Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

PMD: Entorno para el modelado y simulación de sistemas híbridos con aplicación a la prognosis

---

Alumno: Ángel Olivera Crego

Tutor: Anibal Bregón Bregón

PMD: Entorno para el modelado y simulación de sistemas híbridos con aplicación a la prognosis

Ángel Olivera Crego

Índice general

Índice de figuras

Lista de Tablas

Agradecimientos

Este proyecto no se habría podido llevar a cabo si no es gracias a todas esas personas que han estado a mi lado antes y durante la realización del mismo. Primero me gustaría agradecer a mis padres el apoyo emocional que han ejercidio sobre mí y toda la ayuda aportada,ya que sin ella jamás me encontraría aquí.

Por otro lado me gustaría agradecer a mi pareja el haber estado a mi lado durante todo el proceso final, animándome en los peores momentos y logrando sacarme una sonrisa en los momentos de mayor preocupación.

No querría dejar de mencionar la acogida que me hizo la Universidad de Valladolid y gracias a los magníficos profesores que he tenido durante este año, que me han mostrado el mundo de una forma diferente, más amigable y enriquecedora. Destacando sobretodo a Anibal por su confianza en mi para realizar este proyecto y animarme a seguir adelante. Sin olvidarme de Jose Vicente, quién me enseño a valorarme a mi mismo e incentivó mi creatividad.

Resumen

Debido al crecimiento de los sistemas autónomos y de la necesidad de que estos sistemas puedan realizar las funciones para las que han sido diseñados de la manera más eficiente, se hace fundamental poder calcular el momento en el que va a dejar de funcionar un sistema, conocido como EOL (End Of Life o fin de la vida útil). De ahí nace la idea de realizar este proyecto, el cual servirá de apoyo para el diseño de sistemas híbridos tanto electrónicos como mecánicos, hidráulicos... Ofreciendo al usuario una interfaz intuitiva y sencilla con la que poder modelar estos sistemas y posteriormente permitir aplicar los cálculos a través de scripts python de una forma mucho más rápida y gráfica.

Palabras claves: simulación, prognosis, sistemas híbridos, electrónica, física.

Capítulo 1

Descripción del proyecto

Se tratará brevemente de explicar cómo se organiza la Memoria del Trabajo Fin de Grado (TFG), del posible contenido de cada uno de los capítulos y secciones, así como de contenidos mínimos exigibles y algunas recomendaciones prácticas.

1.1. Introducción

El continuo desarrollo tecnológico e industrial ha llevado a la creación de sistemas cada vez más complejos, lo cual, junto con la necesidad de que estos sistemas sean lo más independiente posibles, hace necesaria la existencia de mecanismos para predecir cuando estos sistemas van a fallar o se van a quedar sin energía, conociendo algunas variables proporcionadas en tiempo real.

A día de hoy es bastante común que los dispositivos móviles con los que convivimos dispongan de una cierta tecnología que nos ayuda a determinar cuando se va a quedar sin batería, utilizando para ellos algoritmos que van memorizando nuestro comportamiento con ese dispositivo y prediciendo de ese modo en qué momento se va a quedar sin batería. Por otro lado, sin tener relación a simple vista con la informática tal y como la conocemos, los coches modernosy no tan modernos pueden predecir cuanta distancia podemos recorrer con lo que nos queda de combustible.

Si bien estas estimaciones pueden ser adecuadas para el usuario normal, en el sectores mas avanzados de la industria se hace necesario un conocimiento más preciso del momento en el que los sistemas van a dejar de funcionar. Casos críticos pueden ser drones de reconocimiento que o bien no consigan regresar a la base por falta de energía o bien regresen antes de terminar el máximo numero de misiones por temor a quedarse sin energía. O casos de plataformas petrolíferas situadas en medio del océano, donde llevar repuestos puede ser una labor que requiera mucho tiempo de desplazamiento, que en caso de tener una estimación muy próxima del momento del fallo el transporte se podría realizar con suficiente previsión.

1.2. Motivación

Una vez detallada la diferencia de los sistemas de predicción de fallo, se procede a detallar un ejemplo para defender el sistema que se quiere desarrollar.

Primero imaginemos una situación trágica: en una determinada zona ha desaparecido una persona sin motivo aparente. Para su localización se emplean todo tipo de medios disponibles, incluyendo entre ellos el lanzamiento de uno o varios drones de reconocimiento que a través de cámaras infrarrojas e IA puedan detectar la presencia de humanos.

Estos drones disponden de mecanismos para conocer el tiempo de vuelo que les queda basándose en la catidad de carga y el tiempo de vuelo que llevan (pero no tienendo en cuenta si el viento va a estar a favor o en contra). Al cabo de un rato el dron detecta que su batería se va a acabar y cree que no va a poder acudir a rastrear la siguiente localización por lo que decide regresar a la estación base. El tiempo que el dron pierde mientras regresa de su posición actual a la base, se cambia o se recarga la batería y retorna para proseguir con su misión, puede ser vital para una persona que se encuentra en una situación de riesgo.

De ahí nace la idea de tratar de mejorar los actuales cálculos de la prognosis por cálculos más avanzados donde entren varias variables, permitiendo calcular de forma más precisa el momento en el que el sistema va a dejar de funcionar correctamente.

Con este proyecto se pretende crear una interfaz gráfica amigable que permita la realización de diagramas de distintos tipos de sistemas, a tarvés de los cuales se pueda facilitar el cálculo de dicho problema.

Su principal aplicación está pensada para empresas de electrónica o sector industrial. Con el fin de agilizar los procesos de cálculo del tiempo de vida estimado para esos sistemas. Cabe remarcar que las posiblidades pueden ser amplias ya que el propio usuario final podrá añadir, elminar y modificar cualquier componente del diagrama y añadirle cualquier tipo de ecuación, variable...

A través de este documento se indicarán las metodologías empleadas, así como de las tecnologías.

1.3. Objetivos del trabajo

El principal objetivo de este trabajo es la realización de una aplicación con interfaz gráfica que permita el diseño y estudio de sistemas principalmente electrónicos e hidráulicos. El principal motivo para realizarla es la no existencia de una aplicación que nos permita realizar este tipo de cálculos, si bien existen muchas aplicaciones que permiten hacer simulaciones pocas son multiplataforma, de código abierto y desde luego no conozco ninguna que permita realizar estudios de sistemas híbridos, lo cuál hace realmente necesario que exista esta aplicación.

Su funcionalidad principal es la de permitir crear diagramas de circuitos electrónicos, hidráulicos u otros pudiendo el usuario final definir tanto nuevos componentes como nuevos tipos de sistemas. Con esto se cubre la necesidad de una aplicación amigable para realizar el estudio de la prognosis.

Para facilitar su utilización en diversos sistemas, se va a realizar utilizando un lenguaje de programación el cual permita producir aplicaciones para varios tipos de Sistemas Operativos. El idioma principal de la aplicació será el inglés.

1.4. Estado del arte

Aunque existen multitud de herramientas pensadas para la creación de diagramas principalmente electrónicos: Fritzing, Eagle, Oregano, Kicad; ninguna permite realizar los cálculos para el estudio de la prognosis. Aparte no todas están pensadas para que el usuario pueda definir sus propios componentes ni tampoco son multiplataforma y tampoco permiten diseñar sistemas híbridos. De ahí nace la idea de la creación de una aplicación que nos permita diseñar diferentes modelos para posteriormente transformarlos en archivos que puedan ser procesados por los scripts de cálculo existentes.

No obstante se tendrán en cuenta algunos aspectos del diseño de algunas de estas herramientas, ya que manteniendo un diseño similar la curva de aprendizaje será mucho menor que tener que habituarse a una interfaz nueva.

Como normalmente trabajo en linux voy a analizar dos aplicaciones relacionadas con el diseño de diagramas. Una relacionada con la electrónica, puesto que la aplicación a desarrollar tendrá bastante importancia en el campo de la electrónica y otra con la radio, que aunque no tiene mucho que ver con la temática de este TFG si resulta interesante analizar el tipo de interfaz que tiene y me resulta a mi mucho más fácil ya que es con la aplicación que tengo más experiencia.

1.4.1. KiCad

KiCad es una herramienta de código abierto, programada en c++, desarrollada principalmente para diseñar circuitos electrónicos. Sus principales ventajas son la flexibilidad y la adaptabilidad, dispone de una gran biblioteca de componentes electrónicos y además permite añadir nuevos componentes. Además también permite diseñar circuitos impresos para su posterior producción.

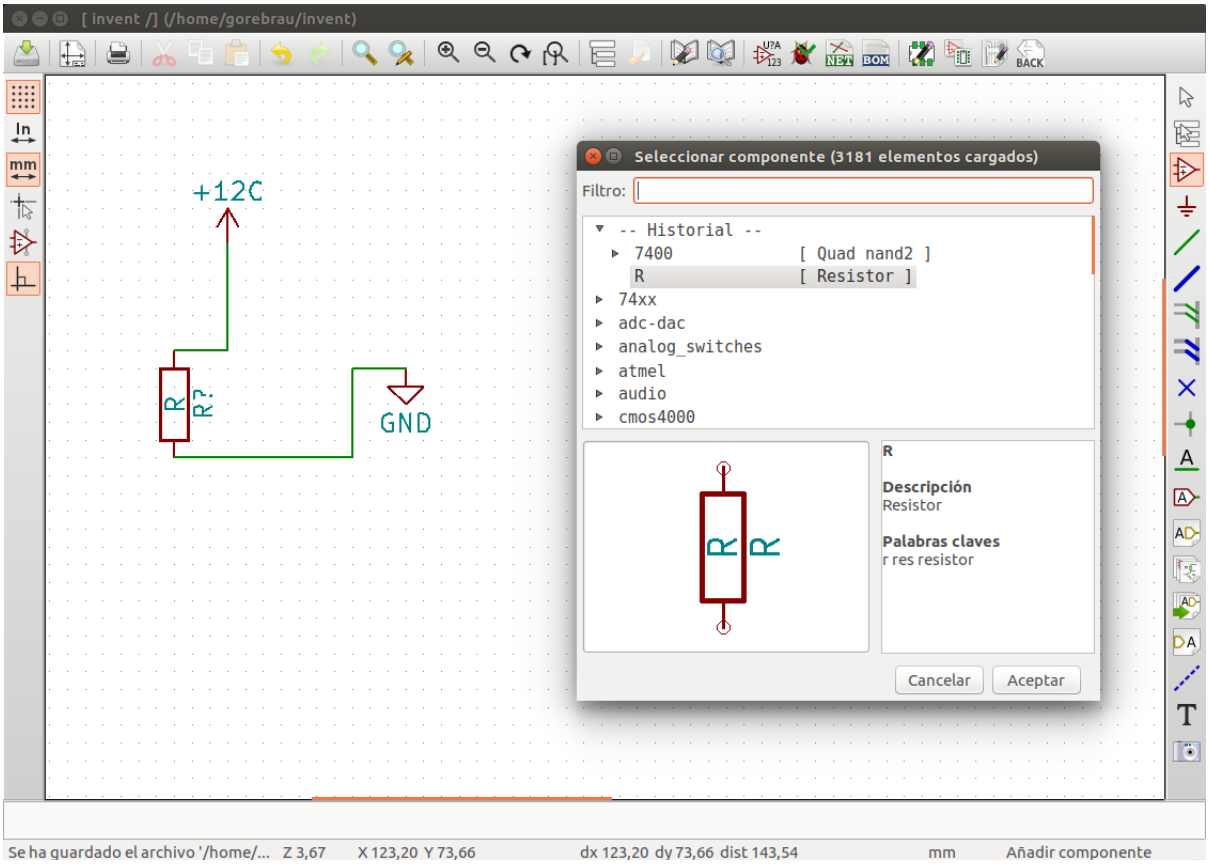


Figura 1.1: KiCad

De la interfaz de KiCad sacamos una idea de cómo puede ser la forma de unir componentes y de representarlos en la pantalla, aunque no nos agrada del todo la iteración con el usuario ya que los cursores cambian poco y es fácil equivocarse, además resulta poco práctico el eliminar componentes. Por otro lado la lista de componentes está muy bien esctructurada aunque no nos termina de convencer que sea a través de una ventana emergente.

1.4.2. GNU Radio

GNU Radio es una herramienta de código abierto, programada en python, que provee bloques de procesamiento de señal para implementar en sistemas de radio definida por software. A través de este software se puede trabajar con señales de radiofrecuencia ya bien sea a través de ficheros capturados como de dispositivos hardware "diseñados"para tal proposito. Permitiendo aplicar diferentes algoritmos a esas señales con los que obtener otras señales.

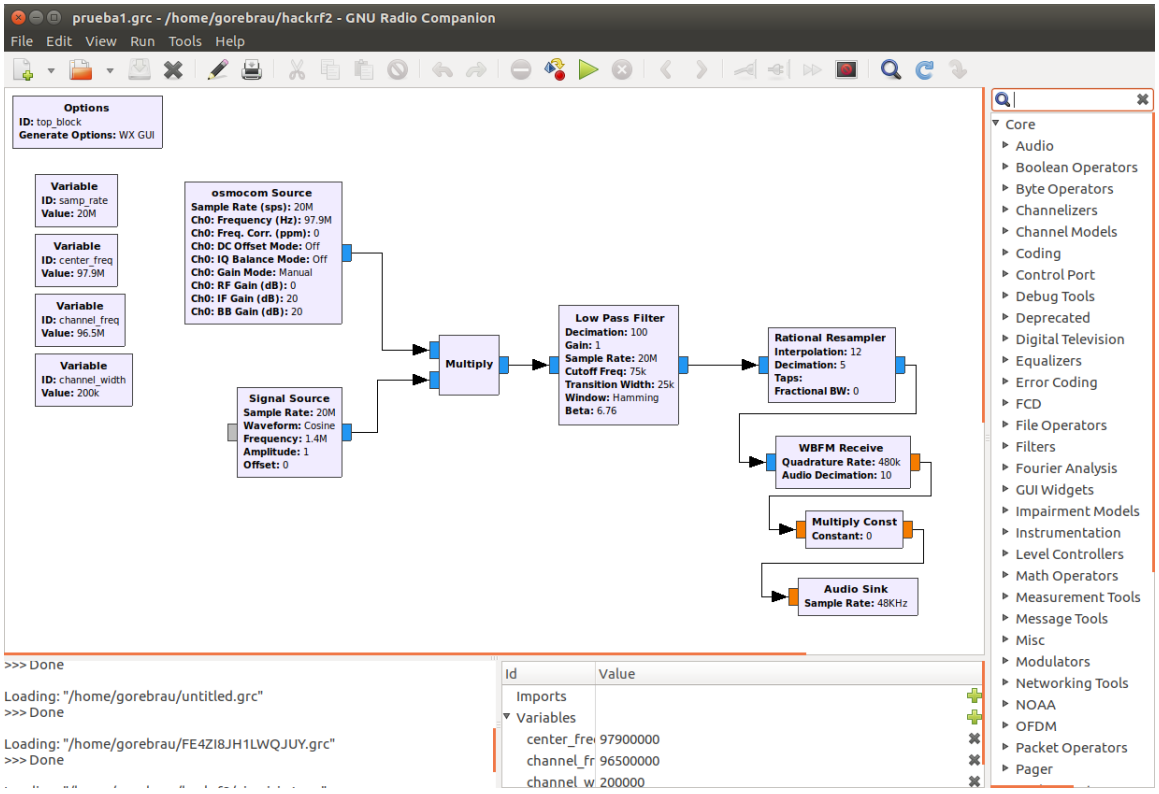


Figura 1.2: GNU Radio

De la interfaz de GNU radio nos quedamos con el diseño simplista y el listado de los diferentes componentes. Nos gusta la interfáz de drag and Drop que tiene.

1.5. Organización de la memoria

En este apartado se pretende hacer un repaso de la extructura de la memoria para que el lector pueda tener un resumen de los contenidos de ella. Para ello se van a explicar brevemente en que consisten los 9 capítulos que conforman este documento.

En el capítulo 1 se centra en la introducción del proyecto. Con él se pretende poner al lector en la situación del desarrollo de este proyecto. Para ello se explicará con una

introducción en qué consiste el proyecto, las motivaciones para su realización, los objetivos que se pretenden cumplir y el actual estado del arte.

En el capítulo 2 se explica las metodologías empleadas para la realización de este proyecto, detallando el proceso de software y aspectos relacionados con las herramientas empleads, lenguajes de programación, etc.

En el capítulo 3 se detalla la planificación del proyecto a través de estimaciones económicas y temporales, indicando al final los presupuestos obtenidos para la realización del mismo.

En el capítulo 4 se realiza el análisis de la aplicación a través de análisis de requisitos y su posterior transformación en casos de uso y clases de análisis.

En el capítulo 5 se detallan los aspectos del diseño de la aplicación, se detallan aspectos relativos a la arquitectura, asi como a los aspectos relacionados con la interfaz.

En el capítulo 6 se detallan los aspectos de implementación empezando por un breve análisis de las herramientas utilizadas y prosiguiendo con un análisis de las partes de código más relevantes.

En el capítulo 7 se detallan las pruebas a las que se ha sometido el software. Mostrando posibles problemas y sus consecuentes correcciónes.

En el capítulo 8 se detallan los manuales de usuario, instalación y compilación, para facilitar al usuario la utilización de este software.

En el capítulo 9, se finaliza el proyecto indicando las conclusiones que hemos obtenido durante su desarrollo, así como futuras lineas de desarrollo.

Finalmente se añaden algunos anexos donde se incluyen algunos casos de uso no detallados en la memoria del proyecto.

Capítulo 2

Metodología

2.1. Proceso de desarrollo

La aplicación se realizará como un desarrollo iterativo incremental, basado en prototipos no desechables los cuales se van a ir refinando hasta conseguir una versión definitiva. Una vez terminada una iteración se obtenemos un hito el cual será una versión ejecutable y entregable al cliente. Esta metodología nos permitirá ahorrar tiempo y coste cuando se requieran aplicar cambios.

Durante este modelo las fases de anlálsis y diseño cobran más fuerza durante las primeras iteraciones, mientras que las de coficiación y pruebas tendrán más impacto en las últimas iteraciones.

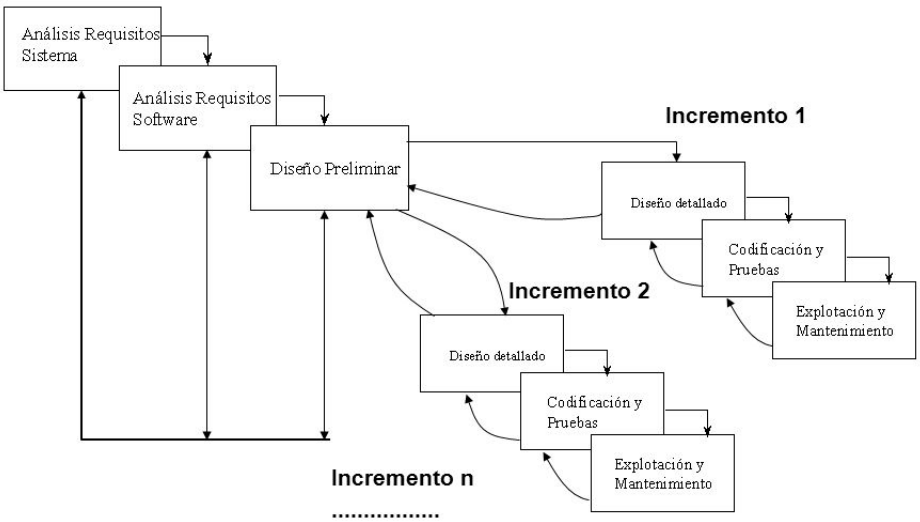


Figura 2.1: Modelo de desarrollo iterativo

Para el correcto desarrollo y fácil crecimiento se realizará la aplicación utilizando el paradigma de programación orientada a objetos.

2.2. Herramientas utilizadas

Las principales herramientas utilizadas son:

2.2.1. Lenguajes de programación

Se decide realizar el Software empleando como lenguaje de programación Java 1.8 debido a su versatilidad y posiblidad de poder ejecutar las aplicaciones realizadas con Java en multitud de sistemas operativos distintos. Además se empleará Swing para desarrollo de las interfaces gráficas.

Por otro lado para realizar los cálculos se empleará Python debido a su potencia a la hora de realizar cálculos matemáticos y a que las fórmulas matemáticas que se van a utilizar ya se encuentran programadas en Python.

2.2.2. Tipos de archivo

La aplicación trabajará principalmente con archivos de imágenes los cuales vienen a representar los diferentes elementos. Asímismo se utilizarán archivos XML como archivos para guardar tanto los elementos sueltos como los diagramas. El programa generará también archivos de Python para su posterior estudio y ejecución.

2.2.3. Programas

Como IDE de desarrollo principal se empleará Netbeans debido a que es el que mejor integra Java y los proyectos se pueden importar/exportar sin problemas. El código Java se analizará utilizando la herramienta de análisis estático de código PMD. Para el diseño y verificación de los archivos XML se empleará XML Copy Editor. La documentación se elaborará utilizando el lenguaje de tipado LaTEX, la suite ofimática Libre Office, StartUML y OpenProj.

2.2.4. Sistemas Operativos

El SO principal en el que se va a desarrollar el Software será Ubuntu Linux, principalmente por ser un SO de código abierto y libre.

2.3. Definición de siglas y abreviaturas

- IDE (Integrated Development Environment) Entorno de desarrollo integrado.
- XML (eXtensible Markup Language) Lenguaje de Marcado Extensible
- DTD (document type definition) Definición de tipo de documento
- SO Sistema operativo.
- JDK Java Delopement Kit

Capítulo 3

Planificación

3.1. Estimación del esfuerzo

Para la planificación temporal y de costes se va a aplicar el modelo de estimación mediante puntos de función (PFNA) y posteriormente el modelo constructivo de costos (COCOMO).

3.1.1. Estimación por Puntos de función

La técnica de estimación por puntos de función es una técnica bastante fiable que nos permite medir el tamaño real del software, independientemente del entorno tecnológico y de la metodología. Para llevar a cabo dicha estimación hay que evaluar los siguientes parámetros:

- **Número de entradas** Son los datos que el usuario aporta al sistema (nombres de ficheros, menús de selección)
- **Número de salidas** Son los datos que el sistema aporta al usuario (informes, mensajes)
- **Número de ficheros lógicos internos** Ficheros o bases de datos internos al sistema (sólo los utiliza el sistema, ficheros maestros)
- **Número de ficheros externos** Ficheros o bases de datos externos al sistema (accesibles por otras aplicaciones)
- **Número de consultas externas** Entradas que requieren de una respuesta por parte del sistema.

Para realizar los cálculos primero se ha de contar el número de elementos de cada clase, luego se clasifican según el grado de complejidad y por último se obtienen los PFNA mediante la suma ponderada de esas cantidades con los pesos que aparecen en la siguiente tabla:

Parámetro significativo	Complejidad baja	Complejidad media	Complejidad alta
Entradas	x3	x4	x6
Salidas	x4	x5	x7
Ficheros internos	x7	x10	x15
Ficheros externos	x5	x7	x10
Consultas externas	x3	x4	x6

Después de haber obtenido los PFNA se han de ajustar mediante un factor de ajuste (FA). Para ello existen 14 factores que contribuyen a la complejidad de una aplicación, cada uno de ellos valorados dentro de una escala de 0 a 5. El FA se obtiene a partir de la suma de los 14 factores de complejidad (FC) mediante la siguiente ecuación:

$$FA = (0{,}01 * \sum FC) + 0{,}65 \tag{3.1}$$

Factores de Complejidad (FC)	0-5	Factores de Complejidad	0-5
Comunicación de datos		Funciones distribuidas	
Rendimiento		Gran carga de trabajo	
Frecuencia de transacciones		Entrada on-line de datos	
Requisitos de manejo del usuario final		Actualizaciones on-line	
Procesos complejos		Utilización con otros sistemas	
Facilidad de mantenimiento		Facilidad de operación	
Instalación en múltiples lugares		Facilidad de cambio	

Posteriormente obtenemos los puntos de función ajustados (PF) utilizando la siguiente fórmula:

$$PF = PFNA * FA \tag{3.2}$$

Y por último se obtienen las lineas de código multiplicando PF por el número de lineas de código por punto de función para el lenguaje que vamos a utilizar en el proyecto. Como el lenguaje va a ser JAVA,tenemos que 1 punto de función equivale a 53 lineas de código.

