UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
GRADO EN INGENIERÍA INFORMÁTICA

DESARROLLO DE UN ENTORNO INTERACTIVO DE VISUALIZACIÓN DE ALARMAS PARA LA OPERACIÓN DE LA RED DE DISTRIBUCIÓN DE ENERGÍA ELÉCTRICA

Francisco Javier Álvarez Sánchez
Tutor: Rosa María Romero Gómez
15/07/2014

Resumen

El presente trabajo se enmarca dentro del contexto de la **operación de la red de distribución de energía eléctrica**. La red de distribución es la parte del sistema de suministro eléctrico cuya función es el suministro de energía desde la subestación de distribución, pasando por líneas de distribución y centros de transformación, hasta los usuarios finales.

Consecuentemente, el propósito principal de esta red es garantizar el suministro a usuarios finales, punto en el que aparece la **labor de operación**, es decir, la supervisión y control de la red de distribución de energía eléctrica por operadores humanos, desde salas de control. Un artefacto de control clave para llevar a cabo esta labor son las **alarmas**, percibidas por los operadores a través de **representaciones visuales o visualizaciones**. Su función principal es advertir a los operadores acerca de una condición que se desarrolla cuando el proceso de control se desvía significativamente del modo normal de operación.

Sin embargo, en numerosas ocasiones, estas visualizaciones resultan más una carga que una ayuda desplegando volúmenes ingentes de alarmas, lo cual ha sido conceptualizado como "*alarm flooding*" ("inundaciones de alarmas"). Antes estas situaciones, el operador no es capaz de procesar la información que recibe y por consiguiente, de tomar decisiones efectivas. Esta problemática puede desembocar en situaciones de operación problemáticas e incluso en apagones eléctricos los cuales poseen un impacto altamente negativo en la actividad socio-económica de un país.

La solución adoptada para la resolución de esta problemática en el presente trabajo fin de grado es un entorno interactivo de visualización de alarmas para la operación de la red de distribución de energía eléctrica. En concreto, las contribuciones de este proyecto consisten en un entorno que permite la exploración tanto de volúmenes pequeños de alarmas como de volúmenes grandes a través de los distintos elementos eléctricos que componen la infraestructura de red mediante una vista general de las alarmas registradas en la infraestructura de red a través de un mapa que permita situar espacialmente los elementos causantes de las incidencias, y un conjunto de vistas detalladas que están organizadas de acuerdo a los tres elementos eléctricos principales que componen esta red. Además, el entorno permite la extracción de tendencias espacio-temporales sobre las alarmas a fin de ayudar en la detección de problemas en la infraestructura de red mediante las vistas detalladas, que consisten en una serie de gráficos que muestran las alarmas de manera espacio-temporal, de los tres elementos eléctricos principales, a distintos niveles de detalle e interconectados entre ellos mediante técnicas de interacción.

Abstract

This work is framed within the context of the **electric distribution network operation**. An electric distribution network is the part of the electric power system that delivers electric power from distribution substations, through distribution feeders and transformers to the end users or consumers.

Consequently, the main purpose pursued by this network is to ensure the delivery of electricity to end users, which is supervised and controlled by electric power operators from control rooms. Within this context, **alarms** have become key control artifacts for carrying out this operation, which are presented to the operators through **alarm system displays**. The principal function of these **alarm system displays** is to warn operators about a condition that develops when the control process deviates significantly from the normal operation.

However, in many cases, these displays are more a burden than help, displaying large volumes of alarms, which has been formally conceptualized as "alarm flooding". To these situations, the operator is not able to process all this alarm information, and therefore to make effective decisions. This problem can lead to problematic operating inefficiencies and even critical operating situations, which can have a highly negative impact on the socio- economic activity of a country.

The adopted solution to solve this problem in this final thesis is an alarm interactive visualization environment for the electric distribution network operation. Specifically, the contribution of this project is an environment that enables the exploration of both small and large alarms volumes through distinct electrical elements of the network infrastructure with a general view of alarms registered in the infrastructure through a map that allows spatially locate elements causing incidents alarms, and a set of detailed views that are organized according to the three main electrical components of this network. In addition, the environment allows the extraction of spatial and temporal trends on alarms to help detecting problems in the network infrastructure by the detailed views, which are series of graphs showing the alarms in a space-time way, of the three main electrical elements at different levels of detail and interconnected by interaction techniques.

Índice de contenidos

Índice de figuras

Índice de tablas

Glosario de términos y acrónimos

Catálogo de términos y acrónimos específicos del contexto del trabajo.

- **Alarma**: artefacto de control clave para llevar a cabo la labor de operación de la red de distribución de energía eléctrica.
- **API**: *Application Programming Interface* (Interfaz de Programación de Aplicaciones). Conjunto de funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro software en una capa superior.
- **CPU**: *Central Processing Unit* (Unidad Central de Procesamiento). Es el componente principal del ordenador y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.
- **C#**: es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, diseñado para la infraestructura de lenguaje común.
- ***Framework***: estructura conceptual y tecnológica de soporte definido, con el fin de que un proyecto software pueda ser organizado y desarrollado.
- ***Grid***: componente de las tecnologías web de Microsoft que define un área de cuadrícula flexible que está compuesta de columnas y filas.
- **HTML5**: *HyperText Markup Language* (lenguaje de marcas de hipertexto), versión 5 es la quinta revisión importante del lenguaje básico de la *World Wide Web*, HTML. HTML hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, etc.
- **Interfaz**: la interfaz de usuario, para ser más específicos, es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.
- **Librería**: es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.
- ***Mockup***: en la manufactura y diseño, un *mockup*, o maqueta, es un modelo a escala o tamaño real de un diseño o un dispositivo, utilizado para la demostración, evaluación del diseño, promoción, y para otros fines. Un *mockup* es un prototipo si proporciona al menos una parte de la funcionalidad de un sistema y permite pruebas del diseño.
- **MVVM**: patrón arquitectónico que separa la lógica empresarial, la interfaz de usuario y el comportamiento de la presentación.
- **Operador**: persona que garantiza la labor de operación de la red de distribución de energía eléctrica desde salas de control.
- **Prototipo**: primer ejemplar de alguna cosa que se toma como modelo para crear otros de la misma clase.
- **Red de distribución de energía eléctrica**: es la parte del sistema de suministro eléctrico cuya función es el suministro de energía desde la subestación de distribución,

pasando por líneas de distribución y centros de transformación, hasta los usuarios finales.

- **Visualización**: representación visual de datos abstractos que busca reforzar la cognición humana.
- **WPF**: *Windows Presentation Foundation* es una tecnología de Microsoft, presentada como parte de Windows Vista. Permite el desarrollo de interfaces de interacción en Windows tomando características de aplicaciones Windows y de aplicaciones web.
- **XAML**: *e**X**tensible **A**pplication **M**arkup **L**anguage* (Lenguaje Extensible de Formato para Aplicaciones) es el lenguaje de formato para la interfaz de usuario para WPF y Silverlight, el cual es uno de los "pilares" de la interfaz de programación de aplicaciones .NET en su versión 3.0.

1 Introduction

This work is framed within the context of the **electric distribution network operation**. An electric distribution network is the part of the electric power system that delivers electric power from distribution substations, through distribution feeders and transformers to the end users or consumers. Figure 1 shows a diagram of this distribution network:

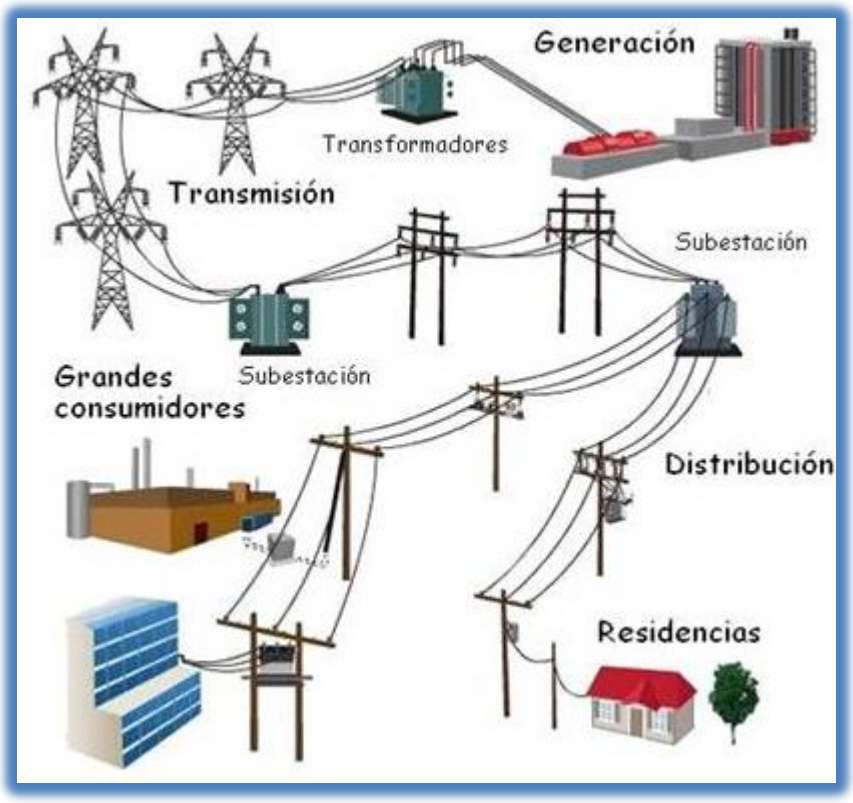


Figure 1. Electric distribution network

Consequently, the main purpose pursued by this network is to ensure the delivery of electricity to end users, which is supervised and controlled by electric power operators from control rooms (hereafter, referred to as operators). Within this context, **alarms** have become key control artifacts for carrying out this operation, which are presented to the operators through **alarm system displays**. The principal function of these **alarm system displays** is to warn operators about a condition that develops when the control process deviates significantly from the normal operation [25].

1.1 Problem definition

The electric distribution network operation aims to ensure that the distribution network is running in normal operating mode. This mode of operation refers to the absence of significant deviations from network performance of the predefined thresholds. With this goal in mind,

operators make use of the alarm system displays, which have become key control devices for this operation labour. In particular, due to the increasing complexity of distribution network, where various parameters from many geographically dispersed components should be supervised, these displays help operators to identify the causes of the problems encountered.

However, in many cases, these displays are more a burden than help, displaying large volumes of alarms [14], which has been formally conceptualized as "alarm flooding". To these situations, the operator is not able to process all this alarm information, and therefore to make effective decisions. This problem can lead to problematic operating inefficiencies and even critical operating situations, which can have a highly negative impact on the socio-economic activity of a country.

1.2 Objectives

The main purpose of this work is the development of an interactive alarm visualization environment for the electric **distribution network operation** that addresses the previously defined problem. This main purpose can be subdivided into the following objectives:

1. This environment should allow the exploration of both small volumes and large volumes of alarms registered on the various elements of the distribution network.
2. This environment should allow the extraction of alarm trends to assist in to identifying the causes of the problems encountered.

1.3 Development phases

The completion of the work involves a number of specific stages of development, which are discussed below.

State of the art

This phase studies the context in which the problem is framed. It reviews both the main alarm system display formats and different technologies in order to select the most suitable to develop this environment.

Problem definition

This phase examines the shortcomings of the alarm system display formats to establish the problem to be addressed by this work.

Solution

At this stage, the solution is developed, identifying its essential features as well as its detailed characteristics.

Evaluation

At this stage, an evaluation to demonstrate the validity of the solution is conducted. It must demonstrate that this solution meets the requirements defined in the previous phase.

1.4 Resources employed

The resources used to develop this work can be divided into two main categories: *literature sources* which have provide the necessary understanding of the technologies used and the alarm display formats, and *development tools*, which have been applied to develop the solution.

The literature sources used were as follows:

- Google scholar.
- University Carlos III of Madrid's library. Escuela Politécnica Superior.

The development tools used were:

- Computers:
  - Personal computer.
  - Laptop.
- Microsoft Visual Studio Professional 2012. Version 11.0.61030.00 Update 4.
- Complementary libraries to Visual Studio:
  - Esri.arcgis.client.
  - Esri.arcgis.client.toolkit.
  - UtilsDEISilverlight.dll.

1.5 Document structure

This document includes a **first chapter** that aims to introduce the work done and the document content.

The **second chapter**, the status of the art, establishes the context in which the problem is framed in this final thesis, reviewing the main alarm system display formats existing nowadays and the technologies that will develop the solution.

The **third chapter**, software project management, reflects the project scope, work plan, resource management, risk management and test plan.

The **fourth chapter**, solution, details the solution conducted, describing the adopted type and the development process.

The **fifth chapter**, evaluation, demonstrates the validity of the developed solution, checking through tests and/or questionnaires, if the problems are resolved and if the objectives exposed in the introduction are satisfied.

The **sixth chapter**, conclusion, provide a summary of work and a personal view of it.

The **seventh chapter**, bibliography, lists references used in the performance of work.

The **eighth and final chapter** contains several annexes on version control, track work order level, prototype and requirements specification.

2 El estado de la cuestión

El objetivo de este apartado reside en establecer el contexto en el que se enmarca el problema a abordar en el presente trabajo fin de grado. Con este propósito, en primer lugar, se revisan los principales formatos de visualización de alarmas existentes en la actualidad, a fin de establecer sus carencias. En segundo lugar, se realiza una revisión de tecnologías para decidir cuál será la seleccionada para elaborar el desarrollo de la solución.

2.1 Revisión teórica de los principales formatos de visualización de alarmas

Una visualización de alarmas puede ser definida como "un método o métodos por los cuáles las alarmas y los mensajes asociados a las mismas son presentados a los operadores de una sala de control" [24]. Como fundamento de esta revisión, se toman las tres categorías de formatos de visualización de sistemas de alarmas para el control de procesos definidas por Stanton y Baber [15], y más recientemente consideradas por Micah Endsley en su teoría acerca de la Consciencia Situacional ("*Situation Awareness*") [14]. Estas categorías incluyen **listados de alarmas** ("*serial alarm displays*", véase la Ilustración 2), **paneles anunciadores de alarmas** ("*annunciator-based alarm displays*", véase la Ilustración 3) y **paneles sinópticos de alarmas** ("*mimic alarm displays*", véase la Ilustración 4). Las ilustraciones incluidas a los largo del apartado muestran un ejemplo de cada una de ellas.

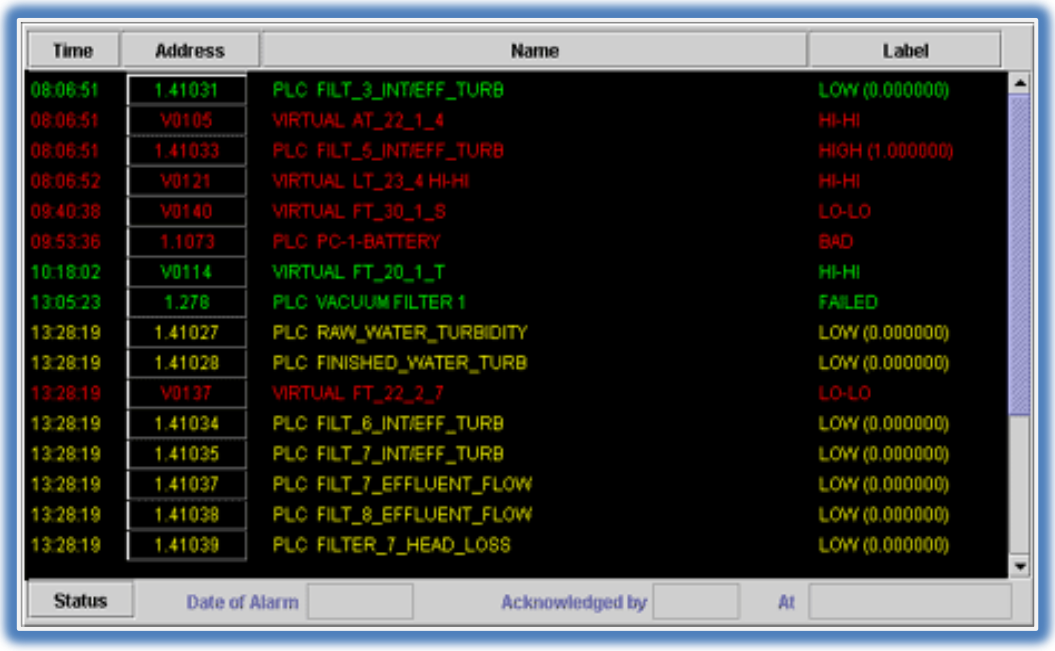


Ilustración 2. Ejemplo de un listado de alarmas

En los listados de alarmas (Ilustración 2), cada alarma se representa tradicionalmente mediante una fila en una lista ordenada de alarmas y una serie de diferentes atributos que se dividen en diferentes columnas. Estos listados se han caracterizado por ser altamente útiles a la

hora de visualizar el orden en que se produjeron las alarmas, pero son considerados como problemáticos por los operadores cuando se registran un gran número de ellas [15][16][17]. En particular, sufren un grave problema conocido como *"alarm flooding"* ("inundaciones de alarmas"); entendiendo *"alarm flooding"* como "*una situación en la que se registra tal número de alarmas en un corto período de tiempo que el operador es incapaz de procesarlas*" [17]. Durante estas situaciones, los únicos mecanismos para hacer frente a tal volumen de alarmas son la paginación y el uso de mecanismos de *scroll*. Sin embargo, aunque estos mecanismos son considerados como un estándar en casi todas las interfaces de usuario, introducen discontinuidad entre la información que ha sido registrada en diferentes períodos de tiempo [26]. Lamentablemente, esta falta de continuidad puede causar altas cargas cognitivas para el operador que debe asimilar mentalmente la estructura general del espacio de información de alarmas [16][14][18].

Ilustración 3. Ejemplo de un panel anunciador de alarmas

Los paneles anunciadores de alarmas (Ilustración 3) muestran varios tipos de alarmas de una sola vez, por lo general, en un panel o pantalla [15]. Estos paneles ayudan a los operadores a percibir altos volúmenes de alarmas al mismo tiempo. Como la misma alarma suele estar representada en el mismo lugar, también favorecen que los operadores extraigan patrones de alarmas [14]. De esta manera, con el tiempo, el operador puede llegar a asociar ciertos patrones de alarmas con ciertos tipos de fallo, por lo que los fallos frecuentes y familiares es probable que sean fácilmente identificables. Sin embargo, como desventaja, no parecen guiar al operador en el descubrimiento de fallos nuevos y poco frecuentes, debido a la manera en que la información está representada [15]. Por otra parte, si ocurren diversos problemas al mismo tiempo, puede resultar difícil saber cuál es el nuevo dispositivo o sistema que está registrando una nueva alarma [14].

