UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR

DE

INGENIEROS DE TELECOMUNICACIÓN

Departamento de Comunicaciones

Proyecto Final de Carrera

MONITORIZACIÓN Y ADQUISICIÓN DE DATOS PARA LA OPTIMIZACIÓN ENERGÉTICA Y REDUCCIÓN DEL IMPACTO MEDIOAMBIENTAL DE UN CENTRO DE PROCESADO DE DATOS

Realizado por:

José Luís López de Pablo Moya

Dirigido por:

José Manuel Mossi García

Noviembre 2013

Agradecimientos

A mi tutor, Dr. José Manuel Mossi García, por su gran ayuda y comprensión mostrada durante el desarrollo de este trabajo ya que en todo momento entendió mi situación facilitándome el estar hoy aquí defendiendo el resumen de tantos años de esfuerzo.

A mi familia y, en especial, a mis padres ya que, aunque no lo exprese, son el pilar sobre el que se sustenta mi vida. Gracias a su esfuerzo, dedicación desinteresada, educación otorgada y valores inculcados a lo largo de mi vida, han hecho de mí la persona que hoy está aquí redactando estas líneas.

A Ana por haberme acompañado a lo largo de todos estos años aceptando mis virtudes y, sobre todo, mis defectos. Hoy termina una etapa de mi vida para empezar un camino juntos.

1. INTRODUCCIÓN

1.1 Motivación

En el mundo actual que nos rodea, tanto en ambientes urbanos como en ambientes rurales, la sociedad continuamente está realizando un consumo de información a través de los múltiples dispositivos dedicados a este fin. Desde los colegios enseñando a los niños el uso de internet, las industrias consultando datos técnicos con los que mejorar sus productos, los hospitales consultando y actualizando el historial médico de los pacientes, los bancos obteniendo el estado de las cuentas de los clientes, los supermercados para facilitar el precio final de la compra al cliente, hasta cualquier transeúnte haciendo uso de su Smartphone o Tablet. Todos tienen en común el uso de datos que no se encuentran físicamente en el mismo lugar que ellos pero, sin embargo, esos datos son necesarios para la realización de sus labores.

Esta continua consulta de datos remotos provoca la necesidad de focalizar toda la información en lugares específicos donde se garantiza el acceso incluso en situaciones críticas. Bajo esta premisa nace un *Centro de Procesamiento de Datos o CPD*[1].

Un *Centro de Procesamiento de Datos* (en adelante CPD) es un edificio o sala destinado a albergar una gran cantidad de equipos electrónicos cuya finalidad es el almacenaje de datos críticos. Debido a dichos datos críticos, un CPD debe ser capaz de garantizar la continuidad de los servicios de los clientes indistintamente del sector al que se dedique el cliente. Garantizar la continuidad de los servicios implica garantizar un acceso restringido a la ubicación física de los equipos electrónicos y, sobretodo, garantizar, en todo momento, la disponibilidad energética necesaria para el correcto funcionamiento de la electrónica destinada al procesamiento de los datos.

Este hecho implica que un CPD es un gran consumidor de energía eléctrica ya que, como se ha indicado anteriormente, se debe garantizar la disponibilidad energética 24 horas al día los 365 días del año.

Figura 1. *Centro de Procesamiento de Datos*

[1] **Centro de procesamiento de Datos:** ubicación física donde se concentran los recursos tanto físicos, lógicos y humanos para el tratamiento de la información de una organización o institución de forma segura.

1.2 Estado actual de los Centros de Procesado de Datos

La actualidad de los Centros de Procesado de Datos pasa por la reducción del consumo energético debido a los altos costes que esto supone y al elevado impacto que tiene sobre el medioambiente. Para este efecto, la Unión Europea ha creado lo que se conoce como *Code of Conduct for Data Centers.* Este código es la respuesta al incremento de gasto energético por parte de los CPDs y es un intento de ser una guía de buenas praxis donde se estimula la búsqueda de eficiencia energética.

El cumplimiento de dicho código es opcional pero, una vez estás aceptado como miembro, se obtiene la certificación como Centro de Procesamiento de Datos eficiente. Actualmente, *Code of Conduct for Data Centers* tiene 82 miembros certificados como eficientes repartidos en diferentes países:

País	Miembros
Reino Unido	32
Francia	15
Paises Bajos	11
Alemania	6
Bélgica	5
Austria	2
España	2
Irlanda	2
Luxemburgo	2
Estados Unidos	1
Grecia	1
Italia	1
Malta	1
Suiza	1

Figura 2. *Países y número de miembros del Code of Conduct for Data Centers*

1.3 Metodología de trabajo

Para la correcta realización de un proyecto necesitamos una exhaustiva planificación y una metodología de trabajo rigurosa donde se garantice el correcto funcionamiento de cada etapa del proyecto.

Este proyecto se ha divido en 4 etapas:

1. Estudio de las características del Centro de Procesamiento de Datos.
2. Despliegue de la red de sensores.
3. Configuración de la red de monitorización.
4. Tratamiento de los datos obtenidos.

Como se ha indicado anteriormente, una rigurosa planificación es esencial en el desarrollo de un proyecto debido a la criticidad de cada etapa. A pesar de ser la primera etapa, el estudio de las características del Centro de Procesamiento de Datos es fundamental ya que, dependiendo de ésta, el despliegue de sensores para la adquisición de datos variará en función de, por ejemplo, la carga de procesamiento de cada zona, el caudal de refrigeración de las diferentes zonas del CPD, criticidad de los servicios alojados, etc.

Una vez finalizado el estudio, se procede al despliegue de los sensores de forma eficiente y su posterior configuración para la adquisición de los datos necesarios. Por último, tras esta recolección de datos, se pasa al procesamiento de dichos datos para poder ver la distribución del consumo energético en la sala.

Figura 3. *Etapas del Proyecto*

1.4 Objetivos

El objetivo principal de este proyecto es el desarrollo de una red de monitorización capaz de recolectar datos como temperaturas, humedades y consumos por toma eléctrica con la intención de encontrar una correlación entre la temperatura y humedad con el consumo energético dentro de un *centro de procesado de datos*. Para ello, se tendrá en cuenta la influencia del diseño de la sala donde se alojan los equipos electrónicos, el sistema de climatización de precisión destinado a mantener la temperatura y la humedad dentro de los márgenes óptimos.

Por lo tanto, los objetivos del presente proyecto pueden ser diferenciados de la siguiente forma:

- Análisis de las características físicas de la *sala técnica*[2].
- Estudio del sistema de climatización ubicado en la sala técnica.
- Despliegue y configuración de la red de sensores para la adquisición de datos.
- Procesamiento de los datos obtenidos.

1.5 Guía de la memoria

Llegados a este punto, procederemos a realizar una breve explicación de los diferentes apartados de la memoria:

- ***Capítulo 2: Estudio previo de las instalaciones***

[2] **Sala técnica**: sala destinada a albergar todos los equipos electrónicos necesarios para el procesamiento de datos de las diferentes empresas.

En este capítulo se procederá a realizar todo el estudio necesario para poder abordar de forma satisfactoria el despliegue de la red de monitorización.

- ***Capítulo 3: Desarrollo de la red de monitorización***
  Despliegue de la red de sensores ubicándolos en posiciones estratégicas.

- ***Capítulo 4: Desarrollo del entorno de evaluación***
  Desarrollo de la GUI desde donde se podrá tratar los datos recogidos por la red de monitorización.

- ***Capítulo 5: Resultados***
  Obtención de los resultados tras los tratamientos de éstos.

- ***Capítulo 6: Conclusiones y Líneas Futuras***
  Conclusiones y líneas futuras que nacen de la visión global otorgada gracias a los anteriores puntos.

- ***Anexo I: Sistemas de Comunicaciones Inalámbricos.***
  Explicación de los diferentes sistemas de comunicaciones inalámbricos alojados en un CPD..

- ***Anexo II: Sistemas de Comunicaciones Alámbricos.***
  Explicación de los diferentes sistemas de comunicaciones alámbricos alojados en un CPD..

- ***Anexo III. Funciones.***
  Código de algunas funciones utilizadas para la elaboración del presente proyecto.

- ***Anexo IV. Índice de figuras.***
  Listado de todas las figuras encontradas en el presente documento.

- ***Anexo V. Glosario.***
  Listado de todos los términos poco conocidos o de difícil interpretación en el contexto del proyecto.

- ***Anexo VI: Bibliografía***
  Documentación utilizada para la realización del presente texto.

2. ESTUDIO PREVIO DE LAS INSTALACIONES

2.1 Objetivo

En este capítulo abordaremos las principales características de las que dispone un CPD. Dentro de esto, analizaremos tanto las características físicas de construcción como las características de los diferentes sistemas que conviven para el correcto funcionamiento del CPD. Haremos mayor hincapié en tres de sus principales sistemas como son:

- Sistema de distribución energética
- Sistema de comunicaciones.
- Sistema de seguridad

2.2 Características típicas de los CPDs

Debido a la alta importancia de los servicios que aloja un CPD, éste cuanta con unas características muy concretas a la hora de su diseño. El diseño de un CPD pasa, en primer lugar, por la elección de su ubicación física la cual necesita un perfecto equilibrio entre varios factores como pueden ser:

- Inversión económica para la construcción del edificio que alojará el CPD.
- Riesgos medioambientales tales como inundaciones, terremotos, tormentas eléctricas, incendios.
- Infraestructuras disponibles en la zona.
- Acometidas eléctricas.
- Medidas de seguridad de la zona.
- Posición final del edificio a:
    - 0.4 Km como mínimo de aeropuertos, ríos, costas o presas con reservas de agua.
    - 0.8 Km como mínimo de bases militares.
    - 1.6 Km como mínimo de centrales nucleares y fábricas de armamento.
    - Nunca se ubicará de forma adyacente a una embajada extranjera.
    - Menos de 0.8 Km de una autopista.

Junto a esto, se debe tener en cuenta otros factores como sistemas de seguridad en las instalaciones, accesos biométricos, identificaciones mediante tarjetas personalizadas e intransferibles, sistemas de climatización de precisión, cableado de red, etc. Todo esto hace que el diseño de un CPD se traduzca en una obra de ingeniería compleja debido a la diversidad de factores que se deben controlar / estudiar y sus diferentes naturalezas.

2.3 Principales Sistemas de los CPDs

Como se ha citado anteriormente, los CPDs están dotados de múltiples sistemas que deben trabajar de forma precisa para garantizar el correcto funcionamiento del CPD a nivel físico y lógico.

Por este motivo, procederemos en las siguientes líneas a explicar la función e importancia de cada sistema ubicado dentro de un CPD.