3.1.1.1. Estimación aplicación

- Entradas
  - Datos de entrada creación elemento: complejidad media
  - Panel de edición de sistemas: complejidad alta
  - Datos medición sensores: complejidad media
  - Datos edición de valores de variables y parámetros: complejidad baja
  - Datos de listado de elementos: complejidad baja
  - Datos definición tipo de sistema: coplejidad baja
  - Datos de guardado de archivos: complejidad baja
  - Menús: complejidad baja
- Salidas

  - Lista de elementos: complejidad baja
  - Ventanas de error: complejidad baja
  - Representación gráfica de elementos: complejidad media
- Ficheros lógicos internos
  - Archivos XML: complejidad media
  - Imágenes: complejidad baja
- Ficheros lógicos externos
  - Archivos XML: complejidad media
  - Imágenes: complejidad baja
  - Modelos: complejidad media
- Consultas Externas
  - Elementos: complejidad baja
  - Tipos: complejidad baja
  - Sistemas: complejidad baja

	Complejidad			
Parámetro significativo	Baja	Media	Alta	Total
Entradas	5x3	2x4	1x6	29
Salidas	2x4	1x5	0x7	13
Ficheros internos	1x7	1x10	0x15	17
Ficheros externos	1x5	2x7	0x10	19
Consultas externas	3x3	0x4	0x6	9
			PFNA	87

Obtenemos un total de puntos de función no ajustados de 87. Ahora calculamos el factor de ajuste FA de acuerdo a los factores de complejidad de la siguiente tabla:

Factores de Complejidad (FC)	0-5	Factores de Complejidad	0-5
Comunicación de datos	0	Funciones distribuidas	0
Rendimiento	3	Gran carga de trabajo	1
Frecuencia de transacciones	1	Entrada on-line de datos	0
Requisitos de manejo del usuario final	5	Actualizaciones on-line	0
Procesos complejos	5	Utilización con otros sistemas	3
Facilidad de mantenimiento	3	Facilidad de operación	2
Instalación en múltiples lugares	5	Facilidad de cambio	3
		Total FC	31

$$FA = (0{,}01 * 31) + 0{,}65; PA = 0{,}97 \tag{3.3}$$

Ahora podemos calcular los puntos de función ajustados según el factor de ajuste:

$$PF = 87 * 0{,}97; PF = 84{,}39 \tag{3.4}$$

Por último podemos calcular el número total de lineas de código:

$$LDC = PF * LCJAVA; LDC = 84{,}39 * 53 = 4473 \tag{3.5}$$

3.1.2. Estimación por COCOMO del esfuerzo y coste

COMOMO es un mecanismo de estimación que nos permite medir el esfuerzo del desarrollo de un sistema. Como nos encontramos en el inicio del proyecto y no disponemos de detalles vamos a emplear el modelo de COCOMO básico ya que nos permite realizar los cálculos a partir del tamaño del programa, el valor LDC previamente calculado.

Los procedimientos para la realización del cálculo de COCOMO básico son:

- Para el esfuerzo
  - $E = a(KLDC)^b m(x)$
  - E es el esfuerzo en personas - mes implicadas en el diseño y desarrollo de una aplicación.
  - $KLDC$ son los miles de lineas de código que se prevé.
  - $m(x) = 1$ para cada uno de los 15 atributos conductores de coste.
- Para el tiempo
  - $TD = cE^d$
  - TD es el tiempo de desarrollo en meses
  - a, b, c y d depende del tipo de proyecto.

Proyecto de software	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semiorgánico	3.0	1.12	2.5	0.35
Integrado	3.6	1.20	2.5	0.32

Para ajustar los factores de coste se puede emplear la siguiente tabla:

Factores Conductores del Coste		Valor de los factores					
		Muy bajo	Bajo	Medio	Alto	Muy Alto	Extra
Software	Fiabilidad del software requerido	0.75	0.88	1.00	1.15	1.4	
	Tamaño de la base de datos		0.94	1.00	1.08	1.16	
	Complejidad del software	0.70	0.85	1.00	1.15	1.30	1.65
Hardware	Restricciones de rendimiento en tiempo de ejecución			1.00	1.11	1.30	1.66
	Restricciones de memoria			1.00	1.06	1.21	1.56
	Volatilidad del entorno de la máquina virtual		0.87	1.00	1.15	1.30	
	Tiempo de respuesta requerido		0.87	1.00	1.07	1.15	
Personal	Capacidad de los analistas	1.46	2.29	1.00	0.86	0.71	
	Experiencia con el tipo de aplicación	1.29	1.13	1.00	0.91	0.82	
	Experiencia con el hardware	1.21	1.10	1.00	0.90		
	Experiencia con el lenguaje de programación	1.14	1.07	1.00	0.95		
	Capacidad de los programadores	1.42	1.17	1.00	0.86	0.70	
Proyecto	Técnicas modernas de programación	1.24	1.10	1.00	0.91	0.82	
	Utilización de herramientas software	1.24	1.10	1.00	0.91	0.83	
	Restricciones en la planificación temporal del desarrollo	1.23	1.08	1.00	1.04	1.10	

A parte de utilizar el modelo básico de COCOMO se utilizará el tipo orgánico ya que el proyecto a realizar es de tamaño pequeño.

3.1.2.1. Estimación por COCOMO de la aplicación

Durante el proceso de estimación por puntos de función obtuvimos un resultado de 445 líneas de códigoo lo que es lo mismo 0,445 KLDC. Con este valor se puede realizar la Estimación por COCOMO:

- Esfuerzo nominal = $2{,}4 * (4{,}473)^{1{,}05} = 11{,}57$ personas/mes
- Esfuerzo = $11{,}57 * 1{,}15$ (fiabilidad del software) $*0{,}95$ (Experiencia con el lenguaje de programación) $*0{,}91$ (utilización de herramientas software) $*0{,}83$ (Experiencias con el tipo de aplicación) = 12,51 personas/mes
- tiempo de desarrollo = $2{,}5 * 12{,}51^{0{,}38} = 6{,}53$ meses
- Número medio de personas = $12{,}51/6{,}53 = 1{,}9 \approx 2$ personas

3.2. Planificación temporal

En la planificación de tareas se tendrán el cuenta los objetivos y requisitos de la aplicación, así como la estimación del esfuerzo, pero también puede tenerse en cuenta la distinción en tareas de análisis, diseño, implementación y pruebas, en cada una de las posibles iteraciones del ciclo de vida del proyecto, y el marco temporal (plazo de entrega del proyecto).

El correspondiente diagrama de Gantt permite observar de forma gráfica la distribución temporal de las tareas:

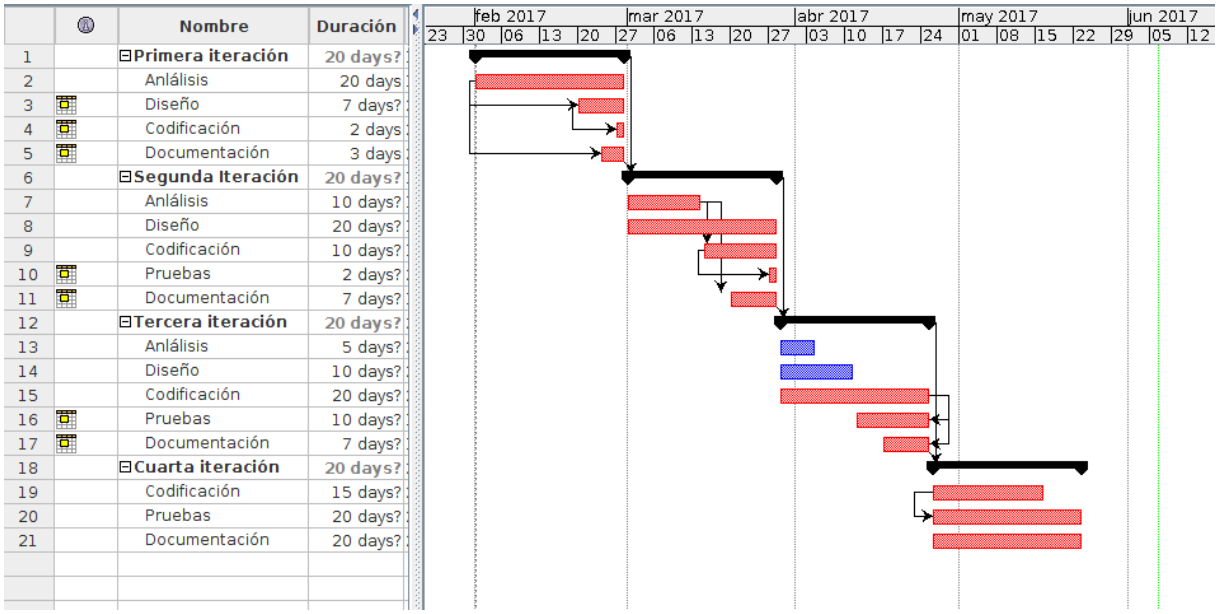


Figura 3.1: Diagrama de Gantt

3.3. Presupuesto económico

Para la estimación del presupuesto se tendrán en cuenta las herramientas utilizadas (hardware y software, con los factores de impacto que correspondan a la duración del proyecto), junto con los recursos humanos necesarios según la planificación de tareas, y el tipo de rol (analista, programador, etc) correspondiente a cada tarea. Para ello conviene tener una tabla actualizada del coste por hora de cada tarea del proyecto, además del estudio previo de la planificación de tareas.

3.3.1. Hardware

Para poder llevar a cabo este proyecto se va a emplear el siguiente Hardware:

- Un ordenador personal para llevar a cabo el análisis, diseño, codificación, pruebas y documentación del proyecto.
- Algunos dispositivos periféricos para agilizar el desarrollo tales como un ratón óptico y un monitor de apoyo.
- Conexión a internet para la búsqueda de información e investigación.
- Una impresora para imprimir la documentación relativa a la documentación del proyecto.
- CD'S para grabar la documentación, ejecutables y código fuente.

Componente hardware	Uso (%)	Coste (€)	Coste total (€)
Ordenador	50	1200	600
Periféricos	40	130	52
Conexión a Internet	400	20	80
Impresión	1000	5	50
CDS	600	0.20	1.20
Total			783.2

Figura 3.2: Tabla presupuesto Hardware

3.3.2. Software

Para poder llevar a cabo este proyecto se va a emplear el siguiente software:

- Ubuntu Linux (Sistema operativo principal)

- NetBeans IDE (Entorno de desarrollo)
- JavaSDK (Lenguaje de programación)
- REM (Análisis de Requisitos)
- OpenProj (Planificación)
- TexMaker (Editor latex)
- LaTeX core (Lenguaje de tipado para la documentación)
- LibreOffice (algunos diagramas y presentaciones)
- StarUML (Anlálisis y diseño de diagramas)
- Pencil (Diseño de interfaces)

Componente software	Uso (%)	Coste (€)	Coste total (€)
Ubuntu Linux	80	0	0
NetBeans IDE	60	0	0
JavaSDK	60	0	0
REM	10	0	0
OpenProj	10	0	0
TexMaker	20	0	0
LaTeX core	20	0	0
LibreOffice	10	0	0
StarUML	10	0	0
Pencil	10	0	0
Total			0

Figura 3.3: Tabla presupuesto Software

3.3.3. Recursos humanos

Este proyecto va a tener una duración de 4 meses y será realizado por una única persona. Teniendo en cuenta que realizará el trabajo en jornadas de 8 horas durante 21 días que compone un mes, el cálculo de la mano de obra sería el siguiente:

$horasTotalesTrabajo = 4meses * 8horas * 21dias = 672$

Para el cálculo del presupuesto de recursos humanos se utiliza un salario base de 10€ brutos a la hora.

Trabajadores	Horas	Salario (€)	Coste total (€)
Ángel Olivera	672	10	6720
Total			6720

Figura 3.4: Tabla presupuesto RRHH

3.3.4. Presupuesto total

Para obtener el presupuesto total necesitaremos sumar el total de todos los presupuestos anteriormente analizados:

	Hardware (€)	Software (€)	RRHH (€)	Total (€)
Estimación	783.2	0	6720	7503,2
Final	783.2	0	6720	7503,2

Figura 3.5: Tabla presupuesto Total

Capítulo 4

Análisis

4.1. Requisitos

4.1.1. Requisitos de usuario

La aplicación contará con un único usuario el cual podrá diseñar los diagramas, definir nuevos elementos y generar modelos.

4.1.2. Requisitos funcionales

FRQ-0001 Creación de nuevos elementos: El sistema deberá permitir al usuario crear sus propios elementos para su estudio

FRQ-0002 Representación de elementos: El sistema deberá permitir representar elementos gráficos a partir de imágenes.

FRQ-0003 Elementos por defecto: El sistema deberá disponer de una colección de elementos predefinidos, especialmente para sistemas electrónicos

FRQ-0004 Verificación de ecuaciones: El sistema deberá poder verificar si una ecuación está bien formada teniendo en cuenta los valores de las posibles variables del sistema

FRQ-0005 Creación de diferentes sistemas: El sistema deberá permitir al usuario definir varios tipos de sistemas con 2 variables de sistema, revisable a ser ampliado en un futuro el número de variables

FRQ-0006 Creación de archivos xml: El sistema deberá almacenar información relativa a los elementos, sistemas y diseños en formato XML.

FRQ-0007 Lectura de archivos XML: El sistema deberá leer la información referente a los sistemas y componentes guardada en archivos XML.

FRQ-0008 Vincular unos elementos con otros: El sistema deberá permitir unir unos elementos con otros en un sistema.

FRQ-0009 Representación gráfica de los vínculos de los elementos: El sistema deberá representar de forma gráfica los vínculos entre elementos.

FRQ-0010 Establecer variables de los elementos: El sistema deberá permitir el establecer los valores de las variables de cada elemento

FRQ-0011 Establecer parámetros de los elementos: El sistema deberá permitir al usuario modificar los parámetros de los elementos

FRQ-0012 Listado de elementos: El sistema deberá listar los elementos guardados en un archivo para facilitar su búsqueda a la hora de crear sistemas.

FRQ-0013 Filtrado de elementos en la lista: El sistema deberá permitir al usuario buscar los elementos de una forma rápida a partir de el tipo de elemento y el nombre del elemento.

FRQ-0014 Eventos de edición: El sistema deberá facilitar al usuario diferentes opciones de edición tales como copiar, cortar y pegar

4.1.3. Requisitos no funcionales

NFR-0001 Software Universal: El sistema deberá poder ser ejecutado en todo tipo de sistemas operativos, independientemente de que sean Windows, Linux o MacOs.

NRF-0002 Código Abierto: El sistema deberá estar desarrollado bajo una licencia de código abierto.

NFR-0003 Interfaz intuitiva: El sistema deberá disponer de una interfaz intuitiva para el usuario

NFR-0004 Idioma inglés: El sistema deberá de estar traducido al inglés ya que está enfocado a usuarios más técnicos.

NFR-0005 Manejo de excepciones: El sistema deberá manejar todos las excepciones que puedan ocurrir durante la ejecución.

4.1.4. Requisitos de la información

IRQ-0001 Lista de elementos: El sistema deberá almacenar la información correspondiente a los diferentes elementos a representar en los diagramas almacenando las diferentes fórmulas según los estados, la descripción del elemento, el tipo del elemento y una imagen para poderlo representar.

IRQ-0002 Archivos de guardado: El sistema deberá almacenar la información correspondiente a los sistemas generados por el usuario, incluyendo posición y unión de los elementos, valores definidos por el usuario y cálculos realizados.

4.1.5. Descripción general de los casos de uso

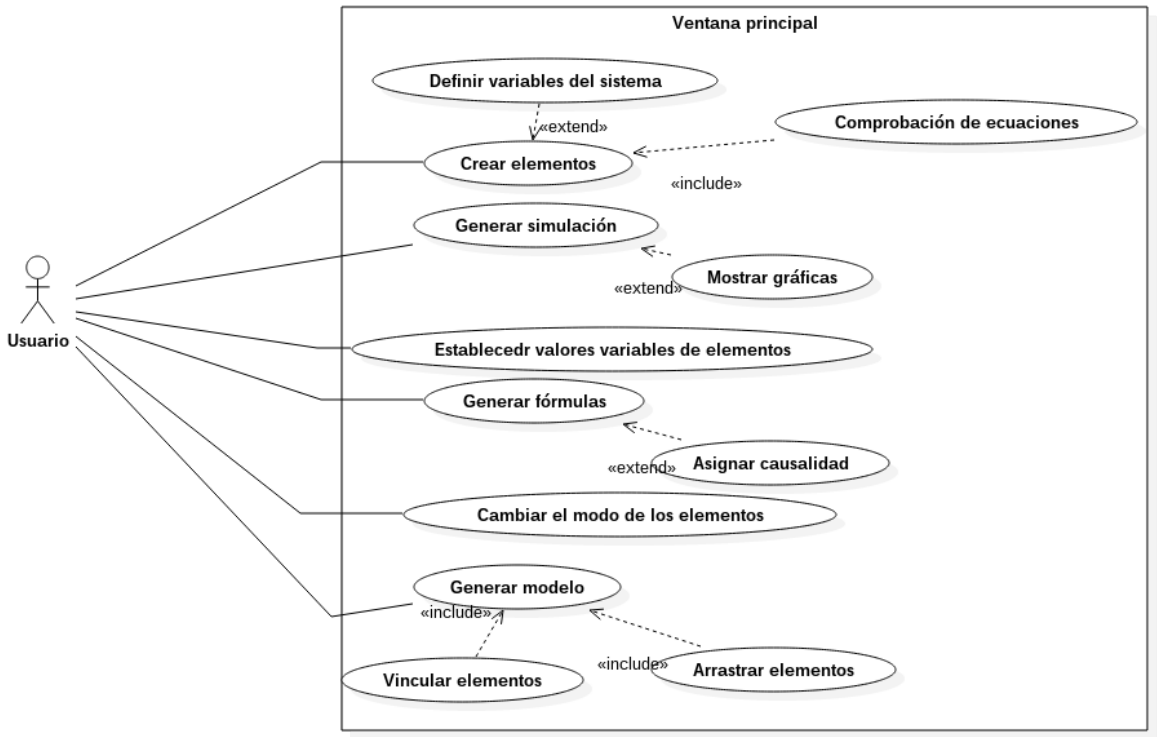


Figura 4.1: Diagrama de casos de uso

4.1.6. Casos de uso

A continuación se describen los casos de uso más importantes.

Nombre e ID del CU	CU-01 Crear nuevo elemento
Actor	Usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario pulsa el botón de crear un nuevo elemento.
Precondiciones	
Postcondiciones	POST-1. El elemento es guardado en la lista de elementos

Flujo normal	FN1 El actor rellena el formulario facilitado con la información que desea definir. FN2 Si el usuario selecciona crear un nuevo tipo para el elemento, se realiza el caso de uso Definir un nuevo tipo de elemento CU-0004 FN3 El actor intruduce las diferentes fórmulas teniendo en cuenta las variables que se han definido para el sistema. FN4 El actor indica que quiere asociar una imagen para el elemento FN5 El sistema muestra una ventana para seleccionar archivos de tipo imagen. FN6 El actor presiona el botón para guardar el elemento. FN7 El sistema verifica que los datos introducidos son correctos. FN8 Si los datos son correctos, el sistema guarda la información del elemento en el disco duro en formato XML. FN9 El sistema oculta la ventana de crear elemento y añade el elemento creado a la lista de elementos.
Flujo alternativo 1	FA8 Si los datos son incorrectos se informa al usuario del error indicando cuales son los datos incorrectos.
Excepciones	E1 El usuario ha dejado campos requeridos sin rellenar. E2 Ya existe un elemento con el mismo nombre y el mismo tipo. E3 Las ecuaciones introducidas no son correctas.
Prioridad	Alta
Otra info	Si no se selecciona una imagen el sistema asignará una por defecto.

Tabla 4.1: CU-01. Crear nuevo elemento

Nombre e ID del CU	CU-02 Crear Modelo
Actor	Usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera generar un modelo.

Precondiciones	PRE-1. Deben existir elementos en la biblioteca de elementos.
Postcondiciones	POST-1. Aparece al representación gráfica del sistema, se habilitan los botones de guardar proyecto
Flujo normal	FN1 El actor filtra los elementos de acuerdo a sus necesidades. FN2 El sistema muestra la lista de elmentos según las los parámetros definidos por el usuario. FN3 El actor pincha sobre el elemento que quiere añadir al sistema y lo arrastra a la zona visible. Y se inicia el caso de uso arrastrar elementos. FN4 El sistema muestra el elemento en la pantalla de diseño mostrando en la leyenda el identificador del elemento. FN5 El sistema incrementa el valor del indentificador para el próximo elemento.
Flujo alternativo 1	FA3 Si el elemento seleccionado es de tipo port o tipo input, el sistema preguntará al usuario qué variables del sistema va a utilizar o el número de puertos de salida respectivamente.
Excepciones	E1 El usuario no ha indicado el tipo de sistema o número de puertos de salida. E2 El usuario ha indicado un número erróneo de puertos de salida.
Prioridad	Alta

Tabla 4.2: CU-02. Crear modelo

Nombre e ID del CU	CU-03 Vincular elementos
Actor	Usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera unir elementos en la ventana de edición.
Precondiciones	PRE-1. Deben existir elementos en la ventana de edición.
Postcondiciones	POST-1. Los elementos se unen y se muestra una representación de ese enlace

Flujo normal	FN1 El actor clickea sobre un puerto de salida de un elemento. FN2 El sistema dibuja un alinea desde el puerto de salida hasta la posición del cursor, la cual va cambiando según cambia la posición de éste. FN3 El actor clickea sobre un puerto de entrada de un elemento. FN4 El sistema une los elementos y muestra esa unión de forma gráfica.
Flujo alternativo 1	FA3 Si el puerto de salida escogido previamente pertenece a un elemento que ya ha sido vinculado a otro elemento, no siendo este un elemento de tipo Sensor, entonces el sistema no permite seleccionar ese puerto de entrada. FA3 Si el puerto de entrada ya está vinculado a otro puerto de salida, el sistema no permite escoger ese puerto. FA3 Si el elemento del puerto de origen paretenece a un tipo diferente al del elemento de destino, el sistema no permite realizar el enlace. FA3 Si el elemento de destino es un sensor, el sistema muestra un diálogo permitiendo al usuario indicar sobre qué variable se va a realizar la muestra.
Excepciones	E1 El usuario intenta unir un puerto de salida con otro puerto de salida. E2 El usuario intenta unir un puerto de salida ya vinculado con otro elemento no de tipo Sensor. E3 El usuario intenta unir 2 elementos de diferente tipo. E4 El usuario intenta unir un puerto de salida con otro de entrada que ya tiene asignado un elemento.
Prioridad	Alta

Tabla 4.3: CU-03. Vincular elementos

4.2. Diagramas

4.2.1. Modelo conceptual de datos

Aunque la aplicación no va a utilizar bases de datos relacionales sí que va a almacenar datos en forma de archivos XML, por lo que se cree adecuado crear un modelo conceptual de estos datos.

En el diagrama representado en la Figura 4.2 se pretende representar como ha de ser almacenada la información relativa a un elemento.

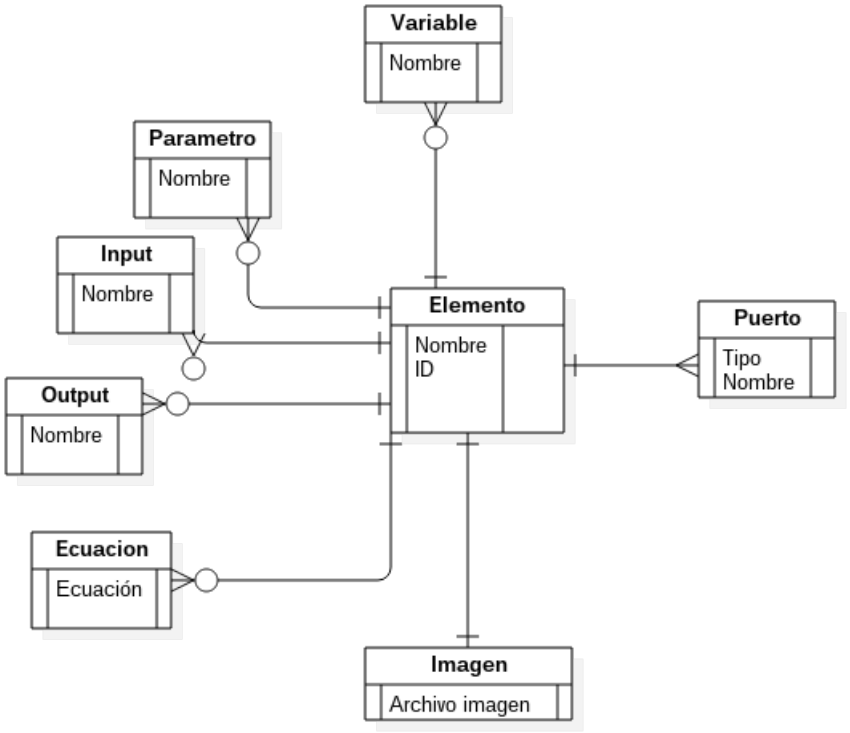


Figura 4.2: Diagrama entidad relación Elemento

Por otro lado el diagrama de la figura 4.3 representa la forma de almacenar la información para los archivos de respaldo.

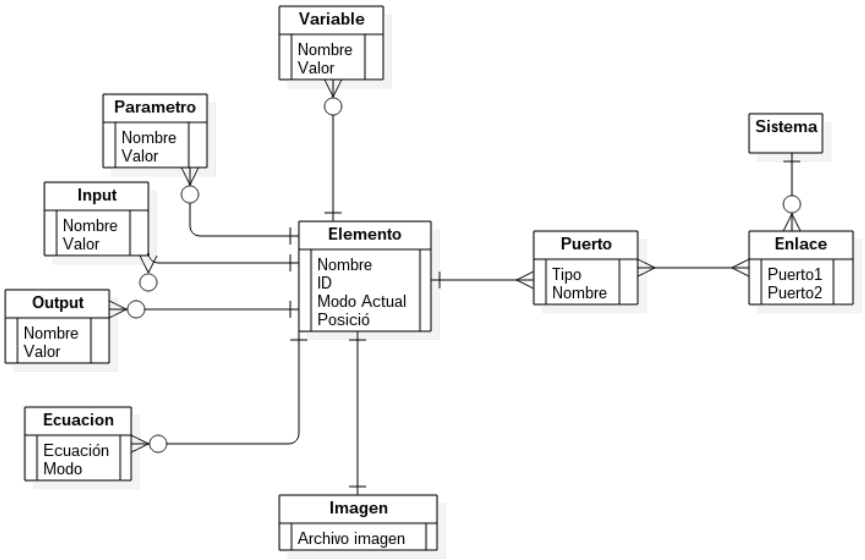


Figura 4.3: Diagrama entidad relación del Sistema

4.2.2. Diagramas de clases

A continuación se presentan los diagramas de clases de análisis de la aplicación, es un primer acercamiento al modelo de dominio y el cual será detallado en profundidad en el diagrama de clases de diseño.