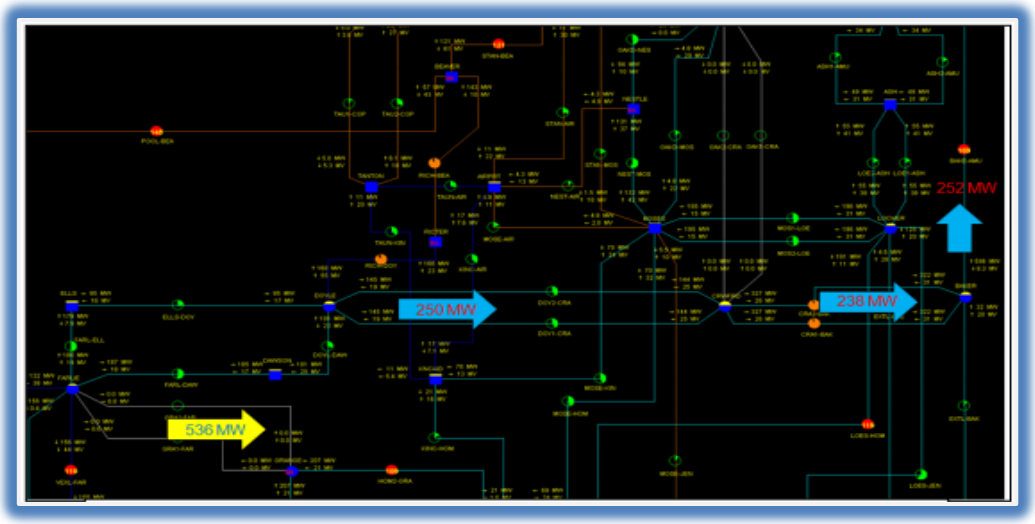


Ilustración 4. Ejemplo de un panel sinóptico de alarmas

Finalmente, los paneles sinópticos de alarmas (Ilustración 4) intentan mostrar las alarmas sobre un patrón que se corresponde con una representación pictórica del sistema subyacente [14]. Estos paneles tienen la ventaja de proporcionar un mapeo espacial más directo entre la sala de control y el proceso controlado [15][14]. En particular, mediante el uso de líneas, iconos, etiquetas y colores, tales paneles ayudan a los operadores a determinar cómo un patrón particular de alarmas puede estar relacionado con determinadas condiciones de fallo. Sin embargo, si surgen varias alarmas, se enfrentan a problemas similares a los de las cuestiones de superposición de información de los paneles anunciadores de alarmas. Es más, a través de estos paneles no hay manera de determinar el orden en que se produjeron dichas alarmas.

Consecuentemente, esta revisión muestra que, aunque adecuados para soportar las tareas de gestión de alarmas, tales como tareas de ordenación temporal, tareas de concordancia de patrones o tareas espaciales, respectivamente [15][14], estos formatos de visualización de alarmas poseen algunas deficiencias. En particular, estas visualizaciones permanecen estáticas en lugar de considerar el carácter dinámico del proceso controlado, tales como las variaciones en los volúmenes de alarmas registrados o los objetivos que posee el operador. Del mismo modo, no conservan la continuidad entre la información registrada en diferentes períodos de tiempo, lo que a menudo introduce una elevada carga cognitiva para el operador. Por lo tanto, teniendo en cuenta que los seres humanos sólo pueden gestionar las alarmas de forma fiable si el número es limitado y si el proceso controlado se encuentra razonablemente estable [25], una de las consecuencias de estas limitaciones es que los operadores no son capaces de proyectar condiciones futuras de explotación. Como resultado, esta falta de proyección puede conducir a ineficiencias operativas o incluso a situaciones críticas como apagones eléctricos.

2.2 Elección de la tecnología de desarrollo

El objetivo de este apartado reside en decidir la tecnología con la que se va a implementar la solución. El objetivo principal del trabajo, como ya se ha especificado anteriormente, reside en desarrollar un entorno de visualización de alarmas interactivo para la operación de la red de distribución de energía eléctrica. Por lo tanto, la tecnología seleccionada tiene que proveer altas capacidades de interacción y visualización. Además, teniendo en cuenta el carácter de prueba de

concepto de la solución, sin una implantación inmediata en un entorno de control real, la tecnología seleccionada debe permitir el seguimiento de su desarrollo. Teniendo en cuenta estos requisitos, las tecnologías web han sido caracterizadas como las más adecuadas en esta situación. En concreto, se va a realizar una comparativa entre diferentes tecnologías web de gran uso y aplicación como son *Silverlight*, *Adobe Flash* y *JavaFX*.

La estructura de este apartado es la siguiente. En primer lugar se definen los requisitos a cubrir por la tecnología seleccionada para, a continuación, señalar las características principales de cada tecnología junto con una tabla que indique cómo cubre dicha tecnología cada requisito; en última instancia se selecciona la tecnología con la que se va a desarrollar el trabajo, justificando dicha elección.

Requisitos a cubrir por la tecnología

A continuación se establecen una serie de requisitos fundamentales que han de ser cubiertos, en la medida de lo posible, por la tecnología seleccionada:

- **Multiplataforma:** la tecnología seleccionada tiene que permitir un despliegue independiente de la plataforma para que no haya restricciones a la hora de la implantación. En este caso, al tratarse de una tecnología web, lo que se busca es una independencia de navegador web. Se valorará positivamente una independencia completa.
- **Capacidades gráficas y de interacción:** se trabajarán con múltiples gráficos y/o imágenes de forma simultánea. Se valorará positivamente que le tecnología provea integración con mapas, componentes visuales e interacción rica en la interfaz.
- **Material de ayuda y referencias (APIs, librerías, frameworks):** la tecnología seleccionada debe proveer un adecuado material de ayuda y referencias. En caso de que el acceso sea complicado, se debe garantizar al menos un bajo coste. Además, se debe contar con todo el material posible que ayude al desarrollo y cuyo aprendizaje para el uso sea lo más sencillo posible, entre el que se debe incluir todo tipo de documentación, tanto de fuentes oficiales (libros y artículos de investigadores profesionales) como de no oficiales (foros de consulta en la web). Se valorará positivamente la mayor cantidad de material disponible así como su calidad y fiabilidad.
- **Curva de aprendizaje:** la tecnología seleccionada debe facilitar el tiempo necesario para aprender los lenguajes de programación empleados tanto a alto como a bajo nivel, es decir, los que definen la interfaz de usuario y el comportamiento de los objetos (en el caso de que sean lenguajes distintos). El tiempo necesario también para familiarizarse con el entorno de desarrollo y con las herramientas complementarias. Se valorará positivamente el menor tiempo posible de aprendizaje y/o el conocimiento previo de los lenguajes.
- **Licencia de software:** es deseable que la tecnología, tanto el programa base como las extensiones o herramientas, sea de libre acceso para los usuarios, tanto para los desarrolladores como los que van a hacer uso de su disfrute, aspecto que se valorará positivamente para su elección.

Principales características de las tecnologías y adaptación a los requisitos a cubrir

En este apartado se describen las principales características de las tecnologías seleccionadas, acompañadas de una tabla que indica cómo cubren los requisitos definidos en el apartado anterior.

Silverlight

Microsoft Silverlight es una herramienta para crear y desarrollar aplicaciones de Internet enriquecidas y aplicaciones Web con funciones multimedia como la reproducción de vídeos, gráficos vectoriales, animaciones e interactividad. La base de su programación es XAML y el acceso a los objetos está dado por C# y VisualBasic. La primera versión de Silverlight fue lanzada en septiembre de 2007 por Microsoft y actualmente su versión 5.0 se distribuye de forma gratuita [3][4].

Multiplataforma	Compatible con todos los navegadores en Windows y Mac OS. También compatible con Linux a través de Moonlight.
Capacidades gráficas y de interacción	Provee integración con mapas, componentes visuales e interacción rica en la interfaz.
Material de ayuda y referencias	Amplio, tanto de fuentes oficiales como no oficiales. Al ser una tecnología con menos recorrido en el mercado, en comparación con competidores directos, la calidad del material es de un grado menor.
Curva de aprendizaje	No se cuenta con experiencia previa en el entorno de desarrollo ni en los lenguajes de programación. La curva de aprendizaje tiene una gran pendiente, aunque luego el desarrollo es muy estable y muy rápido.
Licencia de software	Libre y gratuita.

Tabla 1. Requisitos cubiertos por Silverlight

Adobe Flash

Adoble Flash es una aplicación de creación y manipulación de gráficos vectoriales con posibilidades de manejo de código mediante un lenguaje de scripting llamado ActionScript. Los archivos reproducibles de Adobe Flash pueden aparecer en un navegador web, como animaciones en sitios web multimedia, y más recientemente en Aplicaciones de Internet Ricas. La base de su programación es C++. La primera versión de Flash fue desplegada en 1996 por Macromedia y actualmente Adobe distribuye la última versión de forma gratuita [5][6].

Multiplataforma	Compatible con todos los navegadores en Windows, Mac OS y Linux. No compatible con dispositivos Apple.
Capacidades gráficas y de interacción	Provee componentes visuales e interacción rica en la interfaz, pero no permite integrar mapas.
Material de ayuda y referencias	Amplio, tanto de fuentes oficiales como no oficiales. Calidad garantizada gracias al largo recorrido de la tecnología en el mercado.
Curva de aprendizaje	No se cuenta con experiencia previa en el entorno de desarrollo ni en los lenguajes de programación. La curva de aprendizaje tiene una gran pendiente, aunque luego el desarrollo es muy

	estable y muy rápido [6].
Licencia de software	Libre y gratuita.

Tabla 2. Requisitos cubiertos por Adobe Flash

JavaFX

JavaFX es una tecnología dedicada a la creación de Aplicaciones de Internet Ricas, incluyendo aplicaciones multimedia interactivas. Al estar la mayoría de lenguajes script orientados a las páginas web, JavaFX surge para competir con Flash y Silverlight, centrándose en el desarrollo de interfaces altamente animadas. La base de su programación es Java. Sun Microsystems desplegó la primera versión en 2007, y en 2008 ya lo hizo de forma libre [7].

Multiplataforma	Compatible con todos los navegadores en Windows, Mac OS y Linux.
Capacidades gráficas y de interacción	Provee componentes visuales de baja calidad, interacción rica en la interfaz pero más limitada y no permite integrar mapas.
Material de ayuda y referencias	Amplio, tanto de fuentes oficiales como no oficiales. Calidad garantizada gracias al largo recorrido de la tecnología en el mercado.
Curva de aprendizaje	Se cuenta con gran experiencia en el entorno de desarrollo y en el lenguaje de programación.
Licencia de software	Libre y gratuita.

Tabla 3. Requisitos cubiertos por JavaFX

2.3 Justificación de la tecnología seleccionada

Teniendo en cuenta la revisión previa, la tecnología elegida para la implementación de la solución es **Silverlight**, seguida muy de cerca por Adobe Flash. **JavaFX** ha sido descartada en seguida debido al poco seguimiento que tiene de usuarios y de desarrolladores. No es una tecnología que disponga del potencial de las otras dos en términos gráficos y de interacción, no ofrece tanto material de ayudas o referencias, ni tiene muchas expectativas en el mercado a día de hoy, aparte de contar con una cantidad muy limitada de recursos visuales, punto fundamental a cubrir.

Con respecto a **Flash**, esta tecnología es compatible con todos los navegadores pero no es compatible con dispositivos Apple, ni tiene expectativas de hacerlo en un futuro debido a su compromiso con HTML5. Flash cuenta con infinidad de librerías que apoyan al desarrollo además de muchos expertos y publicaciones a los que acudir. Desde el punto de vista de los recursos visuales, Flash aporta bastante material, pero no cuenta con integración con mapas, requisito fundamental que se ha de cumplir. Desde un punto de vista menos técnico, Flash cuenta con más años de experiencia, pero se encuentra en un estado en el que comienza un ligero declive que le puede llevar a declararse obsoleta.

Con respecto a **Silverlight**, esta tecnología no ofrece compatibilidad con la totalidad de navegadores. Aun así, Silverlight cuenta con *Moonlight* para dar soporte a Linux. Silverlight cuenta con una cantidad de librerías y de publicaciones a la altura de Flash, pero la gran ventaja de esta tecnología reside en los recursos visuales que aporta: integración con mapas, componentes visuales e interacción rica en la interfaz. Teniendo en cuenta las características del presente trabajo, estas capacidades han sido determinantes a la hora de seleccionar esta tecnología para el

desarrollo del entorno. Desde un punto de vista menos técnico, Silverlight ofrece buenas expectativas de futuro, ganándose cada vez más la confianza de los desarrolladores, facilitando así la garantía de poder ofrecer tanto un servicio actual al cliente como un soporte en el futuro para solucionar problemas y evolucionar el producto.

3 Gestión de proyecto software

El objetivo de este apartado es realizar una simulación de la gestión del proyecto software desarrollado. Esta gestión consiste en una estimación hipotética del proyecto, como si de uno real se tratase, realizado con las condiciones habituales del entorno empresarial. El proyecto ha tenido una duración de 10 meses, desde septiembre de 2013 hasta junio de 2014. En los apartados siguientes se incluye una estimación de los recursos empleados durante estos meses así como del presupuesto final del proyecto; además, se incluye la planificación seguida para el desarrollo del mismo. Esta planificación está detallada en el Anexo II. Seguimiento del trabajo fin de grado.

3.1 Estimación de tareas y recursos

En este apartado se presentan los costes asociados al desarrollo del proyecto. Los costes que se tienen en cuenta son los siguientes: costes de personal y costes de hardware y software.

Costes de personal

Los costes de personal incluyen los salarios del equipo participante en el proyecto. En primer lugar, se definen los roles que los empleados han tenido asignados [19]:

- **Jefe de proyecto**: es la persona que administra y controla los recursos asignados a un proyecto, con el propósito de que se cumplan correctamente los planes definidos.
- **Analista**: es la persona que trabaja con el cliente para realizar el análisis y especificación del sistema, es decir, dividir el problema a resolver en sub-problemas de menos complejidad.
- **Diseñador**: es la persona encargada de generar el diseño arquitectónico del sistema, el diseño detallado y los prototipos del sistema.
- **Programador**: es la persona encargada de convertir la especificación del sistema en código fuente ejecutable utilizando uno o más lenguajes de programación, así como herramientas de software de apoyo a la programación.

Por tanto, este equipo contará con suficiente personal como para cubrir los roles designados. Dependiendo del rol, la jornada de trabajo varía, así como su salario: el jefe de proyecto trabajará media jornada al estar involucrado generalmente en más proyectos; un analista también trabajará media jornada debido a que sus tareas solo son requeridas durante una parte del proyecto; los diseñadores y programadores trabajarán jornadas completas. La Tabla 4 recoge los salarios mensuales de cada empleado en función de su rol. El salario bruto mensual se calcula de la siguiente forma:

$$Salario\ bruto\ mensual = \frac{coste}{hora} \times \frac{horas}{día} \times \frac{días}{mes}$$

Empleado	Coste / hora (€)	Horas / día	Días / mes	Salario bruto mensual (€)
Jefe de proyecto	60	4	22	5280
Analista	35	4	22	3080
Diseñador	16	8	22	2816
Programador	16	8	22	2816

TOTAL (€)	13992

Tabla 4. Salario bruto mensual del equipo de trabajo

Tras calcular los salarios mensuales de los empleados, se calculan los costes asociados a los 10 meses del proyecto, teniendo en cuenta la cotización a la seguridad social, cuyo porcentaje es del 23,6%. La Tabla 5 muestra los datos exactos. El coste total por empleado y rol se calcula de la siguiente manera:

$$Coste\ total = (salario\ bruto\ mensual + cuota\ SS) \times meses$$

Empleado	Salario bruto mensual (€)	Cuota seguridad social (€)	Meses de trabajo	Coste total (€)
Jefe de proyecto	5280	1246,08	10	65260,80
Analista	3080	726,88	2	7613,76
Diseñador	2816	664,576	8	27844,608
Programador	2816	664,576	8	27844,608
TOTAL (€)			128563,776	

Tabla 5. Costes totales de personal

Costes de hardware y software

Los costes de hardware y software incluyen aquellos asociados a los recursos hardware y software empleados para el desarrollo del proyecto.

Los recursos hardware empleados han sido los siguientes:

- Ordenador de sobremesa *HP Pavilion*.
- Monitor *Packard Bell* para el ordenador de sobremesa *HP*.
- Ordenador portátil *HP Pavilion dv6*.
- Ordenador de sobremesa *Mac Pro Quad-Core Intel Xeon* 3.2GHz.
- Monitor *Ostendo Curve* 43”.
- Servidor *PowerEdge* R515.

Los recursos software empleados han sido los siguientes:

- Entorno de desarrollo Microsoft Visual Studio 2012.
- Microsoft Office 2010.

La Tabla 6 muestra los datos exactos de los costes de los productos sin IVA:

	Recurso	Coste (€)	Unidades	Coste total (€)
Hardware	**Sobremesa HP**	450	1	450
	Monitor PB	50	1	50
	Portátil HP	550	1	550
	Mac	650	1	650
	Monitor Curvo	6500	1	6500
	Servidor	1049	1	1049
Software	**Visual Studio**	603,31	1	603,31

	Office	355,38	2	710,76
TOTAL (€)			10563,07	

Tabla 6. Costes de hardware y software

Presupuesto

Tras realizar el desglose de costes de personal y de recursos hardware y software, el coste final asociado al proyecto aparece indicado en la Tabla 7.

Concepto	Coste (€)
Costes de personal	128563,78
Costes de hardware y software	10563,07
TOTAL	139126,85

Tabla 7. Costes totales

Para finalizar este apartado, la Tabla 8 muestra el presupuesto final del proyecto, teniendo en cuenta margen de riesgos, beneficios y el IVA:

Concepto	Coste (€)
Costes totales	139126,85
Riesgos (10%)	13912,69
Beneficios (20%)	27825,37
TOTAL (sin IVA)	180864,91
IVA (21%)	37981,44
TOTAL	218846,35

Tabla 8. Presupuesto final

El presupuesto total de este proyecto asciende a la cantidad de DOSCIENTOS DIECIOCHO MIL OCHOCIENTOS CUARENTA Y SEIS CON TREINTA Y CINCO CÉNTIMOS (218846,35 €).

Leganés a 12 de Abril de 2014

El ingeniero proyectista

Fdo. Francisco Javier Álvarez Sánchez

4 Solución

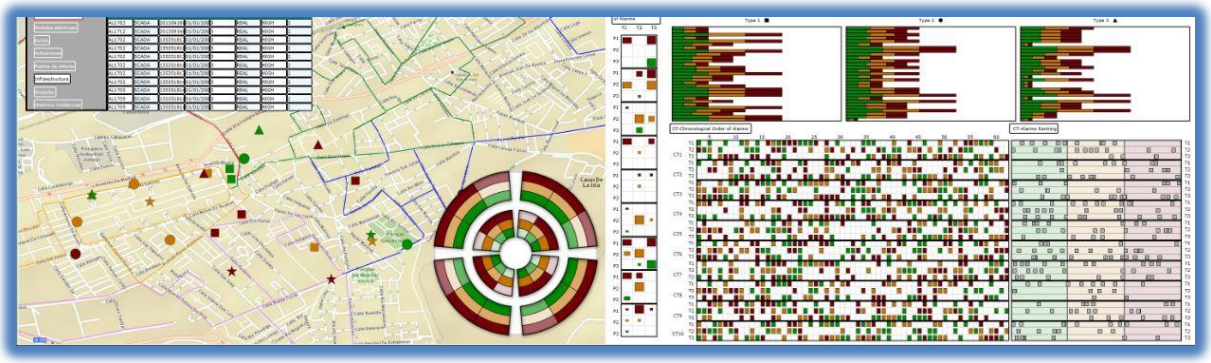
El objetivo de este apartado consiste en presentar la solución propuesta al problema expuesto en el apartado Problem definition. El apartado está dividido en dos sub-apartados: el primero está orientado a la descripción de la solución, exponiendo la forma en la que se abordan los objetivos principales del trabajo presentados en el apartado Objectives; el segundo apartado se focaliza en el proceso de desarrollo, detallando las fases de desarrollo realizadas para la creación del entorno interactivo de visualización de alarmas: la fase de análisis, la fase de diseño y la fase de implementación.

