si una armadura gira entre dos polos magnéticos fijos, la corriente en la armadura circula en un sentido durante la mitad de cada revolución, y en el otro sentido durante la otra mitad (10).

1.2.1 Componentes definidos

El estándar seguido para la utilización de los símbolos eléctricos está definido por la Comisión Electrotécnica Internacional (*IEC*, por sus siglas en inglés), dicha organización se encarga de la normalización en los campos: eléctrico, electrónico y tecnologías relacionadas. Los gráficos de la simbología eléctrica se rigen en la norma europea EN 60617 aprobada por el Comité Europeo de Normalización Electrotécnica (CENELEC) bajo la Norma Internacional IEC 61082 (11).

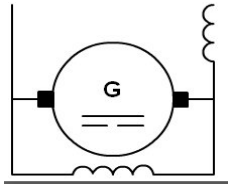


Fig. 1 Generador de corriente continua con excitación compuesta corta

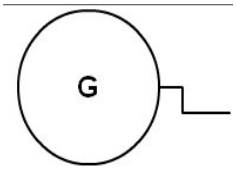


Fig. 2 Magneto Generador Manual

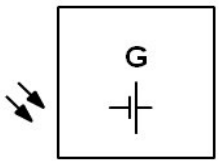


Fig. 3 Generador fotovoltaico

Fig. 4 Generador no rotativo. Símbolo general

Las imágenes de los restantes generadores eléctricos definidos se encuentran en el Anexo 2.

1.2.2 Soluciones similares

En la actualidad existe una variedad de sistemas *SCADA* que poseen una Interfaz Hombre-Máquina que permite la representación de generadores eléctricos como componentes gráficos, entre ellos se encuentran:

- ***AggreGate SCADA* / *HMI*:** es un sistema para visualizar y operar procesos, flujos de producción, máquinas y plantas. AggreGate *SCADA* / *HMI* y otros productos basados en AggreGate reconocen una amplia gama de tareas de ingeniería energética y ofrecen una solución rentable para diversos objetivos de gestión energética (12).
- ***Citect SCADA Schneider Electric*:** brinda soluciones que le permiten concentrarse en aumentar la eficiencia de capital y la efectividad del operador, maximizar el rendimiento y la confiabilidad de los activos, y administrar la producción, la seguridad y el cumplimiento (13).

Los sistemas anteriormente expuestos, se caracterizan por poseer licencias privativas, lo cual imposibilita el empleo de sus componentes. Además de emplear lenguajes de programación (C#, JAVA) distintos al establecido en el *SCADA* SAINUX en su versión 2.0. Sin embargo, el empleo de la herramienta de edición "*Building*

Operation Graphics Editor" en su versión 1.9 del *Schneider Electric* proporcionó las siguientes características:

- Definición de las propiedades de los componentes gráficos (color de base, posición, ancho, largo, opacidad).
- Representación y agrupación de los generadores eléctricos como paleta de componentes gráficos.
- Visualización de los generadores eléctricos.

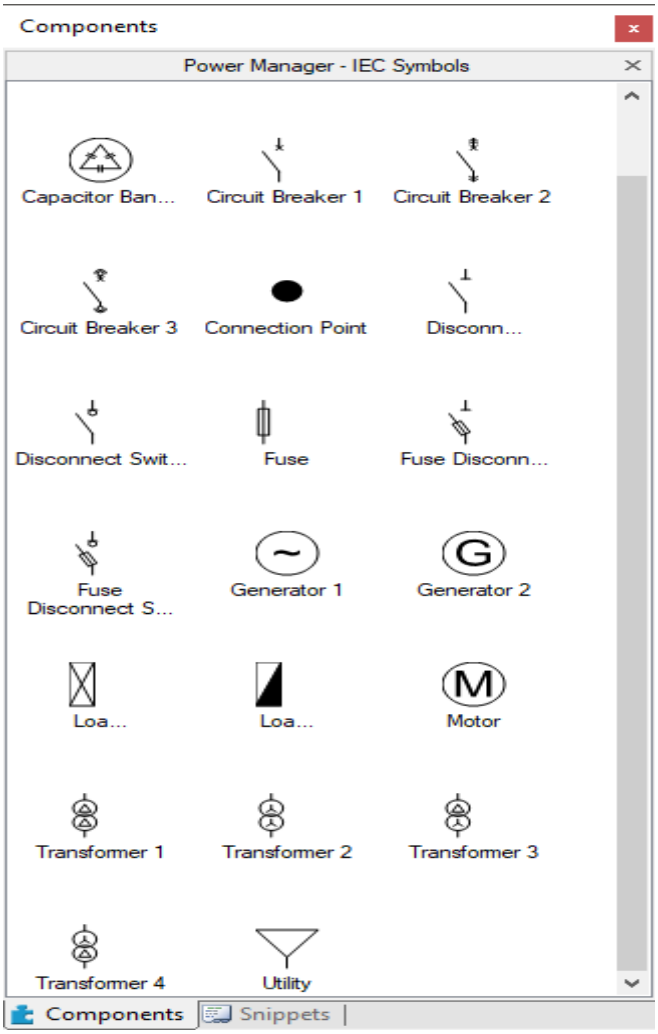


Fig. 5 Paleta de los componentes gráficos en el Citect.

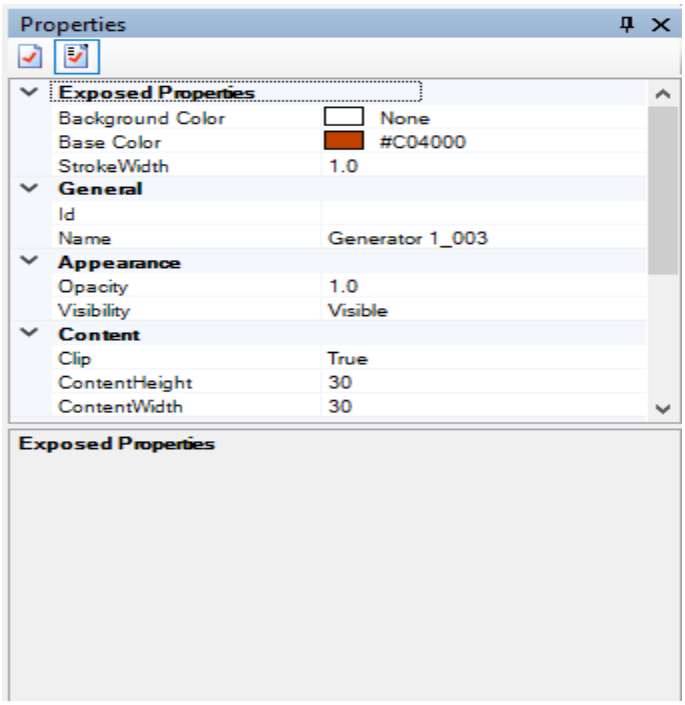


Fig. 6 Representación de las propiedades en el Citect.

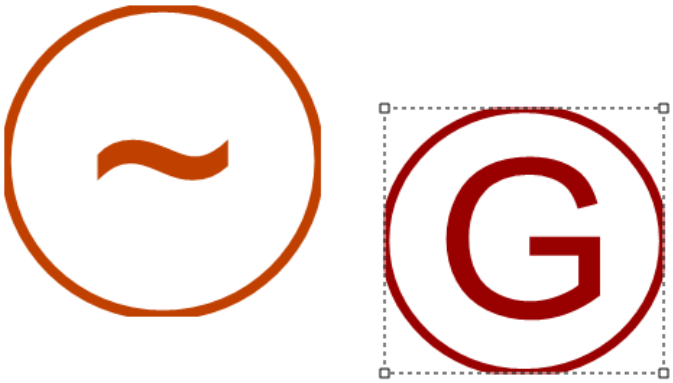


Fig. 7 Área de edición del Citect, visualizando los generadores eléctricos.

1.3 Métodos, herramientas y tecnologías a utilizar

Las herramientas y tecnologías escogidas para el desarrollo de un proyecto deben ser seleccionadas cautelosamente, ya que pueden suponer el fracaso de éste o pueden aumentar su complejidad, para lo cual se deben conocer cuáles son las distintas alternativas y las necesidades del proyecto. La selección de varias herramientas y tecnologías está fundamentada por su utilización en el desarrollo del proyecto Continuidad del Sistema de Automatización Industrial UX (SAINUX 2.0).

1.3.1 Gráficos Vectoriales Escalables (SVG)

SVG[3] (del inglés, *Scalable Vector Graphics)*, que se podría traducir libremente al español como gráficos basados en vectores escalables. En otras palabras, decir que es un formato gráfico basado en XML para crear archivos vectoriales en 2D, con un lenguaje de marcado por medio de etiquetas. Entre sus posibilidades, se puede señalar la capacidad de usar tres tipos de objetos gráficos: formas de vectores gráficos (entre las que se incluyen líneas, polígonos, polilínea, rectángulos, círculos o elipses), imágenes y texto. Además, a los objetos gráficos se les puede aplicar transformaciones (traslaciones, escala), animaciones y efectos de filtro (muy variados, como los que se conoce en programas de diseño como *Photoshop*) (14).

Algunas ventajas de este formato son: no pierde calidad si se hace zoom, se puede escalar, se muestra de forma progresiva, no se tiene que esperar a que se descargue todo el archivo, los vectoriales por ser solamente parámetros matemáticos, suelen ser mucho menos pesados que una imagen de píxeles. Los gráficos vectoriales pueden ser transformados (estirar, rotar, mover, distorsionar) de una manera más sencilla y con menos requerimientos de memoria en el equipo (14).

Para la visualización de los componentes gráficos que representan específicamente los generadores eléctricos se empleará el estándar SVG.

1.3.2 Inkscape

[3] Gráficos Vectoriales Escalables (SVG, del inglés *Scalable Vector Graphics*)

Es un *software* de vectores gráficos de calidad profesional para Windows, Mac OS X y GNU/Linux. Es usado por diseñadores profesionales y aficionados de todo el mundo para crear una gran variedad de gráficos como ilustraciones, iconos, logos, diagramas, mapas y diseños web. *Inkscape* es un *software* libre y de código abierto, que utiliza SVG *(Scalable Vector Graphic)*, el estándar abierto de W3C, como formato nativo (15).

1.3.3 Lenguaje de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. C++ a pesar de que es un lenguaje orientado a objetos, sigue muy ligado al *hardware* subyacente, manteniendo una considerable potencia para la programación a bajo nivel. Permite utilizarlo para escribir controladores de dispositivos y otro *software* que se basen en la manipulación directa de *hardware*, bajo restricciones de tiempo real. C++ fue diseñado para que cada característica del lenguaje se pudiera utilizar en virtud de limitaciones severas de tiempo y espacio (16).

1.3.4 Marco de trabajo

Un *framework* es una abstracción de código común que provee funcionalidades genéricas que pueden ser utilizadas para desarrollar aplicaciones de manera rápida, fácil, modular y sencilla, ahorrando tiempo y esfuerzo. Qt es un *framework* multiplataforma orientado a objetos, se utiliza para desarrollar *software* que utilicen interfaz gráfica de usuario, así como también diferentes tipos de herramientas para la línea de comandos y consolas para servidores que no necesitan una interfaz gráfica de usuario. Incluye clases, bibliotecas y herramientas para la producción de aplicaciones usando el lenguaje C++ de forma nativa. Es multiplataforma ya que funciona en las principales plataformas y tiene un amplio apoyo operando en varios sistemas como Unix (Linux, MacOS X, Solaris) o incluso toda la familia de Windows. El principal motivo por el que Qt se ha hecho tan popular es por sus métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros métodos para el manejo de ficheros, además de estructuras de datos tradicionales (17).

1.3.5 Entorno Integrado de Desarrollo

Para el desarrollo de este producto se requiere de un entorno de desarrollo integrado (*IDE*, por sus siglas en inglés), el cual puede denominarse como un entorno de programación que consiste en un editor de código y un compilador. El *IDE* seleccionado para el desarrollo de la aplicación es *Qt Creator* en su versión 2.5.0. Qt Creator es un *IDE* multiplataforma para el desarrollo de aplicaciones que pueden o no tener interfaz gráfica. Este se centra en proporcionar características que ayudan a los nuevos usuarios del *IDE* a aprender y comenzar a desarrollar rápidamente.

Existen otras características importantes que presenta este *IDE* como la disponibilidad de código fuente, la excelente documentación organizada que provee en el *Qt Assistant* y un editor para el diseño de formularios denominado *Qt Designer* (18).

Qt Creator cuenta con:

- Un editor de código con soporte para C++.
- Herramientas para la rápida navegación por el código.
- Resaltado de sintaxis y auto-completado de código.
- Control estático de código y estilo a medida.
- Soporte para refactorización de código.
- Paréntesis coincidentes y modos de selección.

1.3.6 Herramienta *CASE*

Las herramientas *CASE* (del inglés, *Computer Aided Software Engineering*), en español Ingeniería de *Software* Asistida por Computadora, son aplicaciones informáticas utilizadas en el proceso de desarrollo de *software* en tareas como: realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente a partir del diseño, compilación automática, documentación o detección de errores entre otras.

Visual Paradigm for UML es una herramienta *CASE* profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este *software* ayuda a una más rápida construcción

de aplicaciones de calidad y a un menor coste. Las características principales de esta herramienta son las que a continuación se enuncian (19):

- Soporte de *UML*[4].
- Ingeniería inversa: Código a modelo, código a diagrama.
- Generación de código: Modelo a código, diagrama a código.
- Generación de bases de datos: Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos: Desde Sistemas Gestores de Bases de Datos existentes a diagramas de Entidad - Relación.
- Editor de figuras.

1.4 Metodología de desarrollo

Para desarrollar un buen *software* se depende de un gran número de actividades y etapas, donde se debe elegir la mejor metodología para un equipo en un determinado proyecto, lo cual es trascendental para el éxito del producto. Al no existir una metodología de *software* universal toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, tiempo, recursos) exigiéndose así que el proceso sea configurable. Estas se pueden dividir en dos grupos de acuerdo con sus características y los objetivos que persiguen: ágiles y tradicionales.