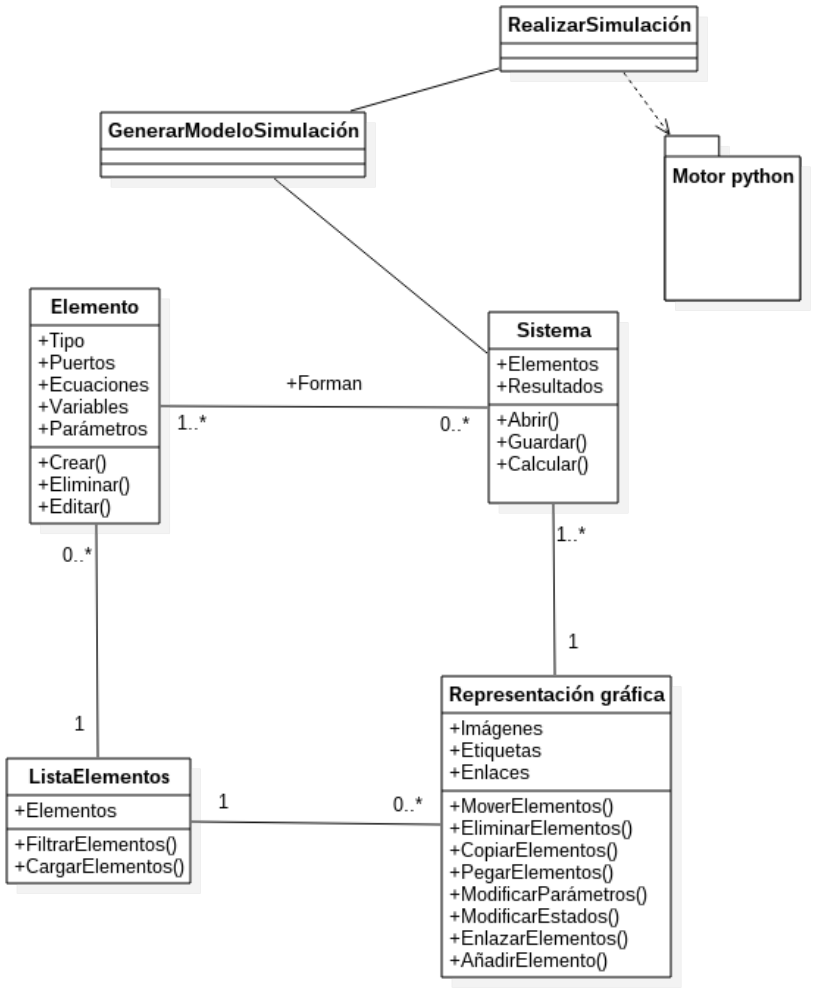


Figura 4.4: Diagrama clases de análisis

4.2.3. Diagramas de estados

A través de los diagramas de estados se pretende describir los estados por los que puede pasar un componente determinado del sistema, los cambios que ocurren al pasar de un estado a otro y los eventos que se activan o inicial cada estado.

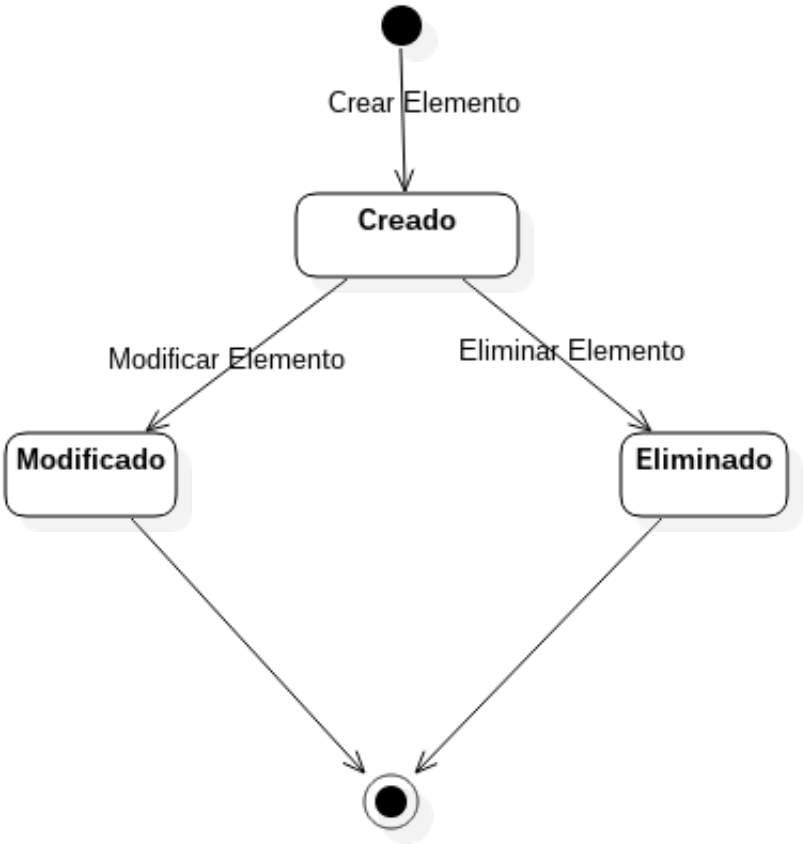


Figura 4.5: Diagrama de estados de Elemento

Este diagrama detalla los diferentes estados por los que puede pasar un elemento. Después de haber sido creados, los elementos pueden ser modificados o eliminados.

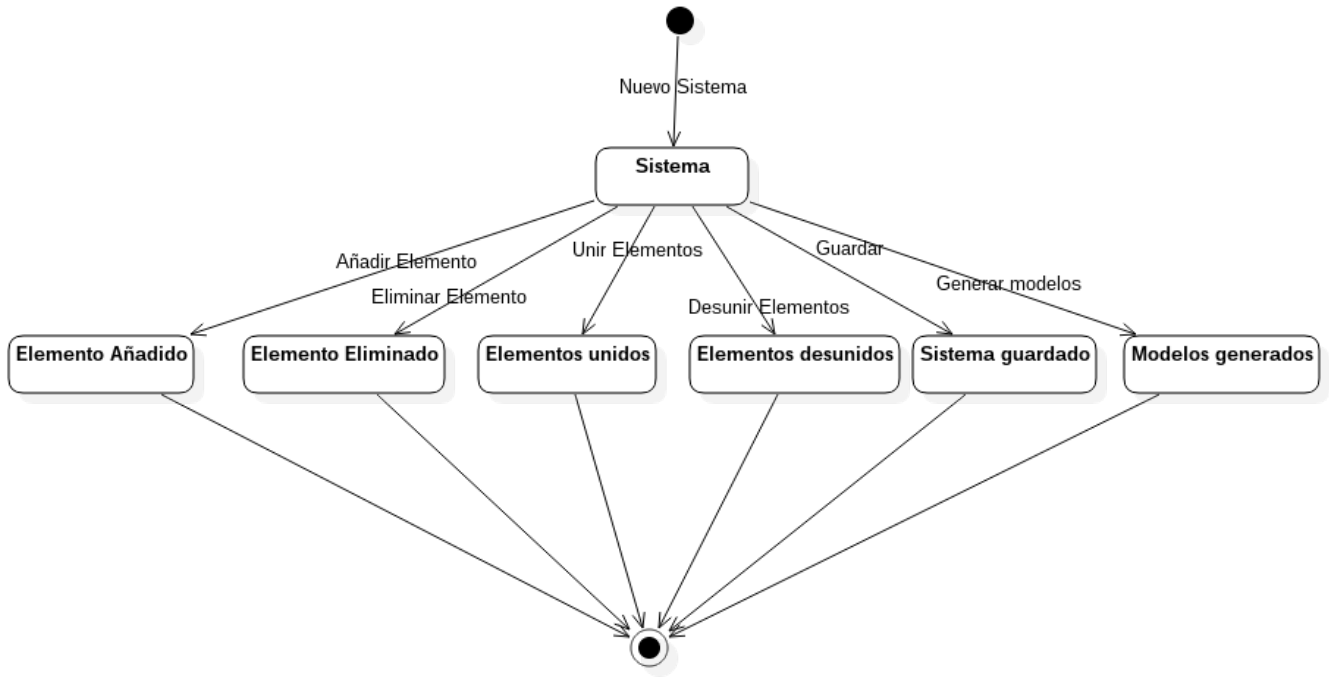


Figura 4.6: Diagrama de estados de Sistema

Este diagrama detalla los diferentes estados por los que puede pasar un sistema. Después de haber creado un sistema nuevo, se pueden añadir elementos, eliminar elementos, unir elementos, desunir elementos, guardar el sistema y generar modelos.

4.3. Atributos de calidad

En la siguiente sección se van a analizar los diferentes atributos de calidad que son deseables en para nuestra aplicación:

Rendimiento: PER-1: El tiempo que tarda la aplicación en generar modelos ha de ser lo más rápido posible. **PER-2:** El tiempo que ha de invertir la aplicación en listar los elementos de la biblioteca ha de ser inferior a 3 segundos.

Seguridad: SEQ-1: La aplicación no permitirá de ninguna manera pegar información desde el portapapeles de forma que pueda interferir con el correcto funcionamiento de esta.

Robustez: ROB-1: Las ecuaciones se han de verificar antes de ser guardadas. **ROB-2**: Si no existen datos de configuración la aplicación los creará automáticamente.

Capítulo 5

Diseño

5.1. Diseño de datos

La aplicación no utilizará una base de datos como tal para su funcionamiento, pero sí empleará diversos mecanismos para almacenar información con persistencia.

Los principales archivos que se necesitarán crear son los que almacenan la información relativa a los elementos, las variables de los sistemas y los archivos de guardado. Para ello se emplearán archivos XML que se parsearán para obtener los datos necesarios.

5.1.1. Variables del sistema

Este archivo se utiliza para guardar la información acerca de los tipos de sistema y sus variables. La mayoría de sistemas van a emplear únicamente 2 variables, por ejemplo los sistemas eléctricos utilizan Voltaje e Intensidad, aunque la extructura del archivo permitiría en un futuro indicar más variables para un único sistema. Como el usuario también puede crear sus propios sistemas es necesario que este archivo se pueda modificar.

```
<!ELEMENT types (type+)>
<!ELEMENT type (name, variables)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT variables (variable+)>
<!ELEMENT variable (Small, Full)>
<!ELEMENT Small (#PCDATA)>
<!ELEMENT Full (#PCDATA)>
```

Codigo 5.1: DTD variables del sistema

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE types SYSTEM "./types.dtd">
<types>
  <type>
    <name>Electrico</name>
    <variables>
      <variable>
        <Small>V</Small>
        <Full>Voltaje</Full>
      </variable>
      <variable>
        <Small>I</Small>
        <Full>Intensidad</Full>
      </variable>
    </variables>
  </type>
</types>
```

Codigo 5.2: XML ejemplo variables del sistema

Detalles de la extructura:

- **types** elemento el cual hace referencia a una lista de elementos de tipo type.
- **type** almacena la información con relación a cada tipo de sistema. Este tiene como elementos hijos: name y variables.
- **name** indica el nombre del tipo de sistema.
- **variables** representa una lista de elementos variable, donde indican las variables del tipo de sistema.
- **variable** representa a cada variable y tiene dos hijos: Small y Full.
- **Small** identifica el nombre corto para la variable ej: *V*.
- **Full** indica el nombre completo de la variable ej: *Voltaje*.

5.1.2. Elementos

Por cada elemento se va a crear un archivo específico donde se almacenarán los detalles de estos elementos. Estos archivos deben de poder ser modificados por el usuario, bien sea a través de la aplicación o manualmente. La elección de utilizar archivos XML para almacenar elementos se debe a su versatilidad, a no tener que depender de bases de datos y por la facilidad de poder importar/exportar elementos a diferentes sistemas.

```
<!ELEMENT system (elements, links)>
<!ELEMENT elements (element*)>
<!ELEMENT element (#PCDATA|(name, type, image, ports,
   parameters, variables, inputs, outputs, equations, mode,
   location, varValue, parValue))>
<!ATTLIST element id CDATA>
<!ELEMENT name (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT image (#PCDATA)>
<!ELEMENT ports (in,out)*>
<!ELEMENT parameters (parameter*)>
<!ELEMENT parameter (#PCDATA)>
<!ELEMENT variables (variable*)>
<!ELEMENT variable (#PCDATA)>
<!ELEMENT inputs (input*)>
<!ELEMENT input (#PCDATA)>
<!ELEMENT outputs (outpu*)>
<!ELEMENT output (#PCDATA)>
<!ELEMENT equations (equation*)>
<!ELEMENT equation (content, state)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT mode (#PCDATA)>
<!ELEMENT location (x,y)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT varValues (#PCDATA)>
<!ELEMENT parValues (#PCDATA)>
<!ELEMENT links (link*)>
<!ELEMENT link (org, dest)>
<!ELEMENT org (element, port)>
<!ELEMENT dest (element, port)>
<!ELEMENT port (#PCDATA)>
```

Codigo 5.3: DTD elemento

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE element SYSTEM "./elements.dtd">
<element id="Cb">
  <name>Circuit breaker</name>
  <type>Electrico</type>
  <image>elements/Electrico/images/Circuit breaker.png</image>
  <ports>
    <in>p1</in>
    <out>p2</out>
  </ports>
  <parameters/>
  <variables/>
  <outputs/>
  <inputs/>
  <equations>
    <equation>
      <content>0 = {id}.p1.I</content>
      <state>Abierto</state>
    </equation>
    <equation>
      <content>0 = {id}.p2.V</content>
      <state>Abierto</state>
    </equation>
    <equation>
      <content>{id}.p1.I = {id}.p2.I</content>
      <state>Cerrado</state>
    </equation>
    <equation>
      <content>{id}.p1.V = {id}.p2.V</content>
      <state>Cerrado</state>
    </equation>
  </equations>
</element>
```

Codigo 5.4: XML ejemplo de elemento

Detalles de la estructura:

- **element** almacena la información del elemento, cada elemento tiene un identificador único el cual se define a través del atributo id.
- **name** indica el nombre del elemento.
- **type** indica el tipo del elemento.
- **image** indica la ruta de la imagen asociada a dicho elemento.

- **ports** este elemento guarda la información de los puertos del elemento, puede contener uno o varios elementos hijos los cuales pueden ser o in o out.
- **in** se utiliza para definir el nombre de un puerto de entrada.
- **out** se utiliza para definir el nombre de un puerto de salida.
- **variables** guarda las variable que puede tener el elemento, utilizando nodos variable.
- **variable** se utiliza para definir el nombre de una variable.
- **parameters** guarda los parámetros que puede tener el elemento, utilizando nodos parameter.
- **parameter** se utiliza para definir el nombre de un parámetro.
- **inputs** guarda las entradas que puede tener el elemento, utilizando nodos input.
- **input** se utiliza para definir el nombre de una entrada.
- **outputs** guarda las salidas que puede tener el elemento, utilizando nodos output.
- **output** se utiliza para definir el nombre de una salida.
- **equations** guarda las diferentes ecuaciones del elemento, utilizando nodos equation.
- **equation** se utiliza para definir una ecuación utilizando dos nodos para definirla: content y state.
- **content** se utiliza para definir la ecuación.
- **state** se utiliza para definir el estado del elemento para esa ecuación.

5.1.3. Archivos de almacenamiento

La aplicación debe permitir guardar y recuperar sistemas creados por el usuario. Para ello se crean archivos XML donde se guardan los elementos que forman el sistema, así como las uniones que se realizan entre ellos. Además se ha de guardar la posición de cada elemento y las variables que se hayan definido.

```
<!ELEMENT system (elements, links)>
<!ELEMENT elements (element*)>
<!ELEMENT element (#PCDATA|(name, type, image, ports,
   parameters, variables, inputs, outputs, equations, mode,
   location, varValue, parValue))>
<!ATTLIST element id CDATA>
<!ELEMENT name (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT image (#PCDATA)>
<!ELEMENT ports (in,out)*>
<!ELEMENT parameters (parameter*)>
<!ELEMENT parameter (#PCDATA)>
<!ELEMENT variables (variable*)>
<!ELEMENT variable (#PCDATA)>
<!ELEMENT inputs (input*)>
<!ELEMENT input (#PCDATA)>
```

```
<!ELEMENT outputs (outpu*)>
<!ELEMENT output (#PCDATA)>
<!ELEMENT equations (equation*)>
<!ELEMENT equation (content, state)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT mode (#PCDATA)>
<!ELEMENT location (x,y)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT varValues (#PCDATA)>
<!ELEMENT parValues (#PCDATA)>
<!ELEMENT links (link*)>
<!ELEMENT link (org, dest)>
<!ELEMENT org (element, port)>
<!ELEMENT dest (element, port)>
<!ELEMENT port (#PCDATA)>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<system>
  <elements>
    <element id="R1">
      <name>Relay</name>
      <type>Electrico</type>
      <image>elements/Electrico/images/Relay.png</image>
      <ports>
        <in>p1</in>
        <out>p2</out>
      </ports>
      <parameters/>
      <variables/>
      <inputs/>
      <outputs/>
      <equations>
        <equation>
          <content>{id}.p1.I = 0</content>
          <state>Off</state>
        </equation>
        <equation>
          <content>{id}.p2.V = 0</content>
          <state>Off</state>
```

```
        </equation>
        <equation>
          <content>{id}.p2.I = {id}.p1.I</content>
          <state>On</state>
        </equation>
        <equation>
          <content>{id}.p2.V = {id}.p1.V</content>
          <state>On</state>
        </equation>
      </equations>
      <mode>Off</mode>
      <location>
        <x>589</x>
        <y>81</y>
      </location>
      <varValues/>
      <parValues/>
    </element>
    <element id="I2">
      <name>Inverter</name>
      <type>Electrico</type>
      <image>elements/Electrico/images/Inverter.png</image>
      <ports>
        <in>p1</in>
        <out>p2</out>
      </ports>
      <parameters>
        <parameter>R</parameter>
        <parameter>eff</parameter>
      </parameters>
      <variables/>
      <inputs/>
      <outputs/>
      <equations>
        <equation>
          <content>{id}.p1.I = {id}.p1.V*{id}.p1.I/{id}.eff
              /({id}.p1.V+0.0001)+{id}.p1.V/{id}.R</content>
          <state>normal</state>
        </equation>
        <equation>
          <content>{id}.p2.V = {id}.p1.V*120/24</content>
          <state>normal</state>
```

```
        </equation>
      </equations>
      <mode/>
      <location>
        <x>51</x>
        <y>124</y>
      </location>
      <varValues/>
      <parValues>
        <R>0.0</R>
        <eff>0.0</eff>
      </parValues>
    </element>
    <element id="Cb3">
      <name>Circuit breaker</name>
      <type>Electrico</type>
      <image>elements/Electrico/images/Circuit breaker.png</
        image>
      <ports>
        <in>p1</in>
        <out>p2</out>
      </ports>
      <parameters/>
      <variables/>
      <inputs/>
      <outputs/>
      <equations>
        <equation>
          <content>0 = {id}.p1.I</content>
          <state>Abierto</state>
        </equation>
        <equation>
          <content>0 = {id}.p2.V</content>
          <state>Abierto</state>
        </equation>
        <equation>
          <content>{id}.p1.I = {id}.p2.I</content>
          <state>Cerrado</state>
        </equation>
        <equation>
          <content>{id}.p1.V = {id}.p2.V</content>
          <state>Cerrado</state>
```

```
        </equation>
      </equations>
      <mode>Cerrado</mode>
      <location>
        <x>393</x>
        <y>114</y>
      </location>
      <varValues/>
      <parValues/>
    </element>
    <element id="m1">
      <name>Sensor</name>
      <type/>
      <image>images/sensor.png</image>
      <ports>
        <in>p1</in>
      </ports>
      <parameters/>
      <variables/>
      <inputs>
        <input>R1.p1.I</input>
      </inputs>
      <outputs/>
      <equations/>
      <mode/>
      <location>
        <x>533</x>
        <y>166</y>
      </location>
      <varValues/>
      <parValues/>
    </element>
    <element id="v2">
      <name>Input</name>
      <type>Electrico</type>
      <image>images/input.png</image>
      <ports>
        <out>p1</out>
      </ports>
      <parameters>
        <parameter>V</parameter>
        <parameter>I</parameter>
```

```
        </parameters>
        <variables/>
        <inputs/>
        <outputs/>
        <equations/>
        <mode/>
        <location>
          <x>29</x>
          <y>20</y>
        </location>
        <varValues/>
        <parValues>
          <V>0.0</V>
          <I>0.0</I>
        </parValues>
      </element>
      <element id="p3">
        <name>Parallel</name>
        <type>Electrico</type>
        <image>images/rectangle.jpg</image>
        <ports>
          <in>P1</in>
          <out>P2</out>
          <out>P3</out>
        </ports>
        <parameters/>
        <variables/>
        <inputs/>
        <outputs/>
        <equations/>
        <mode/>
        <location>
          <x>222</x>
          <y>128</y>
        </location>
        <varValues/>
        <parValues/>
      </element>
      <element id="r4">
        <name>resistencia</name>
        <type>Electrico</type>
        <image>elements/Electrico/images/resistencia.png</image
```

```
    >
  <ports>
    <in>p1</in>
    <out>p2</out>
  </ports>
  <parameters>
    <parameter>r</parameter>
  </parameters>
  <variables/>
  <inputs/>
  <outputs/>
  <equations>
    <equation>
      <content>{id}.p2.V = {id}.p1.V - {id}.p1.I*{id}.r</
          content>
      <state>normal</state>
    </equation>
    <equation>
      <content>{id}.p1.V = {id}.p2.V - {id}.r * {id}.p2.I
         </content>
      <state>normal</state>
    </equation>
    <equation>
      <content>{id}.p1.I = {id}.p2.I</content>
      <state>normal</state>
    </equation>
    <equation>
      <content>{id}.p2.I = {id}.p1.I</content>
      <state>normal</state>
    </equation>
    <equation>
      <content>{id}.p1.I=0</content>
      <state>anormal</state>
    </equation>
  </equations>
  <mode>anormal</mode>
  <location>
    <x>330</x>
    <y>180</y>
  </location>
  <varValues/>
  <parValues>
```

```
        <r>0.0</r>
      </parValues>
    </element>
    <element id="m5">
      <name>Sensor</name>
      <type/>
      <image>images/sensor.png</image>
      <ports>
        <in>p1</in>
      </ports>
      <parameters/>
      <variables/>
      <inputs>
        <input>r4.p2.I</input>
      </inputs>
      <outputs/>
      <equations/>
      <mode/>
      <location>
        <x>488</x>
        <y>211</y>
      </location>
      <varValues/>
      <parValues/>
    </element>
  </elements>
  <links>
    <link>
      <org>
        <element>Cb3</element>
        <port>p2</port>
      </org>
      <dst>
        <element>R1</element>
        <port>p1</port>
      </dst>
    </link>
    <link>
      <org>
        <element>Cb3</element>
        <port>p2</port>
      </org>
```

```
    <dst>
      <element>m1</element>
      <port>p1</port>
    </dst>
  </link>
  <link>
    <org>
      <element>v2</element>
      <port>p1</port>
    </org>
    <dst>
      <element>I2</element>
      <port>p1</port>
    </dst>
  </link>
  <link>
    <org>
      <element>I2</element>
      <port>p2</port>
    </org>
    <dst>
      <element>p3</element>
      <port>P1</port>
    </dst>
  </link>
  <link>
    <org>
      <element>p3</element>
      <port>P2</port>
    </org>
    <dst>
      <element>Cb3</element>
      <port>p1</port>
    </dst>
  </link>
  <link>
    <org>
      <element>p3</element>
      <port>P3</port>
    </org>
    <dst>
      <element>r4</element>
```

```
        <port>p1</port>
      </dst>
    </link>
    <link>
      <org>
        <element>r4</element>
        <port>p2</port>
      </org>
      <dst>
        <element>m5</element>
        <port>p1</port>
      </dst>
    </link>
  </links>
</system>
```

Detalles de la estructura, como los nodos element comparten gran parte de la estructura con los archivos que definen los elementos, sólamente se indicarán las diferencias.

- **system** almacena la información de los elementos y los enlaces del sistema. Tiene como hijos nodos elemens y links.
- **elements** se utiliza para almacenar la lista de elementos del sistema. Tiene como hijos nodos de tipo element con estructura similar a los elementos sueltos.
- **mode** indica el modo actual en el que se encuentra el elemento.
- **location** indica la posisción del elemento, tiene como hijos nodos x e y.
- **x** indica la posición en el eje x.
- **y** indica la posición en el eje y.
- **varValues** se utiliza para definir los valores que pueden tener las variables.
- **parValues** se utiliza para definir los valores que pueden tener los parámetros.
- **links** utilizado para almacenar la información de la lista de enlaces, tiene nodos link.
- **link** se utiliza para almacenar cada uno de los enlaces. Tiene nodos org y dest.
- **org** indica el origen del enlace, tiene nodos element y port.
- **dest** indica el destino del enlace, tiene nodos element y port.
- **port** define el nombre del puerto de destino u origen.

5.2. Diagramas de clase y de secuencia

5.2.1. Diagramas de clases

A continuación se listan los diferentes diagramas de clases, creados a partir del análisis.