4.1 Descripción de la solución

La solución adoptada es un entorno interactivo de visualización de alarmas para la operación de la red de distribución de energía eléctrica que permita resolver la labor de operación por operadores humanos. En concreto, la solución se trata de una aplicación web, es decir, es una herramienta que los usuarios pueden utilizar accediendo a un servidor web a través de Internet mediante un navegador.

A continuación se recuerdan los objetivos del trabajo expuestos en el apartado Objectives y se indica cómo se han cubierto con el entorno:

1. *El entorno debe permitir la exploración tanto de volúmenes pequeños de alarmas como de volúmenes grandes a través de los distintos elementos eléctricos que componen la infraestructura de red.* Esto se cubre con una vista general de las alarmas registradas en la infraestructura de red mediante un mapa que permita situar espacialmente los elementos causantes de las incidencias, y un conjunto de vistas detalladas que están organizadas de acuerdo a los tres elementos eléctricos principales que componen esta red: *subestaciones*, *líneas de distribución* y *centros de transformación*. La distribución en niveles mostrada en la Ilustración 5



corresponde a la misma jerarquía descrita en el apartado Introduction: un primer nivel correspondiente a la subestación eléctrica, instalación destinada a modificar y establecer los niveles de tensión de una infraestructura eléctrica; un segundo nivel correspondiente a las líneas de distribución, los elementos capaces de distribuir la electricidad por toda la superficie, formando la red propiamente dicha; y un tercer nivel correspondiente a los centros de transformación, instalaciones eléctricas que reciben energía en alta tensión y la entregan en media o baja tensión para su utilización por los usuarios finales.

2. *El entorno debe permitir la extracción de tendencias espacio-temporales sobre las alarmas a fin de ayudar en la detección de problemas en la infraestructura de red.* Esto se cubre mediante las vistas detalladas mencionadas anteriormente, que consisten en una serie de gráficos que muestran las alarmas de manera espacio-temporal, de los tres elementos eléctricos principales, a distintos niveles de detalle e interconectados entre ellos mediante técnicas de interacción.

La siguiente ilustración muestra la interfaz definitiva de la aplicación web:

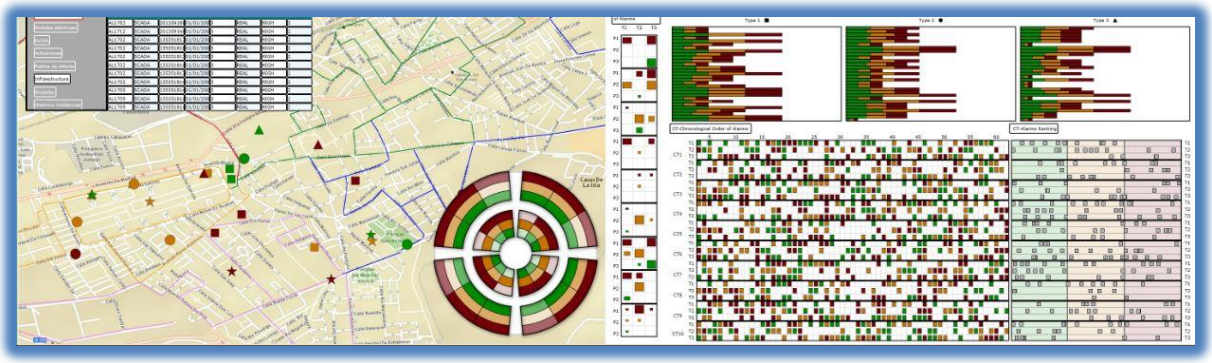


Ilustración 5. Interfaz de la solución

4.2 El proceso de desarrollo

En este apartado se detalla el proceso de desarrollo seguido, detallando el modelo de proceso aplicado para elaborar la solución y las fases de las que consta la solución: la **fase de análisis**, donde se detallan los requisitos; la **fase de diseño**, donde se identifican los componentes del sistema y sus características, y se elaboran prototipos; y la **fase de implementación**, donde se habla de la arquitectura del sistema y su organización.

Modelo de proceso

El ciclo de vida de un proyecto software se define como un marco de referencia en el que se incluyen los procesos, actividades y tareas que forman parte del proceso de desarrollo, mantenimiento y despliegue de un producto software. El modelo de ciclo de vida contempla el estado de las fases que componen un proyecto software, indicando el orden en el que se realiza cada una de las tareas y los criterios de transición entre ellas [20]. El modelo de proceso empleado para la elaboración de la solución ha sido el **modelo basado en prototipos**, perteneciente a los modelos de desarrollo evolutivos. En concreto, en la aplicación de este tipo de modelos el desarrollador construye una primera solución (prototipo) en poco tiempo, usando pocos recursos. El cliente la evalúa para una retroalimentación, gracias a la cual se redefinen los requisitos del software. El momento en el que se produce una evaluación se considera una interacción, la cual se repite a lo largo del tiempo, en periodos cortos, según el desarrollador elabora nuevas versiones que satisfacen las necesidades del cliente. Esto permite que el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo [22]. Además, se consiguen optimizar los tiempos estimados, puesto que no se llegan a puntos en los que es necesario volver atrás, gracias a las continuas correcciones del cliente. La ventaja y motivación de emplear este modelo se basa en que se está trabajando en el contexto de un entorno crítico en el que obtener requisitos del

cliente es difícil, por lo que estos se extraen casi por completo por necesidad del sistema. Además, este modelo reduce el riesgo de fallos al identificar rápidamente errores potenciales. Desde un punto de vista menos técnico, al trabajar de esta forma tan directa entre desarrollador y cliente, se consigue un buen entorno de trabajo, con los buenos resultados que eso implica [13]. Este modelo se ha caracterizado como adecuado en estas situaciones.

A continuación, se incluye la Ilustración 6 que refleja el modelo expuesto anteriormente:

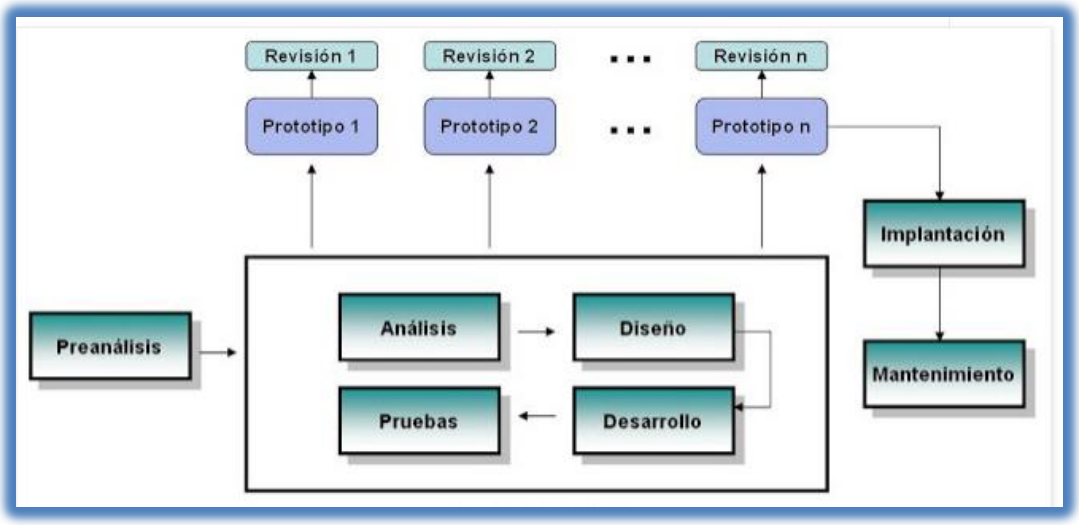


Ilustración 6. Modelo basado en prototipos

Análisis

La fase de análisis está dirigida a conocer las características del problema con el propósito de desarrollar una solución que resuelva el mismo. La realización del análisis implica la representación formal de la información, mediante la definición de requisitos, que guíe la elaboración de la solución. Como se ha explicado en el apartado Modelo de proceso, la extracción de requisitos por parte del cliente es difícil, y además, dada la criticidad del dominio de la red eléctrica, los requisitos se han inferido del propio objetivo del trabajo. La formalización de los requisitos se presenta primero mediante un listado, seguido de la especificación en la que se detallan con la plantilla de *Volere* [11]. La plantilla ha sido adaptada según las necesidades del trabajo, siendo los siguientes campos los relevantes para este proyecto:

- **Identificador**: campo que caracteriza unívocamente a un requisito.
- **Tipo**: campo que define si un requisito es de carácter funcional o no funcional.
- **Descripción**: descripción sencilla de la intención del requisito.
- **Prioridad**: define la importancia alta, media o baja del requisito. Esta medida viene dada por el grado de relación que tenga el requisito respecto de los objetivos del proyecto. Por ejemplo, una relación fuerte implica una prioridad alta.

La Tabla 9 muestra la plantilla empleada para la especificación de requisitos:

Identificador		**Tipo**		**Prioridad**	
Descripción					

Tabla 9. Adaptación de la plantilla de requisitos de Volere

En esta sección tan solo se incluye la definición de requisitos. La especificación completa se detalla en el Anexo III. Especificación de requisitos.

Definición de requisitos

En este apartado se enumeran todos los requisitos funcionales y no funcionales que debe cumplir el sistema. Los requisitos funcionales se etiquetarán mediante **RF-{número de requisito}** y los requisitos no funcionales mediante **RNF-{número de requisito}**.

Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares [27].

RF-1. La interfaz de usuario del entorno interactivo de visualización de alarmas debe comprender dos vistas principales: una general y una detallada.

RF-2. La vista general debe desplegar una vista geoespacial de la infraestructura de red, así como un resumen general de carácter alfanumérico de todas las alarmas registradas en dicha infraestructura.

RF-3. La vista detallada debe desplegar información relacionada con las alarmas registradas en los tres elementos principales que componen la red de distribución de energía eléctrica. Estas vistas estarán compuestas por un conjunto de gráficos. De aquí en adelante, se denominará *vista* al conjunto de gráficos, *elemento* a cada gráfico y *sección* a cada ítem dentro de cada elemento.

RF-4. La información sobre alarmas registradas en la subestación eléctrica debe ser desplegada en un gráfico radial. En particular, éste debe desplegar información relacionada con el volumen, tipología y prioridad de las alarmas.

RF-5. La información sobre alarmas registradas en las líneas de distribución eléctrica vendrán caracterizadas por una matriz de distribución y gráficos de barras. En particular, la matriz debe desplegar información relacionada con la tipología y prioridad de las alarmas; los gráficos de barras deben desplegar información relacionada con la tipología, prioridad y ubicación temporal de las alarmas.

RF-6. La información sobre alarmas registradas los centros de transformación vendrán caracterizados por dos matrices de distribución: una por orden cronológico y otra por ranking de prioridades. En particular, la matriz de orden cronológico debe desplegar información relacionada con la tipología, prioridad y ubicación temporal de las alarmas; la matriz de ranking de prioridades debe desplegar información relacionada con la tipología y prioridad de las alarmas, ordenándolas en función de dicha prioridad.

RF-7. Las secciones vendrán caracterizadas por tipos y prioridades.

RF-8. Los elementos de todas las vistas deben ser interactivos, a nivel de sección y elemento.

RF-9. Las interacciones a nivel de elemento son individuales y a nivel de sección, múltiples.

RF-10. Los elementos y las secciones se podrán deseleccionar a través de capacidades de interacción.

RF-11. La selección de un elemento y de una sección se verá resaltada a través de la interfaz.

RF-12. Todas las vistas tendrán una etiqueta que identificará qué elemento representan.

RF-13. Debe existir coordinación entre vistas para permitir la interoperabilidad entre todos los elementos de las vistas general y detallada.

RF-14. Esta coordinación se producirá de forma unidireccional entre elementos de distintas sub-vistas.

RF-15. Esta coordinación se producirá de forma bidireccional entre elementos de la misma sub-vista.

RF-16. Debe existir coordinación entre el gráfico radial y el mapa y la tabla de la vista general.

RF-17. Debe existir coordinación entre el gráfico radial y todos los elementos de la vista detallada.

RF-18. Debe existir coordinación entre los elementos correspondientes a la vista relacionada con las líneas de distribución y la de los centros de transformación.

RF-19. Debe existir coordinación entre los elementos correspondientes a la vista relacionada con los centros de transformación.

RF-20. Los elementos deben emplear los mismos colores cuando se interactúa con ellos.

Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema [27].

RNF-1. El entorno debe poder desplegarse en cualquier navegador web.

RNF-2. El usuario deberá tener el *plugin* Silverlight instalado, siendo la versión 5 la óptima.

RNF-3. El entorno será visualizado con un monitor de mínimo 43'' Pulgadas.

RNF-4. La proporción de los tamaños de los elementos del entorno se debe adaptar al tamaño del navegador.

RNF-5. La escala de colores empleada para las alarmas seguirá el estándar *Abnormal Situation Management* [23] para la clasificación de alarmas.

RNF-6. Las etiquetas de los gráficos deben estar escritas en inglés para promover la internacionalización.

Diseño

La fase de diseño permite establecer la solución a elaborar, identificando tanto las entidades esenciales de la misma como sus características detalladas. En esta fase se detalla el seguimiento del modelo de proceso de desarrollo expuesto anteriormente, el modelo basado en prototipos, mediante la elaboración de prototipos de distinto nivel de fiabilidad que guíen la definición de la interfaz final. De acuerdo a esto, esta fase se ha dividido en tres sub-fases, cada una de las cuales se verá alimentada por la fase previa.

Fase 1

La fase 1 tiene como objetivo la definición de la estructura o *layout* del entorno interactivo de visualización de alarmas, es decir, la definición de cómo debería presentarse la información correspondiente a las alarmas registradas en los elementos eléctricos que componen la infraestructura de red. En concreto, se define esta estructura de acuerdo a los requisitos previamente definidos en el apartado Definición de requisitos. Además, este primer prototipo se

consigue mediante la elaboración de un prototipo horizontal, aquel que se basa en mostrar las características principales de la interfaz de usuario sin tener un respaldo de la funcionalidad.

La Ilustración 7 muestra este primer diseño:

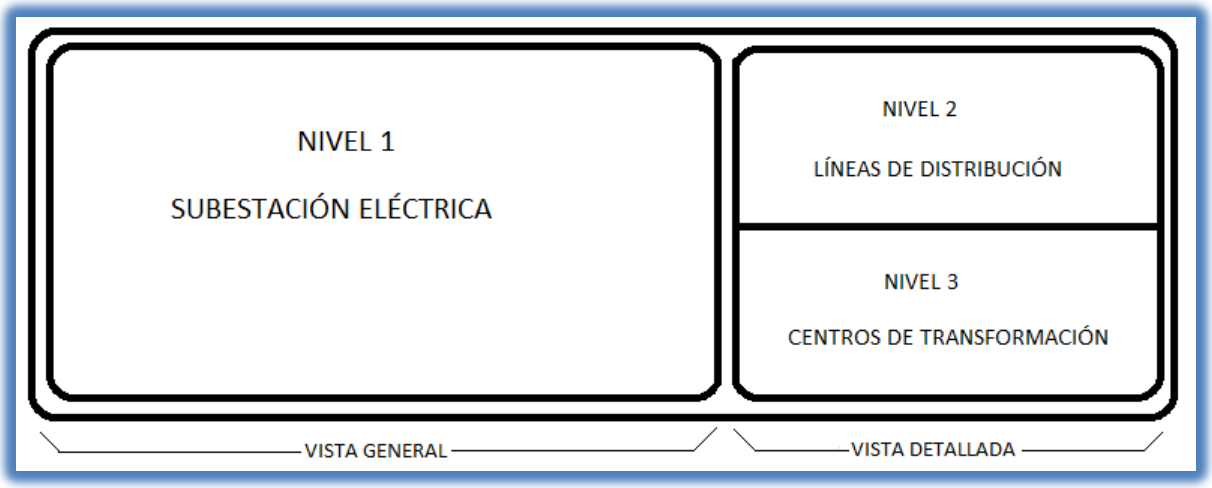


Ilustración 7. *Mockup* de la fase 1 de diseño

El diseño anterior presenta la estructura básica del entorno interactivo de visualización de alarmas. Este diseño se ha desarrollado en base a los objetivos que se desean cubrir en el proyecto, desarrollados en el apartado Descripción de la solución. Se recuerdan brevemente estos objetivos, indicando cómo el diseño lo cubre, centrando la atención en los elementos visuales que lo compondrán:

1. *El entorno debe permitir la exploración tanto de volúmenes pequeños de alarmas como de volúmenes grandes*. Esto se cubre con una vista general (la mitad izquierda del diseño) mediante un mapa, además de un conjunto de sub-vistas detalladas de acuerdo a las *subestaciones* de la red de distribución eléctrica, el primer nivel de la jerarquía. Además, se incluye una vista detallada (la mitad derecha del diseño) que contendrá sub-vistas de acuerdo al resto de elementos principales de la red: *líneas de distribución* y *centros de transformación*, correspondientes a los niveles 2 y 3 de la jerarquía, respectivamente. Nótese que se han dispuesto los elementos visuales respetando los niveles de la jerarquía, de forma que la vista del usuario final los ubique tal y como los espera: en la parte superior izquierda el inicio de la navegación, desplazándose ésta hacia la parte inferior derecha.

Con la realización de este primer prototipo se concluye que a través de la estructura diseñada es posible incluir información de los tres elementos principales de la red eléctrica, cubriendo la necesidad del sistema expuesta en el párrafo anterior.