Las metodologías ágiles tienen como principal característica, el desarrollo rápido de *software*, permiten al equipo adaptar y coordinar tareas, por la necesitad de satisfacer al cliente con la mayor brevedad posible. Deben cumplirse una serie de requerimientos dentro del equipo de desarrollo para poder aplicar un modelo como este.

Las metodologías tradicionales en cambio, se centran especialmente en el control del proceso, mediante una exhaustiva documentación; definiendo roles, actividades, artefactos, herramientas y notaciones para el modelado y una documentación detallada. Estas metodologías son muy efectivas y necesarias en proyectos grandes (20) (21).

[4] *UML* (del inglés, *Unified Modeling Languague)* Lenguaje Unificado de Modelado, que permite modelar el análisis, diseño e implementación del ciclo de desarrollo de un *software*, construyéndose modelos precisos,

1.4.1 Metodología AUP - UCI

Se decidió converger a una metodología que cubra las particularidades de cada una de las ya aplicadas. Esta metodología escogida se adapta a lo que ya la Universidad ha estado proponiendo como ciclo de vida de los proyectos, sin alejarse de lo que hasta el momento se ha trabajado e introducir la menor cantidad de cambios posibles. Esta metodología propuesta es una variación al Proceso Unificado Ágil.

El AUP es una versión simplificada de RUP. Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (22).

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva. Estas fases son:

- **Inicio:** El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- **Elaboración:** El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- **Construcción:** Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- **Transición:** El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

La variación de la metodología AUP, denominada AUP - UCI se adapta al ciclo de vida definido para la actividad productiva de la UCI y es utilizada por el CEDIN en sus productos de *software*. El uso de esta técnica de modelado ágil permite encapsular los requisitos funcionales en Historias de Usuario (HU) o descripción de requisitos por procesos.

Tabla 1: Fases de AUP - UCI (15)

Fases	Objetivos

Inicio	• Planeación del proyecto. • Estudio inicial de la organización cliente. • Alcance del proyecto. • Estimaciones de tiempo, esfuerzo y costo
Ejecución	• Ejecución de las actividades requeridas para desarrollar el *software* • Ajuste de los planes del proyecto. • Modelado del negocio. • Obtención de requisitos. • Se elaboran la arquitectura y el diseño. • Se implementa y se libera el producto.
Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Conclusiones parciales

En el presente capítulo el estudio del estado del arte y las principales soluciones existentes de los componentes gráficos en los diferentes *SCADA*, permitieron sentar las bases teóricas para la futura implementación de la solución propuesta. Después de haber analizado las metodologías, herramientas y tecnologías se decidió utilizar para guiar el proceso de desarrollo de la solución la metodología AUP UCI.

Capítulo 2: Análisis y diseño

Introducción

En el presente capítulo se reflejan las actividades realizadas en los procesos de análisis y diseño de la solución propuesta. Para comprender mejor el funcionamiento del sistema se exponen los artefactos más importantes que describen el flujo normal de eventos que ocurren en el sistema tales como la descripción del modelo de dominio, así como los diagramas correspondientes al sistema, incluyendo la especificación de los requisitos funcionales y no funcionales.

2.1 Modelo de dominio

Un modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos de *software*, muestra a los modeladores clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. Es una representación de las clases conceptuales del mundo real, no de componentes de *software* (23).

A continuación, se presenta el modelo del dominio que corresponde a la solución.

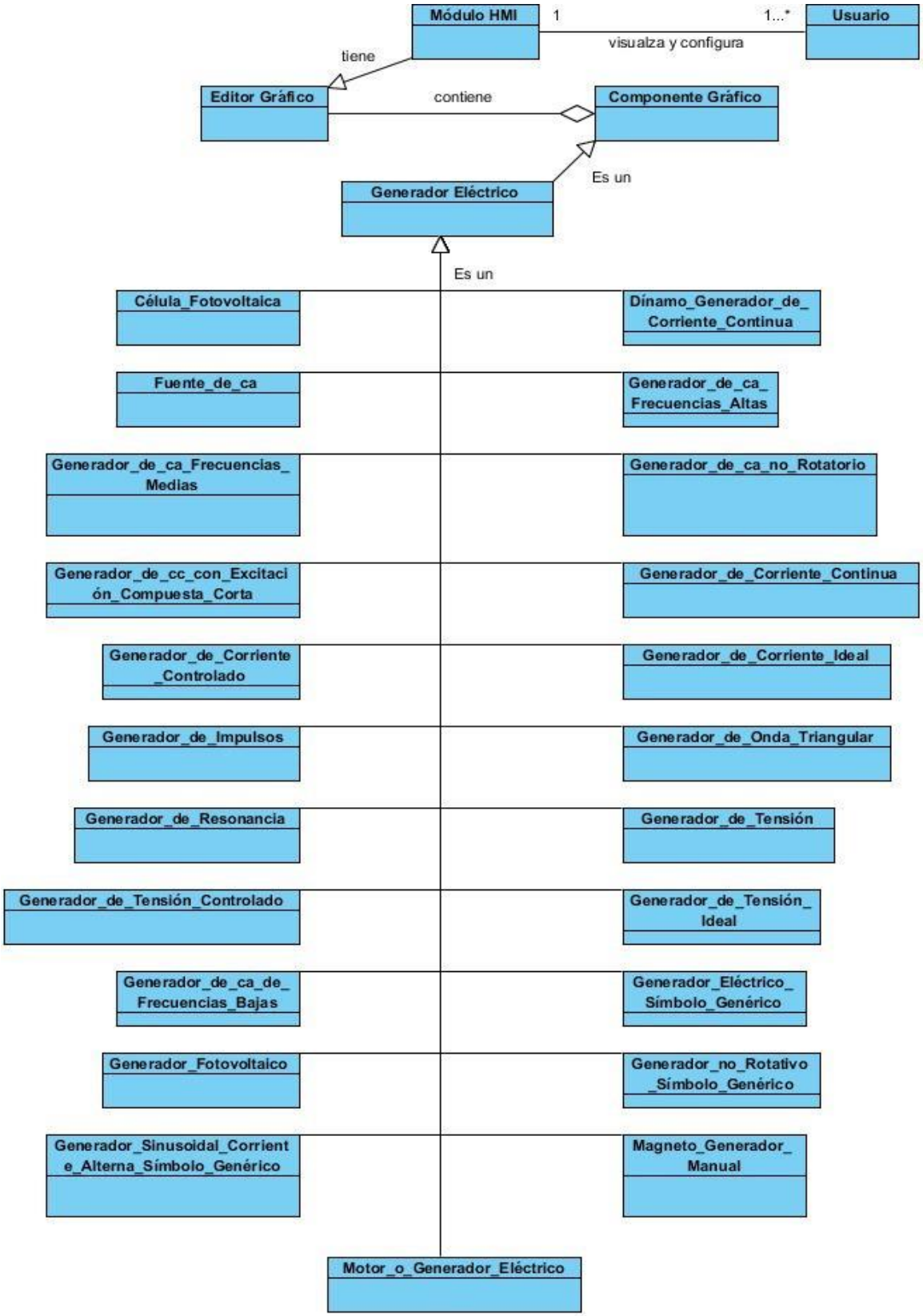


Fig. 8 Modelo Dominio

Se describen a continuación las clases que participan en el negocio:

- **Módulo *HMI*:** es el módulo que se encarga de representar al usuario u operador lo que sucede en el proceso.
- **Editor gráfico:** interfaz donde se muestra la paleta de componentes disponibles para la configuración.
- **Componente Gráfico:** representación abstracta de algún dispositivo del proceso que se está supervisando.
- **Generador Eléctrico:** Clase principal de la cual heredan las clases de los componentes de generadores eléctricos.
- **Célula Fotovoltaica:** Clase que contiene al componente célula fotovoltaica.
- **Fuente de ca:** Clase que contiene al componente fuente de corriente alterna.
- **Generador de ca Frecuencias Bajas:** Clase que contiene al componente generador de corriente alterna de frecuencias bajas.
- **Generador de ca Frecuencias Medias:** Clase que contiene al componente generador de corriente alterna frecuencias medias.
- **Generador de ca Frecuencias Altas:** Clase que contiene al componente generador de corriente alterna frecuencias altas.
- **Generador de ca no rotatorio:** Clase que contiene al componente generador de corriente alterna no rotatorio.
- **Dínamo Generador de Corriente Continua:** Clase que contiene al componente dínamo generador de corriente continua.
- **Generador de Corriente Continua:** Clase que contiene al componente generador de corriente continua.
- **Generador de cc con Excitación Compuesta Corta:** Clase que contiene al componente generador de corriente continua con excitación compuesta corta.
- **Generador de Corriente Controlado:** Clase que contiene al componente generador de corriente controlado.
- **Generador de Corriente Ideal:** Clase que contiene al componente generador de corriente ideal.
- **Generador de Impulsos:** Clase que contiene al componente generador de impulsos.

- **Generador de Onda Triangular:** Clase que contiene al componente generador de onda triangular.
- **Generador de Resonancia:** Clase que contiene al componente generador de resonancia.
- **Generador de Tensión:** Clase que contiene al componente generador de tensión.
- **Generador de Tensión Controlado:** Clase que contiene al componente generador de tensión controlado.
- **Generador de Tensión Ideal:** Clase que contiene al componente generador de tensión ideal.
- **Generador Fotovoltaico:** Clase que contiene al componente generador fotovoltaico.
- **Generador no Rotativo Símbolo Genérico:** Clase que contiene al componente generador no rotativo símbolo genérico.
- **Generador Sinusoidal Corriente Alterna Símbolo Genérico:** Clase que contiene al componente generador sinusoidal corriente alterna símbolo genérico.
- **Magneto Generador Manual:** Clase que contiene al componente magneto generador manual.
- **Motor o Generador Eléctrico:** Clase que contiene al componente motor o generador eléctrico.
- **Generador Eléctrico Símbolo Genérico:** Clase que contiene al componente generador eléctrico símbolo genérico.

2.2 Especificación de requisitos

La ingeniería de requisitos es el conjunto de actividades implicadas en descubrir, documentar y mantener un conjunto de requisitos del *software* a desarrollar. La captura de los mismos es un proceso en el cual los datos son extraídos a partir de la aplicación de técnicas de captura de información como son la tormenta de ideas y las entrevistas (24).

A partir del modelo del dominio presentado se realizó el levantamiento de los requisitos, los cuales están divididos en dos tipos: los funcionales y los no funcionales. Se especifican a continuación los requisitos definidos para el desarrollo de los componentes para generadores eléctricos.

2.2.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (25). Para el desarrollo de los componentes para generadores eléctricos se identifican los siguientes requisitos funcionales:

Tabla 2: Requisitos funcionales

#	Requisitos Funcionales	Descripción	Prioridad	Complejidad	Referencia cruzada
1	Seleccionar componente gráfico.	El sistema debe permitir seleccionar el componente gráfico arrastrándolo de la paleta de componentes hasta el despliegue.	Alta	Alta	HU Seleccionar componente gráfico.
2	Eliminar componente gráfico.	El sistema debe permitir eliminar el componente gráfico.	Media	Media	HU Eliminar componente gráfico.
3	Definir alias del componente gráfico.	El sistema debe de permitir definir el alias del componente gráfico a partir del “Inspector de Propiedades”.	Alta	Alta	HU Definir alias del componente gráfico.
4	Definir descripción del componente.	El sistema debe de permitir establecer una descripción asociada a la funcionalidad o utilidad del componente.	Alta	Alta	HU Definir descripción del componente.

5	Configurar la dimensión del ancho del componente.	El sistema debe permitir establecer el tamaño del ancho del componente gráfico.	Media	Media	HU Configurar la dimensión del ancho del componente.
6	Configurar la dimensión de la altura del componente.	El sistema debe permitir establecer el tamaño de la altura del componente gráfico.	Media	Media	HU Configurar la dimensión de la altura del componente.
7	Definir posición del componente gráfico.	El sistema debe permitir establecer la posición gráfico determinado por las coordenadas X, Y.	Alta	Alta	HU Definir posición del componente gráfico.
8	Definir opacidad del componente gráfico.	El sistema debe permitir establecer el nivel de opacidad del componente.	Baja	Baja	HU Definir opacidad del componente gráfico.
9	Definir rotación del componente gráfico.	El sistema debe permitir establecer la rotación del componente.	Media	Media	HU Definir rotación del componente gráfico.
10	Definir color de base para el componente gráfico.	El sistema debe de permitir modificar el color de base del componente gráfico.	Media	Media	HU Definir color de base para el componente gráfico

2.2.2 Requisitos no funcionales

Un requisito no funcional o atributo de calidad es un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que estos corresponden a los requisitos funcionales, son todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento (20).

Definen propiedades emergentes del sistema, tales como el tiempo de respuesta, la fiabilidad (20).

Requerimientos de *software*

- El sistema debe funcionar en el Sistema Operativo *Debian*, en su versión 8.

Requerimientos de *Hardware*

Debe ser ejecutado en computadoras que tengan como requerimientos mínimos los siguientes:

- Microprocesador: *dualcore* a 1.5 GHz.
- RAM: 2 GB

Restricciones en el diseño y la implementación.

- Se implementará utilizando el lenguaje de programación C++, en su versión 11.
- Se utilizará el IDE Qt Creator en su versión 2.5.0.
- Se empleará el marco de trabajo Qt, en su versión 5.3.1.

Requerimientos de usabilidad.

- La paleta de componentes de los generadores eléctricos debe poseer el nombre de "Generadores Eléctricos" para una fácil identificación, así como la información de cada componente gráfico que contiene.

Rendimiento.

- En el ambiente de despliegue solo se admiten hasta 100 componentes.

2.3 Historias de Usuario

Las Historias de Usuario (HU) son una técnica utilizada en AUP-UCI para especificar los requisitos del *software*. Se realizó una por cada funcionalidad del sistema, fueron empleadas para hacer estimaciones de tiempo y para el plan de iteraciones. Cada historia de usuario que se muestra a continuación se definió lo más comprensible posible para que pueda ser implementada por el desarrollador en pocas semanas (22).