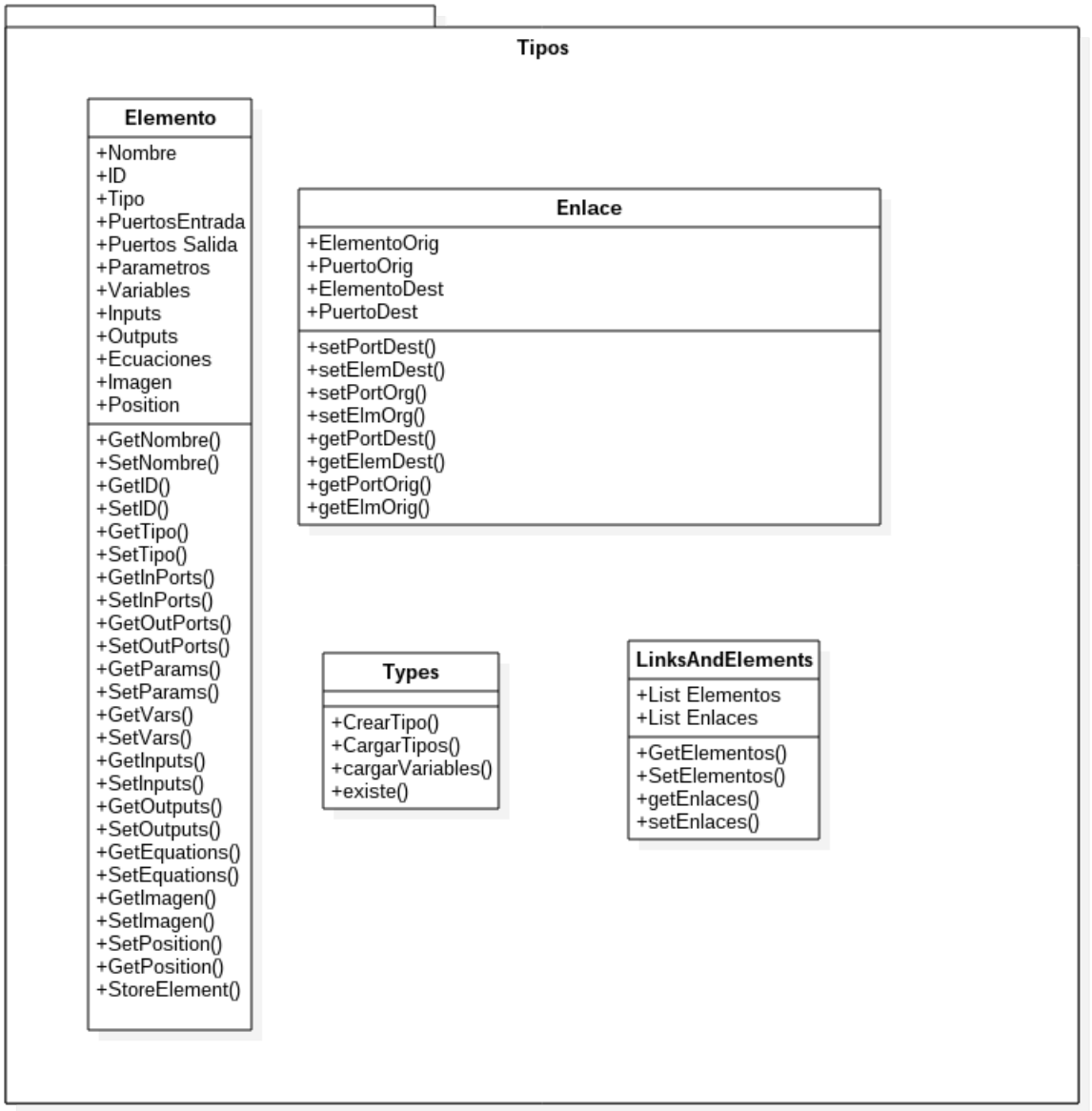


Figura 5.1: Diagramas de clases de los tipos

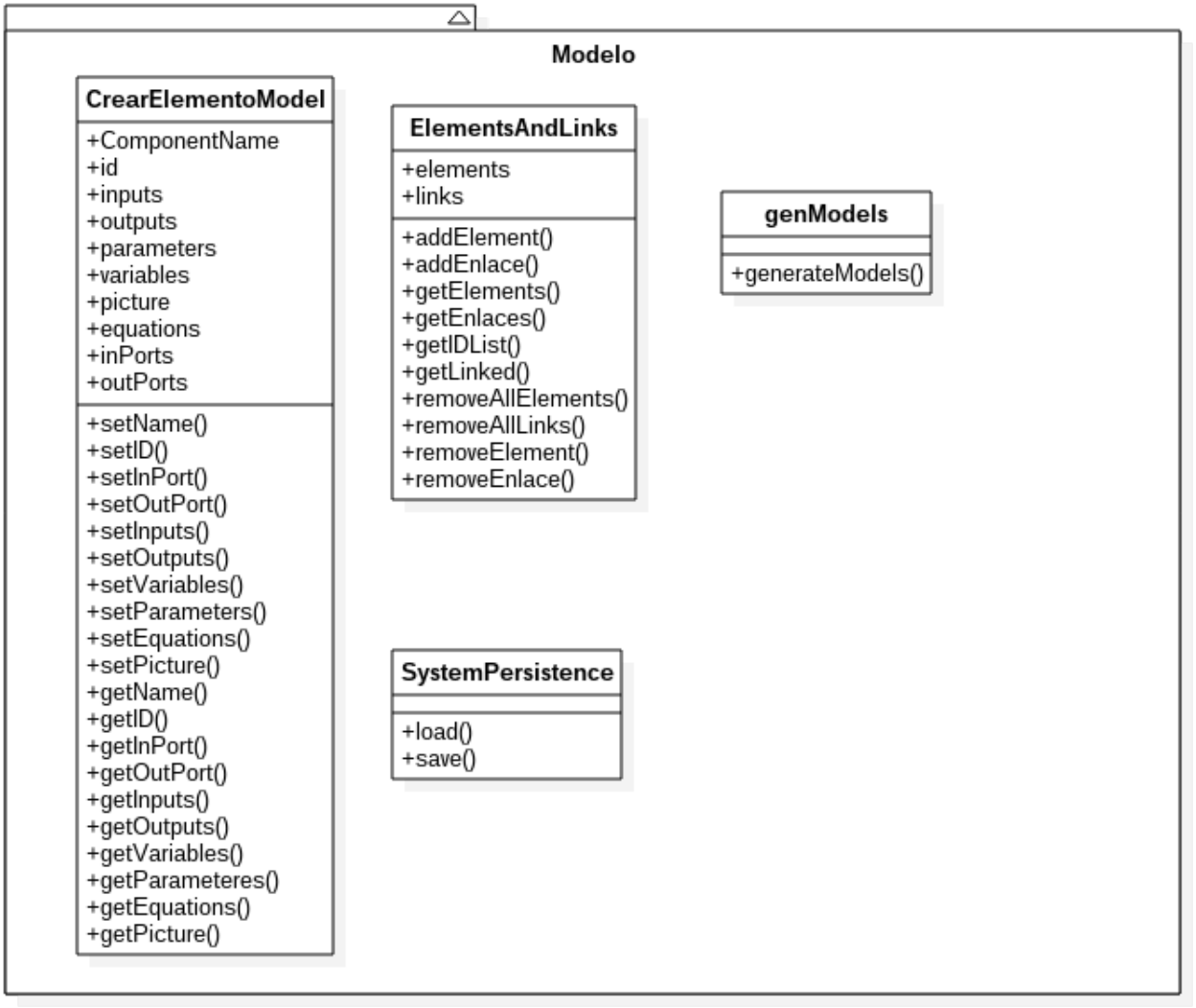


Figura 5.2: Diagramas de clases de los modelos

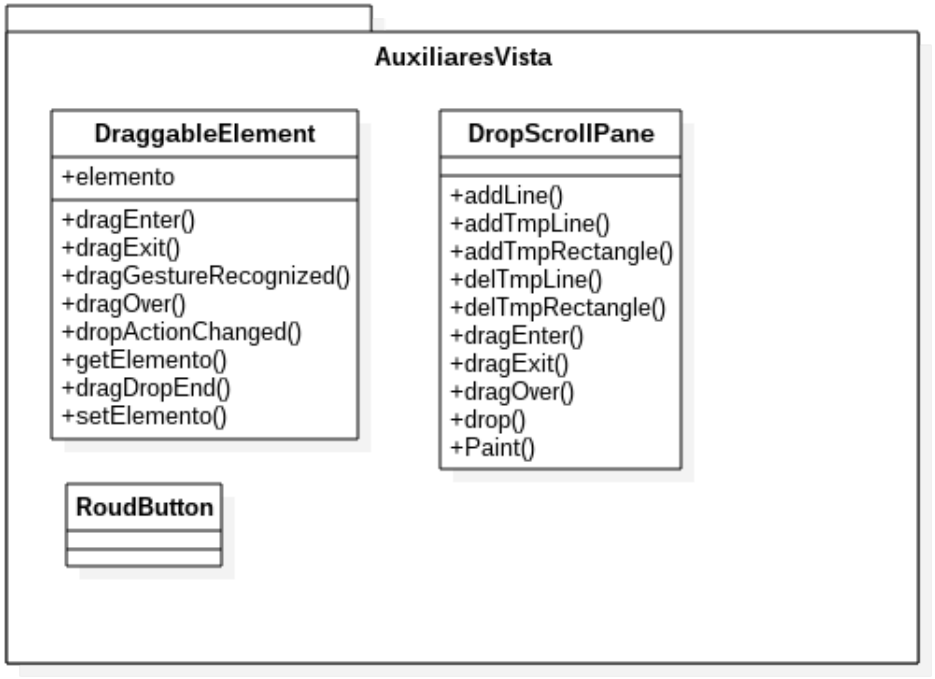


Figura 5.3: Diagramas de clases auxiliares de las vistas

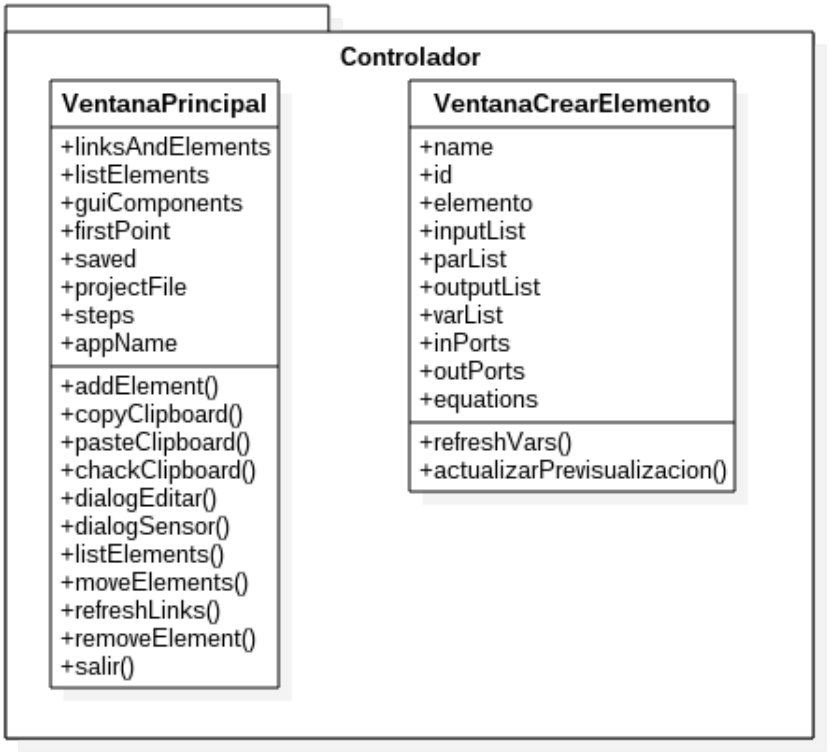


Figura 5.4: Diagramas de clases de los controladores

5.2.2. Diagramas de secuencia

A partir de los casos de uso analizados, lo siguiente es diseñar los diferentes diagramas de secuencia. Estos diagramas de secuencia muestran cómo van a ser las interacciones del usuario con el sistema a lo largo del tiempo. Cada diagrama de secuencia representa un caso de uso:

El siguiene diagrama de secuencia explica cuales son los pasos que se realizan cuando el sistema lista los elementos de la biblioteca.

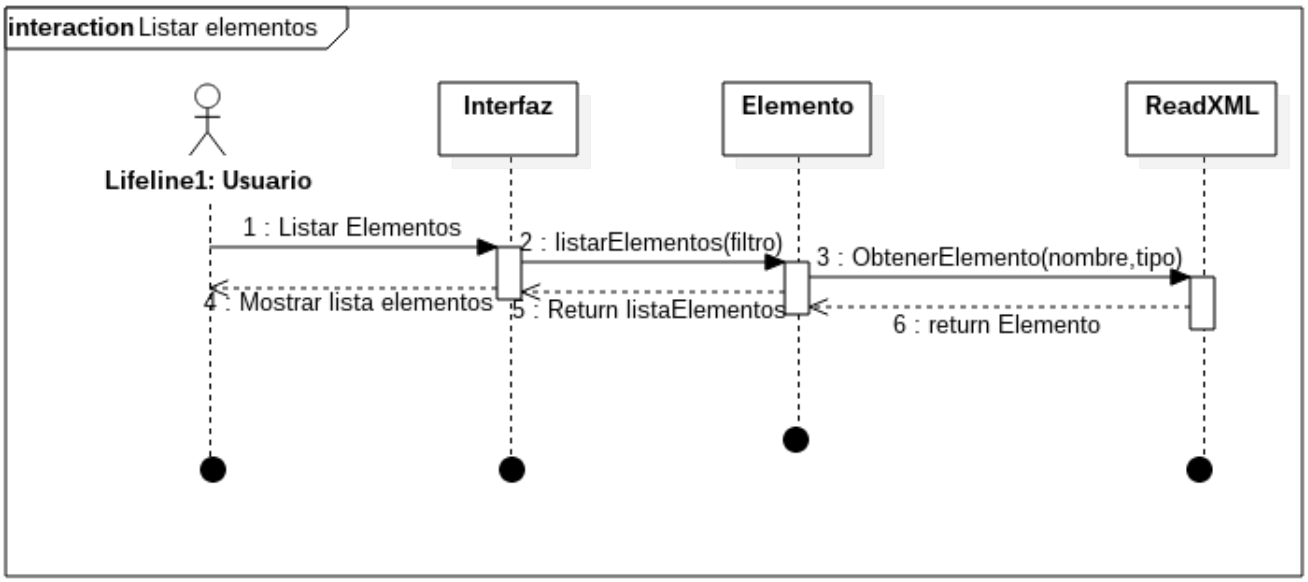


Figura 5.5: Diagrama de secuencia Listar Elementos

Este diagrama representa la secuencia que se ejecuta cuando un usuario crea un elemento.

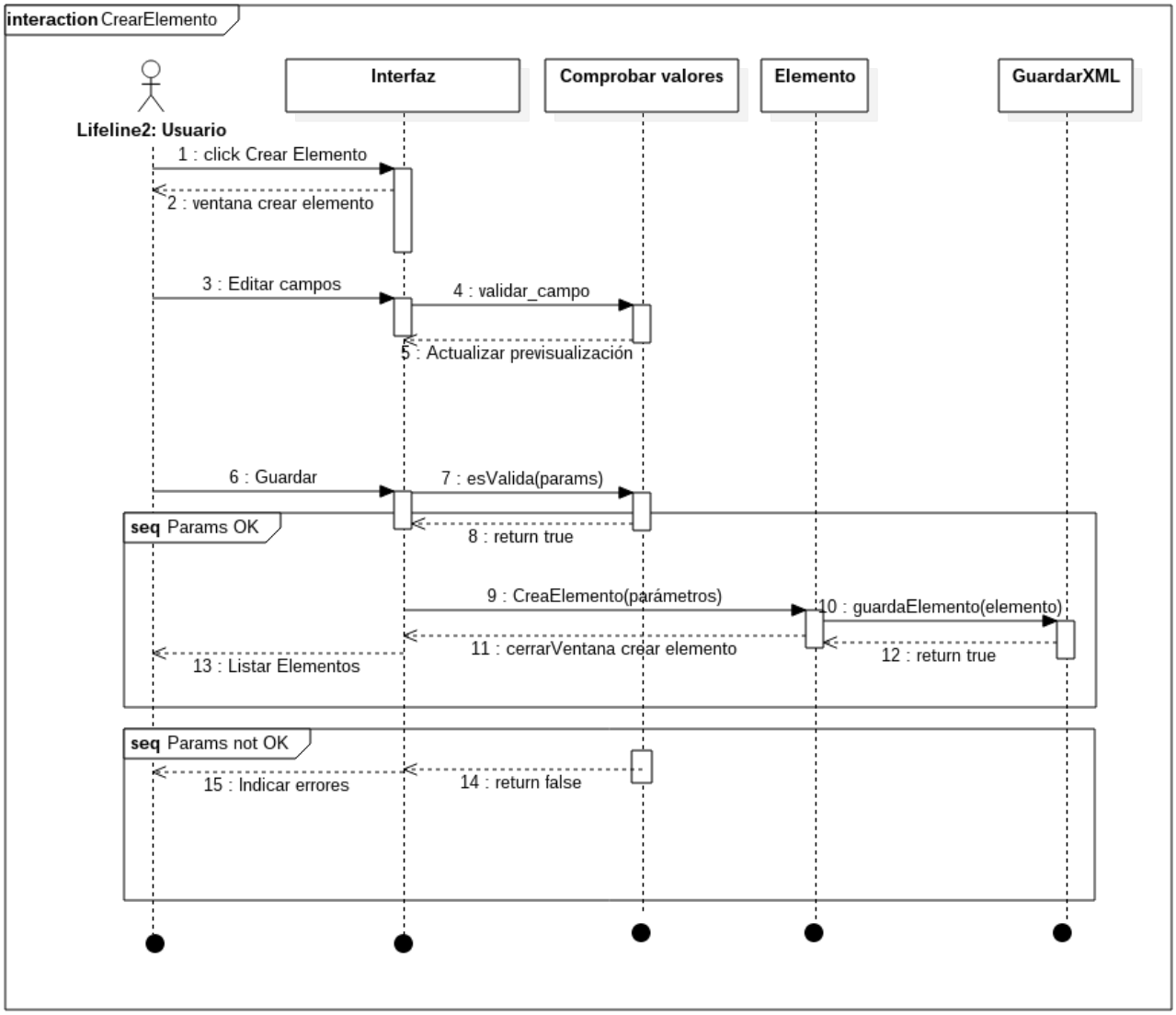


Figura 5.6: Diagrama de secuencia Editar Elemento

Este diagrama representa la secuencia que se ejecuta cuando el usuario crea un modelo, concretamente la acción de añadir nuevos elementos al modelo.

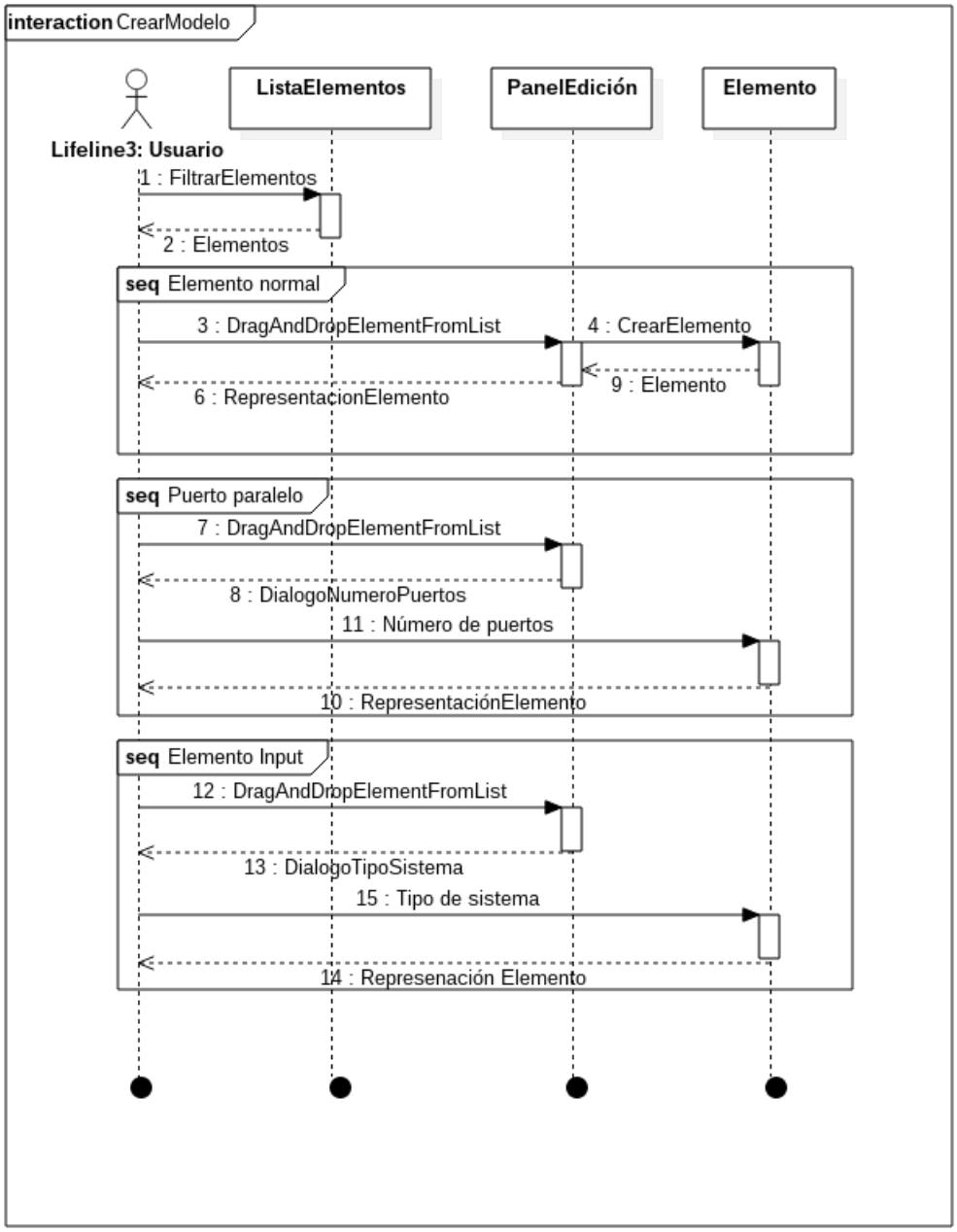


Figura 5.7: Diagrama de secuencia Crear Modelos

Este diagrama representa la secuencia que se ejecuta cuando el usuario enlaza elementos del modelo.

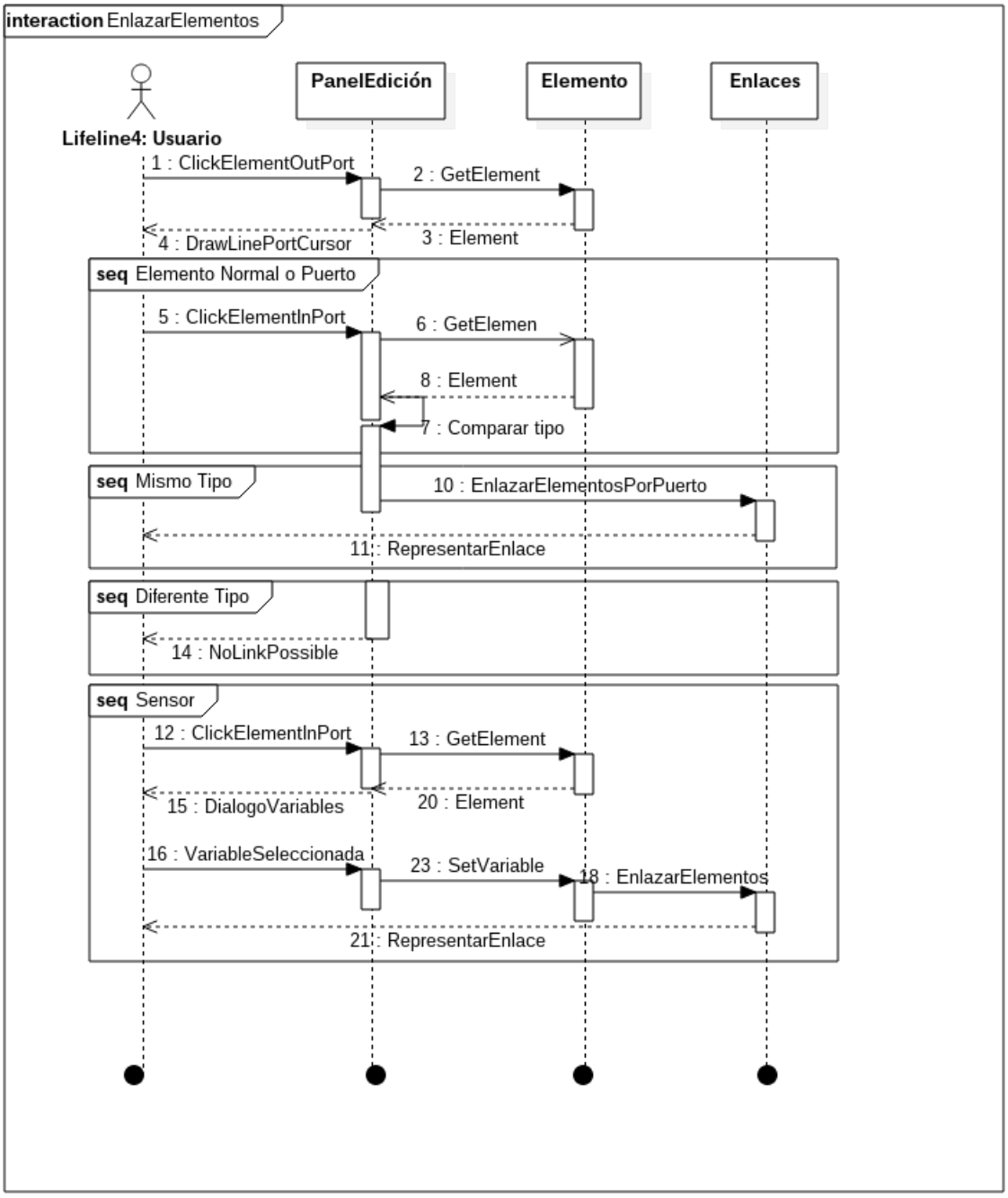


Figura 5.8: Diagrama de secuencia Enlazar Elementos

5.3. Diseño de la interfaz

En este apartado se va a explica cómo va a ser la interfaz principal de la aplicación.