2.3.1 Sistema Anti-incendio

Los sistemas de protección contra incendios pretender responder a diversos objetivos independientes. El primero de ellos se trata de garantizar la integridad física, una vez detectado un incendio, de toda persona ubicada en el interior de las instalaciones en el momento de la detección. En segundo lugar, se busca proteger los activos de los diferentes clientes debido al ser un CPD un punto clave en todas las empresas. El tercer objetivo es garantizar la continuidad de los servicios en el menor tiempo posible en caso de haber un incendio en las instalaciones.

Debido a todo lo anteriormente nombrado, los sistemas anti-incendio instalados en los CPDs deben ser capaces de sofocar un incendio sin provocar lesión alguna a las personas que se encuentren en su interior y, a su vez, deben ser capaces de sofocar un incendio sin dañar los equipos electrónicos alojados en su interior.

Los sistemas anti-incendio dedicados a los CPDs suelen estar compuestos de dos sistemas coexistiendo en busca de un objetivo común:

- **Sistema de detección precoz de humos:** son aquellos sistemas capaces de, en tiempo real, realizar un análisis de la sala técnica y de cada elemento en busca de anomalías y, mediante detectores de alta sensibilidad, poder responder de forma precoz mediante la aspiración de los humos generados.
- **Sistema de extinción mediante agua nebulizada:** el principio básico de este sistema radica en la supresión del fuego aplicando agua proyectada a alta presión con el objetivo de reducir al máximo el tamaño de la gota. Siguiendo este principio se obtiene un excelente poder de refrigeración con un consumo de agua mucho menor que los sistemas convencionales basado en la emisión de agua a baja presión.

Figura 4. *Sistema Anti-incendio Convencional vs. Sistema de agua nebulizada*

2.3.2 Sistema Seguridad

Uno de los sistemas más importantes dentro de un CPD es el sistema de seguridad. Cuando se habla de sistema de seguridad de un CPD, éste no hace referencia únicamente a seguridad física sino, también, a seguridad lógica capaz de garantizar la inaccesibilidad a los datos de los clientes por parte de terceros.

Por lo tanto, y al tratarse de un sistema cuya función está claramente diferenciada, procederemos a analizar por separado ambos.

2.3.2.1 Seguridad Física

Dentro de esta categoría podemos encontrar todas aquellas medidas destinadas a restringir el acceso a las instalaciones a personal no autorizado. Múltiples son los mecanismos instalados para dicha tarea como pueden ser:

- Cámaras perimetrales en la zona exterior.
- Mecanismos de detección y alarma.
- Control de acceso mediante sistemas biométricos, cerraduras magnéticas, tarjetas identificativas intransferibles, etc.
- Registro de acceso a la sala técnica.
- Circuitos cerrados de televisión en la sala técnica.
- Posibilidad de compartimentación de la sala. Cada cliente puede tener la posibilidad de alojar sus equipos dentro de una zona rodeada por una valla donde el acceso solo estará autorizado para el propietario y los técnicos del CPD.

Por tanto, los CPDs son lugares de una alta seguridad donde el acceso a su interior está limitado a personal altamente cualificado.

2.3.2.2 Seguridad Lógica

Esta categoría alberga todas las herramientas destinadas a bloquear el acceso de terceros mediante las redes de comunicaciones para obtener datos de los clientes. Los mecanismos a tener en cuenta dentro de la seguridad lógica pueden ser:

- Sistemas Operativos actualizados debido a la corrección de vulnerabilidades detectadas en sus versiones anteriores.
- Equipamientos de seguridad informática como firewalls, honeypots[3], etc.
- Cifrado: imprescindible garantizar el cifrado de los datos en los puntos más críticos de ser captados por terceros.
- Accesos con permisos especiales para ciertos usuarios debido a que son los encargados de operar los servicios.
- Accesos remotos. Crítico el otorgar un nivel de privilegio adecuado para que cada cliente pueda realizar sus servicios y entender cómo realiza cada cliente su conexión. Los métodos más utilizados son mediante VPN o circuitos MPLS[4] dedicados.

Cabe destacar que estos son algunos de los múltiples mecanismos que garantizan la seguridad lógica de los datos almacenados en un CPD.

2.3.3 Sistema de Climatización

El sistema de Climatización de un CPD juega un papel clave en el buen funcionamiento de las instalaciones debido a que es el encargado de mantener unos niveles de temperatura y humedad constantes en toda la sala técnica. Debido a esta premisa, el sistema de Climatización es uno de los principales focos de consumo energético dentro de un CPD ya que debe contrarrestar el calor disipado por todos los equipos ubicados en su interior garantizando una temperatura de trabajo óptima para dichos equipos.

[3] **Honeypots:** conjunto de computadoras cuya labor es simular una vulnerabilidad para atraer atacantes y, de esta forma, poder analizar las técnicas utilizadas por los atacantes para protegerse frente a verdaderos ataques.
[4] **MPLS (MultiProtocol Label Switching).** Tecnología basada en la virtualización de circuitos sobre redes IP que permite redes privadas virtuales con protección frente a fallos y soporte multiprotocolo.

La evolución de los sistemas de climatización de precisión ha sido vertiginosa debido a la necesidad de cancelar el incremento de calor disipado debido al desarrollo de la tecnología de virtualización con la consiguiente concentración de servidores más potentes en un menor espacio. Esto provocó la aparición de la técnica conocida como *pasillo frío – pasillo caliente* basada en la organización de las caras de los armarios encargados de alojar los equipos de forma que las partes calientes estuviesen enfrentadas y, por consiguiente, las partes frías también estuviesen enfrentadas. En la *Figura 5* se puede observar la técnica de *Pasillo frío – Pasillo Caliente*. Dicha figura muestra la distribución de las caras de los armarios y permite observar la diferencia de temperatura que se logra en cada una de las caras encontrando en las zonas cálidas las partes disipadoras de calor de cada equipo.

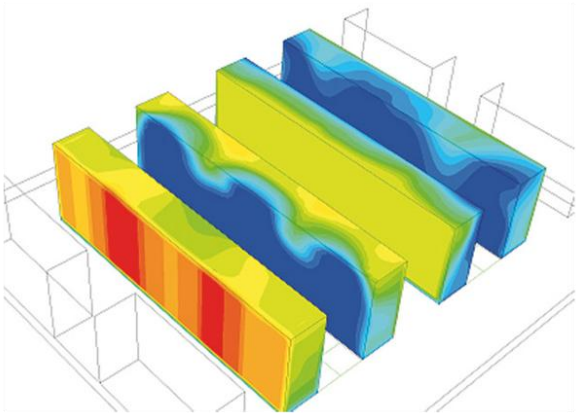


Figura 5. *Ilustración de la técnica de Pasillo Frío – Pasillo Caliente*

Una evolución de esta técnica se basa en la encapsulación de las zonas frías para lograr separar los caudales fríos y calientes y, de esta forma, optimizar el poder frigorífico del caudal suministrado por el sistema de climatización. En la *figura 6* se puede observar el encapsulamiento del pasillo frío y la independencia de caudales siendo el caudal frío suministrado a los racks a través de un suelo técnico y, por el contrario, el caudal caliente recogido por la parte superior debido a su menor densidad.

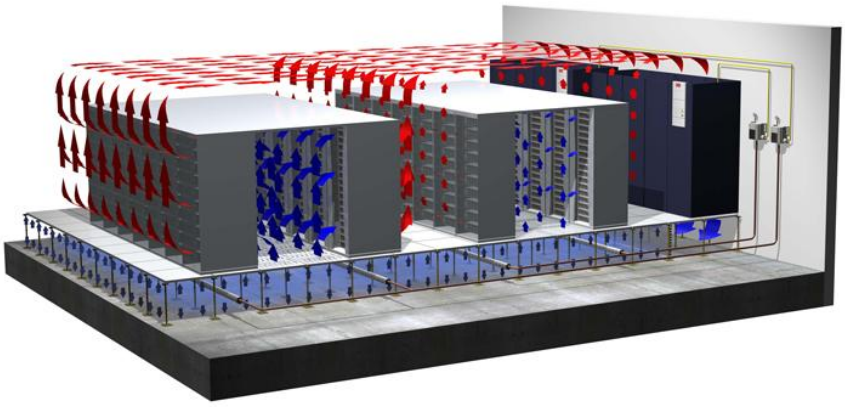


Figura 6. *Ilustración de la técnica de Pasillo Frío – Pasillo Caliente con encapsulamiento del pasillo frío.*

2.3.4 Sistema de Comunicaciones

Como se ha indicado anteriormente, en un Centro de Procesado de Datos coexisten diferentes sistemas que deben trabajar de forma conjunta para garantizar el correcto funcionamiento. Dentro de estos sistemas podemos encontrar el sistema de comunicaciones, el cual aloja cualquier tipo de red alámbrica o inalámbrica disponible en las instalaciones.

A continuación procederemos a describir de forma breve las principales características de dichos sistemas.

2.3.4.1 Sistemas inalámbricos

Los sistemas de radiocomunicaciones son aquellos basados en la transmisión de la información mediante las ondas de radio. Dichas ondas se caracterizan por el movimiento de los campos eléctricos y magnéticos simultáneamente. Al tratarse de una comunicación inalámbrica, ésta es vulnerable a las numerosas interferencias que puede encontrar durante su trayecto. Debido a este problema, es fundamental tener en cuenta diferentes parámetros a la hora de diseñar una comunicación basada en ondas de radio. Para obtener mayor información acerca de los sistemas inalámbricos consultar el Anexo I *Sistemas de Comunicaciones Inalámbricos.*

2.3.4.2 Sistemas Alámbricos

Los sistemas de comunicaciones alámbricos son aquellos en los cuales disponemos de un soporte físico para la transmisión de toda señal eléctrica. Dentro de estos soportes, podemos encontrar los cables de pares trenzados, cables coaxiales o fibras ópticas. Dependiendo del soporte utilizado, las tasas de transferencias y anchos de banda variarán de forma drástica siendo la elección del soporte físico clave para el correcto funcionamiento del sistema de comunicación alámbrica. Para obtener más información a cerca de los sistemas de comunicaciones alámbricos consultar Anexo II *Sistemas de Comunicaciones Alámbricos.*

2.3.4 Sistema energético

Los Centros de Procesado de Datos son centros altamente dependientes de un suministro energético capaz de mantener en funcionamiento los equipos electrónicos que aloja en su interior. No pueden permitir quedarse sin suministro energético en ningún momento.

Bajo esta premisa, los CPD's son salas dotadas de múltiples sistemas energéticos capaces de trabajar en situaciones de fallo de otro sistema por lo que se asegura la disponibilidad de la información en todo momento.