Descripción de Historias de Usuario

- Número: Posee el número asignado a la HU.
- Actor: El usuario del sistema que utiliza o protagoniza la HU.
- Nombre de HU: Atributo que contiene el nombre de la HU.
- Prioridad en Negocio: Evidencia el nivel de prioridad de la HU en el negocio.
- Puntos estimados: Este atributo es una estimación hecha por el equipo de desarrollo del tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo.
- Iteración Asignada: Número de la iteración en la cual se desarrollará la HU.
- Descripción: Posee una breve descripción de lo que realizará la HU.
- Prototipo de Interfaz: Prototipo de la paleta de componentes.

Tabla 3 Historia de Usuario #1 Seleccionar componente gráfico

Historia de Usuario	
Número: 1	**Nombre de Historia:** Seleccionar componente gráfico.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 0.5
Nivel de Complejidad: Alta	**Puntos reales:** 0.5
Descripción: El sistema proporciona una interfaz al usuario donde el operador debe crear el componente gráfico arrastrándolo de la paleta de componentes hasta el despliegue.	
Prototipo de interfaz:	

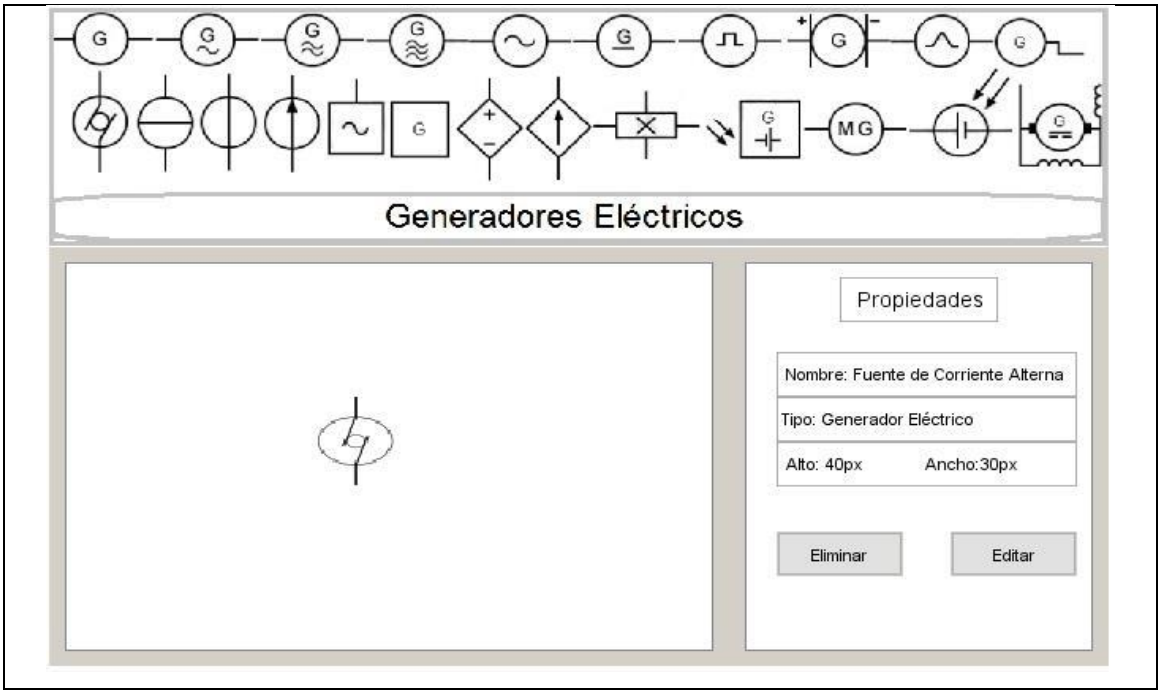


Tabla 4 Historia de Usuario #2 Eliminar el componente

Historia de Usuario	
Número: 2	**Nombre de Historia:** Eliminar el componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Media	**Puntos estimados:** 0.5
Nivel de Complejidad: Media	**Puntos reales:** 0.5
Descripción: El sistema proporciona una interfaz al usuario donde el operador al dar clic izquierdo sobre el componente o al presionar la tecla *Eliminar* en el teclado da la opción de eliminar el componente.	

Prototipo de interfaz:

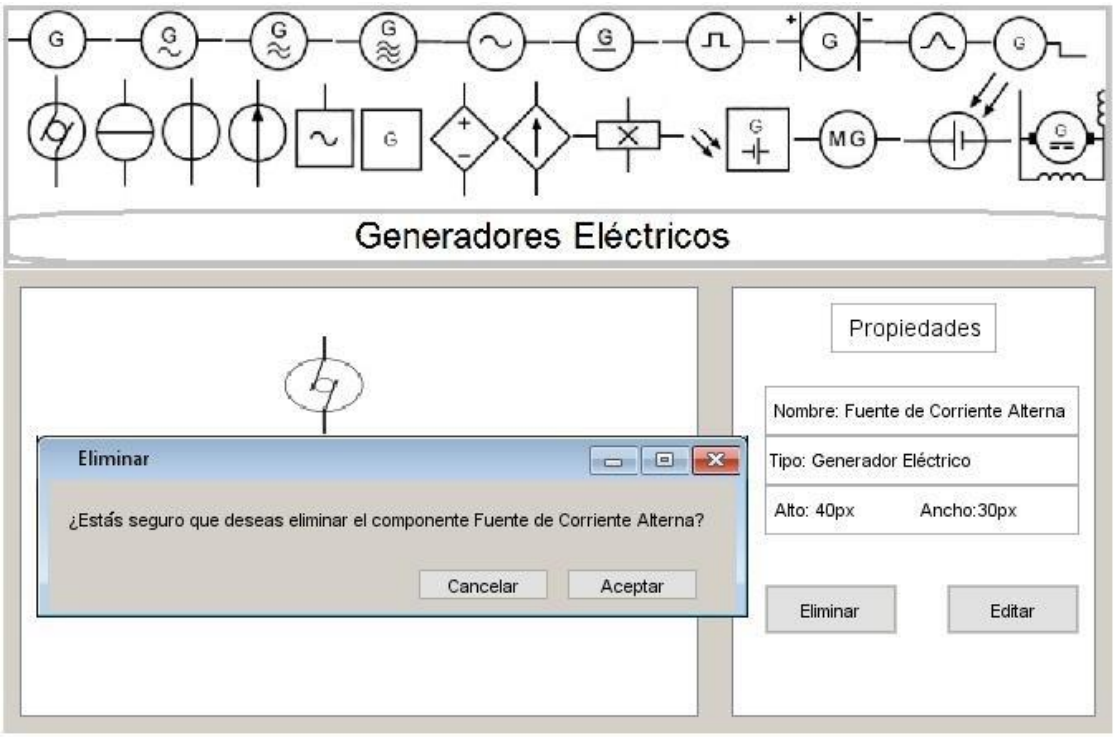


Tabla 5 Historia de Usuario #3 Definir nombre del componente

	Historia de Usuario
Número: 3	**Nombre de Historia:** Definir alias del componente
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 0.5
Nivel de Complejidad: Alta	**Puntos reales:** 0.5
Descripción: El sistema proporciona una interfaz al cliente donde puede escribir el alias del objeto, el mismo no debe exceder de 32 caracteres.	
Prototipo de interfaz:	

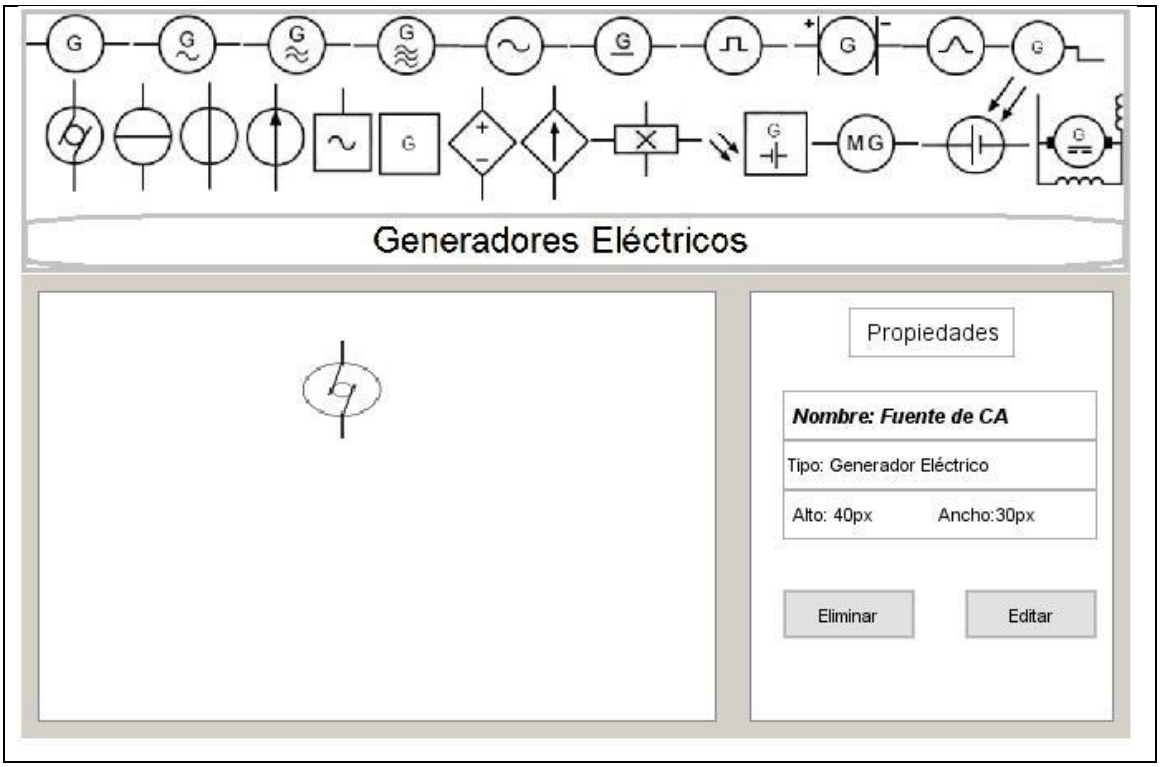


Tabla 6 Historia de Usuario #4 Definir descripción del componente

	Historia de Usuario
Número: 4	**Nombre de Historia** Definir descripción del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 0.5
Nivel de Complejidad: Alta	**Puntos reales:** 0.5
Descripción: El sistema debe de permitir establecer una descripción asociada a la funcionalidad o utilidad del componente.	
Prototipo de interfaz:	

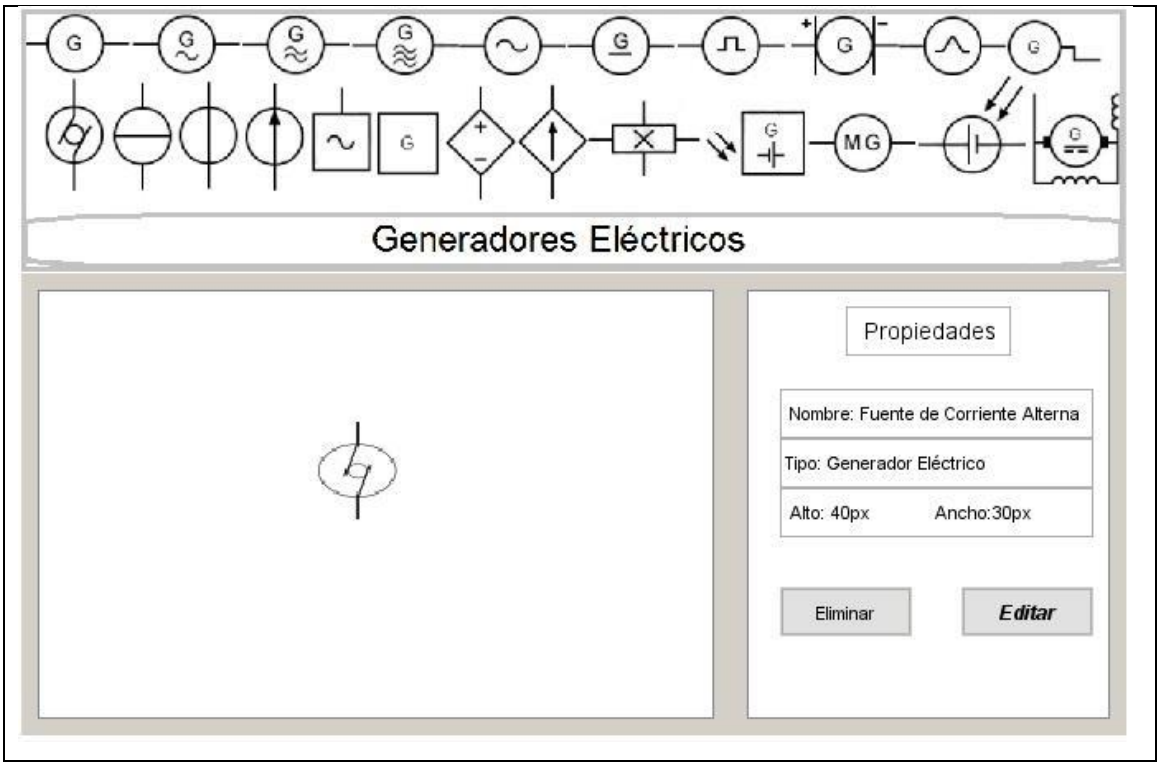


Tabla 7 Historia de Usuario #5 Configurar la dimensión del ancho del componente

Historia de Usuario	
Número: 5	**Nombre de Historia:** Configurar la dimensión del ancho del componente.
Actor: Usuario	**Iteración Asignada:** 2
Prioridad en Negocio: Media	**Puntos estimados:** 0.6
Nivel de Complejidad: Media	**Puntos reales:** 0.6
Descripción: El operador al realizar clic derecho encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde el operador puede definirle el ancho que desea que tenga el componente. También el operador al marcar el componente con un clic, situar el puntero en el borde izquierdo o derecho del componente y presionar el clic izquierdo, puede aumentar o disminuir el ancho del componente a su preferencia.	
Prototipo de interfaz:	