Fase 2

La segunda fase de diseño tiene como objetivo establecer los componentes de la interfaz y los nodos de información específicos. Para poder llevar a cabo este objetivo, se parte del *layout* ya definido en la primera iteración. Por lo tanto, en esta fase se busca detallar un nivel más la forma de visualizar toda la información que requiere un operador. Puesto que esta fase requiere la selección de diferentes componentes visuales, es en esta fase cuando se realiza un prototipo de

mayor nivel de "fiabilidad". La realización de este segundo prototipo implica definir los componentes de las dos vistas del entorno interactivo:

- En la vista general se incluye la vista geoespacial que permita situar espacialmente los elementos causantes de las incidencias, una tabla de carácter alfanumérico que muestre información de varios tipos y un gráfico radial que represente las alarmas correspondientes a las subestaciones eléctricas, agrupadas mediante tipo y prioridad.
- En la vista detallada se incluye un conjunto de sub-vistas correspondientes a los otros dos elementos principales de la red eléctrica: las líneas de distribución y los centros de transformación. Respecto a las líneas de distribución, se incluye un gráfico que agrupe las alarmas en función de su tipo y prioridad, y otro que amplíe la información mediante una ordenación por el instante temporal en el que se produce la incidencia. Respecto a los centros de transformación, se incluyen dos gráficos de apariencia similar: ambos agrupan las alarmas en función de tipo y prioridad, pero uno de ellos las ordena de forma temporal y el otro mediante un ranking en función de su prioridad.

La Ilustración 8 muestra este segundo diseño:

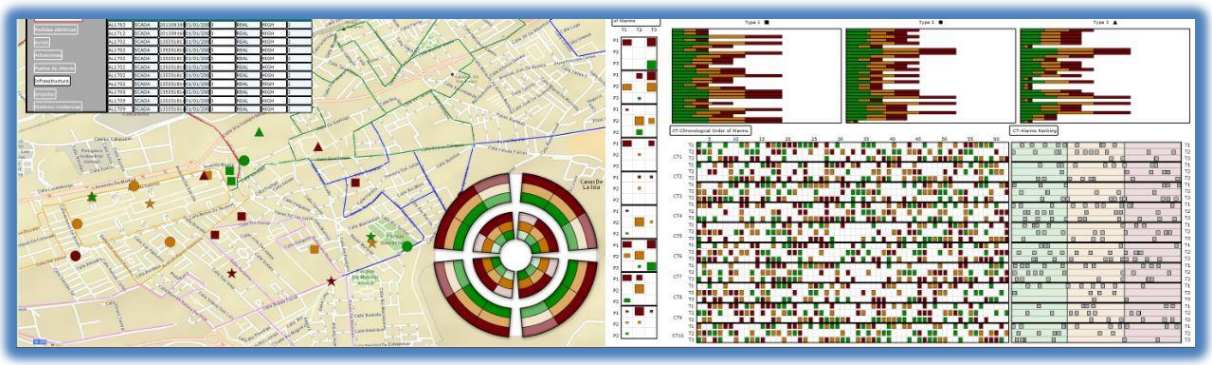


Ilustración 8. *Mockup* de la fase 2 de diseño

Fase 3

La fase 3 presenta un diseño idéntico al de la fase 2, con la diferencia de que en este punto se han incorporado todas las interacciones a los gráficos y la interoperabilidad entre ellos.

La tercera y última fase de diseño tiene como objetivo incorporar toda la interacción a los componentes visuales, permitiendo una coordinación completa. Para alcanzar este objetivo se parte del diseño definido en la segunda iteración. Al dotar de interacción a los componentes, este prototipo ya es de alto nivel de fiabilidad. La interacción, de forma más específica, ha sido de forma individual mediante la selección de elementos y de ítems en cada uno de los gráficos definidos en la fase 2, y de forma múltiple, coordinando todos los gráficos tanto de la vista detallada como de la general, y la tabla de la vista general. Puesto que se sigue tratando de un prototipo, no todos los componentes tienen interacción, siendo la vista geoespacial la que en este punto sigue estática.

Las siguientes ilustraciones muestran las capacidades interactivas añadidas en esta última fase de diseño:

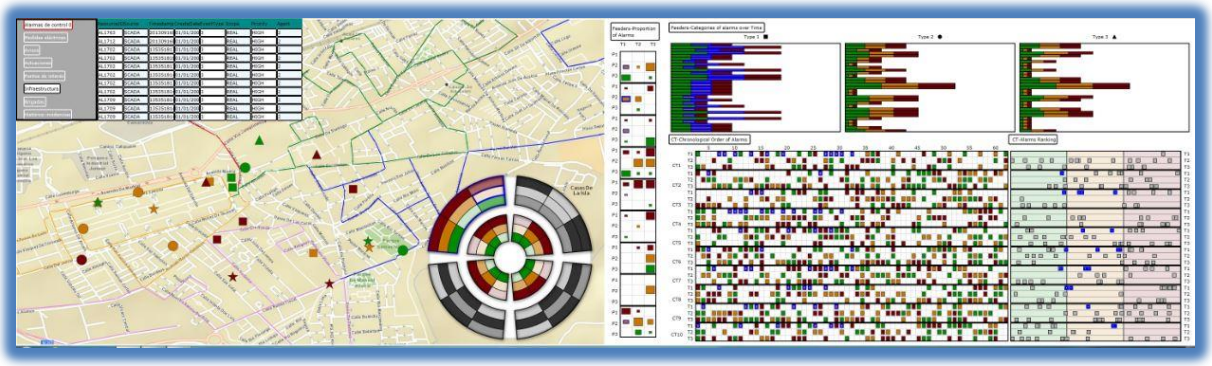


Ilustración 9. Mockup 1 de la fase 3 de diseño

La Ilustración 9 muestra la interacción entre el gráfico radial de la vista general y los componentes de la vista detallada. Al seleccionar una sección del radial, el resto de secciones se visualizan deseleccionadas mediante una gama de colores grises. Al seleccionar un ítem de la sección, este se marca con un borde azul, así como todas las alarmas del resto de gráficos, respetando el tipo y la prioridad de las alarmas.

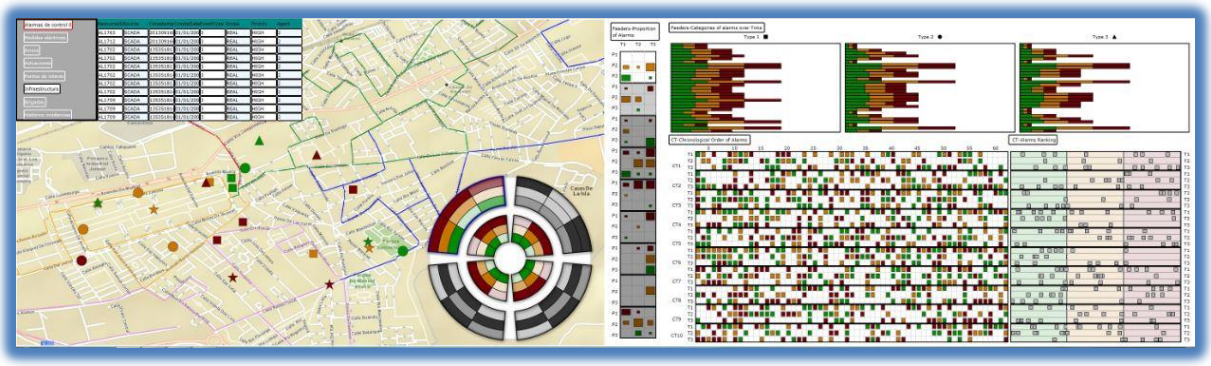


Ilustración 10. Mockup 2 de la fase 3 de diseño

La Ilustración 10 muestra la interacción entre el gráfico de distribución de los hilos y el resto de componentes de la vista detallada. Se observa que el gráfico radial permanece en el mismo estado, a la vez que se selecciona una sección del gráfico de distribución de los hilos, deseleccionando el resto de secciones. En este caso no hay ningún ítem seleccionado por lo que ningún ítem de otro componente muestra un borde azul.

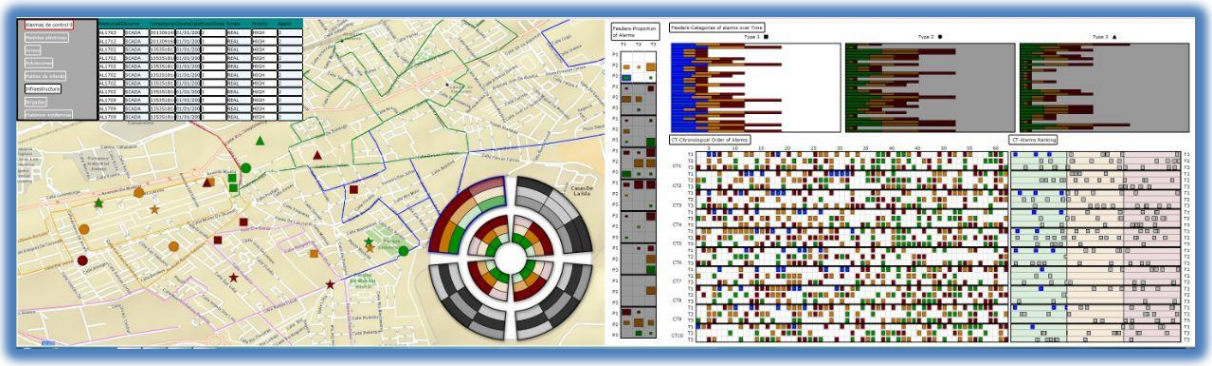


Ilustración 11. Mockup 3 de la fase 3 de diseño

La Ilustración 11 muestra la interacción entre el gráfico temporal de los hilos y los gráficos de los centros de transformación. Se observa que el gráfico radial y el de distribución de hilos permanecen en el mismo estado, a la vez que se selecciona una sección del gráfico temporal de los hilos, deseleccionando el resto de secciones. En este caso, se ha seleccionado un ítem del gráfico de distribución de los hilos, marcado con borde azul, marcando los ítems correspondientes del resto de gráficos, pero no del gráfico radial, comprobando que la interacción se realiza de forma descendiente y unidireccional.

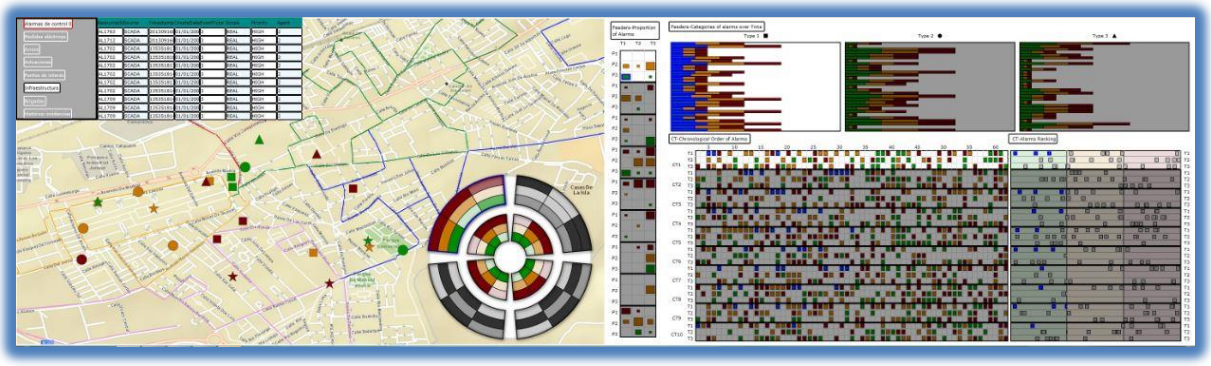


Ilustración 12. Mockup 4 de la fase 3 de diseño

La Ilustración 12 y última de la fase 3 de diseño muestra la interacción entre los gráficos de los centros de transformación. Se observa que el resto de gráficos permanecen en el mismo estado, inclusive los ítems, a la vez que se ha seleccionado una sección del gráfico de distribución del centro de transformación, seleccionando la sección equivalente del gráfico de ranking y deseleccionando el resto de secciones de los dos gráficos.

Implementación

La fase de implementación está dirigida a conocer las características del proyecto desarrollado, identificando la arquitectura del proyecto y la jerarquía de clases de la que se compone. La primera parte de este apartado va a estar dedicada a explicar la arquitectura Modelo-Vista-VistaModelo (*Model-View-ViewModel,* MVVM en adelante) que se ha aplicado en este proyecto así como la justificación de su uso. La segunda parte está destinada a identificar las

características de la arquitectura en el proyecto desarrollado. La tercera parte está destinada a definir la jerarquía de clases empleada en el desarrollo.

Arquitectura del sistema

MVVM es un patrón arquitectónico. Una aplicación que usa MVVM separa la lógica empresarial, la interfaz de usuario y el comportamiento de la presentación [9][10].

Cuando se usa el patrón MVVM, una aplicación se divide en las siguientes capas:

- La capa de **modelo** incluye todo el código que implementa la lógica principal de la aplicación y define los tipos requeridos para modelar el dominio de la aplicación. Esta capa es completamente independiente de las capas de vista y modelo de vista.
- La capa de **vista** define la interfaz de usuario que utiliza marcado declarativo. El marcado de enlace de datos define la conexión entre componentes específicos de la interfaz de usuario y diversos miembros de modelo de vista (y, en ocasiones, de modelo).
- La capa de **modelo de vista** proporciona destinos de enlace de datos para la vista. En muchos casos, el modelo de vista expone el modelo directamente o proporciona miembros que encapsulan miembros de modelo específicos. El modelo de vista también puede definir miembros para realizar un seguimiento de los datos que son relevantes para la interfaz de usuario pero no para el modelo, como el orden de visualización de una lista de elementos.

Estas son las relaciones entre una vista, un modelo de vista y un modelo:

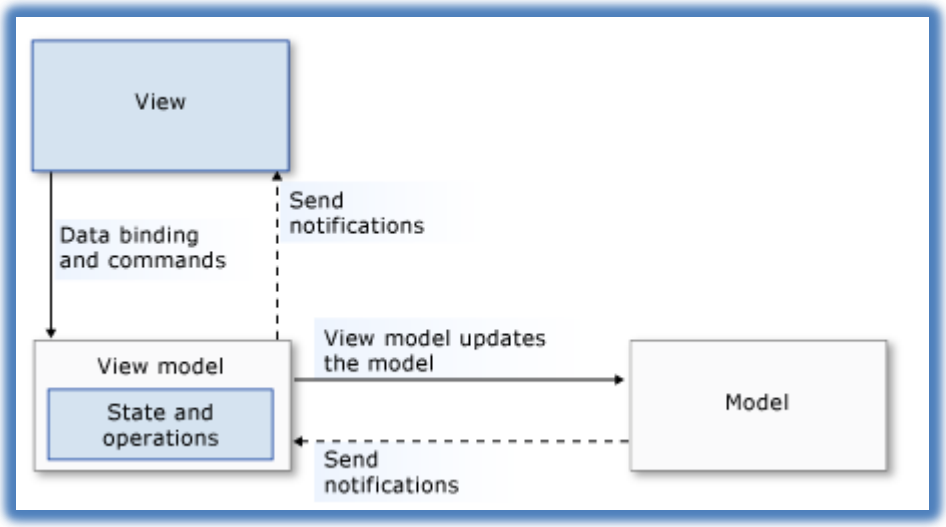


Ilustración 13. Relaciones entre Modelo-Vista-VistaModelo

Las ventajas del patrón MVVM en general y de esta separación en capas en particular, son las siguientes:

- Permite facilitar la comprensión del código. Esto se debe a que el código de características específicas a menudo es independiente de otro código, lo que facilita su aprendizaje y permite reutilizarlo en otras aplicaciones.

- Facilita la realización de pruebas automatizadas del código que no corresponde a la interfaz de usuario. Microsoft Visual Studio admite proyectos de pruebas unitarias, que permiten comprobar el diseño del código durante el desarrollo, así como identificar y diagnosticar errores.
- Una arquitectura estrechamente acoplada dificulta los cambios y el diagnóstico de errores. La principal ventaja de una arquitectura desacoplada es que permite aislar el impacto de los cambios, de forma que es mucho menos arriesgado probar nuevas características, corregir errores e incorporar las contribuciones de los colaboradores.

Por todos estos motivos, la arquitectura MVVM es la adoptada en el presente proyecto.

Relación entre arquitectura y jerarquía de clases

Antes de presentar la jerarquía de clases del proyecto, se va a detallar cómo se han aplicado las características de la arquitectura MVVM en el proyecto. En particular, se incluye la ilustración X, la cual muestra la jerarquía de carpetas del proyecto en el entorno de desarrollo del *Visual Studio*, identificando aquellas correspondientes a las capas de modelo, vista y modelo de vista.

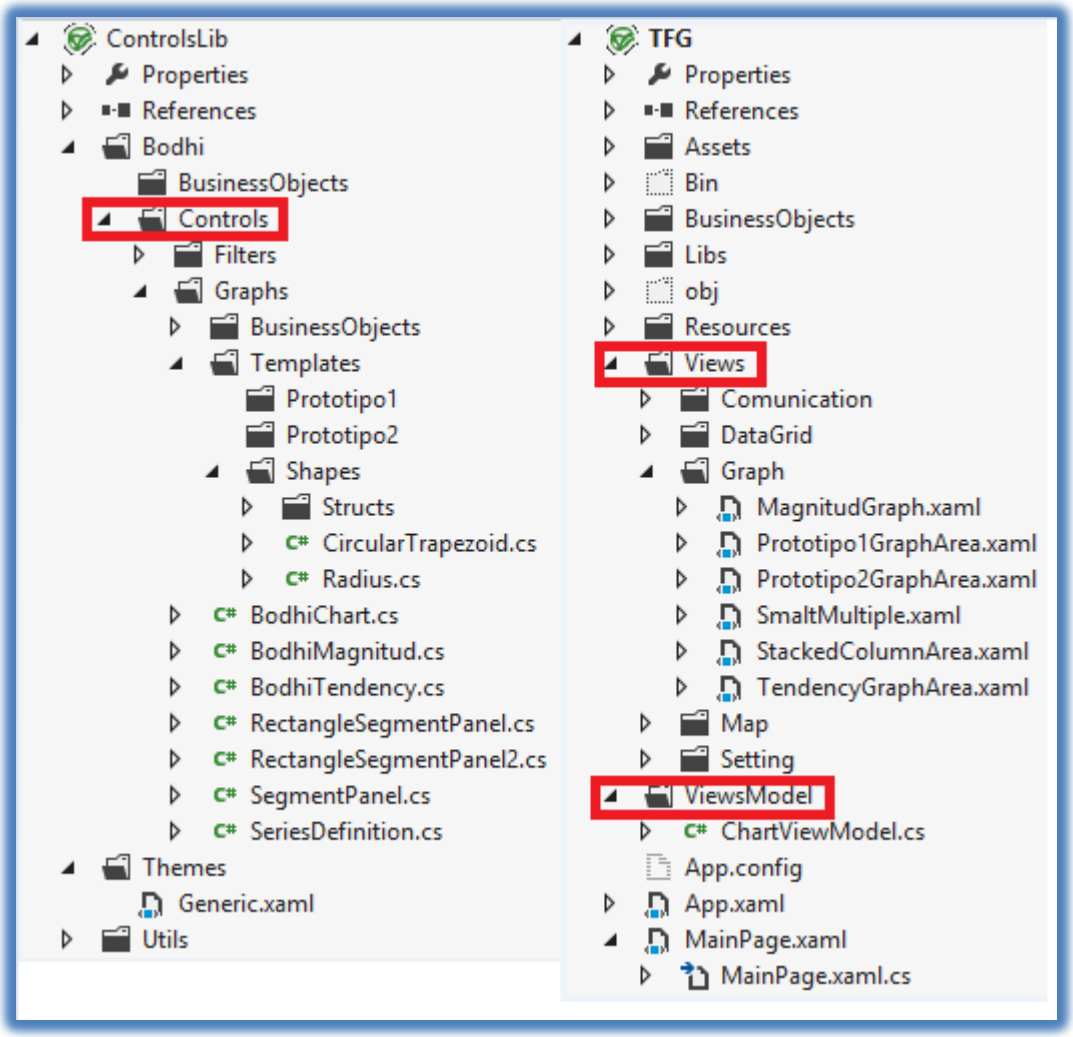


Ilustración 14. Relación entre la arquitectura y la jerarquía de clases

La carpeta *Controls* corresponde a la capa de modelo y contiene las clases que definen la lógica de la aplicación; la carpeta *Views* corresponde a la capa de vista y contiene las clases que definen la interfaz de usuario; la última carpeta, *ViewsModel*, corresponde a la capa de modelo de vista y contiene las clases que proporcionan los datos a la vista.