Los principales sistema de suministro energético serían:

- ***Red eléctrica.***

  Suministro energético principal y más común basado en la recepción de la energía mediante las redes de distribución nacional de alta tensión. Es una fuente de energía contratada a una empresa externa que garantiza el suministro de forma ininterrumpida.

  Habitualmente, los centros de procesado de datos poseen un centro de transformación propio capaz de eliminar las impurezas de la red eléctrica y, por tanto, suministrar una energía más estable a la sala técnica.

- ***Motores de gasoil (Grupos electrógenos)***

  Como existe la posibilidad de fallo en la red eléctrica debido, por ejemplo, a problemas climatológicos, los centros de procesados de datos cuentas con motores de gasoil capaces de soportar toda la carga energética demandada por la sala técnica durante horas.

Dichos motores electrógenos arrancan automáticamente tras detectar una caída de la red eléctrica gracias a la continua monitorización que se hace de dicha red. Gracias a esto, el tiempo transcurrido entre la detección de falta de tensión y el arranque del grupo electrógeno no supera los 30 segundos.

La ubicación de los grupos electrógenos suele ser en exterior debido a la gran cantidad de calor generado lo que produciría un incremento indeseado de la temperatura en la sala técnica.

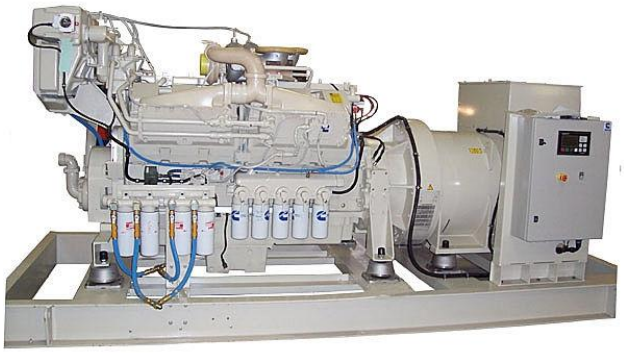


Figura 7. *Grupo electrógeno industrial destinado a garantizar el suministro eléctrico.*

- ***Sistemas de Alimentación Ininterrumpidas o SAIS.***

Los SAIs son, habitualmente, el último dispositivo destinado a garantizar el suministro eléctrico de los equipos de procesamiento de datos. Están dotados de filas de baterías con una autonomía variable dependiendo de la carga que deba soportar cada Sai. Además de garantizar el suministro eléctrico, los Sais también son los encargados de transformar la corriente trifásica de entrada en corriente continua adaptada a los equipos y limpia de todo tiempo de impurezas.

Figura 8. *Sistema de Alimentación Ininterrumpida industrial.*

2.4 Clasificación de los CPDs

La clasificación de los CPDs se basa en una escala de 4 elementos llamados *TIER (Tecnical Independent Evaluation Repor).* Los *TIER I, II, III o IV* son los cuatro niveles de designación creadas por el *Uptime Institute* para clasificar los Centros de Procesado de Datos en función de la disponibilidad de sus servicios.

2.4.4 TIER I

Los TIER I son los CPDs más simples y más vulnerables a fallas energéticas. Sus principales características son:

- Una única fuente de suministro energético.
- Un único SAI encargado de soportar la carga de la sala técnica y el filtrado de impurezas de la red eléctrica.
- Un único proveedor de datos.
- Inexistencia de suelo técnico.
- Errores de operación o fallas en los componentes de su infraestructura implican la interrupción del Data Center.

Como se puede observar, los CPDs TIER I son vulnerables energéticamente hablando debido a que un fallo en la red o el SAI implica una casi segura caída de los equipos alojados.

La disponibilidad garantizada en un TIER I es de 99.671 % del tiempo.

Debido a estas limitaciones, estos CPDs suelen estar destinados a pequeñas empresas o servicios internos.

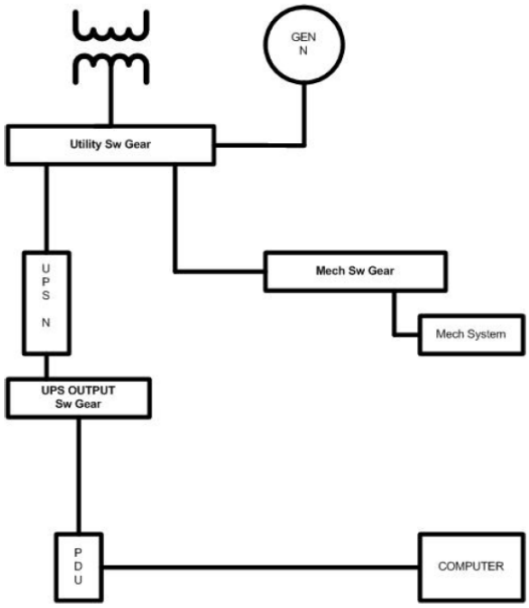


Figura 9. *Infraestructura de un CPD TIER I.*

2.4.5 TIER II

Los CPDs catalogados como TIER II tienen las mismas características que los TIER I además de estar formados por dispositivos con componentes redundantes. La aparición de componentes redundantes aumenta la disponibilidad de los servicios ofertados por la empresa propietaria del CPD hasta un 99.741% del tiempo.

Principales características de los CPDs TIER II:

- Componentes Redundantes
- Existencia de suelo técnico
- Existencia de Sistemas de Alimentación Ininterrumpida y generadores eléctricos
- Existencia de una única línea de distribución eléctrica.
- Diseño N+1. Duplicado de cada componente de la infraestructura.

Los TIER II suelen estar destinados a pequeños negocios, compañías que no ofrecen servicios “online” o “real-time”.

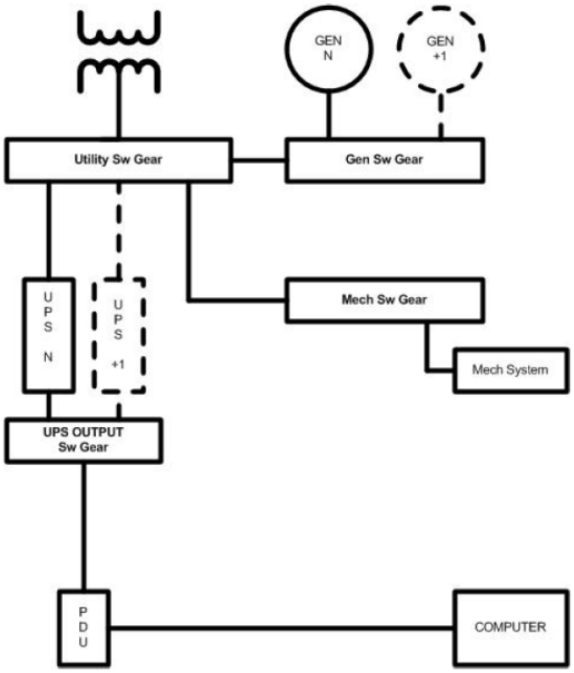


Figura 10. *Infraestructura de un CPD TIER II.*

2.4.6 TIER III

Los CPD's TIER III aparece a principios de los años 80 y se caracterizan por un mantenimiento simultáneo. Son la evolución de los TIER II e incorporan importantes niveles de tolerancia ante fallos al contar con todas sus infraestructuras redundadas. Esto quiere decir que, incluso, el suministro eléctrico está redundado lo que permite una configuración Activo / Pasivo de los diferentes sistemas.

Todos los TIER III se caracterizan por contar, idealmente, con todos sus equipos con doble fuente de alimentación lo que permite un mantenimiento en uno de los ramales sin afectar al servicio.

Principales características de los CPDs TIER III.

- Mantenimiento simultáneo
- Componentes redundantes en estructura N+1.
- Múltiples líneas de distribución eléctrica y de refrigeración aunque una sola activa.
- Susceptibles a actividades no planeadas.
- Disponibilidad de los servicios del 99.982%.

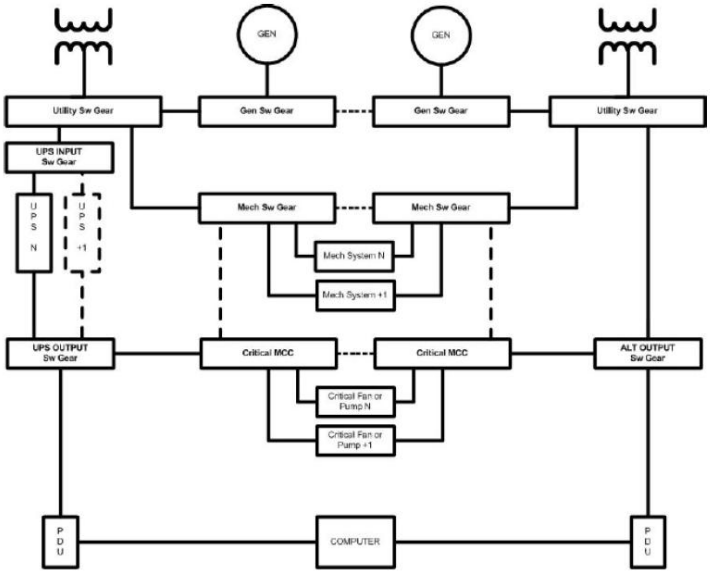


Figura 11. *Infraestructura de un CPD TIER III.*

2.4.7 TIER IV

Nivel más exigente para los CPDs. Un CPD TIER IV es aquel que está diseñado para soportar cualquier fallo sin producirse un corte de los servicios. Cumple con el TIER III e incorpora redundancia 2(N+1)[5] en todos los puntos críticos.

Esto implica que para que un TIER IV deje de dar servicio por un problema en el suministro energético se debería dar el siguiente caso:

- Pérdida de suministro eléctrico en ambas ramas.
- Fallo de 2 o más grupos electrógenos diesel o motores de gas en cada una de las líneas de suministro.

Por lo tanto, un TIER IV es aquel que dispone de múltiples líneas de distribución energética y de refrigeración con componentes redundantes 2(N+1). La disponibilidad de un TIER IV es del 99.995%.