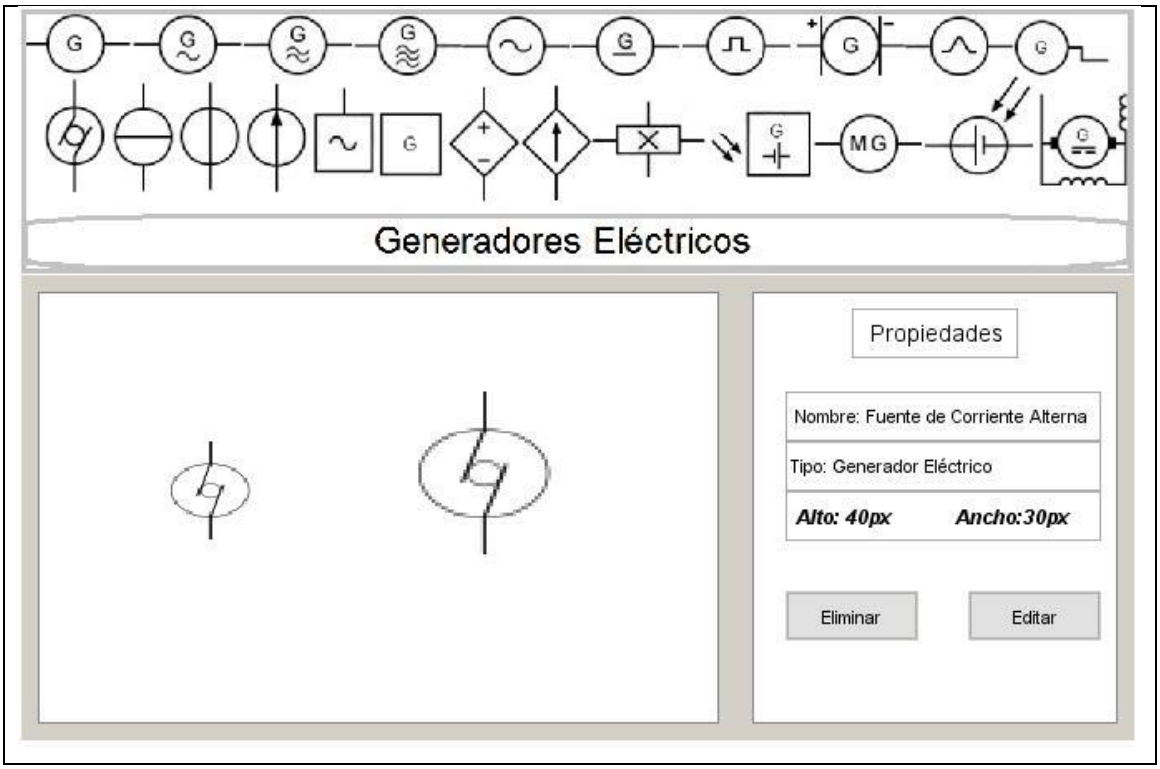


Tabla 8 Historia de Usuario #6 Configurar la dimensión de la altura del componente

	Historia de Usuario
Número: 6	**Nombre de Historia:** Configurar la dimensión de la altura del componente.
Actor: Usuario	**Iteración Asignada:** 2
Prioridad en Negocio: Media	**Puntos estimados:** 0.7
Nivel de Complejidad: Media	**Puntos reales:** 0.7
Descripción: El sistema proporciona una interfaz al usuario donde el operador al dar clic derecho encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde el operador puede definirle la altura que desea que tenga el componente. También el operador al marcar el componente con un clic, situar el puntero en el borde de arriba o debajo del componente y presionar el clic izquierdo, puede aumentar o disminuir la altura del componente a su preferencia.	
Prototipo de interfaz:	

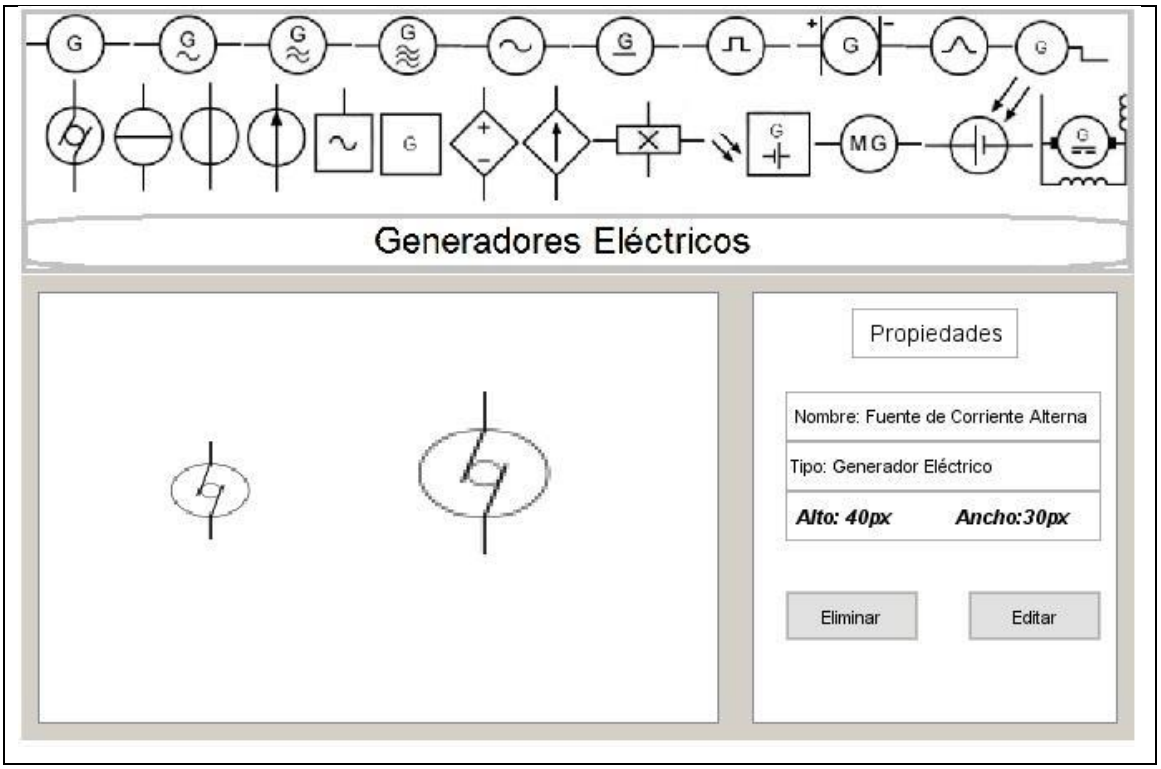


Tabla 9 Historia de Usuario #7 Configurar la posición (X,Y) del componente

	Historia de Usuario
Número: 7	**Nombre de Historia:** Configurar la posición (X, Y) del componente.
Actor: Usuario	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos estimados:** 0.6
Nivel de Complejidad: Alta	**Puntos reales:** 0.6
Descripción: El sistema proporciona una interfaz al usuario donde el operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, donde puede configurar la posición en la que se traslada el objeto sobre el eje X y el eje Y. También el operador puede seleccionar el componente dejando presionado el clic izquierdo sobre este y moverlo según su preferencia.	
Prototipo de interfaz:	

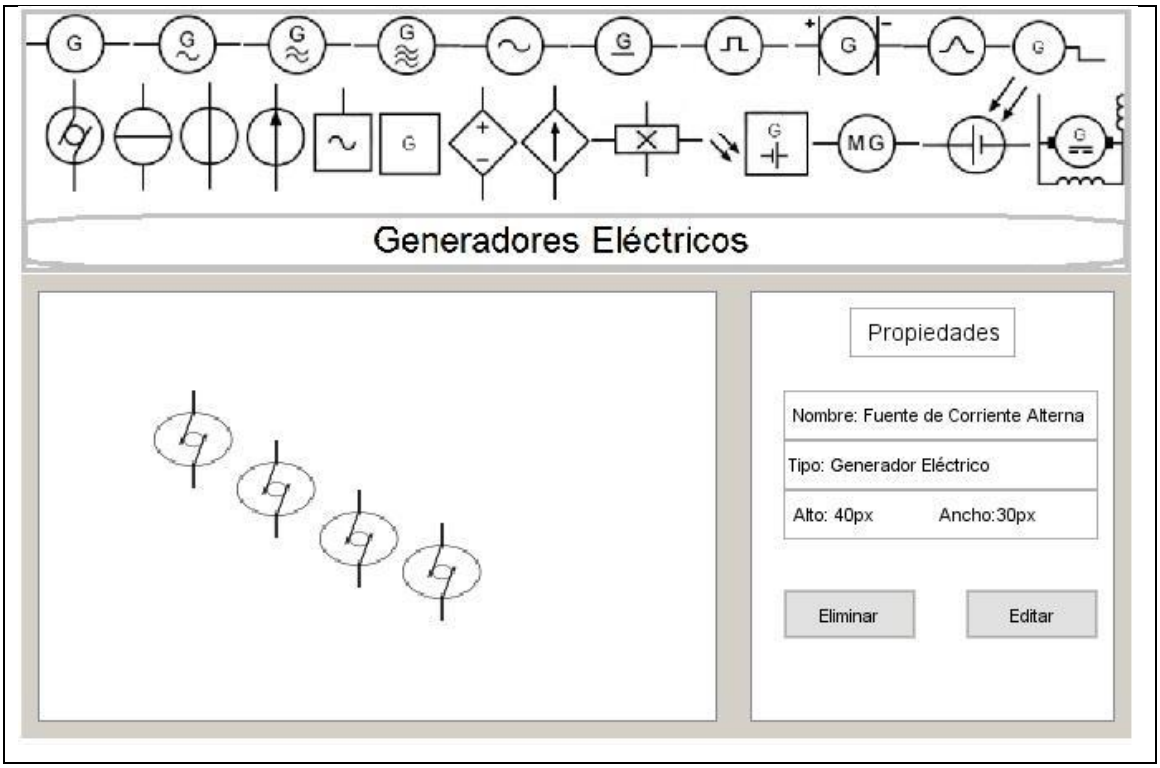


Tabla 10 Historia de Usuario #8 Configurar la opacidad del componente

	Historia de Usuario
Número: 8	**Nombre de Historia:** Configurar la opacidad del componente
Actor: Usuario	**Iteración Asignada:** 2
Prioridad en Negocio: Baja	**Puntos estimados:** 0.4
Nivel de Complejidad: Baja	**Puntos reales:** 0.4
Descripción: El operador al realizar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde el operador puede seleccionar la transparencia que tendrá el componente representado en el rango de 0 a 100.	
Prototipo de interfaz:	

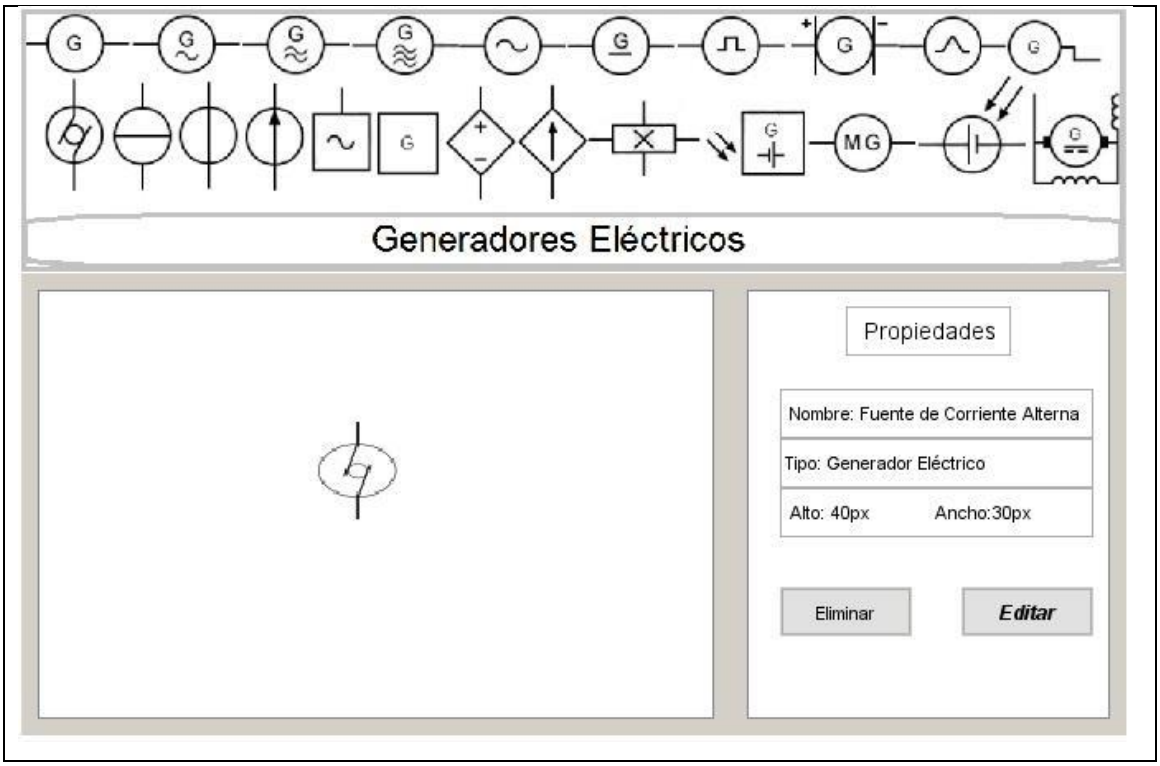


Tabla 11 Historia de Usuario #9 Permitir la rotación del componente

Historia de Usuario	
Número: 9	**Nombre de Historia:** Permitir la rotación del componente.
Actor: Usuario	**Iteración Asignada:** 3
Prioridad en Negocio: Media	**Puntos estimados:** 0.5
Nivel de Complejidad: Media	**Puntos reales:** 0.5
Descripción: Establece un ángulo de rotación. Definir donde localizar el centro de rotación. El centro de rotación se puede localizar en distintas posiciones, en la parte inferior del widget, en la parte superior, en el centro, a la derecha o a la izquierda.	
Prototipo de interfaz:	