Jerarquía de clases

A continuación se presenta la jerarquía de clases del proyecto para tener una visión global del sistema, de forma que la explicación detallada de la funcionalidad se realice siempre teniendo en cuenta el entorno al completo. El apartado se va a exponer de la siguiente forma: en primer lugar, se presenta el diagrama de clases que permita tener una primera idea del proyecto de forma global; en segundo lugar, se comentan las relaciones entre clases para entender de una forma más precisa el funcionamiento del entorno; en último lugar, se detallan todos los componentes visuales, definiendo todos los atributos y métodos que lo componen para que la comprensión del proyecto sea precisa y completa.

La Ilustración 15 muestra la estructura dicha:

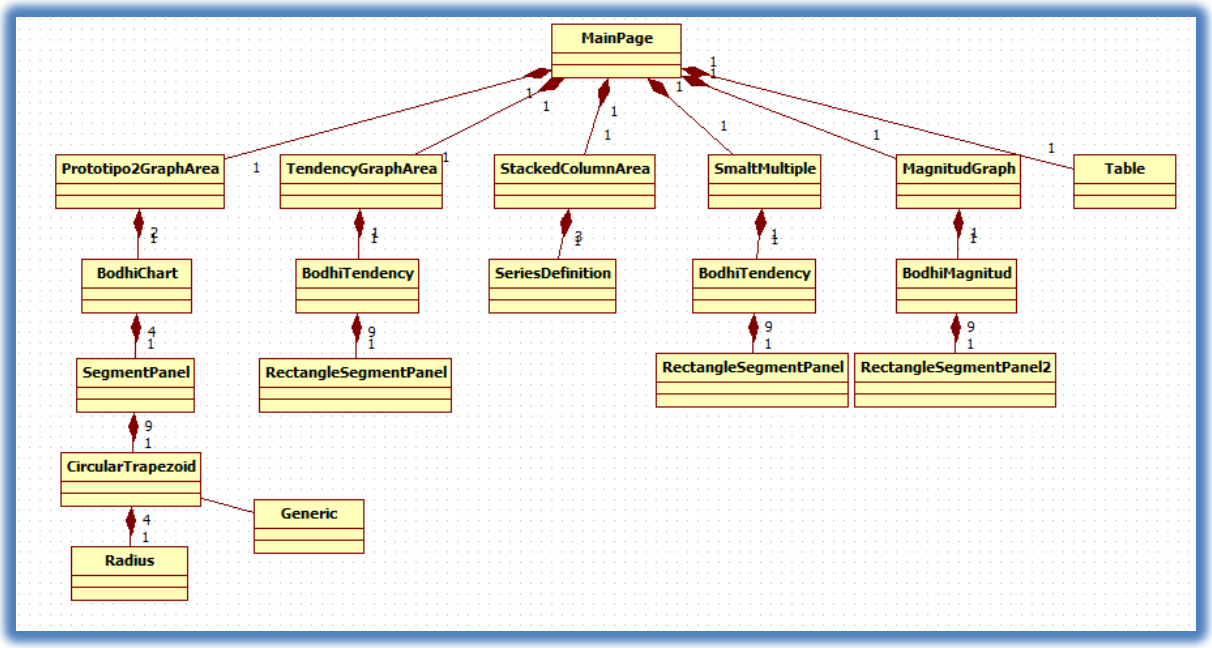


Ilustración 15. Diagrama de clases

A continuación se comentan las relaciones entre clases. Antes de exponerlo, es importante destacar que se está trabajando con un modelo de programación orientada a eventos a través de tecnologías web, en vez con un modelo de programación estructurada o de orientación a objetos. Lo que esto implica es que las clases que se observan en el diagrama no son todas de la misma categoría como se podría esperar: el nodo raíz y los nodos del primer nivel son de formato xaml y el resto de nodos, de formato C#. Lo que esto significa se expone a lo largo del apartado.

Las relaciones, por tanto, son las siguientes:

- La clase MainPage tiene una relación de composición con Prototipo2GraphArea en escala 1-1.
- La clase MainPage tiene una relación de composición con TendencyGraphArea en escala 1-1.
- La clase MainPage tiene una relación de composición con StackedColumnArea en escala 1-1.
- La clase MainPage tiene una relación de composición con SmaltMultiple en escala 1-1.
- La clase MainPage tiene una relación de composición con MagnitudGraph en escala 1-1.
- La clase MainPage tiene una relación de composición con Table en escala 1-1.
- La clase Prototipo2GraphArea tiene una relación de composición con BodhiChart en escala 1-2.
- La clase BodhiChart tiene una relación de composición con SegmentPanel en escala 1-4.
- La clase SegmentPanel tiene una relación de composición con CircularTrapezoid en escala 1-9.
- La clase CircularTrapezoid tiene una relación de composición con Radius en escala 1-4.
- La clase CircularTrapezoid tiene una relación de asociación con Generic.
- La clase TendencyGraphArea tiene una relación de composición con BodhiTendency en escala 1-1.
- La clase BodhiTendency tiene una relación de composición con RectangleSegmentPanel en escala 1-9 (*).
- La clase StackedColumnArea tiene una relación de composición con SeriesDefinition en escala 1-3.
- La clase SmaltMultiple tiene una relación de composición con BodhiTendency en escala 1-1.
- La clase BodhiTendency tiene una relación de composición con RectangleSegmentPanel en escala 1-9 (*).
- La clase MagnitudGraph tiene una relación de composición con BodhiMagnitud en escala 1-1.
- La clase BodhiMagnitud tiene una relación de composición con RectangleSegmentPanel2 en escala 1-9.

(*) Nótese que estas dos relaciones tienen la misma denominación. Se mencionan dos veces porque pertenecen a componentes distintos. Además, Silverlight permite nombrar a los componentes, de forma que aunque tengan la misma base visual, se puedan codificar partes independientes y propias a cada gráfico.

Antes de entrar más en detalle, se reitera que el nodo raíz y los de primer nivel son de formato xaml. En concreto, estos nodos de primer nivel equivalen a cada uno de los gráficos que se observan en el entorno. Es más, de forma más específica, cada eje vertical corresponde a cada gráfico.

Las clases tendrán la extensión .xaml si están destinadas a definir algún formato de interfaz o .cs (C#) si están destinadas a definir el comportamiento de los objetos. Más en concreto, las clases *MainPage, Prototipo2GraphArea, TendencyGraphArea, StackedColumnArea, SmaltMultiple,*

MagnitudGraph, Table y *Generic* son tipo xaml, aunque todas excepto *Generic* tienen su equivalente clase cs. La clase equivalente tiene un formato .xaml.cs, la cual sirve para definir el comportamiento de la clase xaml. El resto de clases son de tipo cs. En las siguientes líneas se detalla el funcionamiento de cada clase para comprender mejor esta jerarquía de clases.

La clase *MainPage* es la principal. Contiene un *grid* de dos zonas. La primera zona integra el mapa y tiene un panel flotante donde se posiciona el gráfico radial (Ilustración 16). La segunda zona está subdividida a su vez en otras dos, la primera de las cuales corresponde al gráfico de proporción de alarmas de los hilos (*feeders*) (Ilustración 17), mientras que la segunda sub-zona corresponde al resto de gráficos (Ilustración 18, Ilustración 19 e Ilustración 20). Sobre la clase *MainPage* se regresará al final del apartado para explicar el funcionamiento de la interacción entre gráficos una vez entendidos los funcionamientos particulares de cada clase. El segundo nivel del diagrama de clases corresponde a todas las clases desplegadas en la clase principal y comparten dos características comunes: sirven como contenedores para llamar a las clases de niveles inferiores, todas ellas de extensión C#, y en ellas se definen las etiquetas correspondientes, tanto etiquetas de nombre de gráfico como de los ejes.

La clase *Prototipo2GraphArea* (Ilustración 16) representa el gráfico radial. Esta clase es de las más complicadas por el siguiente motivo: Silverlight provee infinidad de recursos, pero no incluye nada parecido a trapezoides circulares que generen radiales del estilo. Por tanto, esta figura geométrica se ha de desarrollar desde cero, motivo por el cual hay tantas sub-clases en los niveles inferiores de la jerarquía. La clase en cuestión crea dos objetos *BodhiChart*, es decir, cada uno de los radiales que se observan en la Ilustración 16. La clase *BodhiChart* es una clase C# de tipo control (más adelante se observará que las otras clases C# son de tipo panel, puesto que trabajan con figuras geométricas ya definidas) y genera cuatro objetos de tipo *SegmentPanel*, los cuatro segmentos que se observa forman el círculo. La clase *SegmentPanel* ya contiene la definición de estados, puesto que es a este nivel donde empieza la interacción. Esta clase se compone de filas y columnas (según se indique) de objetos *CircularTrapezoid*. Esta clase representa el nivel máximo de detalle de visualización, contiene interacción propia (una vez que se selecciona un *SegmentPanel*) y es donde se definen las prioridades de las alarmas (véase que cada uno tiene su propio color). La última clase correspondiente a este gráfico es *Radius*, correspondiente al arco del *CircularTrapezoid*, es decir, representa sólo la línea curva. Con dos arcos y dos rectas se obtiene finalmente el trapezoide circular. La clase *Generic*, de extensión .xaml, es la encargada de definir los estilos y las interacciones de todo el gráfico radial, puesto que se está trabajando con clases de tipo control que no proveen ciertas herramientas que sí proveen las clases de tipo panel.

A continuación se detallan todos los atributos y métodos de las clases implicadas en la elaboración de este componente, así como una pequeña descripción de su objetivo para aquellos de cierta relevancia (sólo de aquellas clases cs, puesto que las clases xaml no tienen atributos ni métodos cómo tal):

- ***Prototipo2GraphArea.xaml.cs***:
  - Atributos: no tiene
  - Métodos: no tiene
- ***BodhiChart.cs***:
  - Atributos:
    - Bool isStrech
    - Int innerRadius
    - Int outterRadius
    - Int SegmentCapacity

        - Int RadialAxisCapacity
        - Int ArcAxisCapacity
        - Style SectionStyle
        - EBodhiGraphMode BodhiMode
        - Point center
        - List<SegmentPanel> segmentCollection
        - SegmentPanel selectedSegment
        - List<Radius> radiusCollection
        - Grid SectionPanel
        - Grid CirclePanel
        - Grid RadiusPanel
        - Double ArcGapSpace
        - Double ZoneGapDegrees
        - Double RadiusGapDegrees
        - Random rnd
    - Métodos:
        - void InnerRadiusChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
        - void OutterRadiusChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
        - void SegmentAreasChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
        - void RadialAxisSpacesChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
        - void ArcAxisSpacesChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
        - bool loadSectionCollection()
        - void SegmentOnclick(object sender, MouseButtonEventArgs e)
        - bool loadSegmentCollection()
        - void InitControlData()
        - void InitControlMeasure(Size availableSize)
        - void OnApplyTemplate()
        - Size ArrangeOverride(Size finalSize)
        - Size MeasureOverride(Size availableSize)
- ***SegmentPanel.cs***:
    - Atributos:
        - Double OutterRadius
        - Double InnerRadius
        - Double Angle
        - Bool IsClockwise
        - Double StartAngle
        - Bool IsStrech
        - Bool IsSelected
        - Brush BorderColor
        - Color BackGroundColor
        - Double BackGroundOpacity

        - Int RadialAxisCapacity
        - Int ArcAxisCapacity
        - Style SectionStyle
        - EBodhiGraphMode BodhiMode
        - List<CircularTrapezoid> trapezoidCollection
        - Point center
        - Size lastValidAvailableSize
        - Double ArcGapSpace
        - Double RadiusGapDegrees
        - Random rnd
        - ESegmentState segmentState
    - Métodos:
        - void OuterRadiusChanged(DependencyObject obj, DependencyPropertyChangedEventArgs e)
        - void UpdateChildrenMeasure()
        - void InitControlMeasure(Size availableSize)
        - void OnApplyTemplate()
        - Size MeasureOverride(Size availableSize)
        - Size ArrangeOverride(Size finalSize)
        - bool loadSectionCollection()
        - void MeasaureSectionCollection()
        - void OnClickedEvent(MouseButtonEventHandler OnClickFunction)
- ***CircularTrapezoid.cs***:
    - Atributos:
        - Ranges DensityLevelRange
        - Double DensityContent
        - Point UpRightPoint
        - Point UpLeftPoint
        - Point DownRightPoint
        - Point DownLeftPoint
        - Size ShapeOutterSize
        - Size ShapeInnerSize
        - Bool IsSelected
        - Bool selAux
        - EPrior prior
        - EOpacityModeDensity opacityModeDensity
        - Double fillOpacity
        - Path ShapePanel
    - Métodos:
        - void DensityContentChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
        - void DensityChanged(DependencyPropertyChangedEventArgs e)
        - void IsSelectedChange(DependencyObject d, DependencyPropertyChangedEventArgs e)
        - void DensityStateManager()
        - void PriorStateManager()

        - void createPathGeometry()
        - void OnApplyTemplate()
        - void loadTemplateUIElements()
        - Size MeasureOverride(Size availableSize)
        - void OnMouseEnter(MouseEventArgs e)
        - void OnMouseLeave(MouseEventArgs e)
        - void OnMouseLeftButtonDown(MouseButtonEventArgs e)
        - void NotifyPropertyChanged(String info)
- ***Radius.cs***:
    - Atributos:
        - Point StartPoint
        - Point EndPoint
        - Path ShapePanel
    - Métodos:
        - void createPathGeometry()
        - void loadTemplateUIElements()
        - void OnApplyTemplate()
        - Size MeasureOverride(Size availableSize)

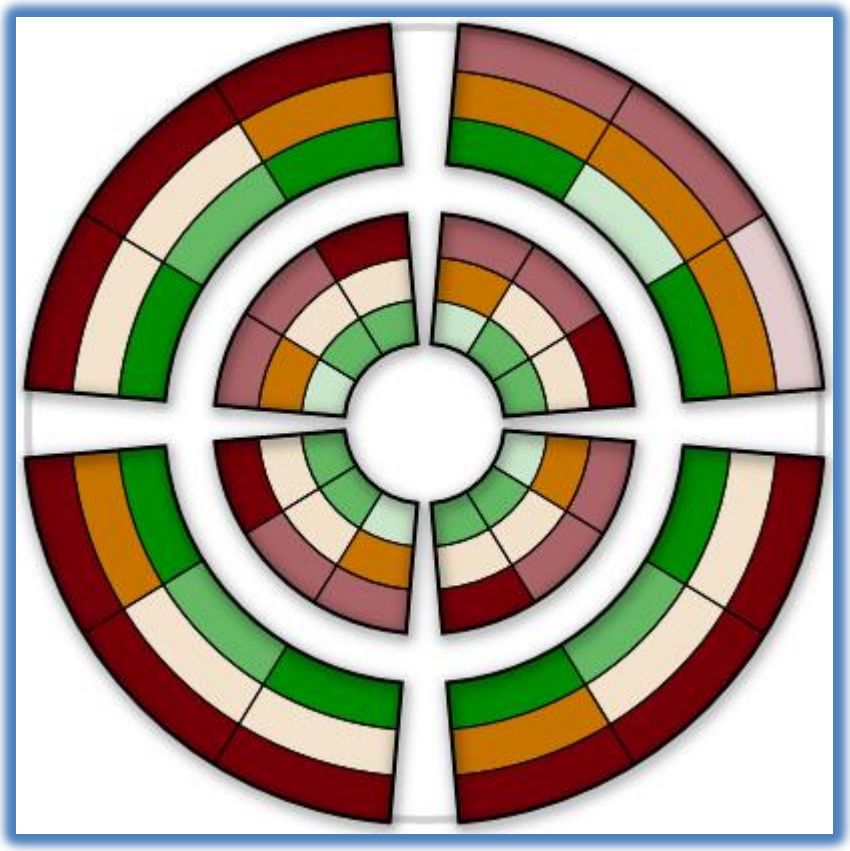


Ilustración 16. Gráfico radial

La siguiente clase es *TendencyGraphArea* (Ilustración 17) correspondiente a la proporción de alarmas en los hilos (*feeders*) de distribución. Esta clase crea un objeto *BodhiTendency* de una columna y nueve filas. Esta clase genera un *grid* (1x9) donde se crean objetos *RectangleSegmentPanel*, los cuales corresponden a los nueve cuadrados que se distinguen en la Ilustración 17 (en la ilustración se observan sólo tres para no emplear espacio innecesario en el presente documento). Esta clase genera dos *grids*, uno externo y global a todo el cuadrado para simular la interacción de nivel global, y otro (una cuadrícula de 3x3) interno para representar los objetos *Rectangle* (ya definidos por *Silverlight*) que representan las alarmas, situadas por prioridades (filas) y tipos (columnas), siendo este nivel el de máximo detalle.

A continuación se detallan todos los atributos y métodos de las clases implicadas en la elaboración de este componente, así como una pequeña descripción de su objetivo para aquellos de cierta relevancia (sólo de aquellas clases cs, puesto que las clases xaml no tienen atributos ni métodos cómo tal):

- ***TendencyGraphArea.xaml.cs***:
  - Atributos:
    - Bool resetTendency
  - Métodos:
    - void Reset(object sender, MouseEventArgs e). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método es el asociado a la etiqueta del componente.
    - void reset(). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método se emplea cuando un componente de una escala superior es seleccionado, de forma que se simule la interacción.
    - void recharge(). Invalida un componente para recargarlo con una disposición distinta de colores, simulando que corresponde a incidencias de elementos eléctricos distintos.
    - void rechargeUp(). Se emplea por cuestiones técnicas para que el método recharge() no se ejecute cada vez que se hace click con el ratón.
- ***BodhiTendency.cs***:
  - Atributos:
    - Rectangle auxOutter
    - RectangleSegmentPanel rectangle
    - Int ColuumnsSegment
    - DependencyProperty ColumnsSegmentProperty
    - Int RowsSegment
    - DependencyProperty RowsSegmentProperty
    - Grid gridPanel
    - List<RectangleSegmentPanel> rectangleSegmentCollection
  - Métodos:
    - void OnApplyTemplate()
    - Size MeasureOverride(Size availableSize)
    - Size ArrangeOverride(Size finalSize)
    - bool loadRectangle()
    - void OnMouseEvent(MouseEventHandler OnMouseFunction)

        - void Reset(object sender, MouseEventArgs e). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método es el asociado a la etiqueta del componente.