5.3.1. Prototipos

A continuación se proponen algunos prototipos de la interfaz gráfica elaborados con Pencil.

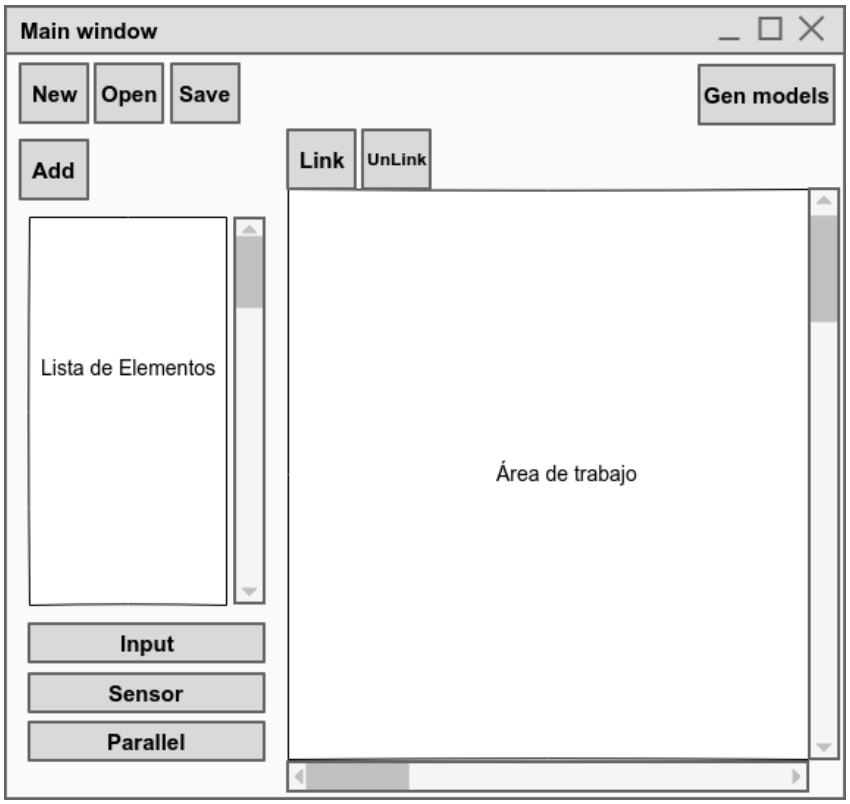
Nombre	Ventana principal
Descripción	Es la ventana principal donde se muestran los controles de edición, la lista de componentes y el panel de diseño.
Evento de Activación	El usuario inicia la aplicación
Boceto	
Eventos	CrearElemento, UnirElementos, AñadirElemento, SeleccionarElmento, Guardar, Cargar, NuevoSistema, GenerarModelos, EliminarElemento, EditarElemento, CopiarElemento, CortarElemento y PegarElemento

Tabla 5.1: PROT-1: Ventana Principal

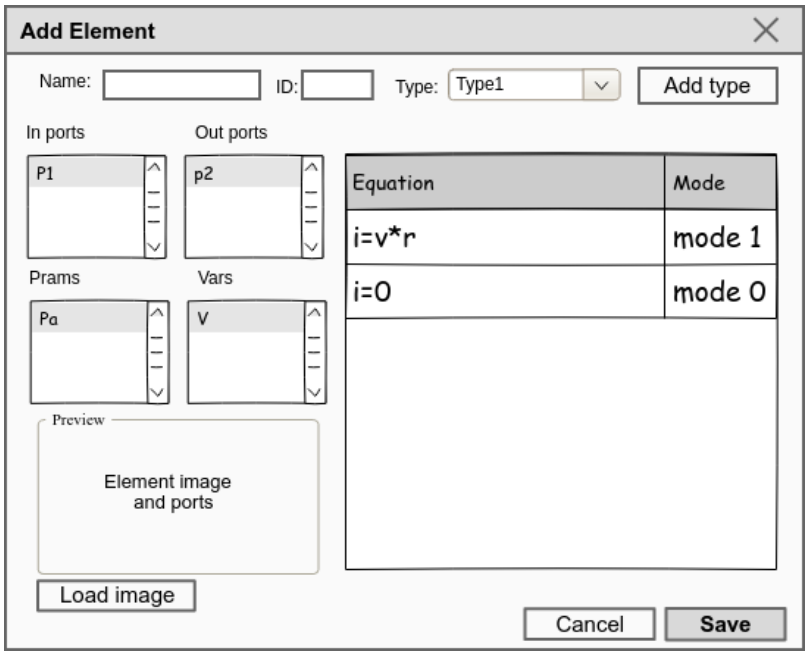
Nombre	Ventana crear elemento
Descripción	Es la ventana que se muestra para que el usuario pueda introducir los datos de los elementos que quiera crear/editar.
Evento de Activación	CrearElemento, EditarElemento
Boceto	
Eventos	AñadirPuertos, AñadirVariables, AñadirParametros, AñadirEcuaciones, AñadirImagen, GuardarElemento, AñadirTipo

Tabla 5.2: PROT-2: Ventana Crear elemento

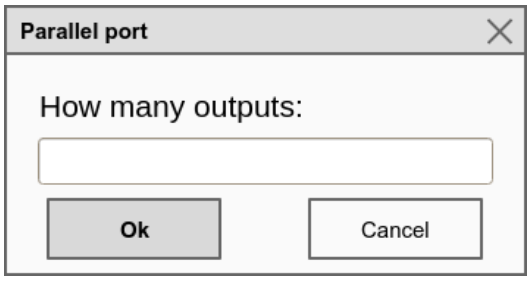
Nombre	Ventana puerto paralelo
Descripción	A través de esta ventana se pueden definir el número de puertos de salida de un puerto paralelo.
Evento de Activación	El usuario arrastra un elemento de tipo puerto paralelo a la zona de Edición
Boceto	
Eventos	AñadirElemento

Tabla 5.3: PROT-3: Ventana puerto paralelo

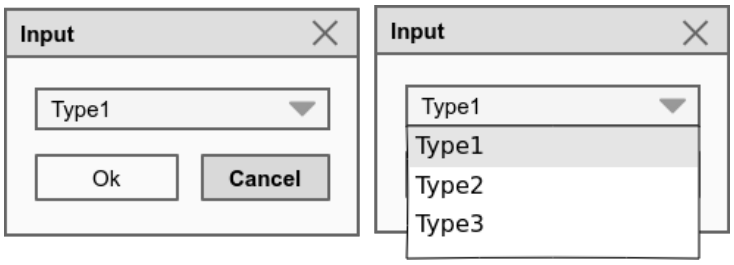
Nombre	Ventana input
Descripción	A través de esta ventana se puede definir el tipo de input se que va a añadir.
Evento de Activación	El usuario arrastra un elemento de tipo input a la zona de Edición
Boceto	
Eventos	AñadirElemento, DefinirTipoElemento

Tabla 5.4: PROT-4: Ventana input

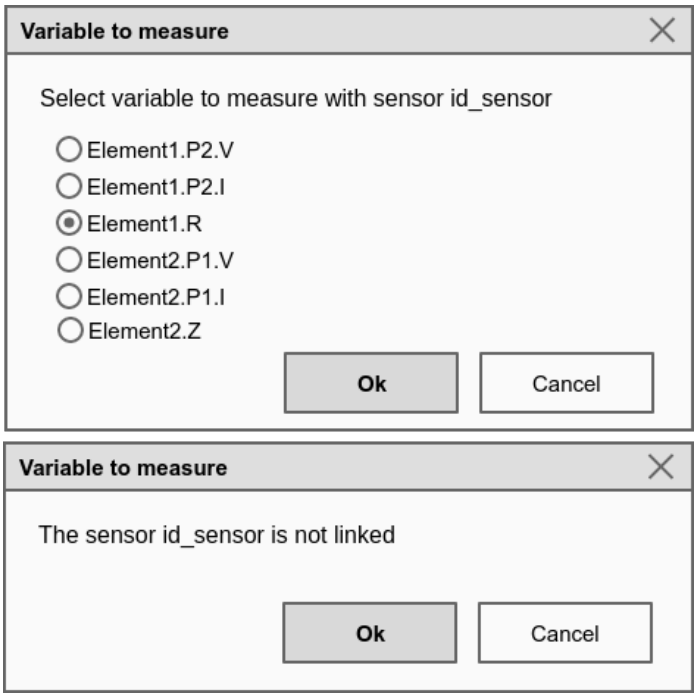
Nombre	Ventana sensor
Descripción	A través de esta ventana se puede definir la variable que va a medir un determinado sensor.
Evento de Activación	El usuario vincula ese sensor a un elemento o el usuario clickea sobre el elemento para definir la variable a medir.
Boceto	
Eventos	DefinirVariable

Tabla 5.5: PROT-5: Ventana sensor

5.4. Arquitectura

5.4.1. Arquitectura lógica

A través de la arquitectura lógica se describen los componentes lógicos del sistema y su relación.

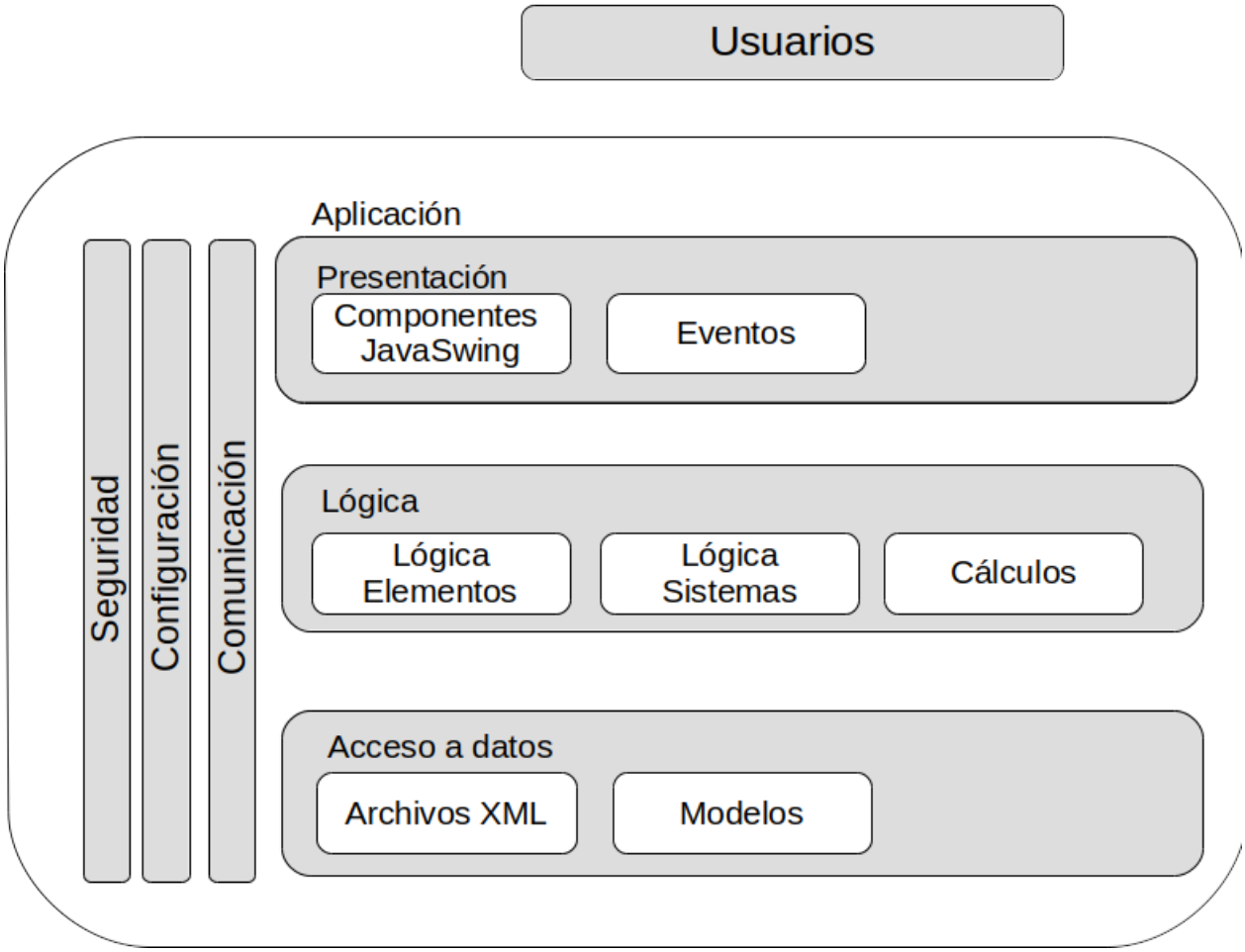


Figura 5.9: Arquitectura Lógica

En la figura 5.9 se representan las diferentes capas lógicas de la aplicación. Para ello se ha empleado el esquema de 3 capas: capa de presentación, la cual se encarga principalmente de la interfaz gráfica y los eventos que realiza el usuario sobre ella; capa de lógica, dónde se realiza la lógica de la aplicación y los cálculos internos y la capa de datos, donde están los componentes que se utilizan para el acceso a datos.

5.4.2. Arquitectura física

Por medio de la arquitectura física se representa cómo han de ser los diferentes componentes lógicos (Hardware) que se van a utilizar en el sistema y la comunicación entre ellos.

En este caso al ser una aplicación de escritorio la arquitectura física se limita a un único componente físico: el ordenador personal donde se ejecutará la aplicación.

Capítulo 6

Implementación

En este apartado explica como se ha realizado la implementación de la aplicación. También se detallarán el hardware y el software empleado y los requisitos necesarios para poder ejecutar la aplicación.

6.1. Tecnología

En esta sección se pretende explicar las tecnologías empleadas para el desarrollo del sistema.

La aplicación se ha desarrollado íntegramente en JAVA utilizando Swing para la realización de la interfaz.

6.2. Herramientas

En esta sección se analizan las herramientas utilizadas para el desarrollo del software.

6.2.0.1. Netbeans 8.2

Netbeans es un IDE de código abierto, desarrollado por Oracle para la realización de aplicaciones programadas en lenguaje de programación Java. Su principal atractivo reside en que es el más apropiado para realizar aplicaciones Java ya que tiene una interfaz adecuada para la programación en dicho lenguaje y recibe soporte oficial de Oracle. Además se trata de un software multiplataforma, lo cual permite exportar e importar proyectos sin problemas en diferentes dispositivos sin importar que tengan corriendo sistemas operativos diferentes.

6.2.1. XML Copy Editor

Es un editor de archivos XML y verificador a través de ficheros DDT. Se trata de un software de código abierto que facilita la edición y verificación de archivos XML y DDT, los cuales serán utilizados como bases de datos y archivos de respaldo.

6.3. Detalles de la implementación

En esste apartado se van a explicar los detalles más relavantes en cuanto al código implementado:

6.3.1. Verificación de Ecuaciones

Para mejorar la fiabilidad de la aplicación ha sido necesario crear un sistema que permita verificar de alguna manera las ecuaciones que introduce el usuario, de tal manera que si son incorrectas no se pueda creer el elemento. Aunque no es fiable del todo si que ayuda en gran medida a evitar introducir datos erroneos. A continuación se detallan algunos aspectos de la clase **FormulaHelper**

```
private static final String allowedOperators[] ={"+","-",
    "*","/","^", "Integral","dot","%","log","tan","sin","
    cos","tanh","sinh","cosh","ln","mod"};
private static final String ALLOWEDSIMBOLS = "
    (\\+|\\*|\\-|\\/|\\^|\\)|\\(|\\[|\\]|\\{|\\}|integral|
    dot|%|log|tan|sin|cos|tanh|sinh|cosh|ln|mod)";
```

Codigo 6.1: Codigo formula operadores

Las dos variables representadas en el código 6.1 las utilizamos para definir las operaciones que consideramos que son las que se pueden emplear en la mayoría de fórmulas.

```
private static boolean isNumeric(String str){
  return str != null && str.matches("[-+]?\\d*\\.?\\d+");
}
```

Codigo 6.2: Codigo comprobar numero

El método representado en el código 6.2 sirve para determinar si una valor de tipo String es un valor numérico.

```
public static String parseToID(String formula, String id,
    ArrayList <String> variables){
    Iterator it = variables.iterator();
    String variable2;
    while(it.hasNext()){
        String variable = (String)(it.next());
        variable2=(variable.replaceFirst("^"+id, "{id}"))
            ;
        if(!variable2.equals(variable)){
            formula = formula.replace(variable, variable2
                );
        }

    }

    return formula;
}
```

Codigo 6.3: Codigo pasar de id

El método representado en el código 6.3 se utiliza para parsear los IDS de los elementos y establecer un identificador genérico ID el cual nos va a permitir cambiarlo por el identificador único de cada elemento cuando este es añadido al modelo. Los argumentos que se le pasa son la fórmula a cambiar, el identificador del elemento y un array con las posibles variables que tiene el sistema. El valor devuelto es la fórmula escrita con IDS genéricos.

```
public static String parseToFormula(String formula,
   String id){
    return formula.replace("{id}", id);
}
```

Codigo 6.4: Codigo pasar a id

El método representado en el código 6.4 se utiliza para cambiar el valor de un ID genérico por el ID específico del elemento. Los argumentos que se le pasan son la fórmula y el identificador del elemento. El valor devuelto es la fórmula escrita con el ID del elemento.

```
public static int isIncorrect(String formula, ArrayList <
   String> variables){
    if(formula.isEmpty())return -1;
    if(formula.charAt(0)== '=' || formula.charAt(formula.
       length()-1) == '=')return 6;
    String equations[];
    String notAlowedSt[] = {"*","/","^","mod"};
    String notAlowedEn[] = {"*","/","^","+","-","Integral
       ","dot","log","tan","sin","cos","tanh","sinh","
       cosh","ln","mod"};
    equations=formula.split("=");
    if(equations.length>2)return 2;
    for (String equation1 : equations) {
        String[] equations2 = equation1.split("((?<=("+
           ALLOWEDSIMBOLS+"))|(?="+ALLOWEDSIMBOLS+"))");
        Stack <Character>  parenthesis = new Stack<>();
        if(Arrays.asList(notAlowedSt).contains(equations2
           [0]))return 4;
        if(Arrays.asList(notAlowedEn).contains(equations2
           [equations2.length-1]))return 5;
        for(String equation: equations2){
            switch(equation){
                case "(":
                    parenthesis.push(')');
                    break;
                case "[":
                    parenthesis.push(']');
                    break;
                case "{":
                    parenthesis.push('}');
                    break;
                case ")":case "]":case "}":
                    if(parenthesis.isEmpty())return 3;
                    char c =parenthesis.pop();
                    if(equation.charAt(0)!=c)return 3;
                    break;
                default:
                    if(Arrays.asList(allowedOperators).
                       contains(equation.trim()));
```

```
                        else if(variables.contains(equation.
                            trim()));
                        else if(!isNumeric(equation)){
                            return 1;
                        }
                        break;
                    }
                }
                if(!parenthesis.isEmpty()){
                    return 3;
                }
            }
            return 0;
        }
    }
```

Codigo 6.5: Codigo verificar ecuaciones

El método representado en el código 6.5 es el método más importante de la clase y es el encargado de verificar que las ecuaciones introducidas son correctas. Para ello se le pasa una fórmula junto con un array con todas las variables que definen un elemento. Primero definimos que símbolos no pueden comenzar una ecuación y cuales no la pueden terminar. Luego se separa la ecuación mediante los símbolos de igual de tal forma que si tiene más de un igual la ecuación no es correcta, además haciendo esto nos permite verificar más facilmente si los símbolos de paréntesis se cierran correctamente. Posteriormente se dividen las ecuaciones resultantes de separarlas del igual por medio de los símbolos de operaciones permitidos, dejando de esta manera arrays que sólo puedan tener operadores, números y/o variables. Comprobamos que el primer y el último valor no tenga operaciones no permitidas para esas posiciones. Luego recorremos todo el array y vamos comprobando que los valores sean correctos. Si en el camino nos encontramos con apertura de paréntesis, llaves o corchetes; se guardará el correspondiente simbolo de cierre en una pila. Cuando se encuentren los símbolos de cierre se comprobarán si concuerdan con el último elemento de la pila. Si no es ningún símbolo de paréntesis, se comprueba que o bien esté dentro de los operadores permitidos, de las variables o que sea un valor numérico. En el caso de que no corresponda con ninguno se indicará que la ecuación es incorrecta. Por último se comprueba que le número de paréntesis en la pila sea 0, pues de esa forma nos indica que se han cerrado correctamente todos los paréntesis.

6.3.2. Clipboard

Para faciitar la edición de los sistemas se ha decidido implementar un portapapeles para que el usuario pueda realizar los métodos de copiar, cortar y pegar, y así agilizar el diseño de los sistemas.

```
static void copyClipBoard(){
    if(DropScrollPane.getSelectedElements().isEmpty())
       return;
    Iterator it = DropScrollPane.getSelectedElements().
       iterator();
    linksAndElements lae = new linksAndElements();
    List <Elements> selElements = new ArrayList();
    while(it.hasNext()){
        selElements.add(((DraggableElement)it.next()).
           getElemento());
    }
    lae.setElementos(selElements);
    List <enlace> tmpEnlaces = new ArrayList();
    it = DropScrollPane.getEnlaces().iterator();
    while(it.hasNext()){
        JButton tmp[] = (JButton[])it.next();
        enlace tmpEnlace = new enlace();
        tmpEnlace.setOrig(((DraggableElement)tmp[0].
           getParent()).getElemento());
        tmpEnlace.setDest(((DraggableElement)tmp[1].
           getParent()).getElemento());
        tmpEnlace.setPortOrig(tmp[0].getToolTipText());
        tmpEnlace.setPortDest(tmp[1].getToolTipText());

        if(lae.getElementos().contains(tmpEnlace.getOrig
           ())&&lae.getElementos().contains(tmpEnlace.
           getDest())){
            tmpEnlaces.add(tmpEnlace);
        }
    }
    lae.setEnlaces(tmpEnlaces);
    try {
        ByteArrayOutputStream bo = new
           ByteArrayOutputStream();
```

```
            ObjectOutputStream so = new ObjectOutputStream(bo
                );
            so.writeObject(lae);
            so.flush();
            String transferable = Base64.getEncoder().
                encodeToString(bo.toByteArray());
            StringSelection stringSelection = new
                StringSelection(transferable);
            Clipboard clipboard = Toolkit.getDefaultToolkit()
                .getSystemClipboard();
            clipboard.setContents(stringSelection, null);

        } catch (IOException ex) {
            Logger.getLogger(VentanaPrincipal.class.getName()
                ).log(Level.SEVERE, null, ex);
        }
    }
```

Codigo 6.6: Codigo copiar en el portapaleles

El método representado en el código 6.6 se utiliza para copiar los elementos seleccionados al portapapeles. Primero se recorren todos los elementos seleccionados y se almacenan en una variable temporal. Luego se recorren los enlaces y se comprueba cuales de ellos enlazan elementos seleccionados. Posteriormente se codifica la variable que contiene los elementos y los enlaces seleccionados en base64 para poder insertarlos en el portapapeles.

```
static  boolean correctClipboard(){
    try{
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        Clipboard clipboard = toolkit.getSystemClipboard
           ();
        String result = (String) clipboard.getData(
           DataFlavor.stringFlavor);
        byte b[] = Base64.getDecoder().decode(result);
        ByteArrayInputStream bi = new
           ByteArrayInputStream(b);
        ObjectInputStream si = new ObjectInputStream(bi);
        linksAndElements lae = (linksAndElements) si.
           readObject();
        if(lae.getElementos().isEmpty())return false;
    } catch (UnsupportedFlavorException | IOException |
       IllegalArgumentException | ClassCastException|
       ClassNotFoundException ex) {
        return false;
    }
    return true;
}
```

Codigo 6.7: Codigo verificar el portapaleles

El método representado en el código 6.7 se utiliza para verificar que el clipboard esté correcto. Para ello se coje la información que hay en el portapapeles y se intenta convertirla a la clase clipboard, si en alguno de los pasos se produce una excepción significa que el contenido del portapales no es compatible con nuestra aplicación.

```
static void pasteClipBoard(Point position){
    Point small = null;
    if(!correctClipboard())return;
    try {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        Clipboard clipboard = toolkit.getSystemClipboard
            ();
        String result = (String) clipboard.getData(
            DataFlavor.stringFlavor);
        byte b[] = Base64.getDecoder().decode(result);
        ByteArrayInputStream bi = new
            ByteArrayInputStream(b);
        ObjectInputStream si = new ObjectInputStream(bi);
        linksAndElements lae = (linksAndElements) si.
            readObject();
        Iterator it = lae.getElementos().iterator();
        DropScrollPane.getSelectedElements().clear();
        while(it.hasNext()){
            Elements elm = (Elements)it.next();
            while(!eal.uniqID(elm.getShortName()+
                panelModel.getCounter())){
                 panelModel.increaseCounter();
            }
            elm.setShortName(elm.getShortName()+
                panelModel.getCounter());
            panelModel.increaseCounter();
            System.out.println(elm.getName());
            if(small==null)small=elm.getLocation();
            else{
                 if(elm.getLocation().x<small.x)small.x=
                     elm.getLocation().x;
                 if(elm.getLocation().y<small.y)small.y=
                     elm.getLocation().y;
            }
            DropScrollPane.getSelectedElements().add(
                addElement(elm));
        }
        if(position!=null){
            System.out.println(position);
            System.out.println(small);
            moveElements(position.x-small.x, position.y-
                small.y);
        }
```

```
            menuCopy.setEnabled(!DropScrollPane.
                getSelectedElements().isEmpty());
            menuCut.setEnabled(!DropScrollPane.
                getSelectedElements().isEmpty());
            menuDelete.setEnabled(!DropScrollPane.
                getSelectedElements().isEmpty());
            refresLinks(lae);

        } catch (UnsupportedFlavorException | IOException |
            ClassNotFoundException ex) {
            Logger.getLogger(VentanaPrincipal.class.getName()
                ).log(Level.SEVERE, null, ex);
        }
    }
```

Codigo 6.8: Codigo pegar desde el portapaleles

El método representado en el código 6.8 se utiliza para pegar elementos desde el portapapeles. Puede recibir un parámetro adicional indicando una posición, el cuál es útil a la hora de pegar los elementos en una zonda indicada con el ratón a través del menú contextual. Al principio el método comprueba que la información almacenada en el portapales sea correcta y si es correcta entonces se descodifica de base64 y se instancia la variable que contiene los elementos y los enlaces. Luego se deseleccionan los elementos ya que los elementos pegados serán los nuevos elementos marcados como seleccionados. Se va iterando por los elementos y se van añadiendo al panel de diseño, comprobando que el id que se va generando no esté siendo utilizado ya por otro elemento; por último se añaden los enlaces. Si se ha definido una posición, entonces los elementos son desplazados hasta esa posición.