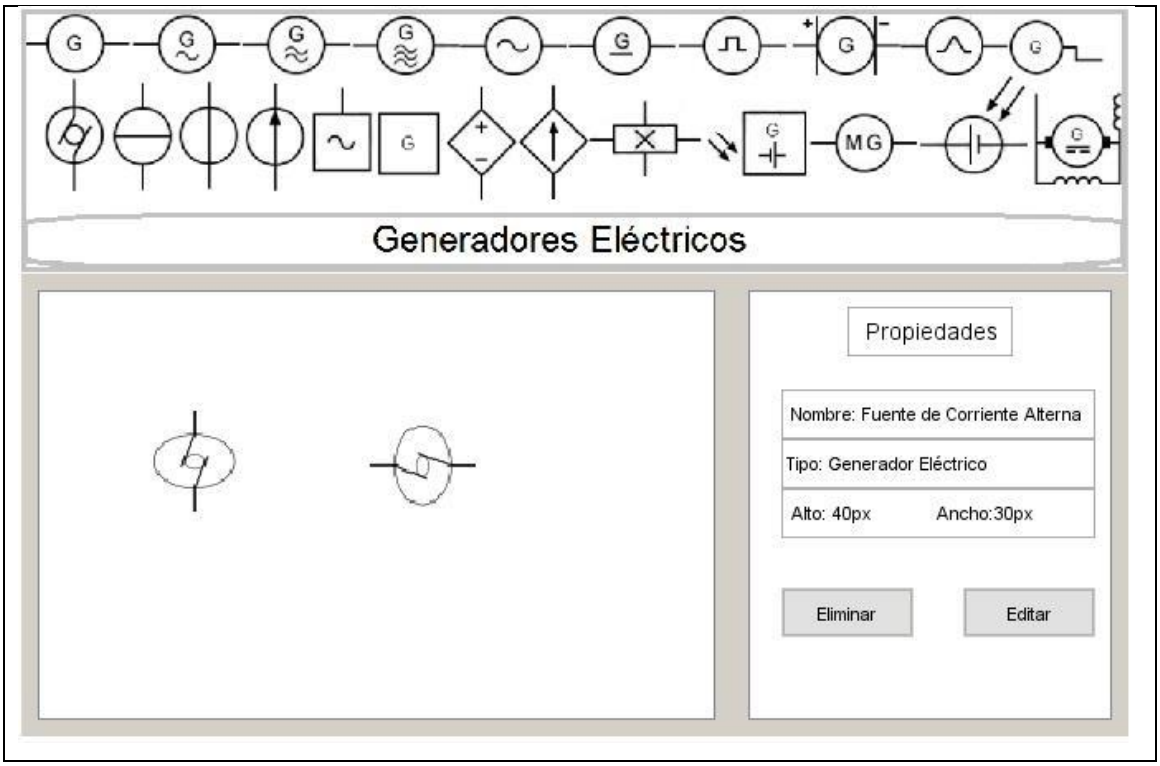
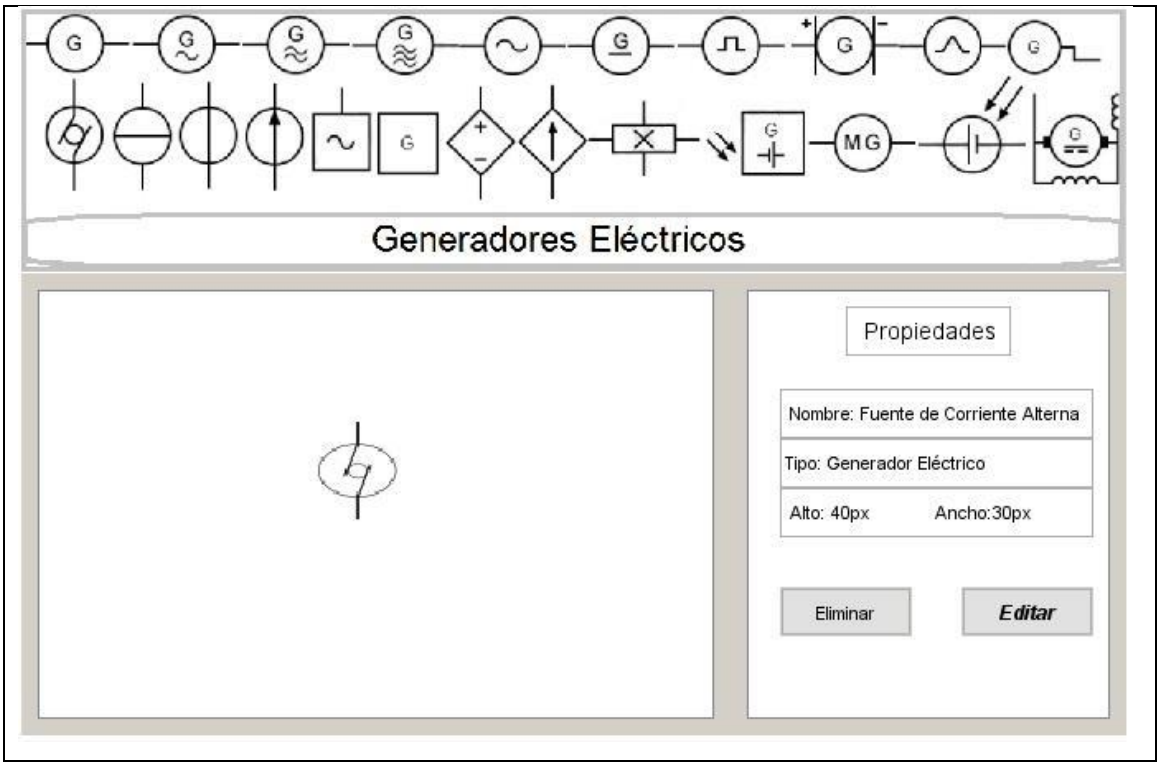


Tabla 12 Historia de Usuario #10 Definir color de base para el componente

	Historia de Usuario
Número: 10	**Nombre de Historia:** Definir color de base para el componente.
Actor: Usuario	**Iteración Asignada:** 3
Prioridad en Negocio: Media	**Puntos estimados:** 0.5
Nivel de Complejidad: Media	**Puntos reales:** 0.5
Descripción: El sistema debe de permitir modificar el color de base del componente gráfico.	
Prototipo de interfaz:	



2.4 Planificación del desarrollo

Esta fase tiene como propósito establecer un acuerdo entre clientes y desarrolladores sobre el menor tiempo en que la mayor cantidad de historias de usuarios puedan ser utilizadas.

2.4.1 Estimación de esfuerzos e iteraciones por Historia de Usuario.

La siguiente tabla muestra la estimación de esfuerzo para cada una de las historias de usuarios definidas en el desarrollo de la solución propuesta, el plan de iteraciones realizado y la duración total de las mismas.

Tabla 13: Estimación de esfuerzo

Historias de Usuario	Puntos de Estimación	Iteración	Duración total de las iteraciones (semanas)
Seleccionar componente gráfico.	0.5	1	2
Eliminar el componente	0.5	1	

Definir alias del componente	0.5	1	
Definir descripción del componente	0.5	1	
Configurar la dimensión del ancho del componente	0.6	2	3
Configurar la dimensión de la altura del componente	0.7	2	
Configurar la posición (X, Y) del componente	0.6	2	
Configurar la opacidad del componente	0.4	2	
Permitir la rotación del componente	0.5	3	1
Definir color de base para el componente	0,5	3	

2.4.2 Plan de iteraciones

Una vez definidas las historias de usuario (HU) se puntualiza la prioridad de cada una de ellas y la estimación de esfuerzo necesario para realizarlas para lo que se llevó a cabo un plan de iteraciones las cuales son mostradas a continuación.

Se definieron 3 iteraciones para el desarrollo de las HU de la aplicación. A continuación, se explican y justifican cada una de las iteraciones:

- Iteración 1: Para esta iteración se desarrollan las HU 1, 2, 3, 4 las cuales se encargan de creación de los componentes correspondientes.
- Iteración 2: En esta iteración se desarrollarán las HU 5, 6, 7, 8 las cuales se encargan de configurar los componentes, garantizando la representación gráfica de los parámetros de cada componente seleccionado.

- Iteración 3: En esta iteración se desarrollarán las HU 9,10 la cual se encarga de realizar animaciones a cada componente gráfico.

2.5 Diagrama de paquetes

Un diagrama de paquetes en el Lenguaje Unificado de Modelado representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre las agrupaciones (25).

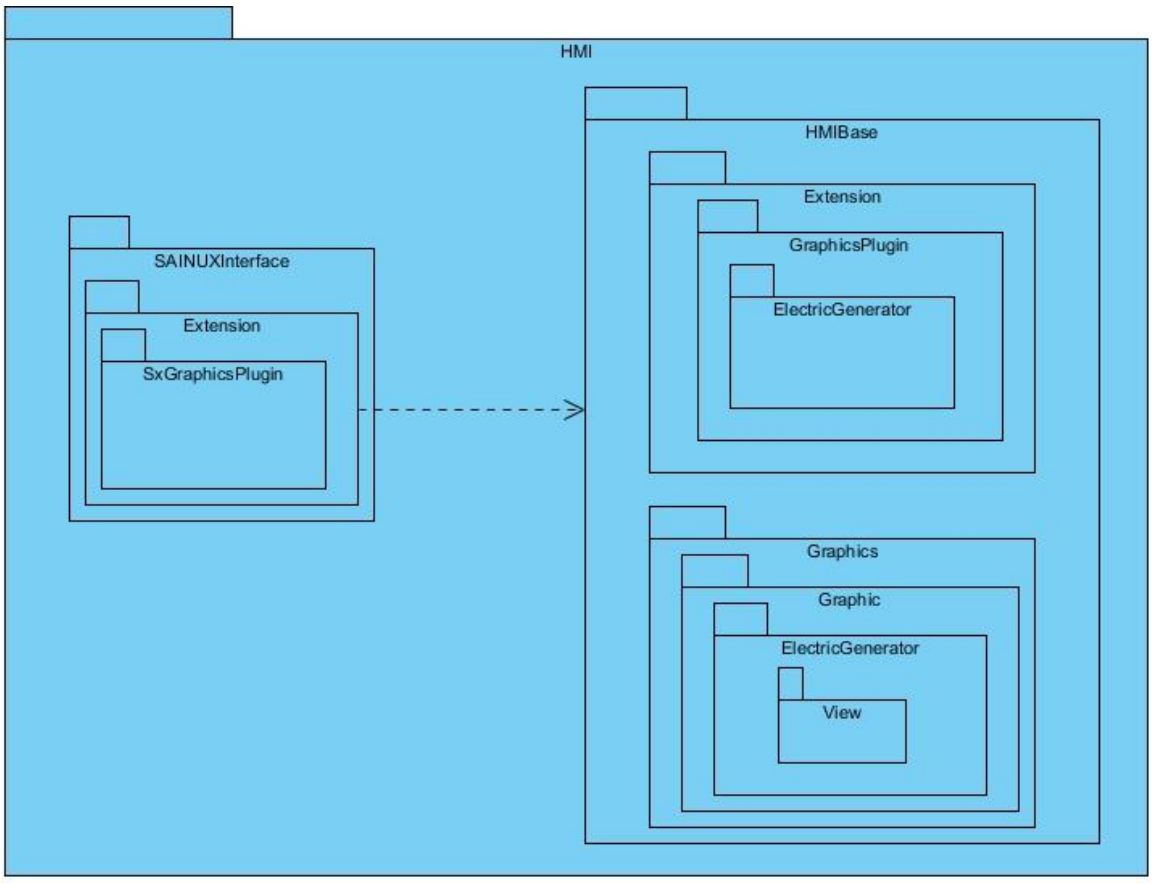


Fig. 9 Diagrama de Paquetes

La solución está estructurada en dos partes; Componentes y Paletas de Componentes. La primera parte contiene la carpeta SAINUXInterface, esta contiene la subcarpeta SxGraphicsPlugin que contiene las entidades referentes a las paletas de los componentes, pero de la interfaz SAINUX. En la segunda parte están las Paletas de Componentes divididas en la carpeta *HMI*Base donde se almacenan las propiedades

comunes de cualquier módulo de un sistema *SCADA* SAINUX, este módulo contiene los paquetes Extensión y dentro de él la subcarpeta GraphicsPlugin, donde se colocan las entidades relacionadas con la paleta de Generadores Eléctricos, y dentro de ellas según el tipo de componente que representa la paleta. El segundo paquete lo conforma Graphics donde se guardan las entidades de la biblioteca de componentes gráficos, dentro de él la subcarpeta Graphic que agrupa las entidades que representan a los componentes Generadores Eléctricos con sus entidades para representar la Vista.

2.6 Diseño de clases

El diseño es un proceso de resolución de problemas cuyo objetivo es encontrar y describir una forma. Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación, ya que una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones (20).