- ***RectangleSegmentPanel.cs***:
    - Atributos:
        - Int ColumnsSegment
        - Int RowsSegment
        - Int ContadorFila
        - Bool IsSelected
        - Bool RandomColors
        - Brush BorderColor
        - Color BackgroundColor
        - Double BackgroundOpacity
        - Style SectionStyle
        - Bool loaded
        - Bool changeGridMini
        - Bool IsMouseCaptured
        - Grid gridPanel
        - Grid gridPanelMini
        - Rectangle auxOutter
        - Random rnd
        - ERectangleSegmentState segmentState
        - EOpacityModeDensityRectangle opacityModeDensityRectangle
        - List<Rectangle> rectangleCollection
        - RectangleSegmentPanel selectedRectangle
    - Métodos:
        - void InitControlMeasure(Size availableSize)
        - void OnApplyTemplate()
        - void loadInitialStructure()
        - Size MeasureOverride(Size availableSize)
        - Size ArrangeOverride(Size finalSize)
        - bool loadSectionCollection()
        - bool loadMiniSectionCollection()
        - void Rectangle_MouseEnter(object sender, MouseEventArgs e)
        - void Rectangle_MouseLeave(object sender, MouseEventArgs e)
        - void Rectangle_MouseClick(object sender, MouseEventArgs e)
        - void unclear(Rectangle auxRectangle)
        - void MeasaureSectionCollection()

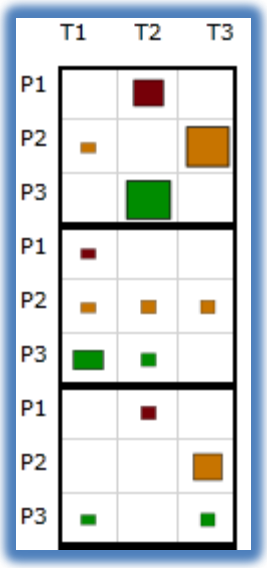


Ilustración 17. Gráfico de proporción de alarmas de los hilos

La siguiente clase es *StackedColumnArea* (Ilustración 18) correspondiente a las alarmas en los hilos de distribución representadas por categorías a lo largo del tiempo, siendo la parte superior la correspondiente al instante inmediato de tiempo, descendiendo hasta treinta segundos por detrás. Esta clase crea tres objetos *SeriesDefinition*, uno por cada tipo de alarmas. Esta clase es equivalente a *RectangleSegmentPanel*, puesto que genera también dos *grids* con sus interacciones correspondientes en los dos niveles. La diferencia es únicamente la forma de representar las alarmas, simulando aquí un gráfico de barras, frente a la cuadrícula del gráfico anterior.

A continuación se detallan todos los atributos y métodos de las clases implicadas en la elaboración de este componente, así como una pequeña descripción de su objetivo para aquellos de cierta relevancia (sólo de aquellas clases cs, puesto que las clases xaml no tienen atributos ni métodos cómo tal):

- ***StackedColumnArea.xaml.cs***:
  - Atributos:
    - Bool addList
    - Bool resetStacked
    - List<SeriesDefinition> seriesCollection
  - Métodos:
    - void stackedSelected(object sender, MouseEventArgs e)
    - void AddList()
    - void Reset(object sender, MouseEventArgs e). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método es el asociado a la etiqueta del componente.
    - void reset(). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método se emplea cuando un componente de una escala superior es seleccionado, de forma que se simule la interacción.

        - void recharge(). Invalida un componente para recargarlo con una disposición distinta de colores, simulando que corresponde a incidencias de elementos eléctricos distintos.
        - void rechargeUp(). Se emplea por cuestiones técnicas para que el método recharge() no se ejecute cada vez que se hace click con el ratón.
- ***SeriesDefinition.cs***:
    - Atributos:
        - Int ColumnsSegment
        - Int RowsSegment
        - Int ContadorFila
        - Bool IsSelected
        - Brush BorderColor
        - Color BackgroundColor
        - Double BackgroundOpacity
        - Style SectionStyle
        - Size lastValidAvailableSize
        - Bool loaded
        - Bool isMouseCaptured
        - Bool changeGridMini
        - Grid gridPanel
        - Grid gridPanelMini
        - Rectangle auxOutter
        - Random rnd
        - ESeriesDefinition segmentState
        - List<Rectangle> rectangleCollection
    - Métodos:
        - void UpdateChildrenMeasure()
        - void InitControlMeasure(Size availableSize)
        - void OnApplyTemplate()
        - void loadInitialStructure()
        - Size MeasureOverride(Size availableSize)
        - Size ArrangeOverride(Size finalSize)
        - bool loadSectionCollection()
        - bool loadMiniSectionCollection()
        - void Rectangle_MouseEnter(object sender, MouseEventArgs e)
        - void Rectangle_MouseLeave(object sender, MouseEventArgs e)
        - void Rectangle_MouseClick(object sender, MouseEventArgs e)
        - void MeasaureSectionCollection()

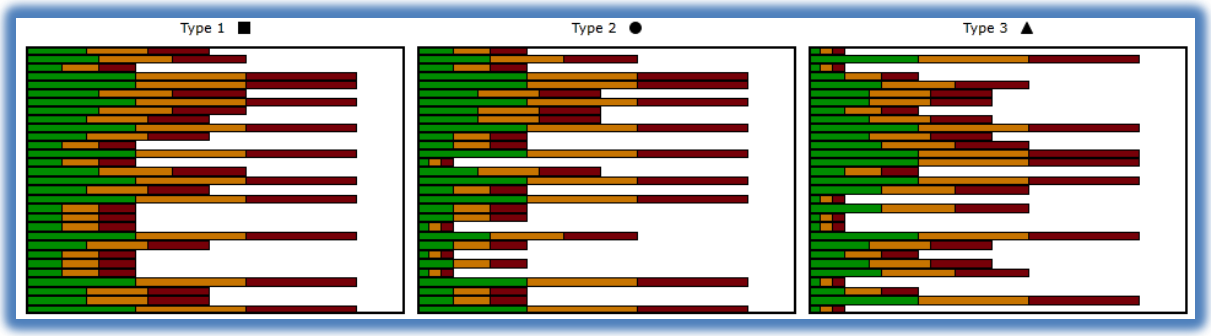


Ilustración 18. Gráfico de categorías de alarmas temporales en los hilos

La siguiente clase es *SmaltMultiple* (Ilustración 19) correspondiente al orden cronológico de alarmas en los centros de distribución. El gráfico es temporal, al igual que el anterior, siendo la parte izquierda el instante actual, hasta la parte derecha correspondiente a sesenta segundos antes. Esta clase es totalmente idéntica, en lo que se refiere a codificación, a *TendencyGraphArea*, creando un objeto *BodhiTendency*, con un *grid*, y con varios objetos *RectangleSegmentPanel*. La diferencia, como se observa, es el tamaño del gráfico y el número de filas y/o columnas.

A continuación se detallan todos los atributos y métodos de las clases implicadas en la elaboración de este componente, así como una pequeña descripción de su objetivo para aquellos de cierta relevancia (sólo de aquellas clases cs, puesto que las clases xaml no tienen atributos ni métodos cómo tal):

- ***SmaltMultiple.xaml.cs***:
  - Atributos:
    - Bool resetMagnitud
  - Métodos:
    - void Reset(object sender, MouseEventArgs e). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método es el asociado a la etiqueta del componente.
    - void reset(). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método se emplea cuando un componente de una escala superior es seleccionado, de forma que se simule la interacción.
    - void recharge(). Invalida un componente para recargarlo con una disposición distinta de colores, simulando que corresponde a incidencias de elementos eléctricos distintos.
    - void rechargeUp(). Se emplea por cuestiones técnicas para que el método recharge() no se ejecute cada vez que se hace click con el ratón.
- ***BodhiTendency.cs***:
  - Atributos:
    - Rectangle auxOutter
    - RectangleSegmentPanel rectangle
    - Int ColuumnsSegment
    - DependencyProperty ColumnsSegmentProperty
    - Int RowsSegment

        - DependencyProperty RowsSegmentProperty
        - Grid gridPanel
        - List<RectangleSegmentPanel> rectangleSegmentCollection
    - Métodos:
        - void OnApplyTemplate()
        - Size MeasureOverride(Size availableSize)
        - Size ArrangeOverride(Size finalSize)
        - bool loadRectangle()
        - void OnMouseEvent(MouseEventHandler OnMouseFunction)
        - void Reset(object sender, MouseEventArgs e). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método es el asociado a la etiqueta del componente.
- ***RectangleSegmentPanel.cs***:
    - Atributos:
        - Int ColumnsSegment
        - Int RowsSegment
        - Int ContadorFila
        - Bool IsSelected
        - Bool RandomColors
        - Brush BorderColor
        - Color BackgroundColor
        - Double BackgroundOpacity
        - Style SectionStyle
        - Bool loaded
        - Bool changeGridMini
        - Bool IsMouseCaptured
        - Grid gridPanel
        - Grid gridPanelMini
        - Rectangle auxOutter
        - Random rnd
        - ERectangleSegmentState segmentState
        - EOpacityModeDensityRectangle opacityModeDensityRectangle
        - List<Rectangle> rectangleCollection
        - RectangleSegmentPanel selectedRectangle
    - Métodos:
        - void InitControlMeasure(Size availableSize)
        - void OnApplyTemplate()
        - void loadInitialStructure()
        - Size MeasureOverride(Size availableSize)
        - Size ArrangeOverride(Size finalSize)
        - bool loadSectionCollection()
        - bool loadMiniSectionCollection()
        - void Rectangle_MouseEnter(object sender, MouseEventArgs e)
        - void Rectangle_MouseLeave(object sender, MouseEventArgs e)
        - void Rectangle_MouseClick(object sender, MouseEventArgs e)
        - void unclear(Rectangle auxRectangle)

- void MeasaureSectionCollection()

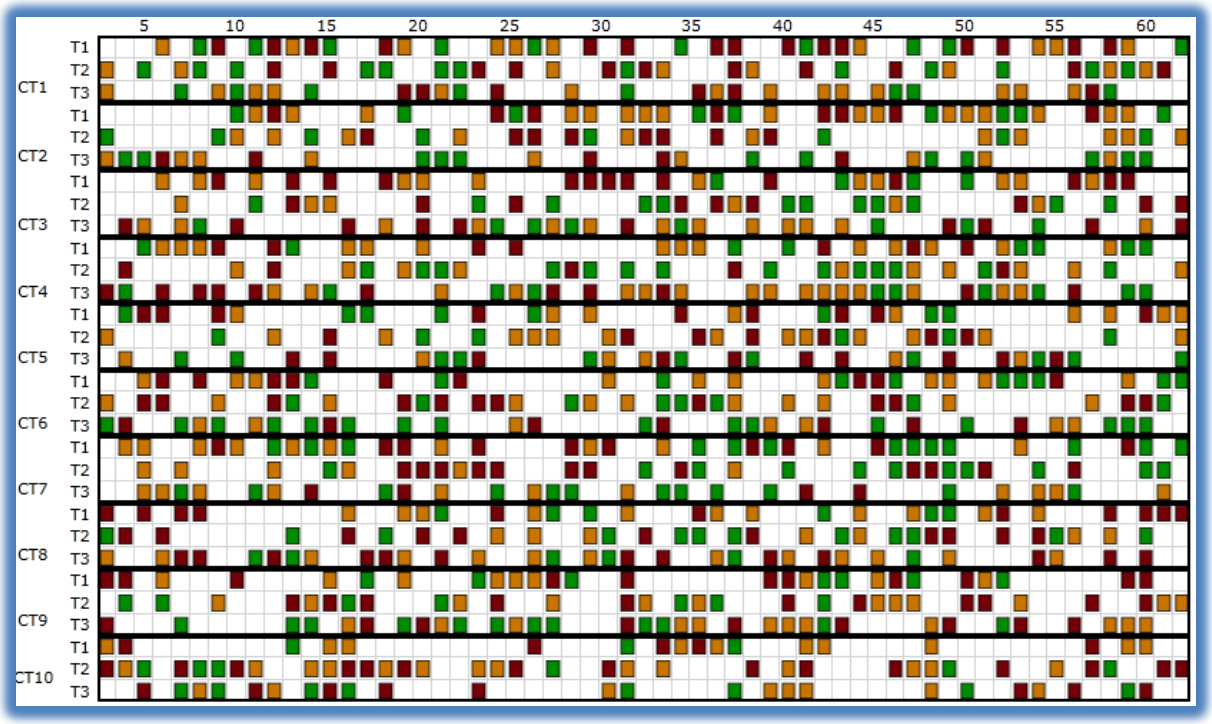


Ilustración 19. Gráfico de orden cronológico de alarmas en centro de distribución

La siguiente clase corresponde a *MagnitudGraph* (Ilustración 20) correspondiente al ranking de alarmas en los centros de distribución. Esta clase es totalmente equivalente a SmaltMultiple, con la diferencia de que se crean objetos *RectangleSegmentPanel2*, de forma que el gráfico se observa visualmente distinto. Estas dos clases representan la misma información (alarmas en centros de distribución) solo que esta última lo hace de forma gradual, de forma que, aun sabiendo que dos alarmas son de prioridad x, saber que la prioridad de una es mayor que la de la otra.

A continuación se detallan todos los atributos y métodos de las clases implicadas en la elaboración de este componente, así como una pequeña descripción de su objetivo para aquellos de cierta relevancia (sólo de aquellas clases cs, puesto que las clases xaml no tienen atributos ni métodos cómo tal):

- ***MagnitudGraph.xaml.cs***:
  - Atributos:
    - Bool resetSmalt
  - Métodos:
    - void Reset(object sender, MouseEventArgs e). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método es el asociado a la etiqueta del componente.
    - void reset(). Deselecciona los elementos del componente, tanto a nivel de sección como de ítem. Este método se emplea cuando un

componente de una escala superior es seleccionado, de forma que se simule la interacción.
        - void recharge(). Invalida un componente para recargarlo con una disposición distinta de colores, simulando que corresponde a incidencias de elementos eléctricos distintos.
        - void rechargeUp(). Se emplea por cuestiones técnicas para que el método recharge() no se ejecute cada vez que se hace click con el ratón.

- ***BodhiMagnitud.cs***:
    - Atributos:
        - RectangleSegmentPanel2 rectangle
        - RectangleSegmentPanel2 rectangleAux
        - Int ColumnsSegment
        - DependencyProperty ColumnsSegmentProperty
        - Int RowsSegment
        - DependencyProperty RowsSegmentProperty
        - Grid gridPanel
        - List<RectangleSegmentPanel2> rectangleSegmentCollection
    - Métodos:
        - void OnApplyTemplate()
        - Size MeasureOverride(Size availableSize)
        - Size ArrangeOverride(Size finalSize)
        - bool loadRectangle()
- ***RectangleSegmentPanel2.cs***:
    - Atributos:
        - int ColumnsSegment
        - int RowsSegment
        - int ContadorFila
        - bool IsSelected
        - bool RandomColors
        - Brush BorderColor
        - Color BackgroundColor
        - double BackgroundOpacity
        - Style SectionStyle
        - bool loaded
        - bool changeGridMini
        - bool isMouseCaptured
        - Grid gridPanel
        - Grid gridPanelMini
        - Rectangle auxOutter
        - Random rnd
        - ERectangleSegmentState2 segmentState
        - EOpacityModeDensityRectangle opacityModeDensityRectangle
        - List<Rectangle> rectangleCollection
        - RectangleSegmentPanel2 selectedRectangle
    - Métodos:

- void InitControlMeasure(Size availableSize)
- void OnApplyTemplate()
- void loadInitialStructure()
- Size MeasureOverride(Size availableSize)
- Size ArrangeOverride(Size finalSize)
- bool loadSectionCollection()
- bool loadMiniSectionCollection()
- void Rectangle_MouseEnter(object sender, MouseEventArgs e)
- void Rectangle_MouseLeave(object sender, MouseEventArgs e)
- void Rectangle_MouseClick(object sender, MouseEventArgs e)
- void unclear(Rectangle auxRectangle)
- void MeasaureSectionCollection()

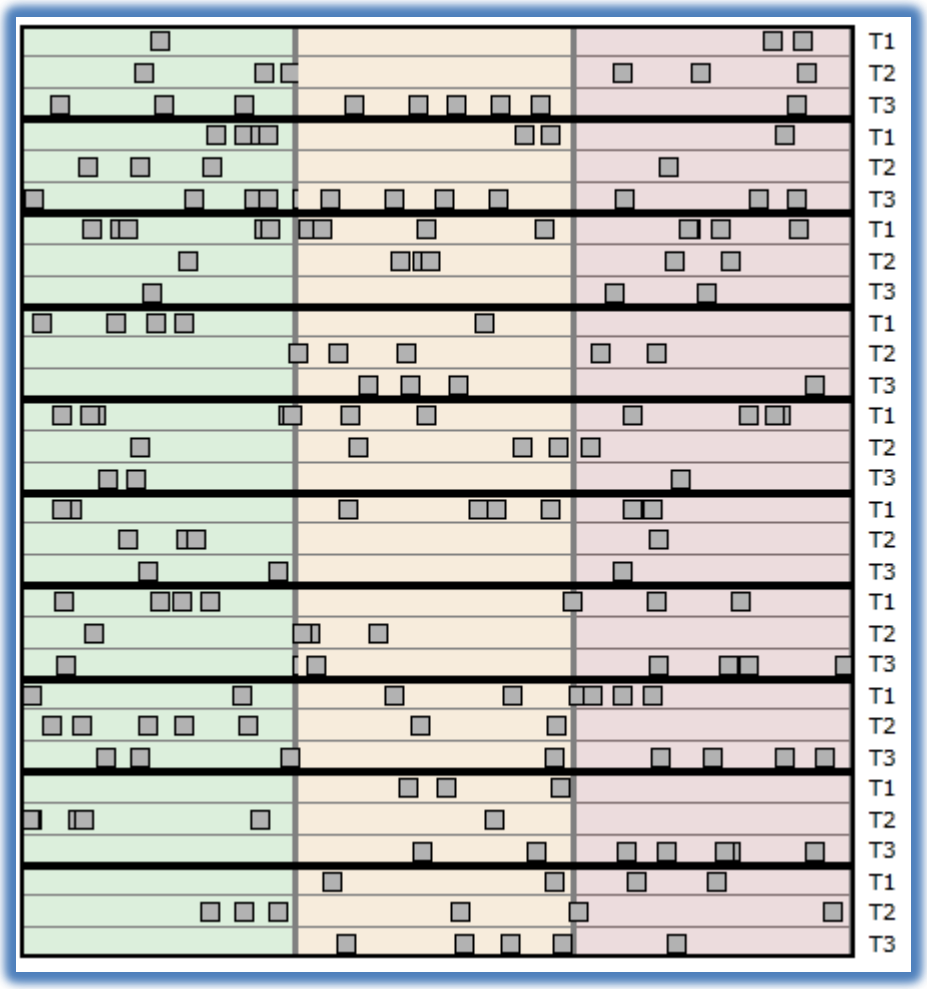


Ilustración 20. Gráfico de ranking de alarmas en centro de distribución

La última clase corresponde a *Table.xaml* (Ilustración 21). Esta clase simula una tabla que muestra datos de distintas características. Puesto que Silverlight no dispone de un objeto "tabla"

como tal (a diferencia de WPF), se ha elaborado un grid con numerosas filas, columnas y elementos *label* ("etiqueta") para rellenar las celdas.