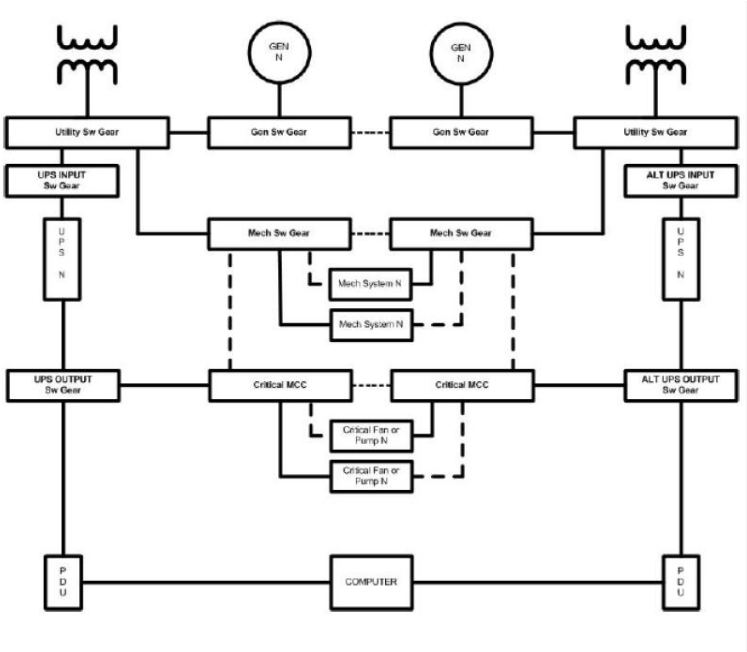


Figura 12. *Infraestructura de un CPD TIER IV*

[5] Configuración 2(N+1): 2 líneas de suministro eléctrico y cada una con redundancia N+1

A continuación se muestra una tabla resumen con las principales características de los cuatro niveles de clasificación para los CPDs.

	Tier I	Tier II	Tier III	Tier IV
Building Type	Tenant	Tenant	Stand-alone	Stand-alone
Staffing shifts Staff/shift	None None	1 Shift 1/Shift	1+Shifts 1-2/Shift	"24 by Forever" 2+/Shift
Useable for Critical Load	100% N	100% N	90% N	90% N
Initial Build-out kW per Cabinet (typical)	<1kW	1-2 kW	1-2 kW	1-3 kW
Ultimate kW per Cabinet (typical)	<1 kW	1-2 kW	>3 kW†,‡	>4 kW*,†
Support Space to Raised-Floor Ratio	20%	30%	80-90+%	100+%
Raised-Floor Height (typical)	12 inches	18 inches	30-36 inches	30-42 inches
Floor Loading lbs/ft (typical)	85	100	150	150+
Utility Voltage (typical)	208, 480	208, 480	12-15 kV	12-15 kV
Single Points-of-Failure	Many + Human Error	Many + Human Error	Some + Human Error	Fire, EPO + Some Human Error
Representative Planned Maintenance Shut Downs	2 Annual Events at 12 Hours Each	3 Events Over 2 Years at 12 Hours Each	None Required	None Required
Representative Site Failures	6 Failures Over 5 Years	1 Failure Every Year	1 Failure Every 2.5 Years	1 Failure Every 5 Years
Annual Site-Caused, End-User Downtime (based on field data)	28.8 hours	22.0 hours	1.6 hours	0.8 hours
Resulting End-User Availability Based on Site-Caused Downtime	99.67%	99.75%	99.98%	99.99%
Typical Months to Plan and Construct	3	3-6	15-20	15-30
First Deployed	1965	1970	1985	1995

Figura 13. *Comparativa entre los diferentes nivel de TIER*

2.5 Consumo Energético

Como ya se ha indicado en ocasiones anteriores, los Centros de Procesado de Datos son grandes consumidores de energía. Este hecho es debido a la gran cantidad de sistemas diferentes que conviven dentro del CPD para garantizar su funcionamiento. Desde el sistema eléctrico pasando por sistemas de refrigeración industrial de precisión, sistemas de alta seguridad, grupo electrógenos, Sais, equipos TI… Todos estos dispositivos dependen de un suministro energético ininterrumpido para su funcionamiento. Muchas empresas abogan por incluir en sus CPDs técnicas descritas por Green IT para reducir el consumo energético mediante la virtualización de los servidores. Aunque es un paso inicial, no es suficiente para reducir los grandes consumos energéticos de los Centros de Procesado de Datos. Para poder ver el gran problema energético desde una visión global, según el New York Times, los CPDs de todo el mundo consumen un promedio de 30.000 millones de Vatios anualmente lo que equivale a un 1,5% de todo el consumo mundial.

Tras este estudio previo de las instalaciones físicas de un CPD podemos decir que el consumo energético es uno de los mayores problemas a los que se enfrentan los Centro de Procesado de Datos debido a que son instalaciones que, necesariamente, deben estar alimentadas ininterrumpidamente para su correcto funcionamiento. A raíz de esta problemática nace la finalidad de este proyecto: Monitorización y Adquisición de datos para la optimización energética y reducción del impacto medioambiental de un Centro de Procesado de Datos. Para ello, dicho proyecto ha sido planteado en tres partes claramente diferencias:

- *Desarrollo de la red de monitorización.*

  En dicho capítulo nos centraremos en el despliegue de la red de monitorización seleccionando las ubicaciones óptimas en función de la carga de los diferentes rack, el caudal de refrigeración que hay en cada zona y la distancia a las diferentes climatizadoras. Junto a esto, se analizará la forma de recoger la información obtenida por los sensores. Este apartado será descrito en profundidad en el capítulo 3 de este documento.

- *Desarrollo del entorno de evaluación.*

  En este capítulo se desarrolla la aplicación gráfica encarga de mostrar al usuario los datos recogidos. Para ello, analizaremos las necesidades de nuestra aplicación. En primer lugar abordaremos la necesidad de dónde ubicar nuestro entorno de trabajo. Esto es, máquina física frente a máquina virtual. A continuación abordaremos la elección del software de monitorización encargado de la recolección de datos. Una vez desplegado el sistema de monitorización trataremos qué tipo de base de datos es más acorde a nuestras necesidades. Realizaremos un pequeño análisis entre MySQL y Oracle. Por último, y no por ello menos importante, seleccionaremos el lenguaje de programación y de desarrollo de la interfaz gráfica más óptimo para el manejo de grandes cantidades de datos. Todo esto se verá detalladamente en el capítulo 4 de este documento.

- *Resultados.*

  En este capítulo analizaremos los resultados obtenidos y el correcto funcionamiento de la red desplegada y analizada. Para ello nos ayudaremos de los resultados obtenidos a través de la interfaz gráfica desarrollada en el capítulo 4.

3. DESARROLLO DE LA RED DE MONITORIZACIÓN

3.1 Objetivo

En este capítulo nos centraremos, una vez analizado las características de nuestra sala técnica, en el despliegue de la red de monitorización.

Para ello se determinará las posiciones más adecuadas teniendo en cuenta parámetros como criticidad de los servicios que alojan, ubicación en la sala, caudal de climatización en las diferentes zonas, etc.

El objetivo final es la adquisición de datos fiables para intentar reducir el consumo energético por lo que la colocación de los sensores en las posiciones óptimas juega un papel fundamental.

3.2 Importancia de la red de Monitorización

Una red de monitorización se puede definir como la utilización de un sistema informático capaz de averiguar el estado de los componentes que conformen nuestra empresa, ya sean hardware o software, con el fin de detectar y notificar un problema o mal funcionamiento en los diferentes componentes.

Debido a la criticidad de los servicios alojados dentro de un CPD, la red de monitorización juego un papel clave ya que es la que nos permite obtener información entendible por el ser humano de los valores registrados por las máquinas. Desde temperaturas, humedades, consumos eléctricos de los equipos, frecuencias de trabajo de los disco duros hasta estado de las baterías de los dispositivos. Son múltiples las variables que se puede monitorización y, por tanto, gestionar y optimizar su funcionamiento en un entorno de producción.

Como consecuencia de los múltiples dispositivos de diferentes naturalezas que es posible monitorizar, son múltiples los protocolos que pueden ser empleados en la obtención de la información requerida.

Algunos ejemplos de estos protocolos serían los siguientes:

- SMTP: protocolo utilizado para la monitorización del estado de un servidor email mediante el envío de un mensaje de prueba en un intervalo de tiempo requerido.

- HTTP: protocolo utilizado para monitorizar el estado de un servidor web mediante el envío de una petición cada x minutos.

- Otros posibles protocolos serían:
    - HTTPS
    - FTP
    - POP3
    - IMAP
    - SSH
    - TELNET
    - TCP
    - UDP
    - SNMP

3.3 Protocolo SNMP

El Protocolo Simple de Administración de Red o SNMP (Simple Network Management Protocol) pertenece a los protocolos de capa de aplicación. Es un protocolo destinado a facilitar el flujo de información entre los diferentes dispositivos que pertenecen a la red y el administrador de la red. Su principal característica es la supervisión del funcionamiento de la red y la búsqueda y resolución de problemas encontrados en la misma. Debido a su sencillez, permite una expansión de la red sin aumentar la complejidad de gestión.

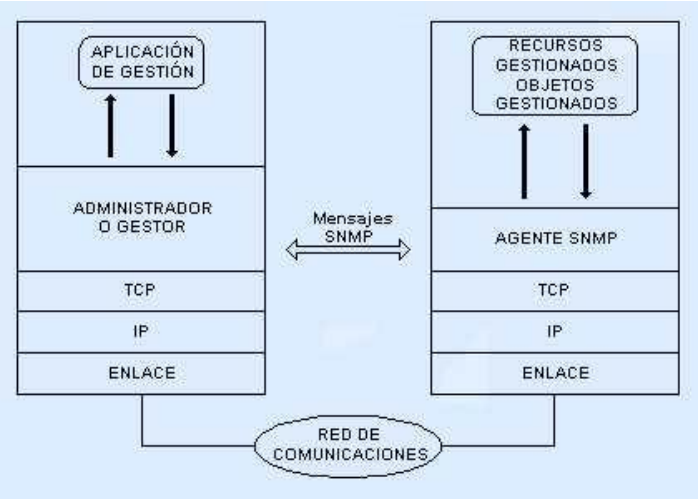


Figura 14. *Capa de utilización de SNMP*

Toda red gestionada a través de SNMP tiene tres puntos en común:

- Elementos administrados: dispositivos que contienen la información del hardware que se desea monitorizar. Serán los elementos preguntados para obtener dicha información.
- Agentes: aplicación software encargada de traducir los valores monitorizados a paquetes SNMP con la finalidad de poder ser transmitidos.
- Sistemas de administración de red: terminal desde el cual los administradores puede realizar las consultas necesarias para conocer el estado de la red.

La interactuación de estos tres elementos se basa en la solicitud de información (polling) por parte del administrador y la respuesta de los agentes a la solicitud. Para realizar dichos trabajos, SNMP consta de cinco tipos de mensajes intercambiados entre Agentes y Administrador:

- Get Request

Una petición del Administrador al Agente para que envíe los valores contenidos en el archivo MIB (Management Information Base).

- Get Next Request

Una petición del Administrador al Agente para que envíe los valores contenidos en el MIB referente al objeto siguiente al especificado anteriormente.