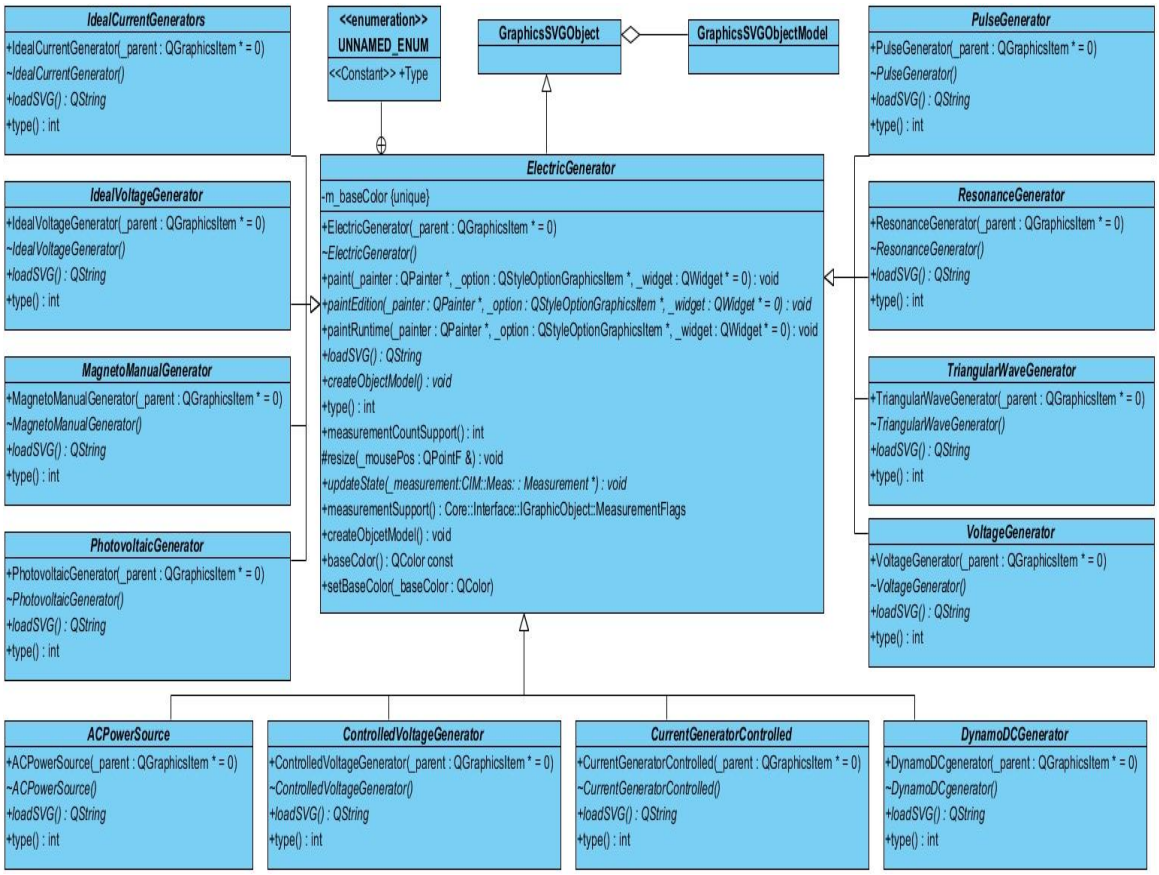


Fig. 10 Diagrama de clases de Generadores Eléctricos

2.7 Patrones de arquitectura

Los patrones de arquitectura son soluciones acertadas a un problema general, que establecen pautas para definir estructuras de diseño en el desarrollo de un *software*. Estos, están más enfocados a los diseños orientados a objetos, pero según dichos patrones a menudo tienen en cuenta características de los objetos tales como la herencia y el polimorfismo que proporciona generalidad (26).

El patrón de arquitectura usado es el Modelo / Vista que propone Qt donde el modelo comunica con una fuente de datos, proporcionando una interfaz para los otros componentes en la arquitectura. La naturaleza de la comunicación depende del tipo de fuente de datos, y la forma en que se implementa el modelo. La vista obtiene índices de modelo; estos son referencias a objetos de datos. Mediante el suministro de los índices del modelo, la vista puede recuperar los elementos de datos del origen de datos. En vistas

estándar, un delegado hace que los elementos de datos, cuando se edita un elemento, el delegado se comunica con el modelo usando directamente los índices modelo (27).

En general, las clases de modelo / vista se pueden separar en los grupos: modelos y vistas. Cada uno de estos componentes se define por las clases abstractas que proporcionan interfaces comunes y en algunos casos, las implementaciones por defecto de características. Las clases abstractas están destinados a ser una subclase con el fin de proporcionar el conjunto completo de funcionalidad esperada por otros componentes; esto también permite que los componentes especializados puedan ser escritos.

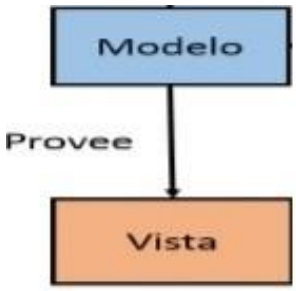


Fig. 11 Patrón de Arquitectura Modelo/Vista

La clase que compone el Modelo es:

- GraphicsSVGObjectModel.

De las clases que componen las vistas se encuentran:

- ACgeneratorHalfFrequencies
- ACgeneratorHighFrequencies
- ACgeneratorLowFrequency
- ACgeneratorNonRotating
- ACgeneratorSinusoidal
- ACpowerSource
- ControlledVoltageGenerator
- CurrentGeneratorControlled
- DCgenerator
- DCgeneratorWithShortCompoundExcitation
- DynamoDCgenerator

- ElectricGeneratorGenericSymbol
- ElectricGeneratorNonRotating
- IdealCurrentGenerator
- IdealVoltageGenerator
- MagnetoManualGenerator
- MotorElectricGenerator
- PhotovoltaicCell
- PhotovoltaicGenerator
- PulseGenerator
- ResonanceGenerator
- TriangularWaveGenerator
- VoltageGenerator

2.8 Patrones de diseño

Los patrones de diseño tratan los problemas que se repiten y que se presentan en situaciones particulares del diseño, con el fin de proponer soluciones a ellas. Se encargan de identificar clases, instancias, roles, colaboraciones entre estas, así como la distribución de responsabilidades. En resumen, es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular (28). Estos patrones de diseño Banda de los Cuatro [Gang of Four, (GoF por sus siglas en inglés)] son clasificados principalmente por tres grupos como son:

- **Patrones de comportamiento:** se utilizan a la hora de definir como las clases y objetos interaccionan entre ellos.
- **Patrones creacionales:** utilizados para instanciar objetos, y así separar la implementación del cliente de la de los objetos que se utilizan. Con ellos se intenta separar la lógica de creación de objetos y encapsularla.
- **Patrones estructurales:** utilizados para crear clases u objetos incluidos dentro de estructuras más complejas.

Para la asignación general de responsabilidades en el *software* existen patrones denominados Patrones Generales de Asignación de Responsabilidades (GRASP, por sus siglas en inglés), que describen los principios fundamentales de asignación de

responsabilidades a objetos, expresados como patrones. Seguidamente se exponen algunos patrones utilizados dentro del contexto de los módulos a implementar (28):

•**Alta cohesión:** Facilita la solución al problema ¿Cómo mantener manejable la complejidad? mediante la asignación de responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. El uso de este patrón se ve reflejado en todo el diagrama debido a que cada clase almacena la información de ella misma de manera coherente.

•**Experto:** Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. De forma general en el diseño del sistema se basa en asignar a cada clase la responsabilidad que solo ellas pueden realizar, pues cada una cuenta con la información necesaria para llevarlas a cabo. Por lo que en todas las clases del sistema utilizan este patrón.

•**Bajo acoplamiento:** Facilita la solución al problema ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización? Reflejada en todo el diagrama.

•**Creador:** Permite asignar el responsable de la creación de una nueva instancia de alguna clase. Explica que clase es la encargada de crear objetos, en determinados escenarios de ejecución y guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito general de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

El patrón GoF utilizado fue el Patrón de Comportamiento:

•**Plantilla** (*Template Method*): Define una operación, el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura. Es utilizado en clase abstracta donde el código común será usado por las clases que heredan de ella permitiendo la reescritura de determinados métodos. Reflejado en el método loadSVG ().

Conclusiones parciales

En el presente capítulo mediante la definición del modelo de domino se reflejó el punto de partida de la solución propuesta. Para un correcto funcionamiento de los componentes se especificaron 10 requisitos funcionales que deben ser cumplidos. El uso del patrón arquitectónico modelo-vista facilitó una mejor definición de la estructura de diseño para el desarrollo e integración de la paleta de componentes. La utilización de patrones GRASP (experto, creador, alta cohesión y bajo acoplamiento) y GoF (plantilla) permitió obtener diseños de clases sólidos al estructurarlos adecuadamente. Con la realización del diagrama de clases, se obtuvo una visión más clara de la solución en términos de implementación.

Capítulo 3: Implementación y prueba

Introducción

En este capítulo se expondrá el estándar de codificación utilizado para el desarrollo de la solución, así como su respectivo diagrama de componentes y de despliegue. Además, se procede a la implementación y prueba de la propuesta de solución, para detectar posibles fallas. Una vez corregidos los errores podrá ser desplegada la paleta de componentes de generadores eléctricos para su utilización.

3.1 Diagrama de componentes

Un diagrama de componentes representa la separación de un sistema de *software* en componentes físicos (por ejemplo, archivos, módulos, paquetes, base de datos, código fuente, binario o ejecutable) y muestra las dependencias y organización existente entre estos componentes. Los diagramas de componentes prevalecen en el campo de la arquitectura de *software*, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema (25).

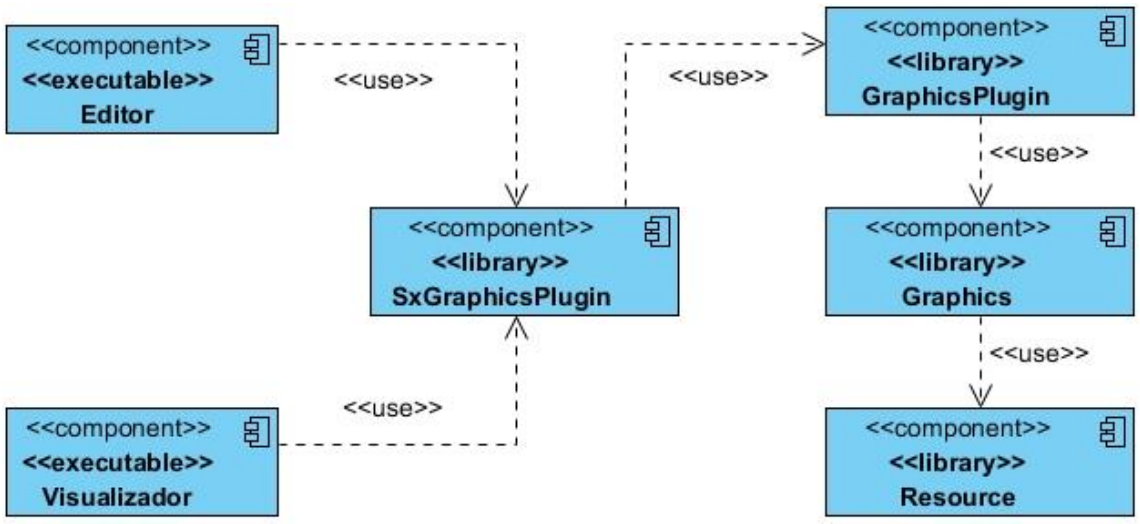


Fig. 12 Diagrama de Componentes

El diagrama de componentes antes representado está estructurado por ejecutables y bibliotecas:

Editor: Es la aplicación donde se trabaja en el entorno de configuración para representar los procesos del campo.

Visualizador: Es una aplicación donde se supervisa el área configurada para interactuar con los componentes gráficos.

Biblioteca Graphics Plugins: En esta biblioteca se almacenan los Plugins de los objetos gráficos.

Biblioteca Graphics: Aquí se encuentran almacenados los objetos gráficos que se utilizan para representar lo componentes gráficos, ya sea de forma simple o componentes complejos.

SxGraphicsPlugins: Es una interfaz de la biblioteca GraphicsPlugins específica para los componentes de SAINUX 2.0.

Resource: Biblioteca donde se encuentran agrupadas cada una de las entidades encargadas de brindar las imágenes, los iconos y los diseños de los componentes en el formato SVG.

3.2 Diagrama de despliegue

Con el objetivo de proveer una descripción de la distribución física del sistema se realiza el modelo de despliegue, mediante el cual se muestran las relaciones físicas de los distintos nodos que componen el sistema. Los diagramas de despliegue son capaces de describir la arquitectura física del sistema durante la ejecución en términos de procesadores, dispositivos y componentes de *software*. Permite una mejor comprensión entre la correspondencia de la arquitectura de *software* y la arquitectura de *hardware* (20).

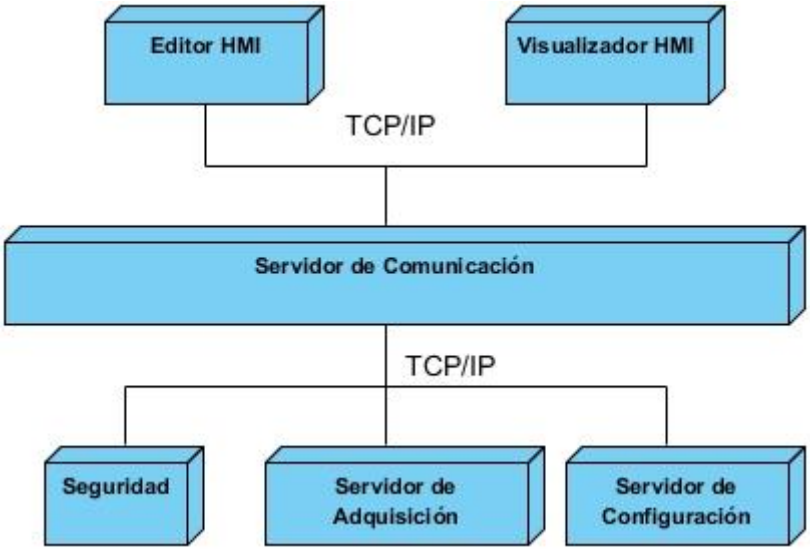


Fig. 13 Diagrama de Despliegue

3.2.1 Descripción del diagrama de despliegue

El módulo *HMI* cuenta con los dos entornos, editor y visualizador, es en este módulo es específicamente donde se encuentra la solución desarrollada. Los módulos adquisición, configuración y seguridad se ejecutan en una PC, que puede ser un servidor para cada uno por separado poniendo en evidencia la arquitectura distribuida del SAINUX, la conexión entre estos módulos se realiza utilizando comunicación TCP/IP.

3.3 Estándar de codificación

Uno de los instrumentos que facilitan el trabajo y la calidad del *software* son la adopción de estilos y estándares de codificación. El uso de estos estándares tiene innumerables ventajas entre ellas lograr un estilo de código homogéneo asegurando su legibilidad y proveer una guía para el encargado del mantenimiento/actualización del sistema, con código claro y bien documentado. Además, ayuda a mejorar el proceso de codificación haciéndolo en gran medida eficiente y en muchos casos reutilizables. La solución propuesta en este trabajo es parte del sistema *SCADA* SAINUX 2.0 el estándar de codificación utilizado fue definido por el proyecto. Algunas de las pautas definidas por el proyecto son (29):

- Es importante especificar el nombre del autor y la fecha de creación de cualquier estructura de código, para ello se utilizan los comandos @autor y @date.