A continuación se detallan todos los atributos y métodos de las clases implicadas en la elaboración de este componente, así como una pequeña descripción de su objetivo para aquellos de cierta relevancia (sólo de aquellas clases cs, puesto que las clases xaml no tienen atributos ni métodos cómo tal):

- ***Table.xaml.cs***:
  - Atributos: no tiene.
  - Métodos: no tiene.

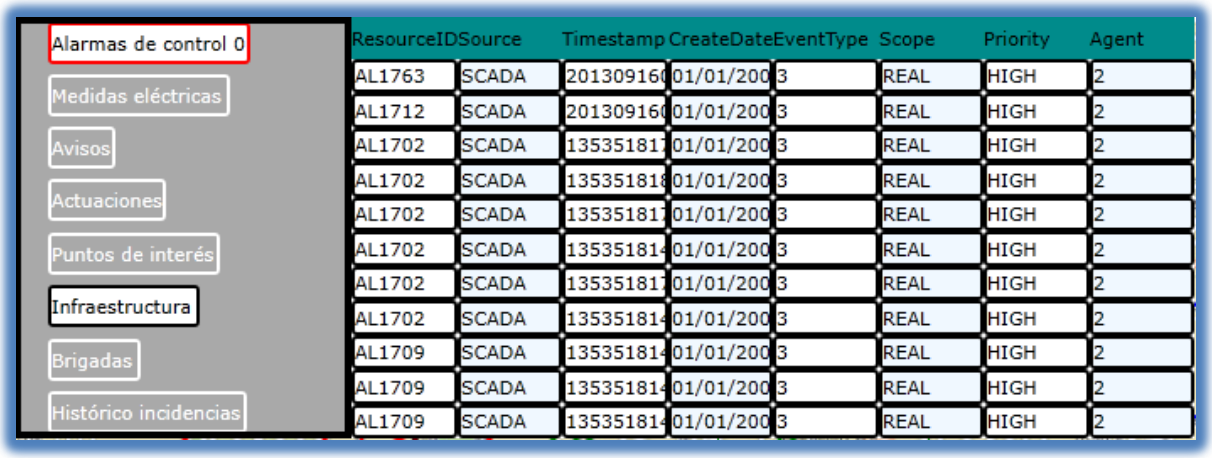


Ilustración 21. Tabla

Una vez que se han detallado las configuraciones de código a nivel individual, la explicación vuelve a la clase *MainPage*, como ya se comentaba al principio del apartado, para detallar la interacción global de todo el entorno. Puesto que esta clase es la única que engloba a todos los gráficos, es necesario codificar ahí todas las interacciones. El primer detalle importante es comentar lo siguiente: todas las clases del segundo nivel en la jerarquía, todas ellas de extensión xaml, tienen, por definición, una clase de extensión C#; estas ampliaciones no siempre contienen implementación propia, pero en este proyecto se ha decidido incorporar una funcionalidad de gran utilidad, que es la de resetear el gráfico, asignando el estado normal a todos los objetos, y la de recargar, actualizando los datos cuando se interactúa con gráficos de nivel superior. La terminología de resetear y recargar se aplicará a continuación.

La interacción global es, por tanto, la siguiente:

- Al seleccionar un segmento externo del gráfico radial, todos los gráficos se recargan y se resetean. Al seleccionar un trapezoide circular, se seleccionan rectángulos individuales del resto de gráficos, respetando tipos y prioridades, y viceversa si se deseleccionan.
- Al seleccionar un segmento del gráfico de distribución en las líneas, los gráficos de nivel inferior se resetean y recargan. Si se selecciona un rectángulo individual, la respuesta es la misma que en el caso anterior.
- Al seleccionar un segmento del gráfico temporal en las líneas, ningún gráfico se resetea y/o recarga puesto que este es el nivel mínimo. Al seleccionar un rectángulo individual, se

seleccionan rectángulos de los dos gráficos inferiores, respetando no sólo tipo y prioridad, sino también el instante temporal en el gráfico temporal de centros de distribución.

- Los dos últimos gráficos son equivalentes por lo que la interacción a este nivel también lo es: si se selecciona un segmento (un centro de distribución) se selecciona el equivalente en el gráfico del ranking. Lo mismo pasa con las alarmas individuales.
- Se recuerda que al seleccionar las etiquetas que denominan a los gráficos, se resetean tanto el gráfico denominado como los de nivel equivalente o inferior.

A continuación, y para finalizar el apartado, se detallan todos los atributos y métodos de las clases implicadas en la elaboración de este componente, así como una pequeña descripción de su objetivo para aquellos de cierta relevancia (sólo de aquellas clases cs, puesto que las clases xaml no tienen atributos ni métodos cómo tal):

- ***MainPage.xaml.cs***:
  - Atributos:
    - Enum VisualizationMode
    - VisualizationMode currentVisualizationMode
  - Métodos:
    - Void onClick(object sender, MouseEventArgs e). Define toda la interacción del entorno global.
    - Void onClickUp(object sender, MouseEventArgs e). Complementa al método anterior, controlando las interacciones de forma que no se produzcan eventos inesperados al hacer click con el ratón.
    - Void resetTendGA(). Gestiona la interacción individual del gráfico de distribución de los hilos y la global con el resto de componentes subyacentes.
    - Void onClickStacked(). Gestiona la interacción individual del gráfico temporal de los hilos y la global con el resto de componentes subyacentes.
    - Void resetBodhiChart(). Gestiona la interacción individual del gráfico radial y la global con el resto de componentes subyacentes.
    - Void resetTendGAUp(). Complementa al método resetTendGA(), controlando la interacción.
    - Void resetBodhiChartUp(). Complementa al método resetBodhiChart (), controlando la interacción.
    - Void Handle_MouseDownB(object sender, MouseEventArgs e)
    - Void Handle_MouseMoveB(object sender, MouseEventArgs e)
    - Void Handle_MouseUpB(object sender, MouseEventArgs e). Estos tres últimos métodos controlan la interacción que permite mover el gráfico radial en el panel flotante donde se ubica.
    - Void Handle_MouseDownT(object sender, MouseEventArgs e)
    - Void Handle_MouseMoveT(object sender, MouseEventArgs e)
    - Void Handle_MouseUpT(object sender, MouseEventArgs e). Estos tres últimos métodos controlan la interacción que permite mover la tabla en el panel flotante donde se ubica.

5 Evaluación

El objetivo de esta apartado consiste en demostrar la validez de la solución elaborada. La solución se considera válida si resuelve los problemas expuestos en el apartado Problem definition y satisface los objetivos definidos en el apartado Objectives. La forma de evaluación se dividirá en dos fases: la primera consiste en seleccionar el método o métodos de evaluación más adecuados, además de ejecutarlos; el segundo apartado consiste en analizar los resultados obtenidos.

5.1 Proceso de evaluación

Este apartado corresponde a la primera parte de la evaluación del sistema. Este se compone a su vez de dos sub-apartados: en el primero se escoge y diseña el método de evaluación a seguir; en el segundo se ejecuta dicho método.

Diseño de la evaluación

La forma de evaluación se basará en la ejecución de un conjunto de escenarios que alberguen una serie de tareas a ejecutar que permitan la verificación de los requisitos. En concreto, se emplearán tres, uno por cada elemento principal de la red de distribución eléctrica: subestación, líneas de distribución y centros de transformación. La formalización de estos escenarios se presenta mediante una plantilla. La plantilla contiene los siguientes campos:

- **Nombre del escenario:** descripción que caracteriza unívocamente a un escenario.
- **Id escenario:** campo que caracteriza unívocamente a un escenario.
- **Descripción:** descripción extendida del escenario de evaluación.
- **Tareas:** actividades que se han de ejecutar para la consecución del escenario de evaluación.
- **Verificación:** resultados obtenidos después de la ejecución del escenario de evaluación.

La Tabla 10 muestra la plantilla empleada para la formalización de los escenarios de evaluación:

Nombre del escenario		**Id escenario**	
Descripción			
Tareas			
Verificación			

Tabla 10. Plantilla de los escenarios de evaluación

Escenarios de evaluación

En este apartado se incluyen los tres escenarios de evaluación mencionados en el apartado Diseño de la evaluación, correspondientes a los tres elementos principales de la red de distribución eléctrica. Para ello, se rellenarán las plantillas correspondientes a la Tabla 10.

Nombre del escenario	Exploración de alarmas	**Id escenario**	ESC-1

	registradas en una subestación		
Descripción	El operador está monitorizando la red y comprueba que la distribución de alarmas de alta prioridad de tipo 2 está aumentando. Acude a la tabla informativa para ampliar la información sobre estas alarmas, buscando, en el listado, patrones que permitan identificar las causas de estas incidencias. Este operario es el jefe de varios agentes, por lo que se centra en encontrar todas aquellas que no tengan un agente asignado, de forma que las distribuya en función de los sectores que cubre cada uno de ellos.		
Tareas	• Consulta de la distribución de alarmas en función de tipo y prioridad en las subestaciones eléctricas. • Seleccionar las que son de prioridad alta y de tipo 2. • Consulta del listado de alarmas en la tabla de la vista general. • Localización de las alarmas que no tengan agente asignado (al ser un prototipo, la tabla no muestra datos apropiados).		
Verificación	• Al seleccionar una subestación eléctrica, el resto se muestran oscurecidas. • Al seleccionar el sector correspondiente a las alarmas de prioridad alta y de tipo 2, este se marca en azul y se despliega información sobre estas alarmas en la tabla.		

Tabla 11. Escenario de pruebas ESC-1

Nombre del escenario	Exploración de alarmas registradas en una línea de distribución	**Id escenario**	ESC-2
Descripción	El operario está monitorizando la red y		

	comprueba que una subestación eléctrica está acumulando muchas alarmas en sus líneas de distribución. Al comprobar que estas alarmas son de prioridad alta, necesita ubicar si las alarmas están dispersas en varias líneas o si solo se producen en una de ellas. Para ello, ha de comprobar los gráficos de distribución de alarmas donde confirma que estas han surgido en una única línea. El operario no está seguro de si se está realizando un seguimiento de estas alarmas y decide comprobar la antelación con la que se han producido. Al ver que en los últimos 15 minutos se ha producido dicho aumento de alarmas, decide encargarse de su resolución.
Tareas	• Consulta de una subestación eléctrica. • Consulta de la distribución de alarmas en las líneas de distribución pertenecientes a la subestación. • Consulta de las alarmas de prioridad alta en el hilo. • Consulta de la distribución temporal de las alarmas en la línea de distribución.
Verificación	• Al seleccionar una subestación en el gráfico radial, se muestra la distribución de alarmas de todos los tipos y prioridades. • Al seleccionar una subestación en el gráfico radial, se muestra la distribución de alarmas de las 9 líneas de distribución que lo componen en el gráfico de distribución de alarmas en los hilos. • Al seleccionar un sector en el gráfico radial, este ítem se marca con borde azul y se muestran las alarmas del mismo tipo y prioridad en los gráficos de los hilos, también con borde azul. • Al seleccionar un hilo en el gráfico de distribución de hilos, se muestra la distribución temporal de estas alarmas.

Tabla 12. Escenario de pruebas ESC-2

Nombre del escenario	Exploración de alarmas registradas en un centro de transformación	**Id escenario**	ESC-3
Descripción	El operario está monitorizando la red y está interesado en las alarmas de prioridad baja de los centros de transformación, que son su especialidad al poder estas, generalmente, solucionarse de forma rápida. En concreto, sólo se centra en las de tipo 1, que son a las que se dedica su sección. En un momento dado, observa un crecimiento masivo de alarmas de estas características en los últimos 5 minutos y en varios centros de transformación. El procedimiento habitual consiste en solucionar las alarmas por orden prioritario, por lo que acude al gráfico de ranking de alarmas de los centros de transformación. Una vez ubicadas las alarmas, las puede seleccionar y solucionar una por una y por orden.		
Tareas	• Consulta de las alarmas en el gráfico de distribución por centros de transformación. • Ubicadas las alarmas de prioridad baja, tipo 1 y sucedidas en los últimos 5 minutos, se seleccionan todas ellas. • Consulta de las alarmas seleccionadas en el gráfico de ranking.		
Verificación	• Al seleccionar un centro de transformación, el resto de centros de transformación se oscurecen, tanto en el gráfico de distribución como en el de ranking. • Al seleccionar las alarmas de tipo 1 y prioridad baja, estas se marcan con borde azul; las alarmas equivalentes del gráfico de ranking también se marcan con borde azul.		

Tabla 13. Escenario de pruebas ESC-3

5.2 Análisis de resultados

Este apartado consiste en analizar los resultados obtenidos tras las ejecuciones de los escenarios de evaluación formalizados en el apartado Escenarios de evaluación.

El análisis de los resultados se va a formalizar mediante una matriz de trazabilidad que relacione cada requisito redactado en el apartado Definición de requisitos con cada uno de los escenarios de evaluación presentados en el apartado anterior. El objetivo de esta matriz es verificar que todos los requisitos han sido cubiertos por el sistema. La verificación será satisfactoria si todos los requisitos están relacionados con un escenario de evaluación. Si hay algún requisito no relacionado, será necesario ejecutar más escenarios de pruebas; por otro lado, si hay requisitos relacionados con más de un escenario, la verificación seguirá siendo satisfactoria aunque posiblemente se hayan ejecutado escenarios de pruebas no del todo adecuados (esto no es necesario puesto que por definición del sistema, determinados requisitos podrían verse siempre relacionados con distintos tipos de pruebas).

Cabe destacar que los requisitos a tener en cuenta son solo los funcionales, aquellos que definen la funcionalidad del sistema.

La Tabla 14 muestra la matriz de trazabilidad en cuestión:

	ESC-1	ESC-2	ESC-3
RF-1	X	X	X
RF-2	X		
RF-3		X	X
RF-4	X		
RF-5		X	
RF-6			X
RF-7	X	X	X
RF-8	X	X	X
RF-9	X	X	X
RF-10	X	X	X
RF-11	X	X	X
RF-12	X	X	X
RF-13	X		
RF-14	X		
RF-15			X
RF-16	X		
RF-17	X		
RF-18		X	
RF-19			X
RF-20	X	X	X

Tabla 14. Matriz de trazabilidad entre requisitos funcionales y escenarios de pruebas

6 Conclusions

The aim of this section is to summarize the main contributions of this project, discuss possible future work based on the current development, list the problems encountered during all phases of the project and, finally, personal reviews of the work are given.

6.1 Contributions

The adopted solution is an alarm interactive visualization environment for the electric distribution network operation that would resolve the operation labour by human operators. Specifically, the solution is a web application, a tool that users can use accessing into a web server through the Internet through a browser.

The electric distribution network operation aims to ensure that the distribution network is running in normal operating mode. This mode of operation refers to the absence of significant deviations from network performance of the predefined thresholds. With this goal in mind, operators make use of the alarm system displays. These displays are more a burden than help, displaying large volumes of alarms [14], which has been formally conceptualized as "alarm flooding". To these situations, the operator is not able to process all this alarm information, and therefore to make effective decisions. This problem can lead to problematic operating inefficiencies and even critical operating situations, which can have a highly negative impact on the socio-economic activity of a country.

The above problem is solved by such an environment through a general view of alarms registered in the infrastructure through a map that allows spatially locate elements causing incidents alarms, and a set of detailed views that are organized according to the three main electrical components of this network: distribution substations, distribution feeders and transformers. These detailed views are a set of graphs showing alarms in a spatiotemporal way at different levels of detail and interconnected by interaction techniques.

6.2 Future work

Possible extensions of the developed prototype are described below.

As already detailed, interactive environment provides a general view and a detailed one. The components of the detail view have a fairly extensive interaction while the general view has fewer capabilities. Specifically, the only existing coordination is between the radial graph and table, which even has its own interaction. Possible improvements are, therefore, provide the table and map interaction at the individual level, in addition to coordinate with the other elements. Thus, the table may have several different views according to the electric elements were selecting in the detail view; this could also be generalized to the map, so to select items (or just alarms) of the detailed view, the exact position on the map was located.

Once a rich interaction at all levels is achieved and in all views, efforts can focus to expedite the work of the operators operation. Therefore, a possible improvement would be to add some design to launch ads in certain situations that would impose for the operators. For example, if operators are concerned about a particular substation, they would configure the system to warn them if the number of alarms of that substation exceeds a certain threshold.

Finally, the last improvement proposal is to take advantage of technological qualities of the moment. Specifically, a mobile app for smartphones and/or tablets that allow operators to be

aware of what happens in the mains could be developed. This application would be more focused on receiving the warnings described in the previous section while the operator is in non-working hours, but could also allow for interactive visualization environment adapted to the screen of the device.

6.3 Problems encountered

The fundamental problems encountered revolve around technology, despite being selected, has led several complications. Silverlight is a relatively new technology since it began shipping in 2007's. The reception that is having among developers is very good and so the documentation can be found is increasing. Even so, being a technology with less travel market, compared to direct competitors, the quality of the material is of a lesser degree.

To the above must be added that the developer of this project did not have previous knowledge of this technology. The general knowledge of web technologies was reduced.

The conflict that arises from these two points, no knowledge of technology and limited documentation, requires that the developer must take a self-taught paper. The department has expert technology staff to support sometimes but without having to reach a situation of teaching as if it were a subject.

Once this barrier is achieved, a small theoretical basis on which to start working is taken. The development of the interface is simple: it is the equivalent part to xaml code and Visual Studio environment helps a lot to designing the user interface, thanks to the display so the view of the code is displayed as the preview visual components. In addition, the corresponding xaml terminology is quite intuitive. The next obstacle arises at the moment to provide interactivity to the visual components, developed in C# language. The language itself is not a problem because it is very similar to C++, Java, etc. The problem consists that these tasks involve a higher degree of difficulty, being necessary the use of documentation to solve this, documentation which quality is so regular. It is easy to find examples of components interaction, but hardly found anything about interaction between different components.

Another problem that has been faced throughout the whole development is compatibility between WPF and Silverlight. In summary, in a little precise way, we can say that these two technologies have a common base but specialize in different branches. This means that the code examples that there are, despite what the author specified, may be compatible for either technologies or just one of them.