- Get Response

La respuesta del Agente a la petición de información lanzada por el Administrador.

- Set Request

Una petición del Administrador al Agente para que cambie el valor contenido en el MIB referente a un determinado objeto.

- Trap

Un mensaje espontáneo enviado por el Agente al Administrador, al detectar una condición predeterminada, como es la conexión/desconexión de una estación o una alarma.

3.3.1 Archivo MIB

Los archivos MIB (Management Informaction Base) son un tipo de base de datos con información jerárquica, estructurada en forma de árbol y define las variables utilizadas por el protocolo SNMP para el control y la supervisión de los diferentes componentes de red. Está basado en la definición de objetos los cuales representan los dispositivos a monitorizar en la red. Cada objeto dentro de una MIB tiene un identificado único el cual se accede descendiendo por la estructura de árbol.

En caso de ser necesaria la monitorización de dispositivos concretos (UPS, PDUs, Climatizadoras,…) , cada fabricante debe proporcionar el archivo MIB para saber cómo poder monitorizar las variables necesarias. Por lo tanto, los archivos MIB son archivos públicos lo cuales podemos encontrar en las propias páginas web de los fabricantes.

Un ejemplo de estructura MIB sería la imagen de la figura :

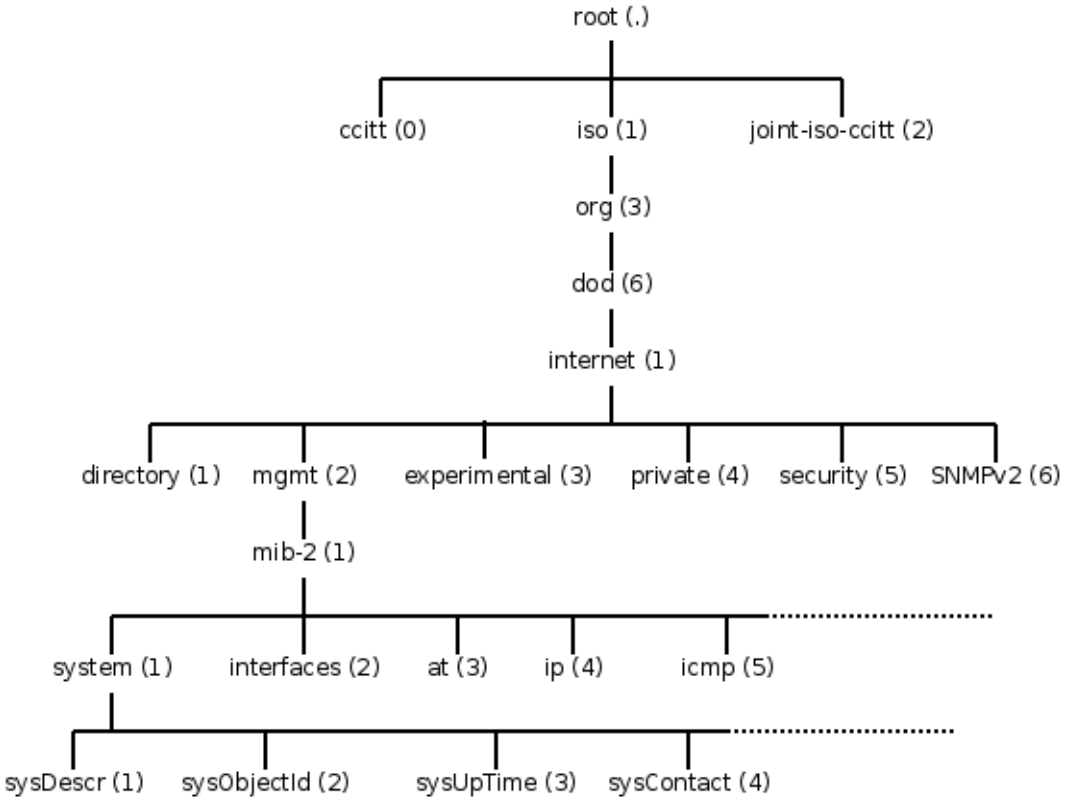


Figura 15. *Estructura de árbol de una MIB*

En el ejemplo anterior, si quisiésemos acceder a la variable *sysUpTime* deberíamos descender por el árbol hasta llegar a su nodo. La forma de identificar cada nodo se corresponde con el número que tiene entre paréntesis. Por lo tanto, para acceder a *sysUpTime* realizaríamos el siguiente camino:

iso(1) - org(3) - dod(6) - internet(1) - mgmt(2) - mib-2(1) - system(1) – sysUptime (3) . Una vez identificado el nodo deberíamos pasar la ruta única al agente para que nos devolvieses el valor requerido. En nuestro caso sería 1.3.6.1.2.1.1.3 . Con esto, el agente entiende que le estamos indicando el nodo correspondiente a la variable *sysUpTime*.

Lógicamente la estructura de árbol se trata de una representación visual del archivo MIB. Por lo tanto, si queremos obtener cierta información de nuestros dispositivos, debemos ser capaces de leer y entender un archivo MIB para poder obtener el identificador único de la variable necesaria.

A continuación se muestra un ejemplo de la estructura que encontramos dentro de las MIBs.

upsmgIdentFamilyName OBJECT-TYPE → Nombre del objeto

SYNTAX DisplayString(SIZE(0..32)) → Tipo de dato que encontraremos

ACCESS read-write → Tipo de acceso a la variable. Sólo lectura o lectura/escritura.

STATUS mandatory → Estado de la variable. Obligatoria, Opcional.

DESCRIPTION → Breve descripción del significado de la variable.

"The UPS family name ('GALAXY 7000')."

::= { upsmgIdent 1 } → Nodo del que cuelga e identificador dentro de ese nodo

Entendiendo la nomenclatura utilizada en una MIB debemos ser capaces de consultar el agente para obtener la información necesaria de la variable requerida.

3.3 Tipos de dispositivos

Dependiendo de la red de monitorización a interrogar, los dispositivos variarán en función de las necesidades de dicha red y, por ende, de dicha empresa.

En el caso concreto de un Centro de Procesado de Datos y, más concretamente la finalidad de dicho trabajo, se basa en la monitorización de la red de sensores desplegada en la sala técnica con la cual poder obtener las temperaturas y humedades para el posterior análisis y procesado de dichos valores. En este caso en particular, los sensores desplegados están conectados a una PDU monitorizable o Power Distribution Unit capaz de recoger los datos de temperaturas, humedades y consumos eléctricos por cada una de las 8 tomas de corriente disponibles.

Figura 16. *PDU o Power Distribution Unit*

Junto a esto, se ha optado por monitorizar también los Sais para poder analizar, posteriormente, la relación que hay entre potencia demandada y calor generado por los diferentes equipos TI en los racks.

Debido a la complejidad tecnológica de un SAI, éstos son capaces de monitorizar gran cantidad de variables. Para nuestro estudio en cuestión únicamente tendremos en cuenta las siguientes variables:

- Tensión a la entrada de la Fase 1
- Tensión a la entrada de la Fase 2
- Tensión a la entrada de la Fase 3
- Potencia a la salida de la Fase 1
- Potencia a la salida de la Fase 2
- Potencia a la salida de la Fase 3
- Carga de la Fase 1
- Carga de la Fase 2
- Carga de la Fase 3
- Estado de la Batería
- Estado del Sai
- Modo Bypass
- Modo Batería
- Modo Normal

4. DESARROLLO DEL ENTORNO DE EVALUACIÓN

4.1 Objetivo

En este capítulo desarrollaremos la herramienta encargada de acceder a los datos monitorizados para, posteriormente y desde la misma aplicación, proceder al tratamiento de éstos.

Para el desarrollo de la aplicación habrá que tener en cuenta la finalidad de dicha aplicación ya que dicho factor será clave para determinar el lenguaje de desarrollo acorde a las prestaciones necesarias.

4.2 Estudio de las necesidades del entorno de evaluación

Para el desarrollo efectivo de nuestro entorno de evaluación debemos analizar en primer lugar cuáles van a ser las necesidades de dicho entorno. Desde un punto de vista global, nuestra aplicación necesitará ser capaz de procesar un gran número de datos. Dichos datos serán los obtenidos a través de SNMP y deben estar ubicados en un lugar accesible por nuestra aplicación. Por lo tanto, necesitaremos crear una base de datos concreta para nuestra aplicación. Junto a esto, la base de datos deberá estar alojada en una máquina, física o virtual, que me permita gestionarla y consultar con una garantía total de disponibilidad.

Por lo tanto, necesitaremos:

- Una máquina dedicada que garantice el acceso y la gestión de la base de datos.
- Software de monitorización vía SNMP.
- Una base de datos capaz de gestionar un gran número de registros en su interior.
- Un lenguaje de programación capaz de trabajar con un gran volumen de datos en cada consulta que realice.
- Un lenguaje de programación capaz de generar una interfaz gráfica con la cual poder interactuar.

4.2.1 Selección de la máquina para alojar la base de datos

En los entornos de producción actuales tenemos varias posibilidades dependiendo de los servicios que vayamos a ofrecer. Debemos analizar los requisitos que necesitará nuestra aplicación para poder optar, de forma eficiente, por el tipo de alojamiento más adecuado.

Factores que deberíamos tener en cuenta serían:

- Disponibilidad del servicio.
- Capacidad de alojamiento necesario.
- Criticidad del servicio.
- Disponibilidad geográfica.
- Disponibilidad económica.

Estos factores serán la clave para poder seleccionar el tipo de alojamiento óptimo. Por ejemplo, si la disponibilidad económica es suficiente, se puede optar por un alojamiento / servidor físico el cual controlaremos por completo. Desde el sistema operativo hasta los recursos disponibles en el mismo. Otra ventaja de los servidores físicos es la despreocupación del software que hay en él debido a que tú mismo eres quien lo gestiona. Junto a la disponibilidad económica debe ir ligada la disponibilidad geográfica ya que, en caso de problema, no tendríamos un acceso rápido a nuestra máquina y, por consiguiente, el servicio se vería afectado.

La otra posibilidad disponible hoy en día y que brinda una gran flexibilidad es la opción de la virtualización. La principal ventaja es la reducción del coste de un servidor virtual frente a un servidor físico ya que no necesitas comprar un equipo y buscar un lugar donde ubicarlo. Adquieres unas prestaciones (espacio de disco duro, memoría RAM, número de procesadores …) y obtienes el servicio. Por el contrario, pierdes el control de dónde está tu servidor ya que, dentro de una plataforma virtual, los equipos virtualizados pueden ser movidos de un cluster a otro sin afectar al servicio.

Habiendo planteado estás diferencias, en nuestro caso se optará por la virtualización debido a la flexibilidad que ofrece y el menor coste de puesta en marcha de la plataforma.