- El código será escrito en inglés y la documentación en español.
- Las variables y funciones comienzan con letra minúscula. Cada palabra consecutiva en el nombre comienza con letra mayúscula.
- Las variables de corto alcance, por ejemplo, un contador, deberían tener nombres cortos y simples
- Los atributos de las clases deben empezar con m_ seguido del nombre del atributo, en el caso de atributos compuestos, la inicial de la segunda palabra debe comenzar con mayúscula.
- Las funciones utilizan la nomenclatura camello.
- Los parámetros que recibe una función deben empezar con _ (guion bajo).
- Todas las funcionalidades y atributos deben seguir el siguiente formato: para documentar @brief Nombre del método.
- Ninguna función debe tener más de 200 líneas.
- Los valores de los numerativos deben ser con letras mayúsculas.
- Las secciones public, protected y private son declaradas en el orden expuesto.

3.4 Solución del problema

En este epígrafe se muestran las interfaces del Sistema *SCADA* SAINUX 2.0, donde se encuentran las bibliotecas de componentes del *HMI* en el entorno de configuración (Editor) y el entorno de visualización (*Runtime*). A continuación se muestra la paleta de componentes básicos, la cual antes se utilizaba para crear los generadores eléctricos en el despliegue y la solución desarrollada de la paleta de componentes gráficos de generadores eléctricos.

3.4.1 Despliegues del sistema

La biblioteca de componentes gráficos del *HMI* SAINUX 2.0 no le provee al mantenedor componentes para la representación de Generadores Eléctricos. Para solventar dicha situación el mantenedor utiliza o combina varias figuras geométricas simples como: polilínea, curvas, rombos, semicírculos y líneas, o el componente imagen. En la siguiente figura se muestran los accesorios que se utilizaban anteriormente.

Fig. 14 Componentes Básicos

En la siguiente imagen se muestran los componentes para la representación de los Generadores Eléctricos, al añadir un total de 23 componentes teniendo estas nuevas funcionalidades, se ve cómo se da cumplimiento a los problemas planteados sobre la representación de generadores eléctricos en el sistema.

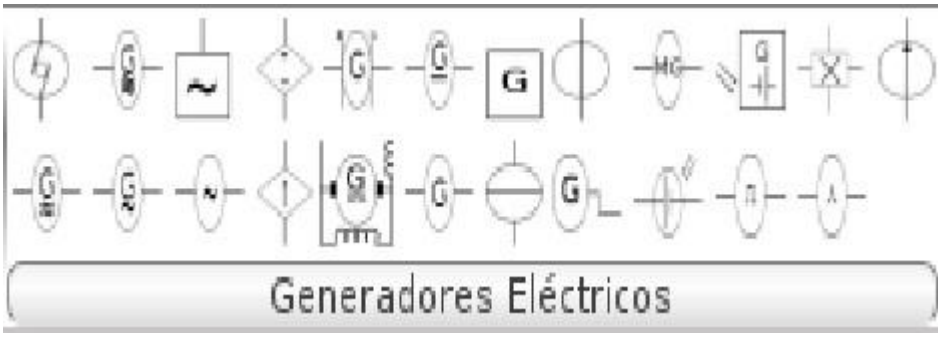


Fig. 15 Paleta de Componentes de Generadores Eléctricos

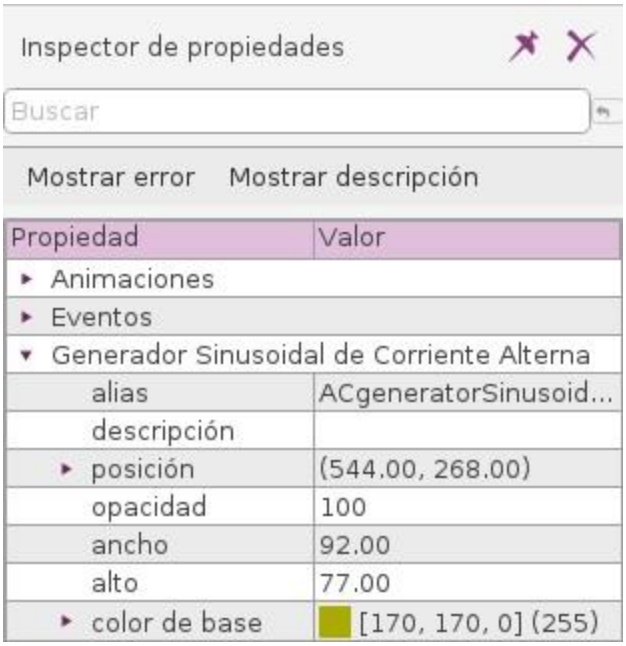


Fig. 16 Inspector de Propiedades

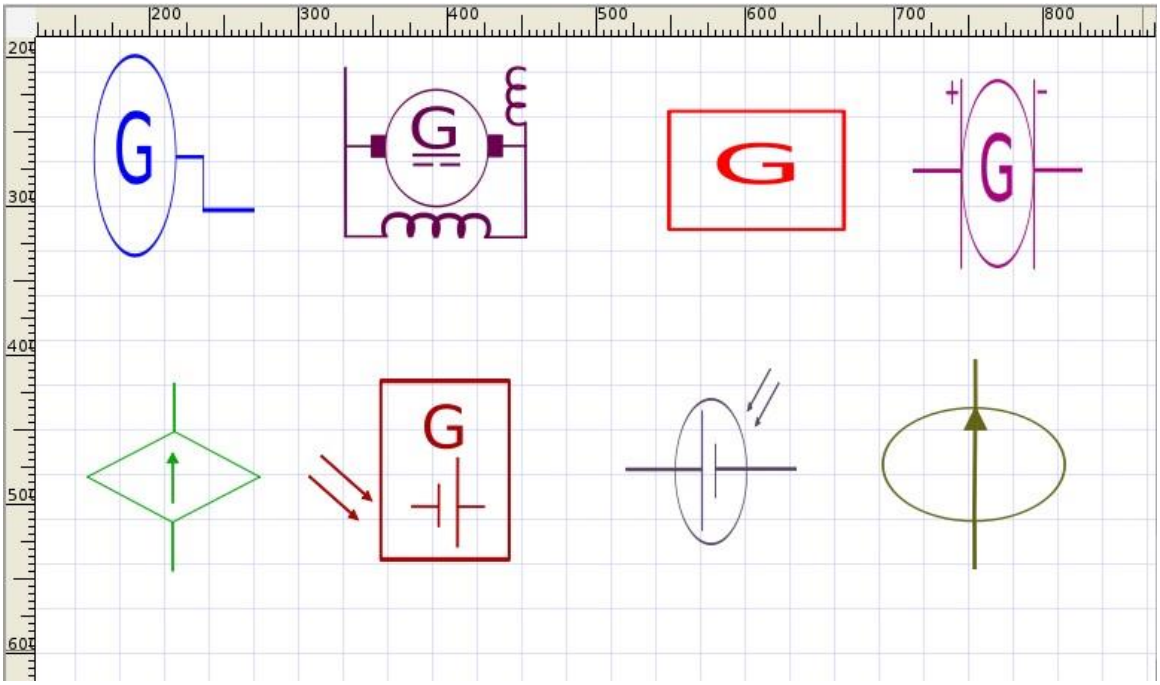


Fig. 17 Componentes en el área de edición.

3.5 Pruebas del sistema

Las pruebas de *software* comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo (30).

Existen diversas clasificaciones de los tipos de pruebas de *software*, por ejemplo pruebas funcionales, pruebas de sistemas, pruebas no funcionales, pruebas de caja negra y pruebas de caja blanca.

Para probar la solución propuesta se empleó el método de Caja Negra.

3.5.1 Prueba de Caja Negra

Son las pruebas que se llevan a cabo sobre la interfaz del *software*. O sea, los casos de prueba (CP) pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene (30).

Estas pruebas permiten encontrar:

- Funciones que estén incorrectas o ausentes.

- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas. Errores de rendimiento.
- Errores de inicialización y terminación

Una vez terminada la fase de implementación de los componentes gráficos se procede a ejecutar las pruebas mediante casos de prueba. Estas pruebas permitirán validar que se cumplan los requerimientos funcionales establecidos. Para garantizar la calidad y el correcto cumplimiento de los requisitos funcionales, se realizarán iteraciones de pruebas hasta que la cantidad de errores encontrados sea cero.

3.5.2 Casos de prueba

Para representar las pruebas de funcionamiento se definieron los siguientes elementos:

- Código: Representa al caso de prueba, incluye el número de HU, de la prueba y si posee diferentes escenarios.
- HU: Número de la HU a la cual pertenece.
- Nombre: Junto al código conforma el identificador del caso de prueba.
- Descripción: Acción que debe realizar el sistema.
- Condiciones de ejecución: Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- Entrada/Pasos de Ejecución: Incluye las entradas necesarias para realizar el sistema, además de los pasos para realizar el caso de prueba.
- Resultados Esperados: Descripción de la respuesta del sistema ante el caso de prueba.
- Evaluación de la prueba: Clasificación de la prueba en satisfactoria o insatisfactoria.

La siguiente tabla muestra las pruebas funcionales de la HU1 perteneciente a la primera iteración de sistema:

Tabla 14 Caso de Prueba Funcionales #1

Caso de Prueba Funcional	
Número: 1	**Historia de Usuario: 1**
Nombre: Permitir seleccionar componente gráfico.	
Descripción: Comprobar que el componente se selecciona de forma correcta.	

Condiciones de Ejecución: El usuario debe comprobar que se puede arrastrar el componente hacia el despliegue.
Entradas/ Pasos de Ejecución: - Seleccionar el componente en la paleta de componentes. - Arrastrarlo hasta el despliegue.
Resultado esperado: El componente en el despliegue.
Evaluación de la prueba: Prueba satisfactoria.

La descripción de las siguientes pruebas funcionales se encuentra en el Anexo 1.

Tabla 15 Diseño de Caso Prueba

Pruebas del Sistema	HU	Iteración	NC	Cerrada
	10	1ra	7	7
		2da	3	3
		3ra	1	1
		4ta	0	0

Una vez realizados los casos de pruebas para un total de 10 historias de usuarios, con tres iteraciones se eliminaron las 11 no conformidades encontradas. Como se muestra en la figura siguiente:

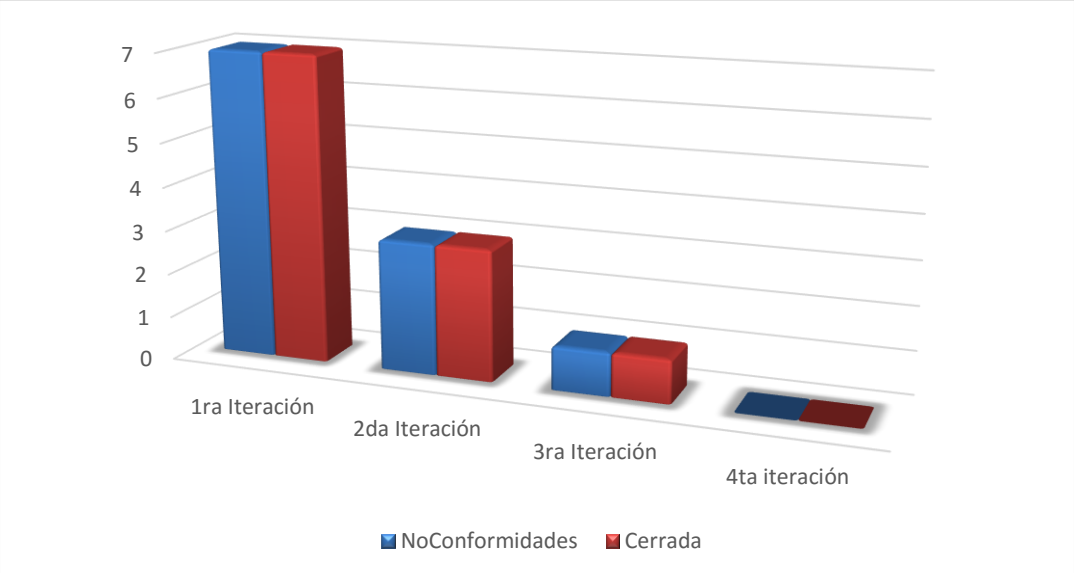


Fig. 18 Plan de Iteraciones

Como resultado se detectó que los 11 errores encontrados fueron de funcionalidades que no realizaban la acción prevista y errores visuales donde el sistema no mostraba lo requerido. El resultado de las pruebas realizadas al *software* fue satisfactorio. Se comprobó que las respuestas fueran las esperadas y que se cumple con las especificaciones definidas a partir de requisitos funcionales planteados con anterioridad.

Conclusiones parciales

Durante el desarrollo de este capítulo se expusieron los principales artefactos del modelo de implementación, lo que permitió mostrar los componentes del sistema y sus relaciones a través del diagrama de componentes. Mediante el diagrama de despliegue se modeló la arquitectura en tiempo de ejecución. Se implementaron los generadores eléctricos como componentes gráficos de la propuesta de solución y se integraron a la paleta de componentes gráficos en el módulo *HMI* lo cual permitió dar solución al problema de la investigación planteado al principio del presente trabajo. Se ejecutaron pruebas funcionales donde se encontraron errores y cada no conformidad detectada fue satisfactoriamente resuelta.