The development of this project has consisted in a web application, a tool that is displayed in a web browser with Internet connection. This is achieved because this is hosted on a web server, displaying all the visual components. The problem that arises here is that the development environment not loaded all of them (the map in particular), thus complicating the development work.

6.4 Personal opinions

The development of this final thesis has been an overall satisfaction in all aspects. In the first place, because it supposes the last door to cross to finish the degree, but above all it supposes a professional and personal satisfaction. In the previous section it has already mentioned the few knowledge of web technologies, so that the knowledge acquired during this year have been very large, especially considering that web technologies are basic elements (among many others) for a good informatics engineering. From a technical view point, the features of the Model-View-

ViewModel architecture have been learned, allowing comparison with the standard model-view-controller.

The knowledge gained was not just technical by the learning of using Silverlight technology, but also of a more practical nature, having faced this learning in a more autonomous way than is customary during the degree. The purpose of this is to train good professionals, able to find solutions without waiting for someone to do it for us (like life itself works).

In addition, this project has put into practice the knowledge of various subjects, having done a project completely, through all its phases. This also reveals the difficulty that supposed and the fundamental importance of planning tasks and follows all carefully: not everything is to develop, but also fully seat bases, defining what it wants to develop, why, etc. It is also learned the importance of tasks with time instead of leaving everything to the last minute. Given the scale of a final project, and comparing the practices of the subjects (less difficult and made in smaller team), in the first instant it is found that this requires a planned and consistent work. To complement the talk about architectures, one can generalize this: it is essential to expand and diversify your skills, so that they know how to analyse problems and select the best tools for their solution.

In summary, and to conclude this section, the overall satisfaction achieved is reiterated: it is gratifying to see how starting from virtually "nothing", that is, some ideas not completely defined, it gets a whole development of a tangible project.

7 Bibliografía

[1] "Silverlight vs Flash vs HTML5". Blogspot. 2009. Disponible [Internet]: http://sketchcode.blogspot.com.es/2010/09/silverlight-vs-flash-vs-html-5.html

[2] Gagliardi, N. and Morales S.: "Comparación de tecnologías para la creación de contenido dinámico". Doctorado en ingeniería de sistemas y computación, Universidad de Málaga, 2007. Disponible [Internet]: http://www.sicuma.uma.es/es/formacion/doctoradoargentina/independientes/argentina08/Gagliardi-Morales/index.htm

[3] "Microsoft Silverlight". Microsoft Corporation. 2014. Disponible [Internet]: http://msdn.microsoft.com/en-us/silverlight/bb187358.aspx

[4] "What is Silverlight". Microsoft Corporation. Disponible [Internet]: http://www.microsoft.com/silverlight/what-is-silverlight/

[5] "Adobe Flash". Adobe Systems Incorporated. 2014. Disponible [Internet]: http://www.adobe.com/es/products/flash.html

[6] "Una breve historia de Flash". Blogspot. 2010. Disponible [Internet]: http://elmundodelflash8.blogspot.com.es/p/una-breve-historia.html

[7] "JavaFX". Oracle and Sun Microsystems. Disponible [Internet]: http://www.oracle.com/technetwork/java/javafx/overview/index.html

[8] "Apple y la posibilidad de dar soporte a Flash". Movilzona. 2011. Disponible [Internet]: http://www.movilzona.es/2011/09/08/apple-y-la-posibilidad-de-dar-soporte-a-adobe-flash/

[9] "Usar el patrón MVVM en Hilo". Microsoft Corporation. Disponible [Internet]: http://msdn.microsoft.com/es-es/library/windows/apps/jj160324.aspx

[10] "Usar el patrón MVVM". Microsoft Corporation. Disponible [Internet]: http://msdn.microsoft.com/es-es/library/windows/apps/jj883732.aspx

[11] Volere, "Plantilla de Especificación de Requisitos", febrero 2006. Disponible [Internet]: http://www.volere.co.uk/pdf%20files/template_es.pdf

[12] Romero, R., Díez, D., Díaz, P. y Aedo, I. "Alarm Trend Catcher". 2014.

[13] "Prototype Model". I answer 4 u. Disponible [Internet]: http://www.ianswer4u.com/2011/11/prototype-model-advantages-and.html

[14] Endsley, M. R. "Designing for situation awareness: An approach to user-centered design CRC Press". 2003.

[15] Stanton, N. A. "Alarm initiated activities. *Human Factors in Alarm Design"*. 1994.

[16] Mattiasson, C. "The alarm system from the operator's perspective. Paper presented at the *Human Interfaces in Control Rooms, Cockpits and Command Centres, 1999. International Conference on,* 217-221". 1999.

[17] Rothenberg, D. H. "*Alarm management for process control: A best-practice guide for design, implementation, and use of industrial alarm systems".* McGraw-Hill Education. 2009

[18] Mikkelsen, C., Johansson, J., & Rissanen, M. Interactive information visualization for sensemaking in power grid supervisory systems. Paper presented at the *Information Visualisation (IV), 2011 15th International Conference on,* 119-126. 2011

[19] "Roles en el desarrollo de software". David Fuller. 2003. Disponible [Internet]: http://webcache.googleusercontent.com/search?q=cache:http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r101415.PDF

[20]"International Standard ISO/IEC 12207. Software Life Cycle Processes". Disponible [Internet]: http://www.abelia.com/docs/12207cpt.pdf

[21]Pressman, R. "Ingeniería del Software: un enfoque práctico" (McGrawHill, 6ª Edición).

[22]"Modelo de prototipos". Wikipedia. Disponible [Internet]: http://es.wikipedia.org/wiki/Modelo_de_prototipos

[23]Rothenberg, D. H. "*Alarm management for process control: A best-practice guide for design, implementation, and use of industrial alarm systems".* 2009. McGraw-Hill Education.

[24]"Standard No. ANSI/ISA-18.2-2009 Management of Alarm Systems for the Process Industries". International Society of Automation. Disponible [Internet]: www.isa.org

[25]Niwa, Y., and Hollnagel, E. "*Enhancing operator control by adaptive alarm presentation*". International Journal of Cognitive Ergonomics, 5(3), 367-384. 2001.

[26]Cockburn, A., Karlson, A., and Bederson, B. B. "*A review of overview detail, zooming, and focus interfaces*". ACM Computing Surveys (CSUR), 41(1), 2. 2008.

[27]Sommerville, I. "*Software Engineering*" (7ª Edición). 2005.

Anexo I. Control de versiones

Versión	Fecha de finalización	Descripción
1.0	03/03/2014	Redacción del apartado 2 “El estado de la cuestión”.
2.0	21/03/2014	Redacción del apartado 1 “Introducción”. Revisión del apartado 2.
3.0	13/05/2014	Redacción del apartado 4 “Solución”. Revisión de los apartados 1 y 2.
4.0	27/05/2014	Redacción del apartado 3 “Gestión de proyecto software” y del apartado 5 “Evaluación”. Revisión de los apartados 1, 2 y 4.
4.1	02/06/2014	Revisión de los apartados 3 y 4.
4.2	09/06/2014	Revisión de los apartados 3, 4 y 5.
5.0	15/06/2014	Redacción del apartado 6 “Conclusiones” y del apartado “Resumen”. Revisión de los apartados 4 y 5. Traducción al inglés del resumen, la introducción y las conclusiones
5.1	19/06/2014	Versión final del documento

Tabla 15. Control de versiones

Anexo II. Seguimiento del trabajo fin de grado

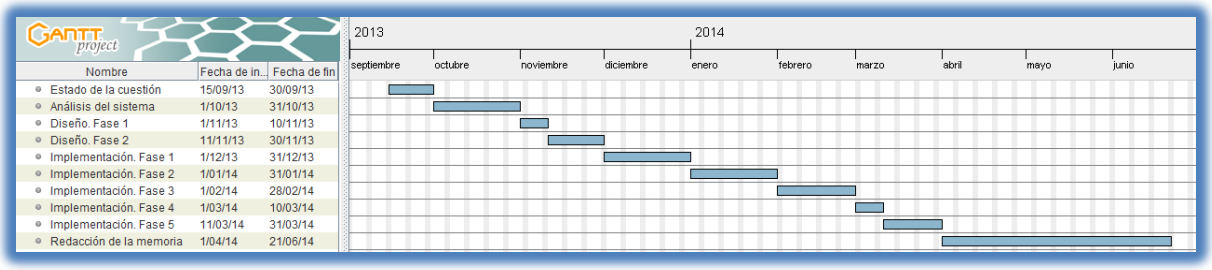
En este apartado se detalla el seguimiento del presente trabajo fin de grado. En primer lugar, se establece la forma de seguimiento que se va a realizar para, a continuación, especificar la planificación inicial del proyecto, especificando las tareas realizadas y sus plazos. El apartado finaliza con unas líneas en las que se indica la fidelidad respecto de la planificación inicial.

Forma de seguimiento

La forma de seguimiento del trabajo fin de grado ha consistido en reuniones quincenales con la tutora del proyecto. Las reuniones se han mantenido durante los 10 meses del proyecto, desde Septiembre de 2013 a Junio de 2014. En estas reuniones se exponían los avances y se establecían los hitos a cumplir para la siguiente reunión. Por cuestiones técnicas y personales, estas reuniones se han podido adelantar o retrasar, aunque nunca se han alargado más de 1 mes.

Planificación inicial

La planificación del trabajo fin de grado se ha formalizado mediante un diagrama de Gantt en el que se especifican las tareas realizadas y las fechas de inicio y de fin. La planificación se ha realizado teniendo en cuenta la ejecución completa del proyecto, pasando por todas sus fases. La Ilustración 22 muestra el diagrama en cuestión:



Nombre	Fecha de in...	Fecha de fin
Estado de la cuestión	15/09/13	30/09/13
Análisis del sistema	1/10/13	31/10/13
Diseño. Fase 1	1/11/13	10/11/13
Diseño. Fase 2	11/11/13	30/11/13
Implementación. Fase 1	1/12/13	31/12/13
Implementación. Fase 2	1/01/14	31/01/14
Implementación. Fase 3	1/02/14	28/02/14
Implementación. Fase 4	1/03/14	10/03/14
Implementación. Fase 5	11/03/14	31/03/14
Redacción de la memoria	1/04/14	21/06/14

Ilustración 22. Diagrama de Gantt

A continuación se describen en qué han consistido cada una de las tareas mostradas en la Ilustración 22:

1. **Estado de la cuestión**: se define el problema a resolver por el proyecto, además de establecer el contexto en el que se sitúa. En última instancia, se realiza una comparativa entre tecnologías para escoger la más adecuada.
2. **Análisis del sistema**: se describen las características del problema a resolver mediante la definición de los requisitos del sistema.
3. **Diseño. Fase 1**: se define la estructura, o *layout*, del entorno interactivo de visualización de alarmas. En concreto, se establecen las zonas y vistas del entorno interactivo, de acuerdo a los requisitos a cubrir.
4. **Diseño. Fase 2**: se define la interfaz definitiva de sistema, especificando todos los componentes (gráficos, mapas, tablas) que se desean desarrollar en el entorno.

5. **Implementación. Fase 1**: se desarrollan aquellos componentes visuales definidos en la fase 2 de diseño correspondientes a la vista detallada del entorno, es decir, los gráficos. En concreto, se implementan únicamente las interfaces de manera estática.
6. **Implementación. Fase 2**: se desarrollan todas las interacciones a nivel individual de los gráficos implementados en la fase 1.
7. **Implementación. Fase 3**: se desarrollan todas las coordinaciones que permitan la interoperabilidad los gráficos implementados en la fase 1.
8. **Implementación. Fase 4**: se desarrollan aquellos componentes visuales definidos en la fase 2 de diseño correspondientes a la vista general del entorno, es decir, se integra el mapa y se implementa la tabla con los datos.
9. **Implementación. Fase 5**: se desarrollan todas las interacciones restantes en el entorno, es decir, se coordinan las vistas general y detallada.
10. **Redacción de la memoria**: se elabora el presente documento, correspondiente a la documentación del proyecto completo.

Planificación final

El presente trabajo fin de grado ha cumplido la planificación inicial, habiéndose finalizado y entregado en fecha, cumpliendo así todos los objetivos propuestos.

Anexo III. Especificación de requisitos

Una vez enumerados todos los requisitos del sistema en el apartado Definición de requisitos, es necesario analizarlos y especificarlos desde el punto de vista del comportamiento, la estructura y la funcionalidad del sistema, para lo que se incluye en este apartado, por cada requisito y respetando el orden, una tabla que detalle dicha información. La Tabla 9 es la plantilla a rellenar por cada requisito.

Requisitos funcionales

Identificador	RF-1	**Tipo**	Funcional	**Prioridad**	Alta
Descripción	La interfaz de usuario del entorno interactivo de visualización de alarmas debe comprender dos vistas principales: una general y una detallada.				

Tabla 16. Requisito funcional RF-1

Identificador	RF-2	**Tipo**	Funcional	**Prioridad**	Alta
Descripción	La vista general debe desplegar una vista geoespacial de la infraestructura de red, así como un resumen general de carácter alfanumérico de todas las alarmas registradas en dicha infraestructura.				

Tabla 17. Requisito funcional RF-2

Identificador	RF-3	**Tipo**	Funcional	**Prioridad**	Alta
Descripción	La vista detallada debe desplegar información relacionada con las alarmas registradas en los tres elementos principales que componen la red de distribución de energía eléctrica. Estas vistas estarán compuestas por un conjunto de gráficos. De aquí en adelante, se denominará *vista* al conjunto de gráficos, *elemento* a cada gráfico y *sección* a cada ítem dentro de cada elemento.				

Tabla 18. Requisito funcional RF-3

Identificador	RF-4	**Tipo**	Funcional	**Prioridad**	Alta
Descripción	La información sobre alarmas registradas en la subestación eléctrica debe ser desplegada en un gráfico radial. En particular, éste debe desplegar información relacionada con el volumen, tipología y prioridad de las alarmas.				

Tabla 19. Requisito funcional RF-4

Identificador	RF-5	**Tipo**	Funcional	**Prioridad**	Alta
Descripción	La información sobre alarmas registradas en las líneas de distribución eléctrica vendrá caracterizadas por una matriz de distribución y gráficos de barras. En particular, la matriz debe desplegar información relacionada con la tipología y prioridad de las alarmas; los gráficos de barras deben desplegar información relacionada con la tipología, prioridad y ubicación				

	temporal de las alarmas.

Tabla 20. Requisito funcional RF-5

Identificador	RF-6	**Tipo**	Funcional	**Prioridad**	Alta
Descripción	La información sobre alarmas registradas los centros de transformación vendrán caracterizados por dos matrices de distribución: una por orden cronológico y otra por ranking de prioridades. En particular, la matriz de orden cronológico debe desplegar información relacionada con la tipología, prioridad y ubicación temporal de las alarmas; la matriz de ranking de prioridades debe desplegar información relacionada con la tipología y prioridad de las alarmas, ordenándolas en función de dicha prioridad.				

Tabla 21. Requisito funcional RF-6

Identificador	RF-7	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Las secciones vendrán caracterizadas por tipos y prioridades.				

Tabla 22. Requisito funcional RF-7

Identificador	RF-8	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Los elementos de todas las vistas deben ser interactivos, a nivel de sección y elemento.				

Tabla 23. Requisito funcional RF-8

Identificador	RF-9	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Las interacciones a nivel de elemento son individuales y a nivel de sección, múltiples.				

Tabla 24. Requisito funcional RF-9

Identificador	RF-10	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Los elementos y las secciones se podrán deseleccionar a través de capacidades de interacción.				

Tabla 25. Requisito funcional RF-10

Identificador	RF-11	**Tipo**	Funcional	**Prioridad**	Media
Descripción	La selección de un elemento y de una sección se verá resaltada a través de la interfaz.				

Tabla 26. Requisito funcional RF-11

Identificador	RF-12	**Tipo**	Funcional	**Prioridad**	Baja
Descripción	Todas las vistas tendrán una etiqueta que identificará qué elemento representan.				

Tabla 27. Requisito funcional RF-12

Identificador	RF-13	**Tipo**	Funcional	**Prioridad**	Media

Descripción	Debe existir coordinación entre vistas para permitir la interoperabilidad entre todos los elementos de las vistas general y detallada.

Tabla 28. Requisito funcional RF-13

Identificador	RF-14	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Esta coordinación se producirá de forma unidireccional entre elementos de distintas sub-vistas.				

Tabla 29. Requisito funcional RF-14

Identificador	RF-15	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Esta coordinación se producirá de forma bidireccional entre elementos de la misma sub-vista.				

Tabla 30. Requisito funcional RF-15

Identificador	RF-16	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Debe existir coordinación entre el gráfico radial y el mapa y la tabla de la vista general.				

Tabla 31. Requisito funcional RF-16

Identificador	RF-17	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Debe existir coordinación entre el gráfico radial y todos los elementos de la vista detallada.				

Tabla 32. Requisito funcional RF-17

Identificador	RF-18	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Debe existir coordinación entre los elementos correspondientes a la vista relacionada con las líneas de distribución y la de los centros de transformación.				

Tabla 33. Requisito funcional RF-18

Identificador	RF-19	**Tipo**	Funcional	**Prioridad**	Media
Descripción	Debe existir coordinación entre los elementos correspondientes a la vista relacionada con los centros de transformación.				

Tabla 34. Requisito funcional RF-19

Identificador	RF-20	**Tipo**	Funcional	**Prioridad**	Baja
Descripción	Los elementos deben emplear los mismos colores cuando se interactúa con ellos.				

Tabla 35. Requisito funcional RF-20

Requisitos no funcionales

Identificador	RNF-1	**Tipo**	No funcional	**Prioridad**	Media
Descripción	El entorno debe poder desplegarse en cualquier navegador web.				

Tabla 36. Requisito no funcional RNF-1

Identificador	RNF-2	**Tipo**	No funcional	**Prioridad**	Alta
Descripción	El usuario deberá tener el *plugin* Silverlight instalado, siendo la versión 5 la óptima.				

Tabla 37. Requisito no funcional RNF-2

Identificador	RNF-3	**Tipo**	No funcional	**Prioridad**	Alta
Descripción	El entorno será visualizado con un monitor de mínimo 42'' Pulgadas.				

Tabla 38. Requisito no funcional RNF-3

Identificador	RNF-4	**Tipo**	No funcional	**Prioridad**	Media
Descripción	La proporción de los tamaños de los elementos del entorno se debe adaptar al tamaño del navegador.				

Tabla 39. Requisito no funcional RNF-4

Identificador	RNF-5	**Tipo**	No funcional	**Prioridad**	Media
Descripción	La escala de colores empleada para las alarmas seguirá el estándar *Abnormal Situation Management* [23] para la clasificación de alarmas.				

Tabla 40. Requisito no funcional RNF-5

Identificador	RNF-6	**Tipo**	No funcional	**Prioridad**	Baja
Descripción	Las etiquetas de los gráficos deben estar escritas en inglés para promover la internacionalización.				

Tabla 41. Requisito no funcional RNF-6