Se ha optado por crear una plataforma virtual de 4 núcleos con 4 GB de RAM y 200 GB de disco duro. Sobre esta plataforma virtual trabajará un Sistema Operativo CentOS 6. Para realizar la conexión a dicha máquina virtual se procederá vía SSH a través del software *Putty*.

4.2.2 Software de monitorización vía SNMP

Una vez tenemos la máquina encargada de alojar nuestros datos, debemos seleccionar el software encargado de realizar las consultas vía SNMP a los agentes de cada dispositivo. Múltiples son los programas encargados de realizar este trabajo por lo que tendremos que buscar el que mejor se acople a nuestras necesidades o nos otorgue una funcionalidad extra.

En nuestro caso se ha optado por utilizar un sistema de monitorización de distribución libre como puede ser *MRTG* (Multi Router Traffic Grapher). La principal ventaja de *MRTG* frente al resto de opciones ya sea, por ejemplo, Nagios, es la capacidad de generar un informe HTML con una representación visual en forma de gráficas con la evolución de la variable a lo largo del tiempo. Esta característica la utilizaremos en caso de emergencia ya que nosotros vamos a crear una interfaz para el procesado de dichos datos.

MRTG puede ser invocado o ejecutado como *daemon* en nuestra máquina a través de la configuración como tarea programada en *cron.* Debido a la continua necesidad de recolectar esos datos, se ha optado por configurarlo como una tarea programada en *cron. MRTG* puede ejecutar cualquier script que se necesite por lo que proporciona una gran potencia a la hora de monitorización.

El período de monitorización con *MRTG* por defecto es de 5 minutos. Este valor puede ser cambiado según las necesidades del sistema. En nuestro caso, se ha optado por tomar la configuración por defecto.

Una vez instalado el software en nuestra máquina virtual, pasamos a la configuración de las variables a monitorizar. Para proceder con dicha configuración, debemos editar el archivo *mrtg.cfg* localizado en el directorio */etc/mrtg.*

Para cambiar la configuración de mrtg debemos editar el fichero mrtg.cfg donde figuran las variables a monitorizar o, también, pueden estar los archivos de configuración que se deben evaluar.

```
[mrtg@          mrtg]$ cat mrtg.cfg
##################################################################
# Multi Router Traffic Grapher -- Example Configuration File
##################################################################
# This file is for use with mrtg-2.0
#
# mrtg.cfg
#--------------------

#Sirve para ejecutar MRTG como daemon del sistema
RunAsDaemon: Yes

Language:spanish
Interval:5
Forks: 8


#Directorios de trabajo de MRTG (web...)
HtmlDir: /var/www/mrtg
ImageDir: /var/www/mrtg
ThreshDir: /var/lib/mrtg
EnableSnmpV3: yes

#Directorio donde MRTG deposita los datos de los elementos a monitorizar
LogDir: /var/lib/mrtg


#Ficheros de configuracion que MRTG carga al inicio
Include:          _pdu.cfg
Include:          _pdu.cfg
Include:          _sai.cfg
Include:          _sai.cfg
```

Como se puede observar, se ha optado por la opción de indicar qué ficheros debe cargar al inicio debido a la gran cantidad de variables a monitorizar.

Si analizamos uno de esos archivos de configuración podemos observar que se ha seguido una estructura definida para darle una coherencia y facilitar la lectura y el procesamiento.

SAMPLE: ITEM ADD

#Target[identificador_dispositivo]: OID 1& OID2:public@IP

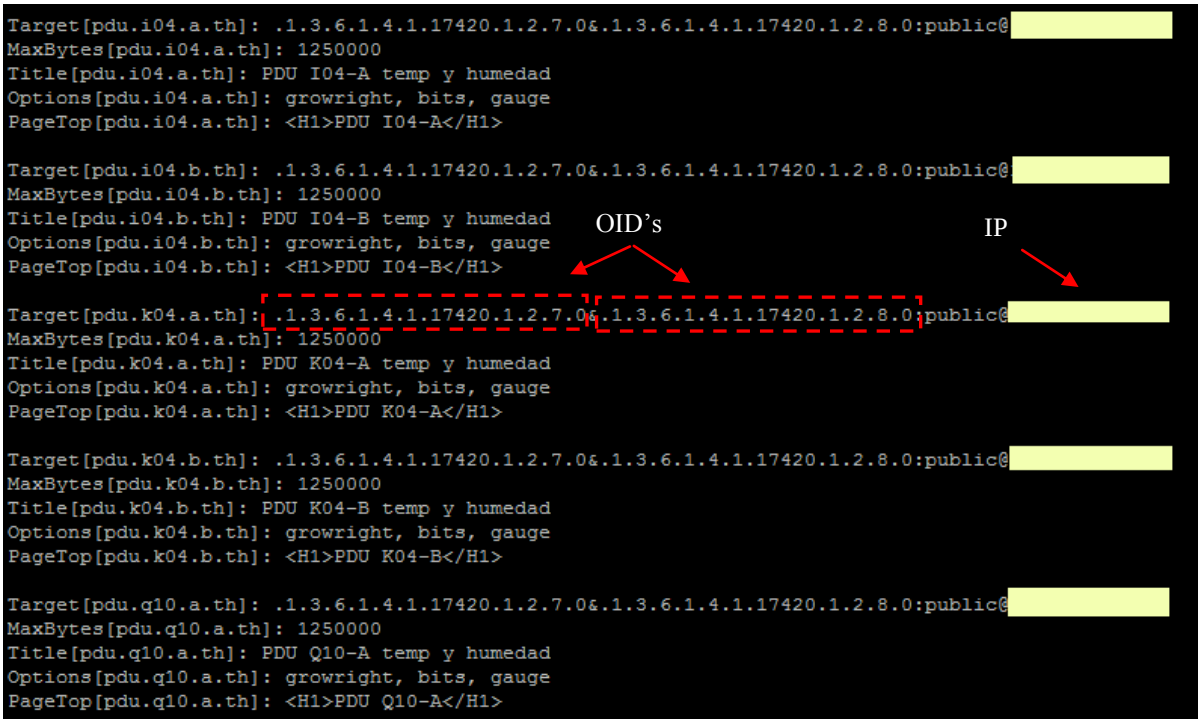
#Options[identificador_dispositivo]: tipo de datos

#MaxBytes[identificador_dispositivo]: tamaño máximo

#Title[identificador_dispositivo]: Título del dispositivo en las gráficas generadas por MRTG

#PageTop[identificador_dispositivo]: <H1>Nombre de la gráfica </H1>

Siguiendo esta estructura en los archivos de configuración obtenemos lo siguiente:

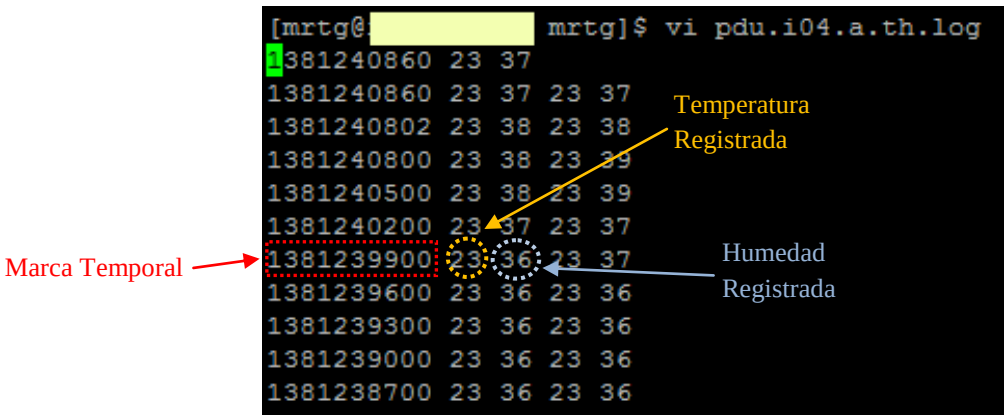


Una vez tenemos todos los dispositivos añadidos en nuestro archivo de configuración, está lista para poder ser ejecutada. En nuestro proyecto se han generado 4 archivos de configuración, 2 por cada sistema de Pdu's y 2 por cada sistema de Sai's a monitorizar. El número total de variables a monitorizar es de, aproximadamente, 450.

El siguiente paso para proceder con la monitorización sería el lanzamiento del proceso demonio que se encargue de ejecutar las variables necesarias. Una vez lanzado este demonio, podemos desplazarnos a la ubicación indicada en la configuración (/var/lib/mrtg) para poder obtener los ficheros .log donde se encuentran los valores monitorizados. Si listamos el directorio anteriormente citado obtendríamos lo siguiente:

```
[mrtg@inframundo mrtg]$ ls *.log
pdu.001.a.th.log  pdu.h10.a.th.log  pdu.p19.a.th.log                                  sai.3.tesla.battery.levelstatus.log
pdu.001.b.th.log  pdu.h10.b.th.log  pdu.p19.b.th.log                                  sai.3.tesla.cargafase1cargafase2.log
pdu.002.a.th.log  pdu.h19.a.th.log  pdu.q04.a.th.log                                  sai.3.tesla.cargafase3.log
pdu.002.b.th.log  pdu.h19.b.th.log  pdu.q04.b.th.log                                  sai.3.tesla.input.tensionfase1tensionfase2.log
pdu.006.a.th.log  pdu.h19.c.th.log  pdu.q07.a.th.log                                  sai.3.tesla.input.tensionfase3.log
pdu.006.b.th.log  pdu.i04.a.th.log  pdu.q07.b.th.log                                  sai.3.tesla.output.potenciafase1potenciafase2.log
pdu.007.a.th.log  pdu.i04.b.th.log  pdu.q10.a.th.log                                  sai.3.tesla.output.potenciafase3.log
pdu.007.b.th.log  pdu.i07.a.th.log  pdu.q10.b.th.log                                  sai.4.edison.battery.levelstatus.log
pdu.008.a.th.log  pdu.i07.b.th.log  pdu.r13.a.th.log                                  sai.4.edison.cargafase1cargafase2.log
pdu.008.b.th.log  pdu.i10.b.th.log  pdu.r13.b.th.log                                  sai.4.edison.cargafase3.log
pdu.009.a.th.log  pdu.i10.c.th.log  pdu.r19.a.th.log                                  sai.4.edison.input.tensionfase1tensionfase2.log
pdu.009.b.th.log  pdu.i10.d.th.log  pdu.r19.b.th.log                                  sai.4.edison.input.tensionfase3.log
pdu.010.a.th.log  pdu.i13.a.th.log  sai.1.franklin.battery.levelstatus.log           sai.4.edison.output.potenciafase1potenciafase2.log
pdu.010.b.th.log  pdu.i13.b.th.log  sai.1.franklin.cargafase1cargafase2.log          sai.4.edison.output.potenciafase3.log
pdu.011.a.th.log  pdu.i16.a.th.log  sai.1.franklin.cargafase3.log                    ups.a.input.tension1tension2.log
pdu.011.b.th.log  pdu.i16.b.th.log  sai.1.franklin.input.tensionfase1tensionfase2.log  ups.a.input.tension3.log
pdu.012.a.th.log  pdu.k04.a.th.log  sai.1.franklin.input.tensionfase3.log            ups.a.nivelbateriatemperatura.log
pdu.012.b.th.log  pdu.k04.b.th.log  sai.1.franklin.output.potenciafase1potenciafase2.log  ups.a.output.potencia1potencia2.log
pdu.013.a.th.log  pdu.k10.a.th.log  sai.1.franklin.output.potenciafase3.log          ups.a.output.potencia3.log
pdu.013.b.th.log  pdu.k10.b.th.log  sai.2.volta.battery.levelstatus.log              ups.b.input.tension1tension2.log
pdu.c19.a.th.log  pdu.m07.a.th.log  sai.2.volta.cargafase1cargafase2.log             ups.b.input.tension3.log
pdu.c19.b.th.log  pdu.m07.b.th.log  sai.2.volta.cargafase3.log                       ups.b.nivelbateriatemperatura.log
pdu.h04.a.th.log  pdu.m13.a.th.log  sai.2.volta.input.tensionfase1tensionfase2.log    ups.b.output.potencia1potencia2.log
pdu.h04.b.th.log  pdu.m13.b.th.log  sai.2.volta.input.tensionfase3.log               ups.b.output.potencia3.log
pdu.h04.c.th.log  pdu.o04.a.th.log  sai.2.volta.output.potenciafase1potenciafase2.log
pdu.h04.d.th.log  pdu.o04.b.th.log  sai.2.volta.output.potenciafase3.log
[mrtg@inframundo mrtg]$
```