6.3.3. Enlaces entre puertos

Una de las cosas más complejas pero más importantes de este proyecto es el enlazado de diferentes elementos de forma gráfica y lógica. Para lograrlo se han creado botones redondos en cada elemento representando cada puerto los cuales procesan varios eventos generados por el ratón.

```
for(int i=0;i<inPorts.size();i++){
    RoundButton con;
    con = new RoundButton();
    con.setToolTipText(inPorts.get(i));
    con.setBackground(Color.yellow);
    con.setPreferredSize(new Dimension(8,8));
    con.setBounds(0,(80/(inPorts.size()+1))*(i+1)
        ,8,8);
    con.addMouseListener(new MouseListener(){

    con.addActionListener(new ActionListener(){

    panel.add(con);
}
for(int i=0;i<outPorts.size();i++){
    RoundButton con;
    con = new RoundButton();
    con.setToolTipText(outPorts.get(i));
    con.setBackground(Color.yellow);
    con.setPreferredSize(new Dimension(8,8));
    con.setBounds(100-8,(80/(outPorts.size()+1))*(i
        +1),8,8);
    con.addMouseListener(new MouseListener(){

    con.addActionListener(new ActionListener(){

    panel.add(con);
}
```

Codigo 6.9: Creación de los puertos

El fragménto de código presentado en 6.9 muestra como se crean los puertos de entrada y los de salida. También se indica que se van a manejar los eventos del ratón. Como los eventos relativos al movimiento del ratón són sólo estéticos no vamos a profundizar en ellos, por lo que nos centraremos en los eventos lanzados al clickear con el ratón.

```
            con.addActionListener(new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent e) {
                    JButton sourceButton=(JButton)e.getSource
                        ();
                    DraggableElement sourcePanel = (
                        DraggableElement)sourceButton.
                        getParent();
                    Elements source = sourcePanel.getElemento
                        ();

                    if(unlink.isSelected()){
                        Iterator it = DropScrollPane.
                            getEnlaces().iterator();
                        while(it.hasNext()){
                            JButton[] tmpBtn=(JButton[])it.
                                next();
                            if(tmpBtn[0]==sourceButton ||
                                tmpBtn[1]==sourceButton){
                                it.remove();
                                break;
                            }
                        }
                        eal.removeEnlace(source, sourceButton
                            .getToolTipText());
                        panelModel.repaint();
                        saved=false;
                        btnSave.setEnabled(!saved);
                        menuSave.setEnabled(!saved);
                    }else if(link.isSelected())
                    if((isButtonLinked(sourceButton)<4)&&(
                        firstButton==null)){
                        firstButton = sourceButton;
                        firstButton.setBackground(Color.blue)
                            ;
                    }else{
                        if(firstButton!=null)
                        firstButton.setBackground(Color.
                            yellow);
                        firstButton=null;
                    }
                }

            });
```


Codigo 6.10: Outport click

En el fragmento de código presentado en 6.10 se ejecuta cuando el usuario clickea sobre un puerto de salida de un elemento. Como los enlaces se hacen siempre primero seleccionando el puerto de salida y luego el puerto de entrada, en este punto el código lo que va a hacer es definir cuál es el puerto de origen, para posteriormente poder definir el puerto de destino y realizar el enlace. Para el correcto funcionamiento primero se comprueba si está seleccionada la opción de unir elementos (o la de desunir). Si se ha seleccionado la de desunir se recorre toda la lista de enlaces y se elimina el enlace equivalente a ese elemento en ese puerto (si es que existe). Si por otro lado está seleccionada la opción de enlazar elementos entonces se llevarán a cabo varias comprobaciones.

- La primera comprobación que hace es determinar si el puerto seleccionado ya se encuentra unido a otro elemento que no sea un sensor (ya que cada puerto sólo permite unir a un único elemento, a menos que sea un sensor).
- Luego se comprueba si ya existe un puerto seleccionado como puerto de origen, en caso afirmativo se cancela el puerto previo.
- Si ha pasado todas las prueba se define ese puerto como puerto seleccionado.

```
            con.addActionListener(new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent e) {
                    //repaint();
                    JButton sourceButton=(JButton)e.getSource
                        ();
                    Elements source = ((DraggableElement)
                       sourceButton.getParent()).getElemento
                        ();
                    if(unlink.isSelected()){
                        Iterator it = DropScrollPane.
                           getEnlaces().iterator();
                        while(it.hasNext()){
                            JButton[] tmpBtn=(JButton[])it.
                               next();
                            if(tmpBtn[0]==sourceButton ||
                               tmpBtn[1]==sourceButton){
                                it.remove();
                                break;
                            }

                        }
                        eal.removeEnlace(source, sourceButton
                            .getToolTipText());
                        panelModel.repaint();
                        saved=false;
                        btnSave.setEnabled(!saved);
                        menuSave.setEnabled(!saved);
                    }else if(link.isSelected()){
                        boolean linked;
                        int isLinked = isButtonLinked(
                            firstButton);
                        if(source.getName().equals("Sensor"))
                            {
                            linked=false;
                        }else{
                            linked=(isLinked==2)||(isLinked
                                ==3);
                        }
                        boolean sameType=false;
```

```
if(firstButton!=null){
    Elements firstElemento = ((
        DraggableElement)firstButton.
        getParent()).getElemento();
    if(firstElemento.getType()==null)
        {
         if(!firstElemento.getName().
            equals("Sensor"))
         firstElemento.setType(source.
            getType());
         sameType=true;
    }else if(source.getType()==null){
         if(!source.getName().equals("
            Sensor"))
         source.setType(firstElemento.
            getType());
         sameType=true;
    }else if(source.getType().equals(
        firstElemento.getType())){
         sameType=true;
    }
    if(!firstButton.getParent().
        equals(sourceButton.getParent
        ()) && sameType  &&!linked &&
        isButtonLinked(sourceButton)
        <3){
         JPanel sourcePanel = (JPanel)
            firstButton.getParent();
         firstPoint = new Point(
            firstButton.getX()+
            sourcePanel.getX()+3,
            firstButton.getY()+
            sourcePanel.getY()+3);
         JButton tmpButton[]= {
            firstButton,sourceButton};
         DropScrollPane.getEnlaces().
            add(tmpButton);
         enlace tmpEnlace = new enlace
            ();
         tmpEnlace.setOrig(((
            DraggableElement)
            firstButton.getParent()).
            getElemento());
```

```
                                    tmpEnlace.setPortOrig(
                                        firstButton.getToolTipText
                                        ());
                                    tmpEnlace.setDest(((
                                        DraggableElement)
                                        sourceButton.getParent()).
                                        getElemento());
                                    tmpEnlace.setPortDest(
                                        sourceButton.
                                        getToolTipText());
                                    eal.addEnlace(tmpEnlace);
                                    panelModel.delTmpLine();
                                    firstButton.setBackground(
                                        Color.yellow);
                                    firstButton=null;
                                    saved=false;
                                    btnSave.setEnabled(!saved);
                                    menuSave.setEnabled(!saved);
                                    panelModel.repaint();
                                    if(tmpEnlace.getDest().
                                        getName().equals("Sensor")
                                        ){
                                          dialogSensor(tmpEnlace.
                                              getDest());
                                    }
                                }
                            }
                        }
                    }

                });
                panel.add(con);
            }
```

Codigo 6.11: Inport click

En el fragmento de código presentado en 6.11 se muestra como se procesa el evento de click de un puerto de entrada, el cuál permitirá completar un enlace. Primero se comprueba si está activado el modo de unión o de desunion, si está seleccionado el de desunión entonces se recorre el array con los enlaces y se eliminan todos los enlaces que están vinculados con el puerto de ese elemento. Por otro lado si está seleccionado el modo

de unión entonces se procede de la siguiente forma:

- Se comprueba si el el puerto pertenece a un sensor.
- Si no es un sensor se comprueba si el elemento de origen no tiene ningún tipo establecido (normalmente usado por los puertos) de tal forma que le asigne el mismo tipo.
- Y si tiene un tipo definido se comprueba que sea del mismo tipo que el elememto al que pertenece el puerto de entrada.
- Posteriormente se verifica que el elemento de origen y destino son del mismo tipo y se comprueba que no estén enlazados con otros elementos que no sean sensores.
- Si todo ha ido bien, entonces el se unen los elementos a través de una clase y se añaden a la lista de enlaces.
- Por último si el elemento al que pertenece el puerto de entrada es un elemento de tipo sensor se muestra un diálogo indicando las variables que se pueden medir con el sensor.

Capítulo 7

Pruebas

En este capítulo se muestran las diferentes pruebas realizadas durante el desarrollo de la aplicación. En este caso nos vamos a centrar en las pruebas de caja negra ya que hemos realizado verificaciones de código con la herramienta **PMD** que nos permite realizar un análisis estático del código fuente de una aplicación JAVA.

7.1. Pruebas de caja negra

Las pruebas de caja negra consísten en verificar que las salidas proporcionadas por la aplicación sean acorde a las entradas proporcionadas por el usuario

Prueba	Resultado Esperado	Resultado Obtenido
Crear un elemento con algunos de los valores obligatorios en blanco	El sistema muestra una ventana de error indicando que se necesita especificar esos valores	Correcto
Crear un elemento con un nombre ya utilizado por otro elemento del mismo tipo	El sistema indica que ese tipo ya está siendo utilizado	Correcto
Crear un elemento especificando nombres de puertos repetidos	El sistema indica que el nombre de los puertos han de ser únicos	Correcto
Crear un elemento especificando ecuaciones incorrectas	El sistema indica que las ecuaciones no son correctas	Correcto

Editar un elemento y definir un identificador que ya está siendo usado en la aplicación	El sistema indica que ese identificador ya está siendo utilizado	Correcto
Enlazar elementos de diferente tipo	El sistema no permite enlazarlos	Correcto
Guardar sistemas para cargarlos posteriormente	El sistema cargado debe ser idéntico al guardado	Los enlaces se guardan mal y los elementos no quedan bien enlazados
Guardar sistemas para cargarlos posteriormente	El sistema cargado debe ser idéntico al guardado	Correcto
Copiar y pegar elementos y enlaces	Los elementos y enlaces tienen que ser una copia del inicial	Correcto
Intentar pegar elementos desde el portapapeles no compatibles con la aplicación	No debe aparecer nada	Correcto
Añadir imágenes de gran tamaño a los elementos	El sistema debe listarlas sin problemas escalándolas	El sistema se ralantiza por el tamaño de las imágenes por lo que se determina que se han de escalar antes de guardarlas en la carpeta donde se guardarán las imágenes.
Añadir imágenes de gran tamaño a los elementos	El sistema debe listarlas sin problemas escalándolas	correcto

Tabla 7.1: Pruebas de caja negra

Capítulo 8

Manuales

En este capítulo se detallarán los diferentes manuales para el correcto uso de la aplicación. Empezando por el manual de compilación, manual de instalación y finalizando por el manual de usuario.

8.1. Manual de compilación

Esta parte no es necesaria ya que la aplicación se distribuye ya compilada, no obstante puede darse el caso de que alguien prefiera compilarla de manera manual por lo que esta sección pretende explicar cómo ha de proceder para realizar esas tareas. Se ha decidido explicarlo de dos formas, una para poderlo compilar desde netbeans y otra para compilarlo desde la consola de comandos, para que el usuario final sea quien decida qué método prefiere.

8.1.1. Usando Netbeans

Para compilar la aplicación con netbeans sólo es necesario bajar la aplicación de Netbeans desde su página web. Se puede descargar cualquiera de las versiones enfocadas a Java, tanto Java SE como Java EE.

Una vez descargado y ejecutado nos saldrá una ventana como esta:

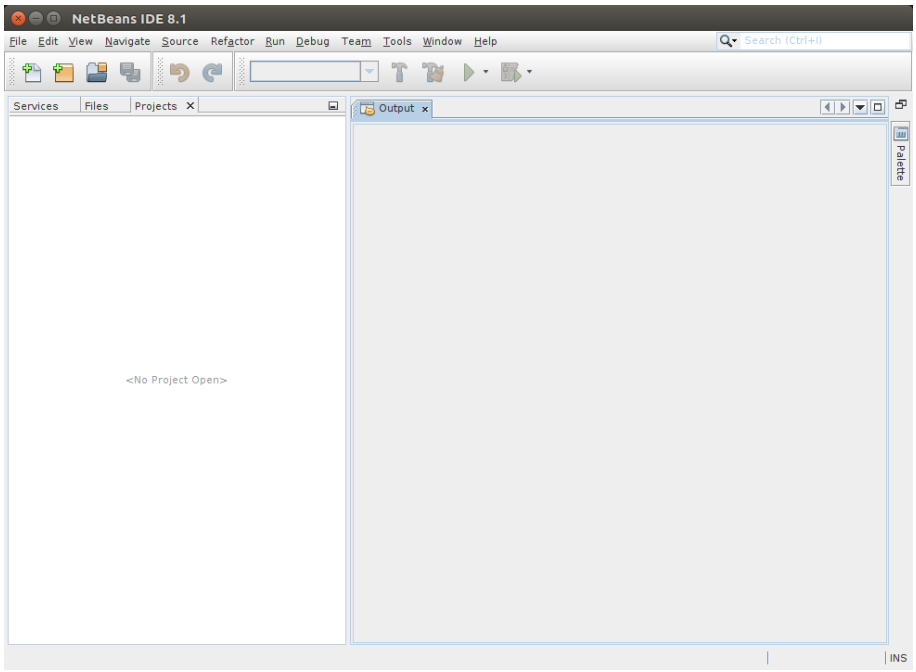


Figura 8.1: Netbeans IDE

Acto seguido presionamos en File ->Open project y nos aparecerá una ventana como la siguiente donde debemos seleccionar el proyecto:

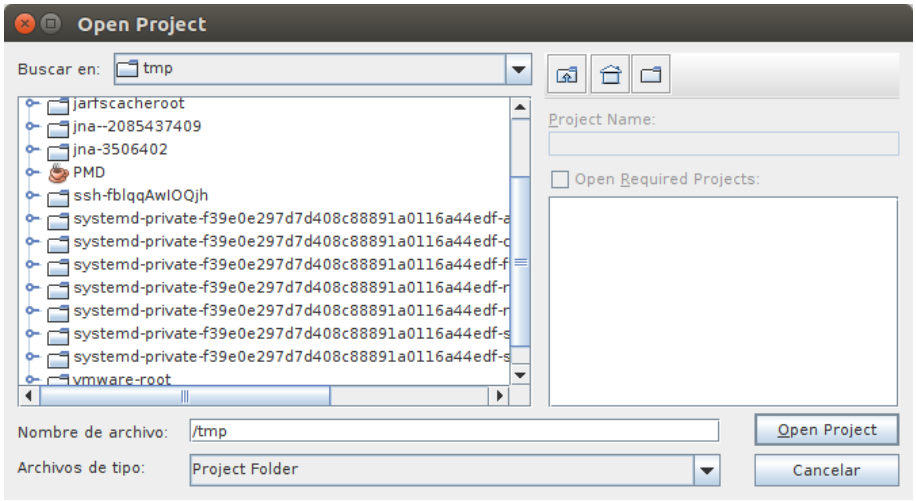


Figura 8.2: Netbeans open

Una vez lo tengamos abierto podemos ejecutarlo seleccionando Run->Run Project, o pulsar F9.

8.1.2. Usando la consola de comandos

Para compilar el programa directamente desde la línea de comandos primero tenemos que tener instalado el sdk de Java, el cual se puede descargar desde la página web de Oracle. Una vez tengamos instalado el JDK necesitamos la herramienta Ant de Apache, la cual se puede bajar desde la página web de apache. Una vez descargado e instalado todo tenemos que desplazarnos al directorio del proyecto, una vez allí podemos ejecutar

```
$ ant jar
```

Con ello se generará el archivo PMD.jar dentro del directorio dist.

8.2. Manual de Instalación

Una de las principales ventajas de esta aplicación es que no necesita ningún tipo de instalador, está pensado que se pueda ejecutar directamente en cualquier sistema operativo, además como no requiere permisos especiales no es necesario que sea ejecutada por un usuario con privilegios elevados. Por eso la forma de distribuirla es directamente como un archivo Java *.jar*, el cual creará los archivos de configuración y carpetas necesarias directamente en el directorio donde se encuentra el archivo *.jar*.

El único requisito es tener instalada una máquina virtual Java (*(*Java SE)), la cual puede descargarse directamente desde la página oficial de oracle. Alternativamente también puede ser ejecutado con openjava, el cual se puede instalar como muestran en su página web.

8.3. Manual de Usuario

En este capítulo se va a desarrollar el manual de usuario final. Para ello se utilizarán diversas capturas de la aplicación en funcionamiento. El principal interés de la realización de dicho manual es facilitarle al usuario final el uso de dicha aplicación y disminuir la curva de aprendizaje.

Al ejecutar la aplicación lo primero que vamos a ver es la siguiente pantalla:

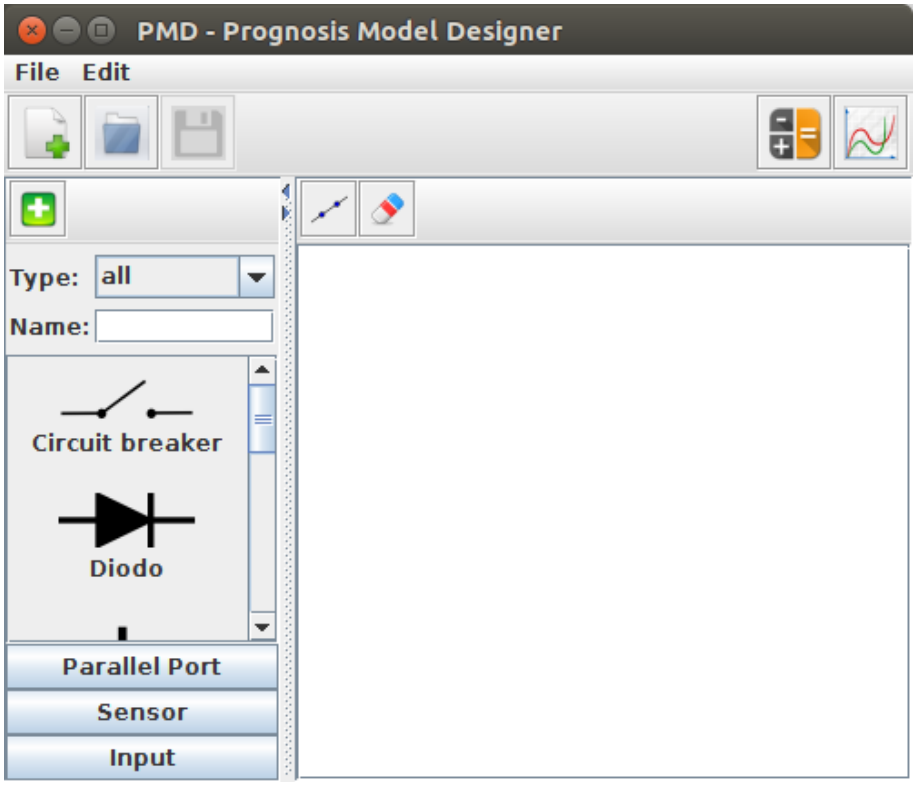


Figura 8.3: Ventana principal

En ella se puede apreciar 3 partes:

- Una zona de menús con las diferentes acciones que se pueden realizar sobre la aplicación.
- Una zona donde aparece la librería de elementos con opciones de filtrado y de creación de nuevos elementos.
- Una zona de edición, que viene siendo la parte más importante de la aplicación y es donde se van a llevar a cabo la mayor parte de la actividad.

Dentro del menú file tenemos las opciones relacionadas con la creación restauración y respaldo del sistema:

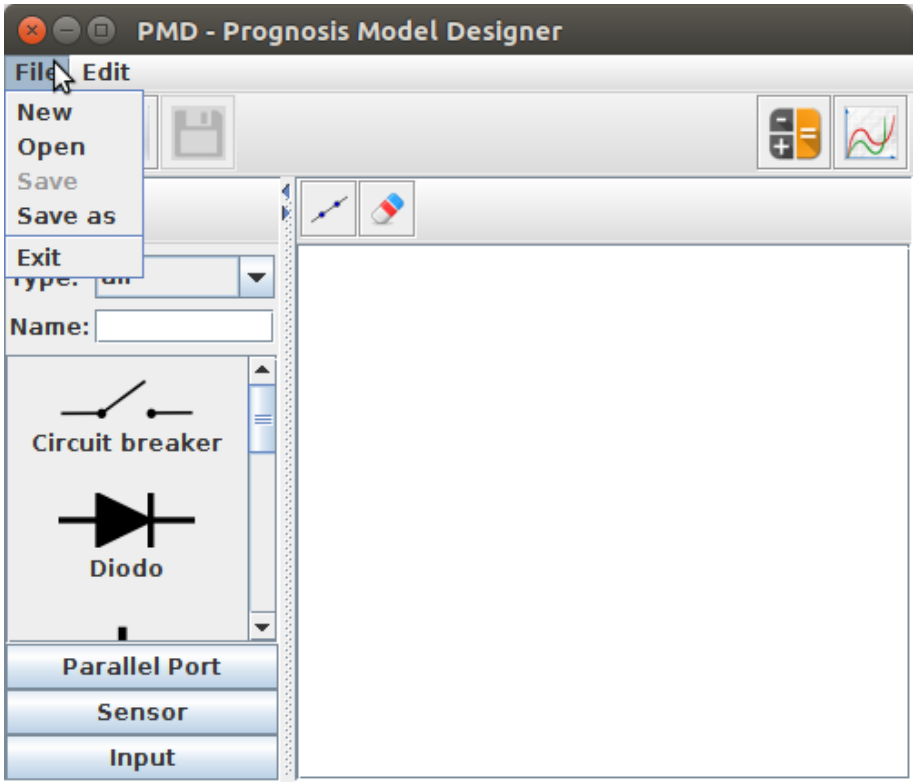


Figura 8.4: Menu file

Ahí se puede apreciar 4 funciones:

- New, Sirve para crear un nuevo sistema, si existiese previamente un sistema en la pantalla la aplicación pregunta si se desea guardar el sistema previo y si se desea cerrarlo. Esta función es igual a la que sale representada en la ventana principal con un icono en forma de hoja en blanco y una cruz verde superpuesta.
- Open, nos permite abrir un archivo de sistema previamente guardado. Para ello muestra un diálogo preguntando por el nombre de archivo. En la ventana principal tiene un icono con similar funcionalidad representado por una carpeta azul.
- Save, nos permite guardar un sistema. Si el sistema no ha sido guardado previamente entonces se preguntará por el archivo de destino a través de un cuadro de diálogo. El botón permanece deshabilitado mientras el sistema no haya sido modificado o no exista ningún elemento. Esta funcionalidad es también accesible directamente desde la ventana principal a través de un botón con icono con forma de disquete.
- Exit, cierra la aplicación, no sin antes preguntar si se desea guardar el proyecto actual.