Conclusiones generales

Con la realización de este trabajo se desarrollaron componentes gráficos para la representación de Generadores Eléctricos sobre la Interfaz Hombre Máquina del *SCADA* SAINUX 2.0. De esta forma se le da cumplimiento al objetivo propuesto al inicio de la investigación, por lo que se llega a las siguientes conclusiones:

- El estudio del estado del arte y las principales soluciones existentes de los componentes gráficos en los diferentes *SCADA*, permitieron sentar las bases teóricas para la futura implementación de la solución propuesta.
- Se definieron los requerimientos funcionales y no funcionales de la solución propuesta, lo cual permitió establecer las pautas para el desarrollo de la paleta de componentes de generadores eléctricos.
- La metodología de desarrollo de *software* utilizada, posibilitó la correcta definición de las Historias de Usuario, logrando así ver con claridad los requisitos funcionales.
- Con la implementación de la solución el *SCADA* SAINUX 2.0 cuenta con la paleta de componentes gráficos para la representación de generadores eléctricos, permitiendo una mejor supervisión de los procesos en el sector eléctrico.
- Mediante la realización de las pruebas funcionales se comprobó que la propuesta de solución cumple con los requerimientos pactados con el centro

Recomendaciones

Se recomienda incluir en el *HMI* del *SCADA* SAINUX 2.0 una paleta de componentes que representen los generadores eléctricos trifásicos, las pilas eléctricas y las baterías eléctricas.

Referencias bibliográficas

1. Concepto de Informática Industrial. *https://es.scribd.com/document/256268436/Concepto-de-Informatica-Industrial.* [En línea] 2017.

2. Dagoberto Monteros, David B. Barrantes y José M. Quirós. *Introducción a los sistemas de control, supervisión y adquisición de datos(SCADA).* Costa Rica : s.n., 2004.

3. Industrial, Centro de Informática. *Manual de Usuario-Editor.* 2017.

4. —. *Manual de Usuario-Visualizador.* 2017.

5. Dayra Iris Hechevarría Rodríguez, Lannie Octavio Herrera Pérez. *Procedimientos para la Gestión de Requisitos en el SCADA SAINUX.* La Habana : s.n., 2012.

6. Universidad Nacional de Quilmes. Introducción a HMI. *HMI.* Quilmes : s.n., 2012.

7. Rojas, Adolfo Yasser Santana. *Visualizador web de la Interfaz Hombre-Máquina del SCADA Guardián del ALBA.* La Habana : UCI, 2015.

8. Universidad Autónoma de Madrid. UAM. *Componentes.* [En línea] 2015. https://www.uam.es.

9. GRUPEL. Energy Everywhere. [En línea] 2017. https://grupel.eu/es/grupel-es/tipos-generadores-de-corriente-electrica/.

10. Generador de corriente alterna y corriente continua. *https://sites.google.com/site/fisicacbtis162/services/3-2-8-generador-de-corriente-alterna-y-corriente-continua-1.* [En línea] 2017.

11. Normalización, Asociación Española de. *Símbolos gráficos para esquemas.* Madrid : AENOR, 1997.

12. Systems, Tibbo. Internet of things integration platform. *AggreGate.* [En línea] http://aggregate.tibbo.com/.

13. Schneider Electric. [En línea] 2017. https://www.schneider-electric.com/ww/en/.

14. Velneo. *Que es el formato SVG.* [En línea] 2017. https://velneo.es/que-es-el-formato-svg.

15. SUSE LLC and others. [En línea] 2017. https://es.opensuse.org/Inkscape.

16. Florentino, Deiry. Historia Del Computador, Arquitectura, Lenguajes de Programación y Algoritmos - reparacion de computadoras. [En línea] http://nectacellcomputer.jimdo.com/historia-del-computador-arquitecturalenguajes-de-programación-y-algoritmos/Esto se llama Cultura General.

17. Importancia de los framework. [En línea] 2016. http://qt.nokia.com/products/developer-tools/

18. Raydel Raúl Viñoso Sosa, Alexander Roquero Figuer. Sistema Gestor de Proceso de Media v2. [En línea] 2015. http://publicaciones.uci.cu/index.php/SC.

19. Visual Paradigm. [En línea] 2016. https://www.visual-paradigm.com/features/.

20. Pressman, Roger S. *Ingeniería del Software. Un enfoque práctico.* D. F. México : MC GRAW HILL INTERAMERICANA, 2010. ISBN: 978-607-15-0314-5.

21. Chimbote, Universidad Católica Los Angeles. *Metodología de Desarrollo de Sotfware.* 2017.

22. *Metodología de Desarrollo para la Actividad Productiva de la UCI.* SÁNCHEZ, TAMARA RODRÍGUEZ. La Habana : RCCI, 2017.

23. Bridón, Yordanis. *Interfaz asíncrona para la comunicación de los controladores lógicos programables utilizando el protocolo industrial Ethernet/IP.* La Habana : s.n., 2008.

24. Tibaquirá, Carolina Martínez. *Ingeniería de Software 1.* Santiago de Chile : s.n., 2012.

25. Sommerville, Ed Ian. *Ingeniería de software.* 7ma. Madrid : Pearson Eduación, 2005. ISBN: 84-7829·074-5.

26. Wesley. *Ingeniería de Software (Parte IV Desarrollo).* 7ma. México : Pearson Addison, 2005. ISBN 84-7829-074-5.

27. The Qt Company. Qt Creator Documentation. [En línea] 2018. http://doc.qt.io/archives/qt-4.8/model-view-programming.

28. Larman. *UML y Patrones.* 2da. Madrid : Pearson Educación, 2002.

29. Lorenzo, Ariel Chávez. *Estándares de codificación para C++.* 2010.

30. Toledo, Federico. *Introducción a las Pruebas de Sistemas de Información.* Montevideo : Kindle Edition, 2014.

31. Simbología electrónica. *www.simbologia-electronica.com.* [En línea] 2017.

Anexos

Anexo 1: Pruebas Funcionales

Tabla 16 Caso de Prueba Funcionales #2

Caso de Prueba Funcional	
Número: 2	**Historia de Usuario: 2**
Nombre: Permitir eliminar el componente.	
Descripción: Comprobar que el componente se elimine.	
Condiciones de Ejecución: El usuario debe comprobar que el componente sea eliminado del sistema.	
Entradas/ Pasos de Ejecución: Seleccionar el Componente. - Clic derecho encima del componente - Selecciona la opción de Eliminar el componente. Otra vía: - Presionar la tecla *Delete* en el teclado y se elimina el componente.	
Resultado esperado: El usuario pudo eliminar el componente utilizando cualquiera de las dos opciones.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 17 Caso de Prueba Funcionales #3

Caso de Prueba Funcional	
Número: 3	**Historia de Usuario: 3**
Nombre: Permitir definir alias del componente.	
Descripción: Comprobar que se puede definir el alias del componente forma correcta sin utilizar números o símbolos.	
Condiciones de Ejecución: El usuario debe comprobar que se puede definir el alias del componente.	
Entradas/ Pasos de Ejecución: - Seleccionar el componente. - Clic derecho con el ratón.	

- Seleccionar la opción Propiedades. - Cambiar el alias.
Resultado esperado: El alias ha sido definido.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 18 Caso de Prueba Funcionales #4

Caso de Prueba Funcional	
Número: 4	**Historia de Usuario: 4**
Nombre: Permitir definir descripción del componente.	
Descripción: Comprobar que se puede permitir establecer una descripción asociada a la funcionalidad o utilidad del componente.	
Condiciones de Ejecución: El usuario debe comprobar que se puede establecer una descripción asociada a la funcionalidad o utilidad del componente.	
Entradas/ Pasos de Ejecución: - Seleccionar el componente. - Clic derecho con el ratón. - Seleccionar la opción Propiedades. - Ir a Descripción. -Escribir descripción.	
Resultado esperado: La descripción ha sido creada.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 19 Caso de Prueba Funcionales #5

Caso de Prueba Funcional	
Número: 5	**Historia de Usuario: 5**
Nombre: Permitir configurar la dimensión del ancho del componente.	
Descripción: Comprobar que se puede configurar la dimensión del componente, definido en el rango (- 2147483647; 2147483647).	
Condiciones de Ejecución: El usuario debe comprobar que el componente puede ser configurado en ancho.	
Entradas/ Pasos de Ejecución: - Seleccionar el objeto.	

- Abrir el inspector de propiedades - Selecciona en el inspector de propiedades la opción para aumentar o disminuir el ancho del componente. Otra vía: - Con el clic izquierdo seleccionar uno de los puntos donde puede redimensionarse el componente para ancharlo o estrecharlo.
Resultado esperado: El componente se redimensiona sin perder calidad utilizando cualquiera de las dos opciones.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 20 Caso de Prueba Funcionales #6

Caso de Prueba Funcional	
Número: 6	**Historia de Usuario: 6**
Nombre: Permitir configurar la dimensión de altura del componente.	
Descripción: Comprobar que se puede configurar la dimensión del componente, definido en el rango (- 2147483647; 2147483647).	
Condiciones de Ejecución: El usuario debe comprobar que el componente puede ser configurado en altura.	
Entradas/ Pasos de Ejecución: - Seleccionar el objeto. - Abrir el inspector de propiedades - Selecciona en el inspector de propiedades la opción para aumentar o disminuir la altura del componente. Otra vía: - Con el clic izquierdo seleccionar uno de los puntos donde puede redimensionarse el componente para agrandarlo o achicarlo.	
Resultado esperado: El componente se redimensiona sin perder calidad utilizando cualquiera de las dos opciones.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 21 Caso de Prueba Funcionales #7

Caso de Prueba Funcional

Número: 7	**Historia de Usuario: 7**
Nombre: Permitir configurar la posición (X, Y) del componente.	
Descripción: Comprobar que el componente se traslada sobre cualquier punto, definido en el rango (- 2147483647; 2147483647).	
Condiciones de Ejecución: El usuario debe comprobar que el componente se puede mover sobre el despliegue.	
Entradas/ Pasos de Ejecución: - Seleccionar el componente y trasladarlo sobre cualquier punto.	
Resultado esperado: El componente se traslada sobre el despliegue en cualquier punto.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 22 Caso de Prueba Funcionales #8

Caso de Prueba Funcional	
Número: 8	**Historia de Usuario: 8**
Nombre: Permitir configurar la opacidad del componente.	
Descripción: Comprobar que al componente puede configurarse su opacidad.	
Condiciones de Ejecución: El usuario debe comprobar que se cumple la propiedad de opacidad para el componente.	
Entradas/ Pasos de Ejecución: - Seleccionar el componente. - Abrir el inspector de propiedades. - Seleccionar opción de opacidad. - Verificar que la opción de opacidad del componente funcione sobre el componente seleccionado.	
Resultado esperado: La opción de opacidad para el componente es configurable.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 23 Caso de Prueba Funcionales #9

Caso de Prueba Funcional	
Número: 9	**Historia de Usuario: 9**
Nombre: Permitir la rotación del componente.	

Descripción: Comprobar que el componente rota.
Condiciones de Ejecución: El usuario debe comprobar que el componente rota.
Entradas/ Pasos de Ejecución: - Seleccionar el componente. - Seleccionar en el menú contextual la opción rotar izquierda. - Seleccionar en el menú contextual la opción rotar derecha.
Resultado esperado: El usuario puede rotar el componente.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 24 Caso de Prueba Funcionales #10

Caso de Prueba Funcional	
Número: 10	**Historia de Usuario: 10**
Nombre: Permitir definir color de base para el componente.	
Descripción: Comprobar que el sistema permita definir el color de base del componente.	
Condiciones de Ejecución: El usuario debe comprobar que el sistema permite definir el color de base del componente.	
Entradas/ Pasos de Ejecución: - Seleccionar el componente. - Abrir el inspector de propiedades. - Seleccionar opción de color de base. - Verificar que la opción de color de base del componente funcione sobre el componente seleccionado.	
Resultado esperado: La opción de color de base para el componente es configurable.	
Evaluación de la prueba: Prueba satisfactoria.	

Anexo 2: Generadores Eléctricos *(31)*

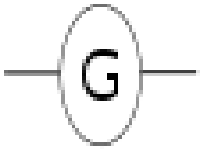


Fig. 19 Generador Eléctrico Símbolo Genérico

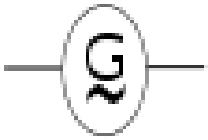


Fig. 20 Generador de CA Frecuencias Bajas

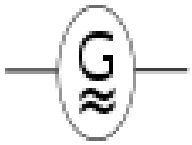


Fig. 21 Generador de CA Frecuencias Medias

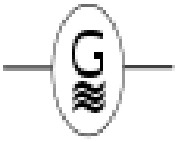


Fig. 22 Generador de CA Frecuencias Medias

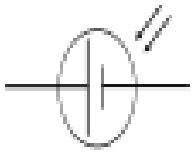


Fig. 23 Célula Fotovoltaica

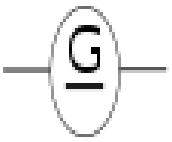


Fig. 24 Dinamo Generador de CC

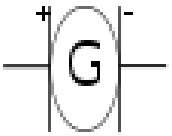


Fig. 25 Generador de CC

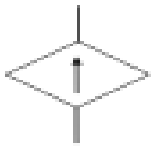


Fig. 26 Generador de Corriente Controlado

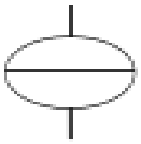


Fig. 27 Generador de Corriente Ideal

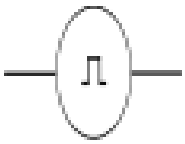


Fig. 28 Generador de Impulsos

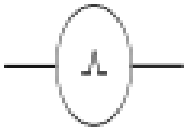


Fig. 29 Generador de Onda Triangular

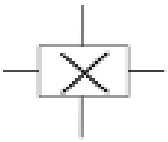


Fig. 30 Generador de Resonancia

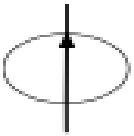


Fig. 31 Generador de Tensión

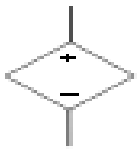


Fig. 32 Generador de Tensión Controlado

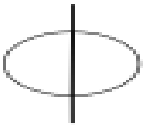


Fig. 33 Generador de Tensión Ideal

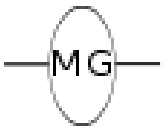


Fig. 34 Generador o Motor Eléctrico

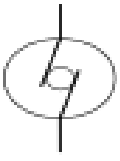


Fig. 35 Fuente de CA

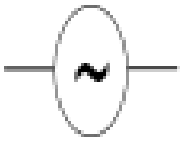


Fig. 36 Generador Sinusoidal CA

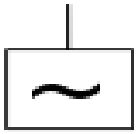


Fig. 37 Generador de CA no Rotatorio