Dicha lista de archivos es la generada gracias a los diferentes archivos de configuración anteriormente creados. En el interior de dichos archivos .log se encuentran los valores monitorizados que, posteriormente, serán tratados. Es papel del desarrollador entender el significado de cada columna así como las unidades de las variables monitorizadas. Si realizamos un vi a cualquier fichero .log podremos ver los valores registrados. Un ejemplo sería el siguiente:



Una vez comprobado que realmente obtenemos los valores demandados, debemos hacer un volcado de dichos valores a una base de datos con el fin de darles un orden y facilitar el manejo del procesado de los datos. Para ello se ha desarrollado un código PHP encargado de leer los diferentes archivos .log e ir copiando dichos valores leídos en la tabla correspondiente. Debido a la adquisición de nuevos valores cada 5 minutos, debemos incluir el código PHP como una tarea programada que se ejecute cada 5 minutos. Para ello, insertamos dicha tarea en el *cron* con el fin de darle carácter de tarea programada.

```
[mrtg@ mrtg]$ cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

Si queremos una ejecución cada 5 minutos todos los días, debemos escribir un 5 en el primer parámetro y asteriscos en el resto. El asterisco marca "todos".

```
[mrtg@            mrtg]$ cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
  5  *  *  *  *
```

Llegados a este punto, tenemos corriendo el software de monitorización encargado de la recolección de los datos y, además, tenemos los *daemons* necesarios para realizar la ejecución y volcado de datos a una base de datos de forma automática cada 5 minutos.

4.2.3 Base de datos

Una vez obtenemos los datos necesarios de los agentes SNMP, debemos poder almacenar dichos datos en alguna ubicación para poder acceder a ellos cuando sea necesario. Para tal fin debemos proveer a nuestra máquina de una base de datos que sea capaz de alojar dicha información.

La primera cuestión a tener en cuenta va a ser la cantidad de información que debemos manejar. En nuestro caso se plantearon dos posibilidades:

- Base de datos MySql
- Base de datos Oracle

En el estudio realizado, se optó por una MySql debido a:

- Número de registros a manejar. Ambos tipos de base de datos pueden gestionarlos sin problema alguno.
- Facilidad de gestión. MySql es un tipo de base de datos ampliamente utilizada y documentada.
- Factor económico. La licencia para trabajar con una base de datos Oracle era inasumible para el fin de dicha aplicación. El importe de la licencia de Oracle frente a MySQL es del orden de 10 veces superior.

Tras estos factores se procedió a la creación y diseño de la base de datos en MySql. Tras estudiar las relaciones entre las variables y las necesidades de diferenciación entre éstas, se ha optado por una base de datos con 6 tablas distribuidas entre Sedes, Dispositivos, Pdus y Sais.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
|                    |
|                    |
|                    |
|                    |
|                    |
|                    |
+--------------------+
6 rows in set (0.00 sec)

mysql> use                    ;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+------------------------------+
| Tables_in_                   |
+------------------------------+
| dispositivo                  |
| pdu                          |
| pdu_seleccionada             |
| sai                          |
| sai_seleccionado             |
| sede                         |
+------------------------------+
6 rows in set (0.00 sec)

mysql> describe sai_seleccionado;
+---------------------+----------------------+------+-----+---------------------+----------------+
| Field               | Type                 | Null | Key | Default             | Extra          |
+---------------------+----------------------+------+-----+---------------------+----------------+
| sai_seleccionado_id | int(1) unsigned      | NO   | PRI | NULL                | auto_increment |
| sai_id              | smallint(6) unsigned | NO   | MUL | NULL                |                |
| last_update         | timestamp            | NO   |     | 0000-00-00 00:00:00 |                |
| battery_level       | float unsigned       | YES  |     | NULL                |                |
| battery_status      | float unsigned       | YES  |     | NULL                |                |
| carga_fase1         | float unsigned       | YES  |     | NULL                |                |
| carga_fase2         | float unsigned       | YES  |     | NULL                |                |
| carga_fase3         | float unsigned       | YES  |     | NULL                |                |
| tension_entrada1    | float unsigned       | YES  |     | NULL                |                |
| tension_entrada2    | float unsigned       | YES  |     | NULL                |                |
| tension_entrada3    | float unsigned       | YES  |     | NULL                |                |
| potencia_salida1    | float unsigned       | YES  |     | NULL                |                |
| potencia_salida2    | float unsigned       | YES  |     | NULL                |                |
| potencia_salida3    | float unsigned       | YES  |     | NULL                |                |
| g_electrogeno       | enum('OK','CRITICAL') | YES  |     | NULL                |                |
| modo_bateria        | enum('OK','CRITICAL') | YES  |     | NULL                |                |
| modo_bypass         | enum('OK','CRITICAL') | YES  |     | NULL                |                |
| modo_normal         | enum('OK','CRITICAL') | YES  |     | NULL                |                |
+---------------------+----------------------+------+-----+---------------------+----------------+
18 rows in set (0.00 sec)
```

Dichas tablas están enlazadas para, seleccionando una sede, poder acceder a los valores de los dispositivos de dicha sede. Junto a esto, al seleccionar un dispositivo también debe realizar un filtrado para mostrar los valores correspondientes a dicho dispositivo. Estas relaciones serán fundamentales a la hora de desarrollar la interfaz gráfica ya que, con una instrucción, podremos seleccionar, de forma correcta, los valores necesarios a mostrar.

4.2.4 Lenguaje de programación

A la hora del desarrollo de la aplicación hay que tener en cuenta el volumen de información que debemos manejar y cuál es la finalidad de la misma. En nuestro caso vamos a manejar un volumen de datos grande debido a que monitorizamos cada 5 minutos durante todos los días del año y, en el peor caso, podemos querer realizar una consulta anual para procesar todos los datos registrados por un sensor. Además, debemos generar una interfaz gráfica que gestione esos datos.

Debido al alto volumen de datos a tratar y la posibilidad de desarrollo de GUI, se ha optado por utilizar el lenguaje de scripts de Matlab ya que, con una única herramienta, tenemos cubiertas las dos necesidades básicas, interfaz gráfica y procesamiento de gran cantidad de datos.

Otro factor determinante es la posibilidad de compilar el programa y su ejecución directa sobre el procesador, lo que aumenta el rendimiento. Este factor fue clave para descartar el lenguaje Java al tratarse de un lenguaje que se ejecuta sobre una máquina virtual.

4.3 Interfaz de Usuario o GUI

Una vez seleccionado el lenguaje de programación destinado al desarrollo de la interfaz gráfica, procedemos al desarrollo de la misma.

Matlab está dotado de una herramienta que permite, de forma clara y sencilla, la edición y desarrollo de un entorno gráfico de trabajo con múltiples botones, paneles, gráficos, menús, etc. Esto facilita la manipulación de los datos obtenidos mediante los programas desarrollados en su lenguaje propio. Al tratarse de un entorno de desarrollo propio, las posibilidades que ofrece son menores en comparación a otros entornos de desarrollo como puede ser Qt Creator para desarrollo de aplicaciones en C++ con la librería Qt.

Para el desarrollo del entorno gráfico, Matlab contiene una herramienta de diseño de GUIs llamada ***GUIDE.*** Para poder abordar el desarrollo de una GUI en Matlab debemos conocer ciertos conceptos específicos del entorno de trabajo para las GUIs.

En los siguientes apartados se explicará, brevemente, cómo se estructura una GUI en Matlab, los controles que podemos agregar a nuestra GUI, la jerarquía de los diferentes elementos dentro de una GUI y, por último, cómo dotar de funcionalidad a los diferentes elementos de control con el fin de obtener los resultados buscados.

4.3.1 Jerarquía Gráfica en Matlab

La jerarquía gráfica de Matlab se basa en una jerarquía de objetos en forma de árbol gracias a la diferencias de los tipos de objetos que podemos encontrar la frase no me cuadra. Dicha jerarquía parte de un nodo raíz llamada pantalla o *screen*. Sólo puede existir un objeto pantalla al tratarse del nodo raíz. Una pantalla puede contener una o múltiples ventanas o *figures*. Dentro de cada ventana se pueden encontrar múltiples ejes o *axes* encargados de visualizar los resultados como, por ejemplo, gráficas o imágenes. El objeto *axes* es muy potente debido a que puede albergar o representar objetos de más bajo nivel. Dentro de las ventanas también se pueden encontrar los controles o *uicontrols* tales como botones, botones de selección, barras de desplazamientos, menús desplegables, etc. Cabe destacar que Matlab sigue una jerarquía de "padres-hijos" en sus objetos lo que implica que, si eliminamos un axes, eliminaríamos todos los hijos contenidos en él (textos, superficies, imágenes…) .