Dentro del menú edit se encuentran las opciones relacionadas con la edición del sistema:

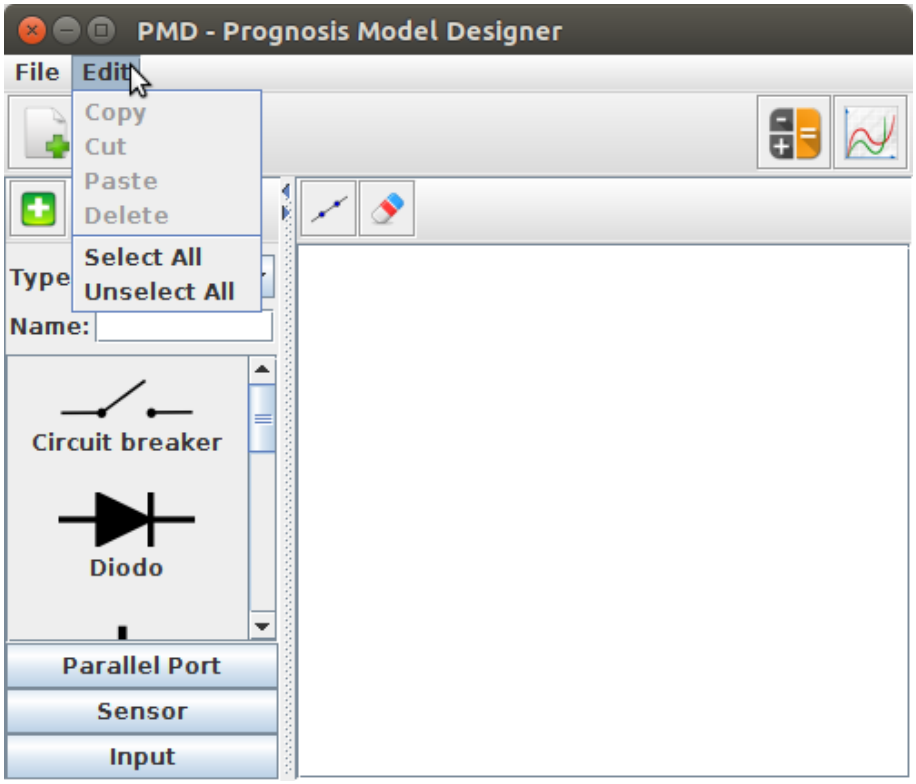


Figura 8.5: Menu edit

Las opciones de este menú son:

- Copy: se utiliza para copiar los elementos seleccionados. El mismo evento se activa al presionar las teclas ctrl+c.
- Cut: se utiliza para cortar los elementos seleccionados. El mismo evento se activa al presionar las teclas ctrl+x.
- Paste: se utiliza para pegar los elementos que se encuentran en el portapapeles y que se han copiado o cortado. El mismo evento se activa al presionar las teclas ctrl+v.
- Delete: elimina los elementos seleccionados. El mismo evento se activa al presionar la tecla supr.
- Select All: Selecciona todos los elementos de la vista.
- Unselect All: Deselecciona todos los elementos de la vista.

Para añadir un nuevo elemento a la lista sólo hay que pulsar directamente el botón con forma de cruz verde que se encuentra encima de la lista de elementos en la ventana principal. Una vez pulsado se nos abrirá una ventana como la siguiente:

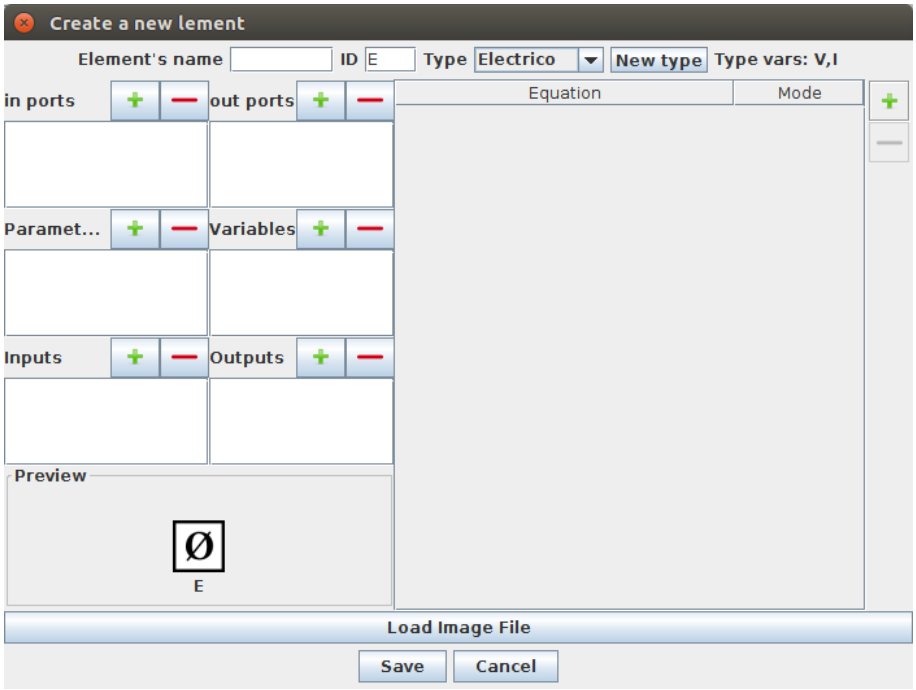


Figura 8.6: Create Element

En esta ventana se permiten definir los elementos con ayuda de un formulario: El primer campo **Element's name** sirve para indicar el nombre del elemento. El segundo campo **ID** nos permite definir un identificador para el elemento. Cuando se introduce el nombre del elemento automáticamente se escribe la inicial en el campo ID para facilitar los pasos (aunque el valor puede ser cambiado por otro más acorde). El siguiente campo **Type** nos permite escoger el tipo del elemento para determinar las variables internas del sistema. Es decir si es de tipo eléctrico trabajará con variables de tipo intensidad y voltaje, mientras que si es de tipo hidráulico las variables pueden ser Caudal y Volumen. La variables se indican en la etiqueta posteriormente mostrada **Type vars**. Aparte de los tipos ya definidos, se pueden definir nuevos tipos pulsado el botón de **New type**, el cual mostrará la siguiente ventana:

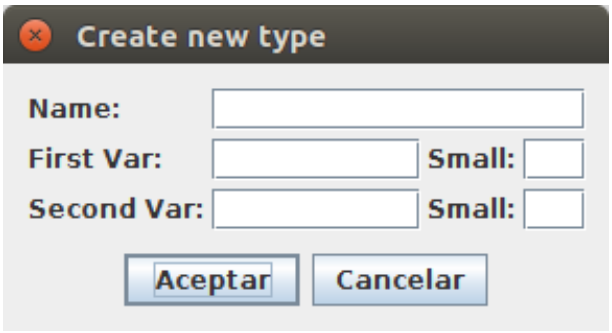


Figura 8.7: Create new type

Los campos a rellenar son: **Name**: Para definir el nombre del tipo **First Var**: para definir el nombre de la primera variable, seguido del nombre de esa variable en forma reducida (ejemplo Intensidad, I) **Second var**: Lo mismo que el campo anterior solo que define la segunda variable del sistema.

Más abajo tenemos unas listas para poder definir los puertos de entrada, salida ,variables, parámetros, etc. Obligatoriamente todos los elementos han de tener al menos un puerto de entrada o salida. Para añadir un puerto de entrada/salida, parámetro, etc hay que clickear el botón que se encuentra a la derecha, encima de cada lista con forma de cruz verde. El sistema nos preguntará el nombre del puerto/parámetro... (el siguiente ejemplo detalla las ventanas que aparecen al añadir un puerto, pero es extrapolable al resto de opciones):

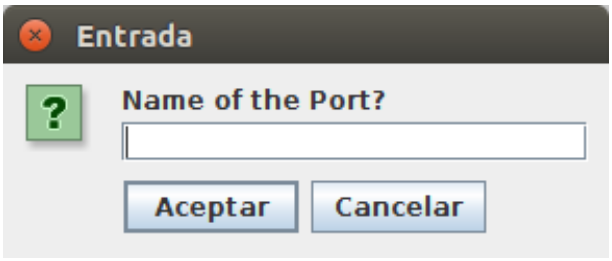


Figura 8.8: Set port name

En el caso de que ya se esté usando ese nombre de puerto en el mismo elemento se muestra la siguiente ventana de advertencia:

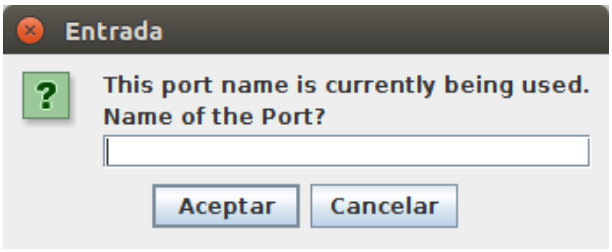


Figura 8.9: Same port name

Si queremos eliminar el puerto tenemos que seleccionar al menos 1 de los elementos de la lista de puertos y posteriormente clickear en el botón con forma de guión rojo. En el caso de no haber seleccionado ninguno, el sistema nos muestra el siguiente error:

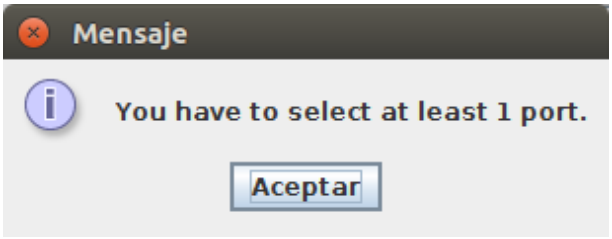


Figura 8.10: Error delete port

A medida que vayamos añadiendo puertos, modificando el ID o añadiendo ecuaciones la previsualización del elemento se irá actualizando.

Una vez tengamos todas las listas rellenadas podemos empezar a añadir ecuaciones. Aunque no es obligatorio hacerlo en ese órden, si que es más práctico ya que las ecuaciones se van a ir comprobando si están correctamente escritas utilizando para ello los nombres de los puertos, variables, parámetros... Para añadir ecuaciones hay que clickear el botón que se encuentra a la derecha de la lista de ecuaciones, representado por medio de una cruz verde. Al pulsarlo se añade una fila en la lista de ecuaciones, la cual tiene 2 campos:

- Equation: Nos permite escribir la ecuación
- Mode: Nos permite definir el modo para el cual se utilizará esa ecuación (por ejemplo modo on y off).

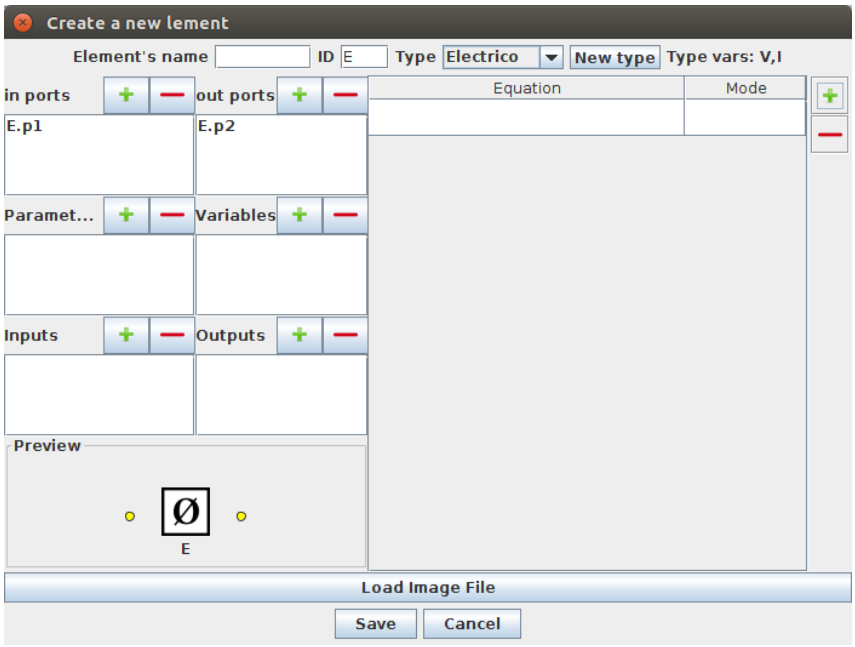


Figura 8.11: Add equation

Según se va escribiendo la ecuación, esta es verificada de forma que si se escribe una ecuación expresada incorrectamente el campo de texto se volverá rojo tal y como se muestra en las siguientes capturas:

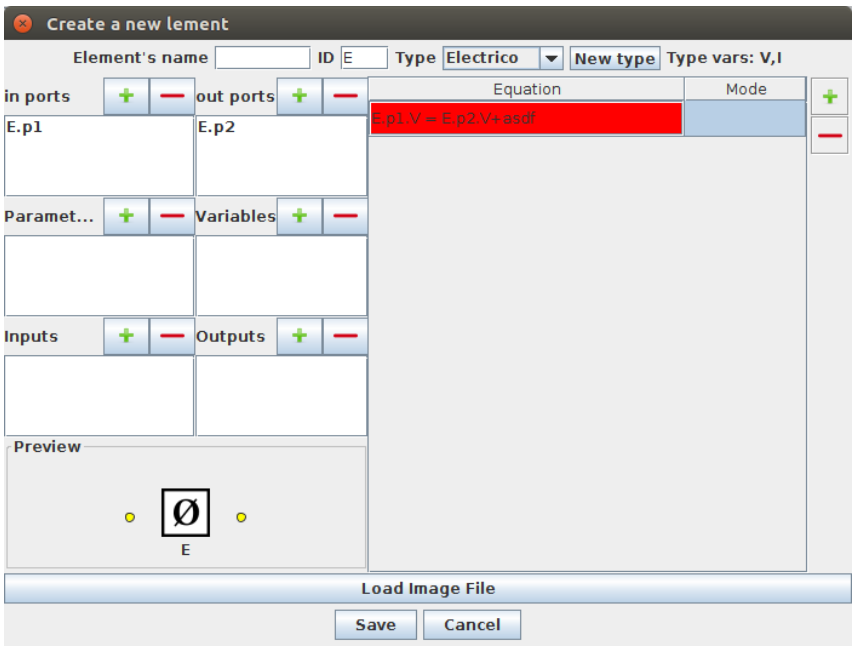


Figura 8.12: Error nombre de variable

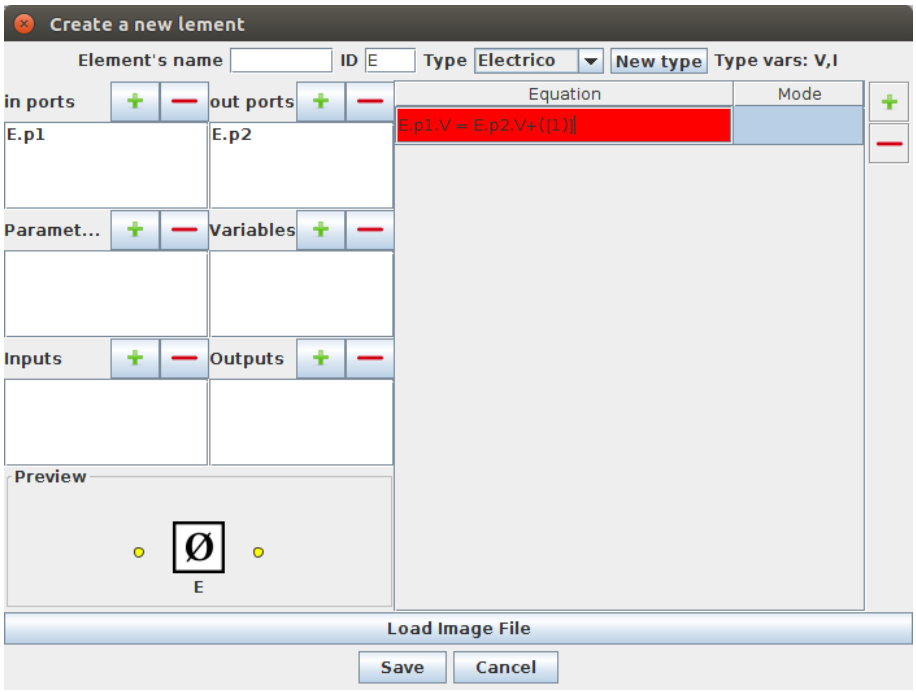


Figura 8.13: Error paréntesis corchetes

Pero si está bien escrita entonces el campo de texto se mantendrá blanco:

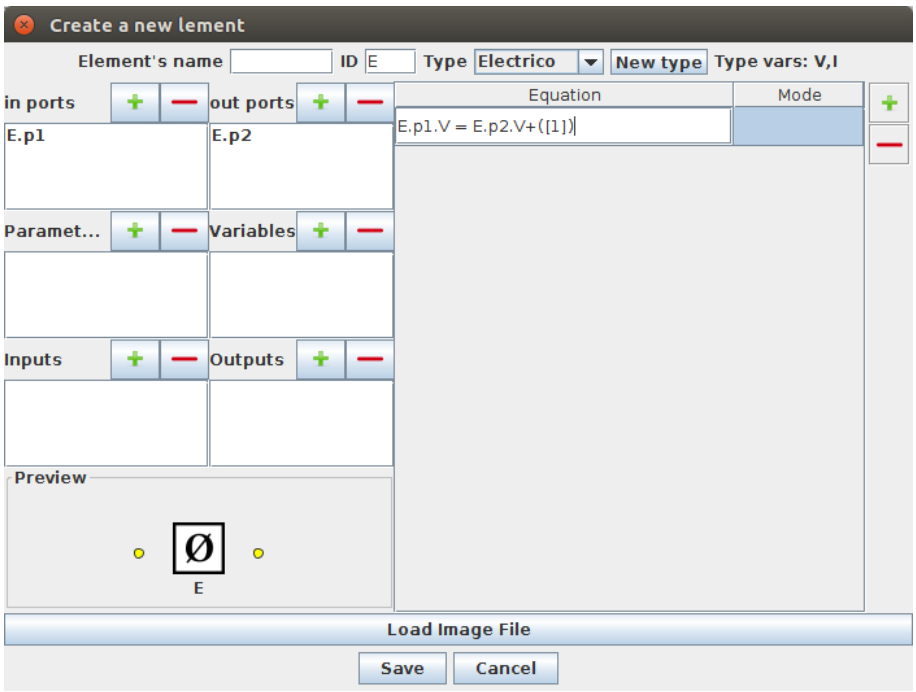


Figura 8.14: Ecuación correctamente escrita

Si por algún casual intentamos guardar el elemento con una ecuación incorrecta aparecerá el siguiente mensaje, indicándonos exactamente donde está la ecuación incorrecta.

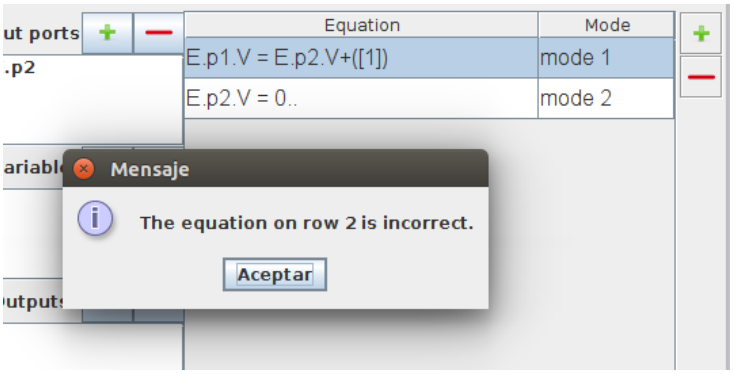


Figura 8.15: Ecuación incorrecta

Por último podemos seleccionar una imagen para asociarla con ese elemento, para ello hay que presionar el botón que se encuentra en la zona de abajo que indica **Load image File** y aparecerá la siguiente ventana a través de la cual se puede seleccionar la imagen:

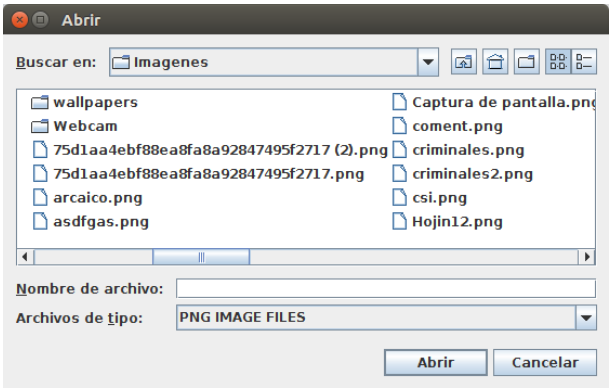


Figura 8.16: Cargar imagen para elemento

Una vez seleccionada la imagen se actualiza la vista preliminar, tal y como se muestra en la siguiente imagen:

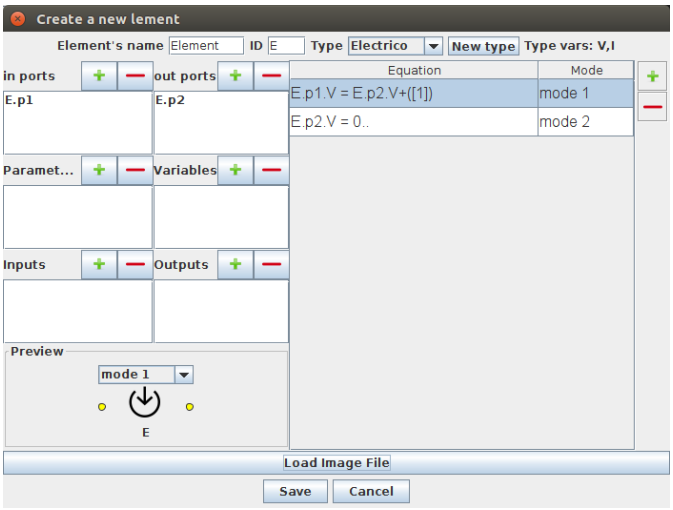


Figura 8.17: Imagen cargada

Para guardar el nuevo elemento se ha de pulsar el botón **Save** y si no se da ningún error (si existe este es notificado al usuario), se vuelve a mostrar la ventana inicial, con el nuevo elemento tal y como muestra la siguiente imagen:

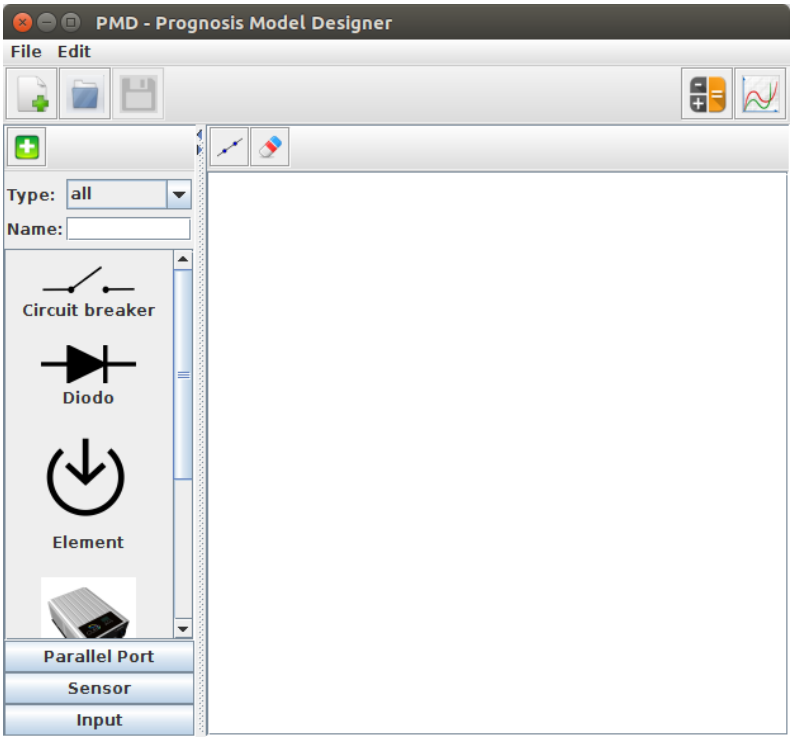


Figura 8.18: Elemento cargado

A partir de ahora empieza la parte más interesante: el diseño del sistema. La intefaz es bastante intuitiva, consiste en arrastrar y soltar. Para añadir elementos al sistema sólo hace falta arrastrarlos desde la lista a la zona de edición; los elementos genéricos, como Input, Sensor y Parallel Port, no aparece en la lista pero sí debajo de esta y el funcionamiento es el mismo: arrastrar y soltarlos en la zona de edición. Si arrastramos un elemento de tipo Input se mostrará una ventana para escoger el tipo de Input que se va a insertar (de acuerdo a los tipos de sistemas que se hayan definido previamente).

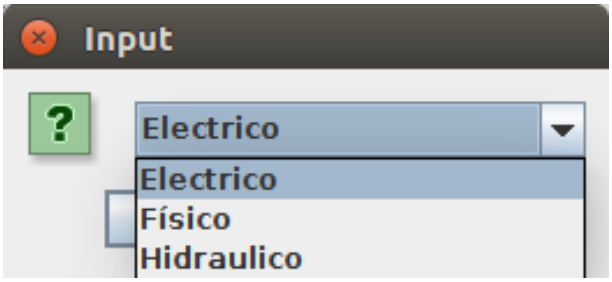


Figura 8.19: Input Type

Si lo que se añade es un puerto paralelo, el sistema pregunta por el número de salidas que va a tener ese puerto:

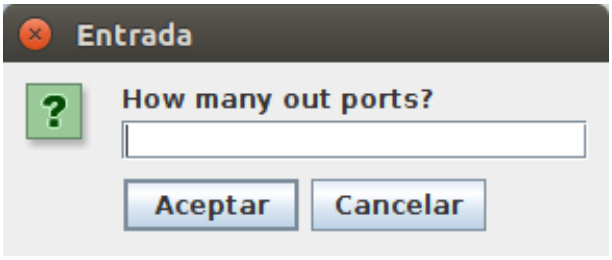


Figura 8.20: Parallel Port Outports

Para enlazar los elementos primero hay que pulsar sobre el botón Link representado por un icono con forma de enlace y situado a la izquierda del botón con forma de goma. Una vez seleccionado podremos enlazar los elementos, para ello primero hay que clickear sobre un puerto de salida (o sobre el elemento), el cual cambiará a color verde en vez de amarillo; en caso de colocar el cursor encima de un puerto de entrada este se marcará en rojo.

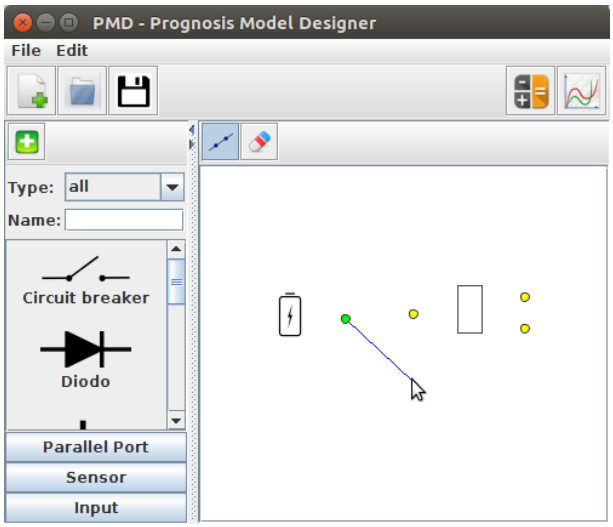


Figura 8.21: Link elements 1

Para enlazar ese puerto de salida con otro elemento hay que posicionarse sobre el puerto de entrada del elemento deseado (o directamente encima del elemento deseado) y el puerto, si pertenece al mismo tipo que el elemento inicial y no se encuentra vinculado a otro elemento diferente de un sensor, se mostrará de color verde indicando que el enlace es posible, en otro caso se mostrará de color rojo, para completar el enlace hay que clickear ya bien sea en el puerto de entrada o en el elemento.

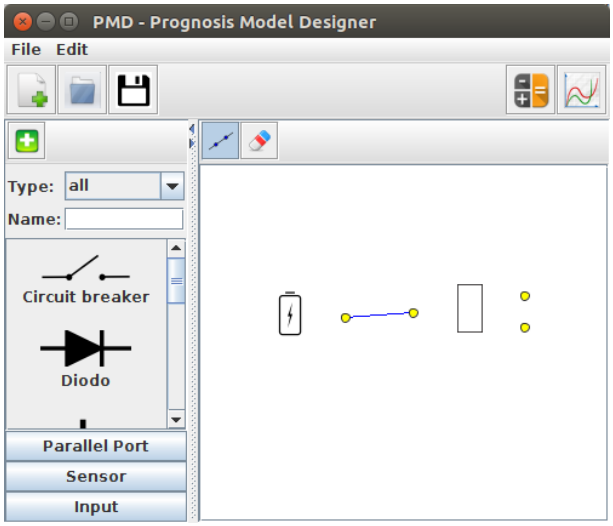


Figura 8.22: Elements Linked

En el caso de enlazar un sensor se nos mostrará una ventana en la que podemos definir la variable a medir:

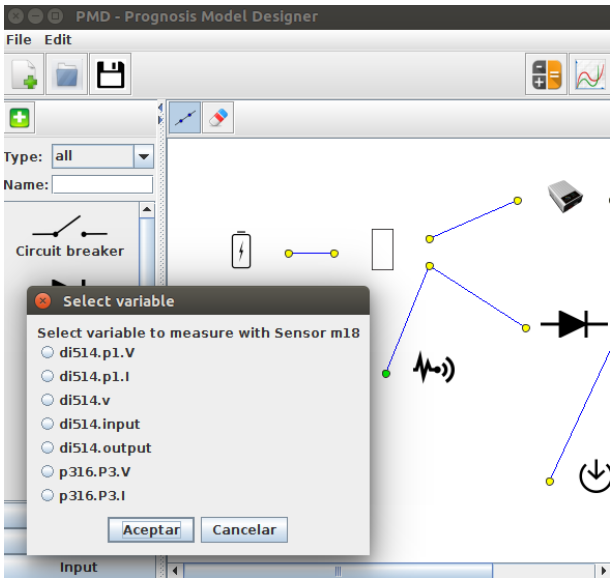


Figura 8.23: Sensor enlazado

Para facilitar la edición se pueden seleccionar múltiples elementos con ayuda del ratón, manteniendo pulsado el botón izquierdo y formando un recuadro con el que cercar los elementos deseados:

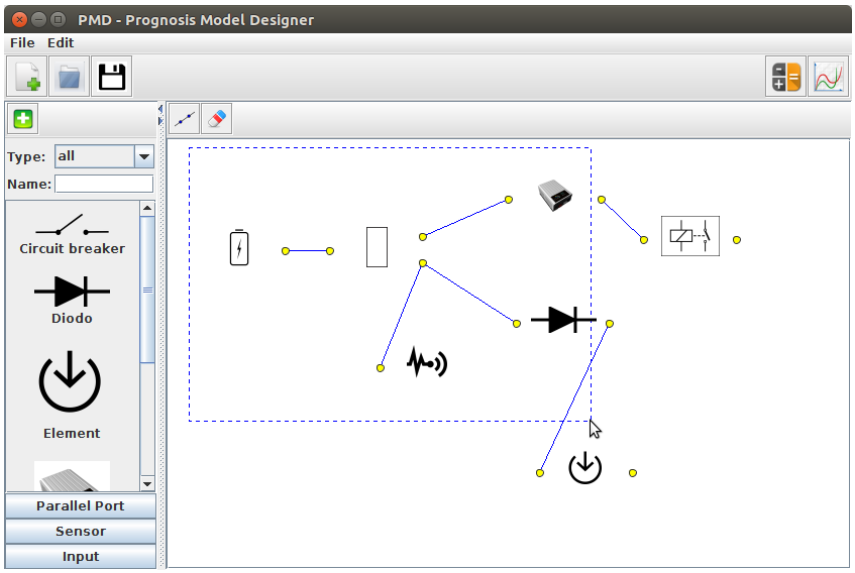


Figura 8.24: Selección múltiple

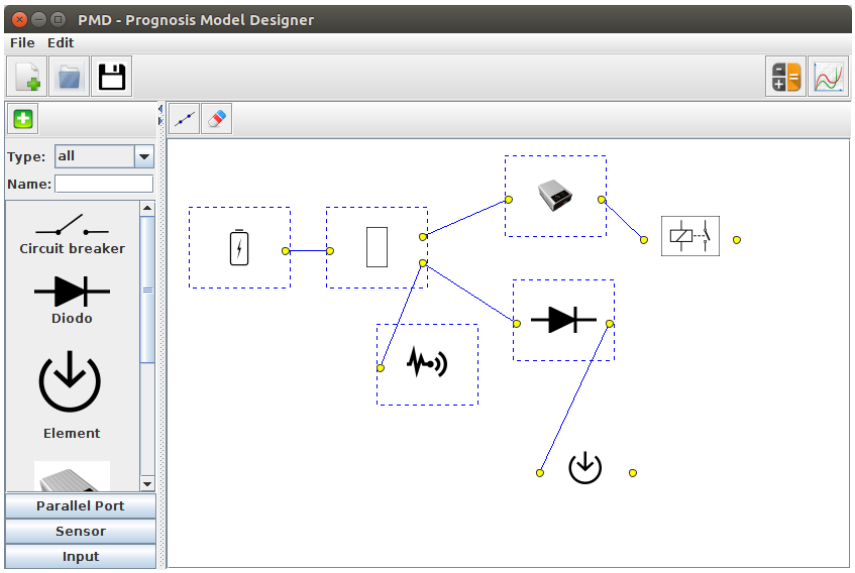


Figura 8.25: Elementos seleccionados

Una vez seleccionados podemos copiarlos y pegarlos, de tal forma que mantendrán sus enlaces. Además podemos copiar y pegar entre varias instancias de la misma aplicación:

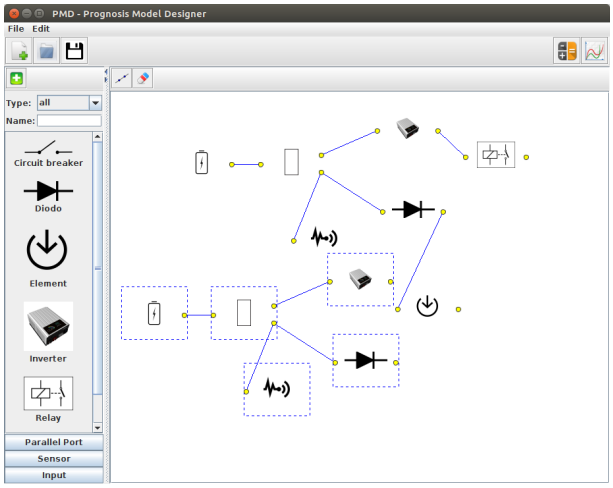


Figura 8.26: Elementos copiados

Los elementos tienen a su vez un menú contextual el cual se muestra al pulsar el botón derecho del ratón sobre el elemento deseado:

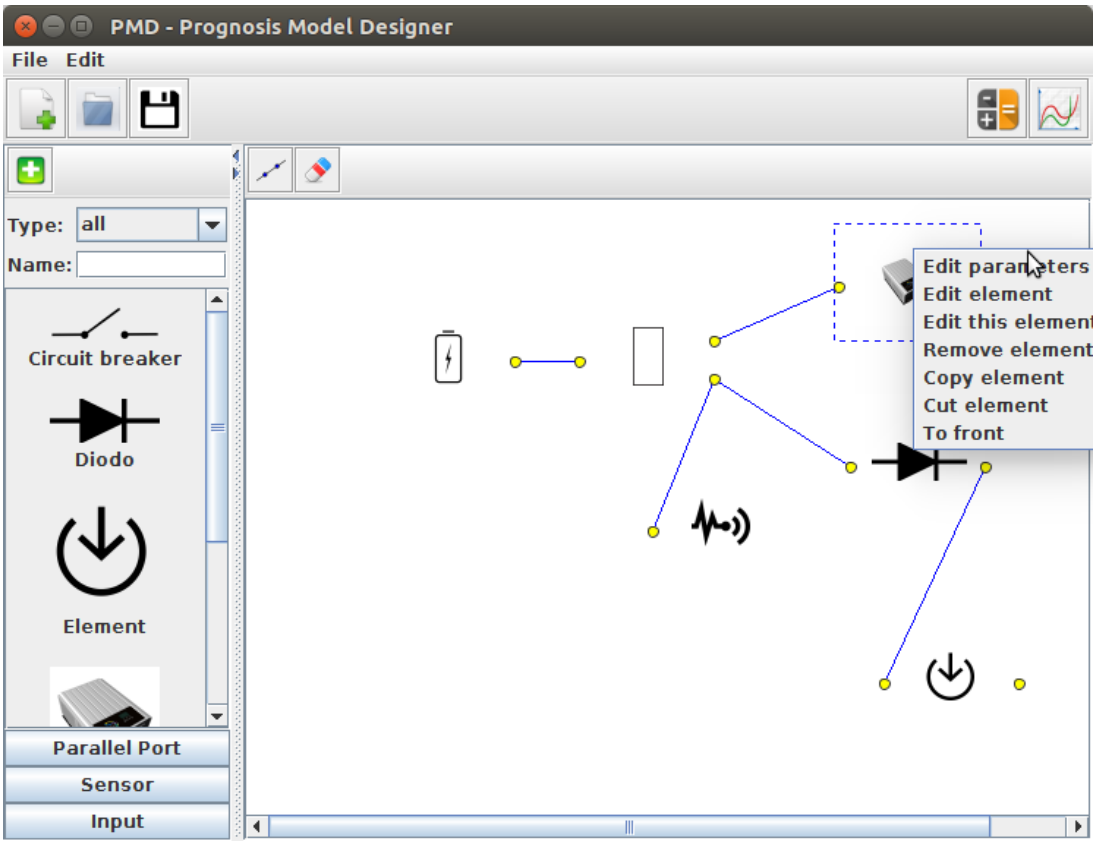


Figura 8.27: Menú contextual elemento

Las diferentes opciones son:

- Edit parameters: permite establecer un valor a los parámetros
- Edit element: permite editar ese elemento de forma global.
- Edit this element: permite editar ese elemento en particular (para casos especiales).
- Remove element: elimina el elemento.
- Copy element: copia el elemento.
- Cut element: corta el elemento.
- To front: trae el elemento al frente, es útil cuando hay dos elementos superpuestos y se quiere traer uno de ellos al frente.

A parte de los elementos con varios modos permiten elegir el modo a través de una lista desplegable:

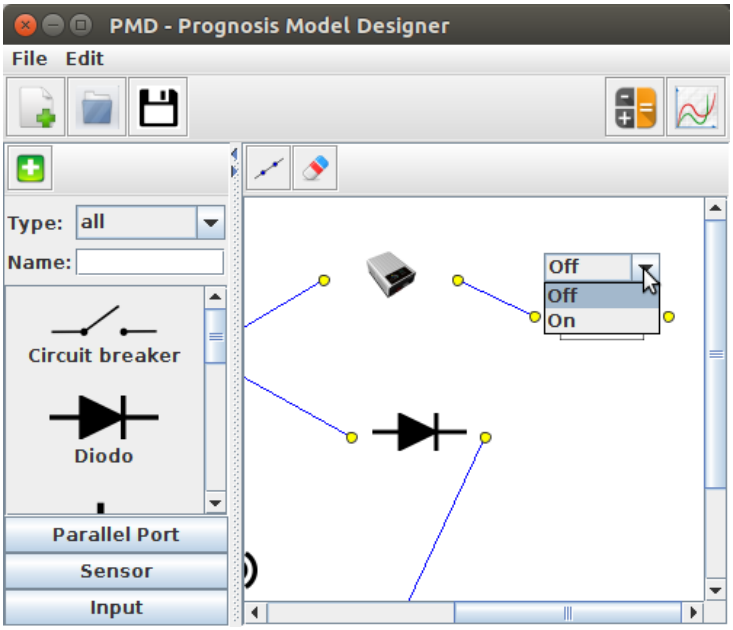


Figura 8.28: Modo elemento

Para guardar el sistema sólo es necesario pulsar sobre File->Save/Save as (o el botón con forma de diskete) y se muestra el siguiente diálogo:

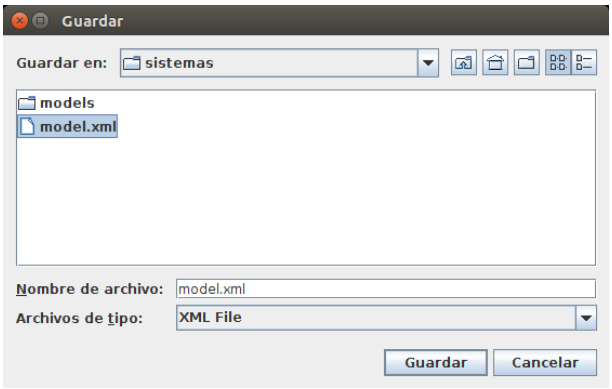


Figura 8.29: Guardar sistema

Y para abrir, File->open o el botón con forma de carpeta:

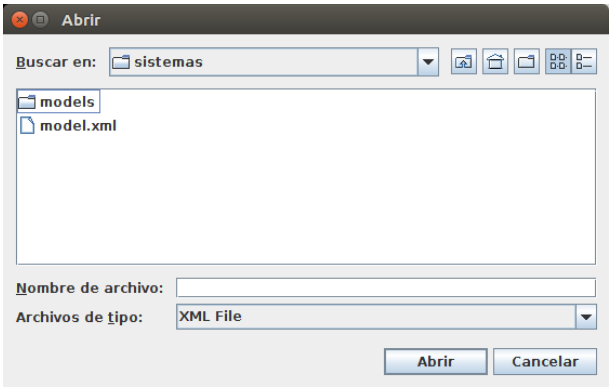


Figura 8.30: Abrir sistema

Por último para generar los modelos hay que pulsar el botón de **Genrate Models** ubicado en la parte superior y con un icono en forma de sistema de ecuaciones. Acto seguido se muestra una ventana indicando dónde queremos guardar los modelos:

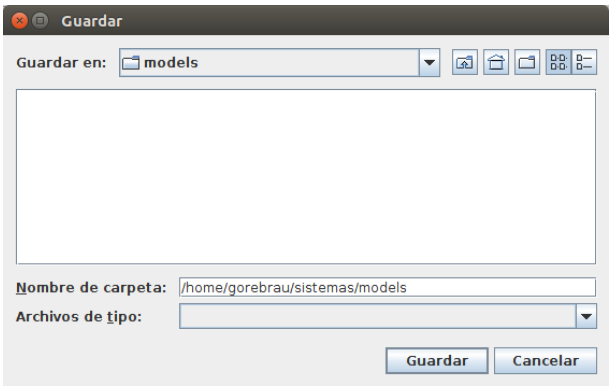


Figura 8.31: Generate models

Con esto se generarán los modelos necesarios para correr las simulaciones:

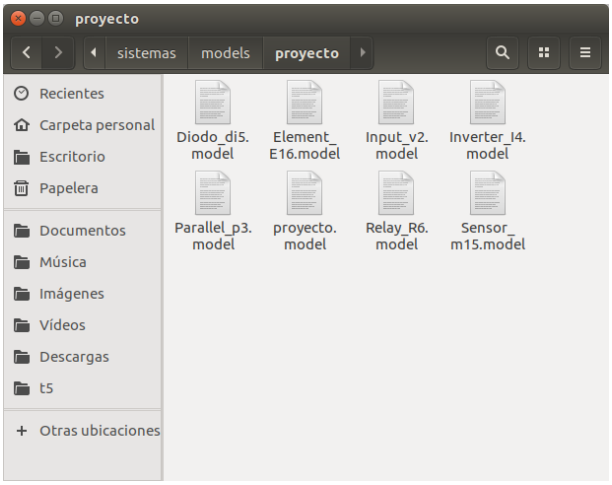


Figura 8.32: Modelos generados

Capítulo 9

Conclusiones

9.1. Conclusiones

Una vez dado por finalizado este proyecto quedo satisfecho por el trabajo realizado, aunque mi naturaleza perfeccionista me indique todas las mejoras que se pueden hacer, la funcionalidad principal de esta aplicación ha sido satisfecha y me siento orgulloso de que su funcionamiento sea correcto.

Durante la realización de este proyecto he tenido la oportunidad de ampliar mis conocimientos sobre el lenguaje de programación Java, ya que siempre he trabajado más con lenguajes diferentes como c o c++. Y mejorar las interfaces de usuario. Además he aprendido a realizar cosas que en un principio consideraba bastante complejas, como ha sido la vinculación de elementos.

Quizás el mayor reto que he tenido ha sido codificar prácticamente toda la aplicación ya que apenas utiliza librerias externas, algo que facilita su ejecución en diferentes sistemas.

9.2. Futuras Mejoras

9.2.1. Añadir el soporte a otros idiomas

Si bien es cierto que el usuario principal al que va enfocada esta aplicación es un usuario más técnico cuyo nivel de inglés va a ser más que suficiente para poder utilizarla, si estaría bien añadir más idiomas para facilitar su comprensión por otras personas.

9.2.2. Permitir compatibilidad con otros sistemas

Estaría bien que la propia aplicación pudiera trabajar con librerías de componentes de otras aplicaciones y/o pudiera cargar modelos generados en otros entornos para poder realizar los cálculos, o exportar estos modelos para que puedan ser interpretados por otros sistemas.

9.2.3. Permitir trabajar con varios modelos

Si bien es cierto que la aplicación se puede abrir tantas veces como sea necesario (no tiene límites de instancias), sí que mejoraría bastante que se incluyera la posibilidad de trabajar con varios proyectos desde la misma instancia de la aplicación.

9.2.4. Realizar las simulaciones

Añadir la funcionalidad de realizar las simulaciones dentro de la aplicación java y no tener que lanzar el script fuera de esta.

Parte I

Apéndices

Apéndice A

Anexos

A.1. Información complementaria

En los anexos se pueden incluir todas aquellas cosas que pueden servir, a nivel teórico o práctico, para ayudar al lector a profundizar en ciertos aspectos de la memoria, como los prerrequisitos teóricos, descripción de alguna herramienta en particular, etc.

A.2. Diagramas y tablas

A.2.1. Casos de uso

Nombre e ID del CU	CU-04 Definir un nuevo tipo de elemento
Actor	Usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario pulsa el botón de definir tipo de elemento en la ventana de creación de elementos.
Precondiciones	
Postcondiciones	POST-1. El conjunto de variables del sistema es guardado en un archivo
Flujo normal	FN1 El actor rellena el formulario facilitado con la información que desea definir. FN2 El actor presiona el botón para guardar el tipo. FN3 El sistema verifica que los datos introducidos son correctos. FN4 Si los datos son correctos, el sistema guarda la información de los tipos en la lista de tipos en formato XML. FN5 El sistema oculta la ventana de crear tipo y añade el tipo creado a la lista de tipos.

Flujo alternativo 1	FA4 Si los datos son incorrectos se informa al usuario del error indicando cuales son los datos incorrectos.
Excepciones	E1 El usuario ha dejado campos requeridos sin rellenar. E2 Ya existe un tipo mismo nombre. E3 Los nombres de las variables son idénticos.
Prioridad	Alta
Otra info	

Tabla A.1: CU-04. Definir un nuevo tipo de elemento

Nombre e ID del CU	CU-05 Comprobación de ecuaciones
Actor	Usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario escriba una ecuación en la ventana de crear elemento.
Precondiciones	
Postcondiciones	POST-1. Determina si la ecuación es correcta o no
Flujo normal	FN1 El actor rellena un campo de fórmula. FN2 El sistema coge esa fórmula y las variables que existan y comprueba que su sintaxis sea correcta. FN3 El sistema determina indica si la fórmula es correcta.
Flujo alternativo 1	
Excepciones	
Prioridad	Alta
Otra info	

Tabla A.2: CU-05 Comprobación de ecuaciones

Nombre e ID del CU	CU-06 Establecer valores variables de elementos
Actor	Usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario indique que quiere modificar una variable de un elemento.
Precondiciones	

Postcondiciones	POST-1. Define el valor de la variable del elemento
Flujo normal	FN1 El actor selecciona un elemento e indica editar variables. FN2 El sistema muestra una ventana de dialogo con los nombres de las variables de los elementos y campos de texto para rellenar. FN3 El usuario rellena los campos. FN4 El sistema guarda el valor de esas variables para ese elemento
Flujo alternativo 1	
Excepciones	
Prioridad	Alta
Otra info	

Tabla A.3: CU-06 Establecer valores variables de elementos

Nombre e ID del CU	CU-07 Generar fórmulas
Actor	Usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario indique que quiere generar las fórmulas según el sistema diseñado.
Precondiciones	
Postcondiciones	POST-1. El sistema genera los archivos de modelo con las fórmulas
Flujo normal	FN1 El actor selecciona la opción de generar modelos. FN2 El sistema comprueba de que haya elementos en la zona de edición. FN3 El sistema muestra una ventana indicando la ubicación donde quiere el usuario que guarde los archivos de fórmulas. FN4 El sistema crea una carpeta con el nombre definido por el usuario y guarda el diagrama en forma de archivos de fórmulas para su posterior procesado.
Flujo alternativo 1	FA2 Si no hay elementos en la zona de edición el sistema informa del error y termina el caso de uso.
Excepciones	
Prioridad	Alta

Otra info	

Tabla A.4: CU-06 Establecer valores variables de elementos

Nombre e ID del CU	CU-08 Cambiar modo de los elementos
Actor	Usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario indique que quiere cambiar el modo de un elemento.
Precondiciones	PRE-1 El elemento tiene que tener más de un modo
Postcondiciones	POST-1. El sistema cambia el modo de ese elemento
Flujo normal	FN1 El actor selecciona el modo dentro del cuadro desplegable del elemento. FN2 El sistema cambia el modo del elemento
Flujo alternativo 1	
Excepciones	
Prioridad	Alta
Otra info	

Tabla A.5: CU-08 Cambiar modo de los elementos

Nombre e ID del CU	CU-09 Editar elemento
Actor	Usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario indique que quiere editar un elemento.
Precondiciones	
Postcondiciones	POST-1. El sistema cambia la información del elemento
Flujo normal	FN1 El actor selecciona editar elemento al pulsar con el botón derecho del ratón sobre un elemento colocado dentro del panel de edición. FN2 El sistema muestra la ventana de crear elementos con los datos del elemento elegido y el campo de nombre deshabilitado. FN3 El usuario modifica los datos que cree conveniente. FN4 El sistema comprueba que los datos sean correctos. FN5 El sistema guarda la información modificada del elemento.

Flujo alternativo 1	
Excepciones	
Prioridad	Alta
Otra info	

Tabla A.6: CU-09 Editar elemento

Apéndice B

Contenido del CD

Dentro del cd se encuentran todos los archivos que componen este TFG, así como todos los documentos generados. La aplicación final es entregada en forma de ejecutable .jar con nombre PMD.jar y el cual se encuentra en la ruta **cd/aplicacion/ejecutable**

Además del ejecutable también se incluye el código fuente de la aplicación, el cual es una carpeta llamada PMD, situada en la ruta **cd/aplicacion/codigofuente**

Este documento de memoria se encuentra en en la ruta **cd/documentacion/** bajo el nombre de memoria.pdf .

La plantilla LaTeX de este documento se encuentra en una carpeta llamada PMD en la ruta **cd/documentacion/plantilla**

Los archivos generados para la documentación se encuentran en la carpeta **cd/anexos/**, siendo:

- **interfaz** los archivos de interfaz diseñados con pencil
- **openproj** el archivo de openproject.
- **diagramas** los archivos de los diferentes diagramas en el formato de starUML
- **imagenes** algunas imágenes generadas con LibreOffice paint.

Bibliografía

[1] P. Bourque y R.E. Fairley. *Guide to the Software Engineering Body of Knowledge, Version 3.0*. ACM-IEEE. 2014.

[2] M. Daigle, A. Bregon e I. Roychoudhury. "A Structural Model Decomposition Framework for Hybrid Systems Diagnosis". En: *Proceedings of the 26nd International Workshop on Principles of Diagnosis*. Paris, France, sep. de 2015.

[3] M. J. Daigle, A. Bregon e I. Roychoudhury. "Distributed Prognostics Based on Structural Model Decomposition". En: *IEEE Transactions on Reliability* 63.2 (2014), págs. 495-510. ISSN: 0018-9529. URL: `http://ieeexplore.ieee.org/document/6782677/`.

[4] *Function Point Languages Table*. Quantitative Software Management. Agosto 2015. URL: `http://www.qsm.com/resources/function-point-languages-table`.

[5] *Java Platform Standard Edition 8 Documentation*. URL: `https://docs.oracle.com/javase/8/docs/index.html`.

[6] R. Pressman. *Ingeniería del Software*. McGraw-Hill, 2014.