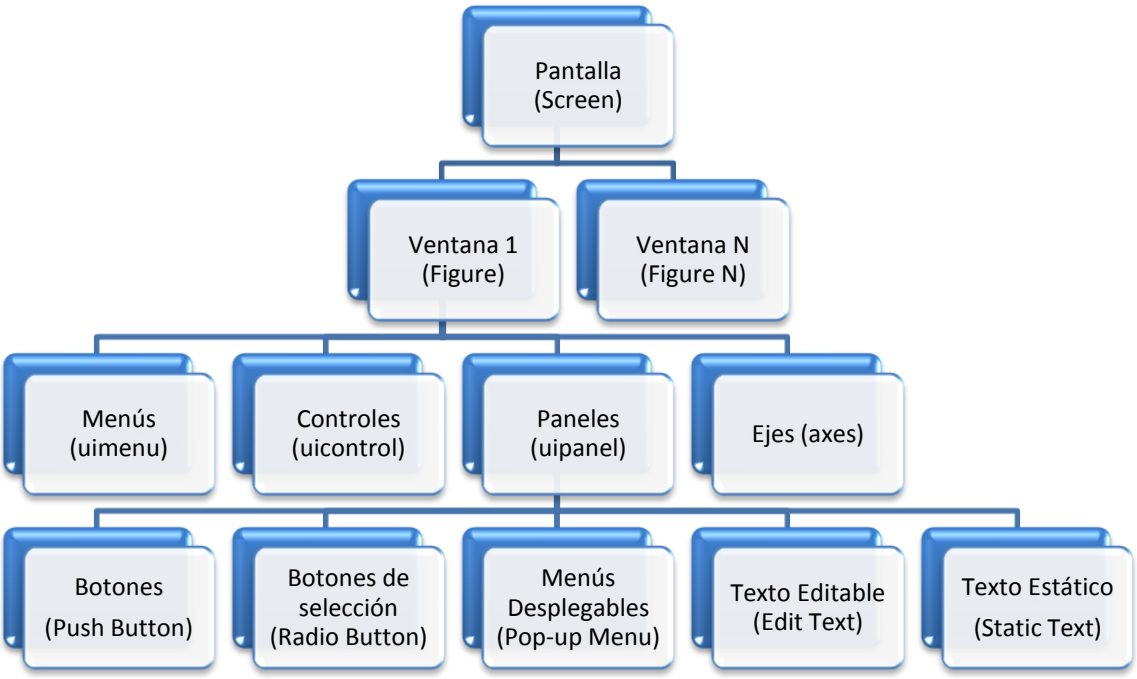


Figura 17. *Jerarquía gráfica en Matlab*

Cada objeto creado está asociado a un identificador único (handle). Gracias a este identificador podemos modificar las características de un objeto gráfico desde el inicio o en cualquier instante de ejecución. Como se ha indicado anteriormente, podemos encontrarnos en la situación de tener diversas ventanas abiertas a la vez pero si necesitamos modificar algún elemento de la ventana activa, cómo la identificamos?. Para ello Matlab nos proporciona tres herramientas capaces de devolvernos el identificador del objeto necesario:

- gcf: devuelve un entero correspondiente al identificador de la ventana activa.
- gca: devuelve un entero correspondiente al identificador de los ejes activos.
- gco: devuelve un entero correspondiente al identificador del objeto activo.

Si conocemos los identificadores de los objetos podemos modificarlos de forma sencilla. Para ello podemos utilizar alguna de las siguientes instrucciones:

- **set (id):** muestra en pantalla todas las propiedades del objeto con identificador *id*.
- **get (id):** produce un listado de las propiedades y de sus valores.
- **set (id, 'propiedad', 'valor'):** establece un nuevo valor para la propiedad del objeto con identificador id. Se pueden establecer varias propiedades en la misma llamada a set incluyendo una lista de parejas 'propiedad', 'valor' en la llamada.
- **get (id, 'propiedad'):** obtiene el valor de la propiedad especificada.
- **delete (id):** borra el objeto cuyo identificador es id y todos sus hijo

4.3.2 Estructura de una GUI en Matlab

Para la realización de la GUI en Matlab debemos ejecutar en la ventana de comandos la instrucción *guide*. Este comando provocará la aparición del cuadro de diálogo de selección del tipo de GUI que queremos desarrollar.

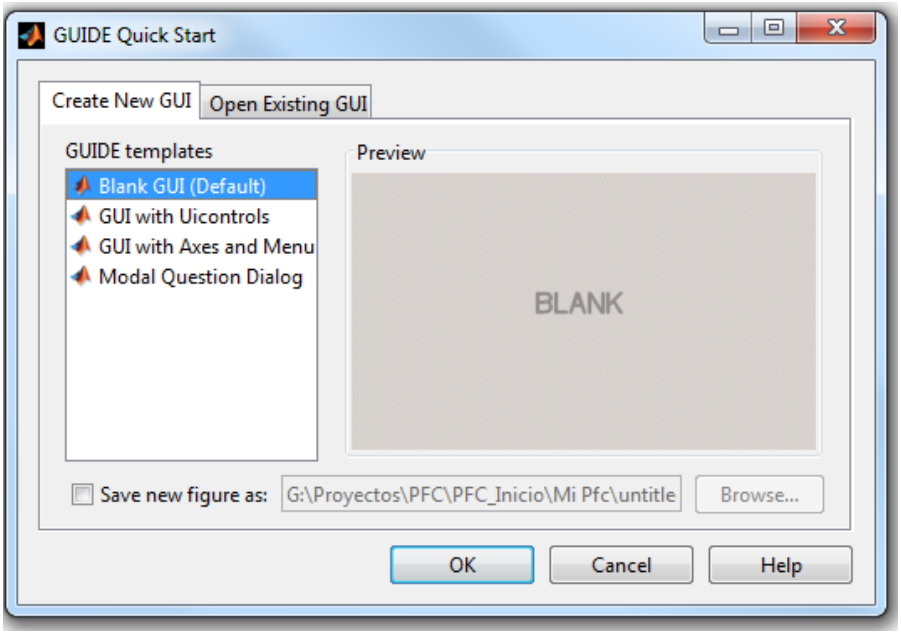


Figura 18. *Jerarquía gráfica en Matlab*

Se puede observar cuatro opciones diferentes:

- **Blank GUI (default)**

  La opción por defecto generará un formulario nuevo vacío donde podemos diseñar nuestra interfaz gráfica.

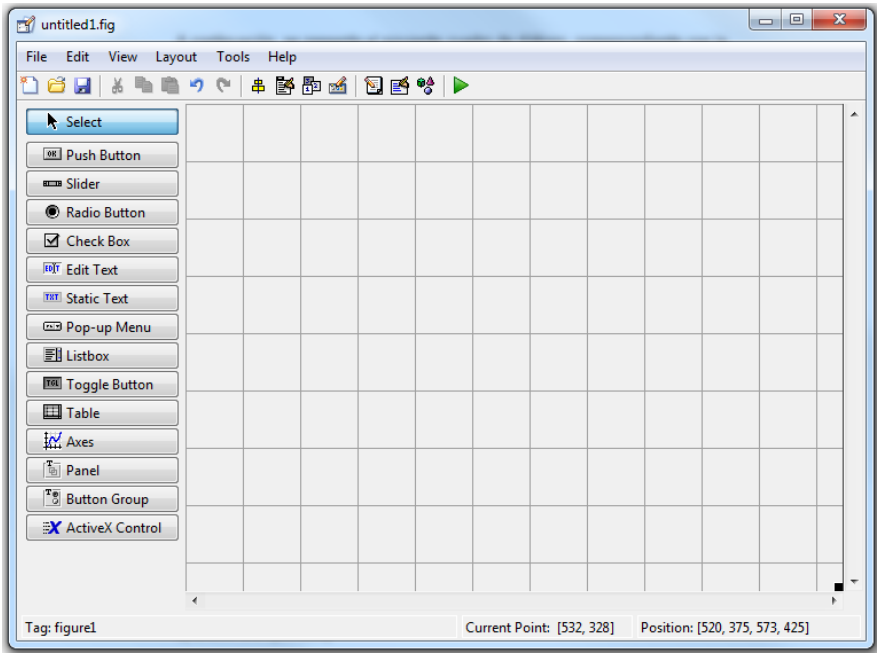


Figura 19. *Formulario vacío de desarrollo de GUI*

- **GUI with controls**

  Esta opción muestra un ejemplo de una interfaz gráfica con controles. Se puede ejecutar para probar su funcionamiento.

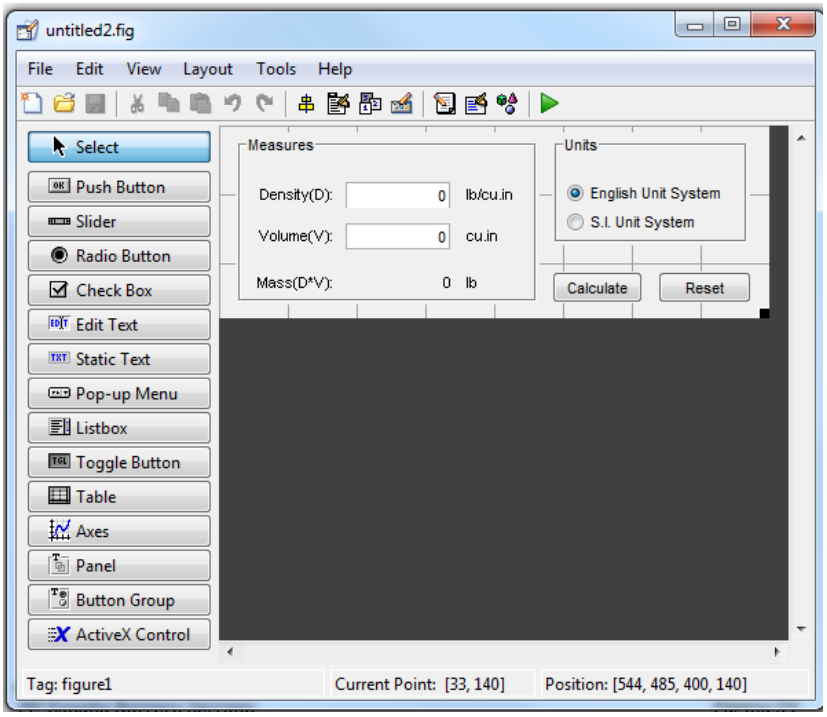


Figura 20. *Ejemplo GUI con controles*

- **GUI with Axes and Menu**

  Esta opción muestra un ejemplo de una interfaz gráfica que contiene un menú desplegable y un axes donde obtener las diferentes gráficas.

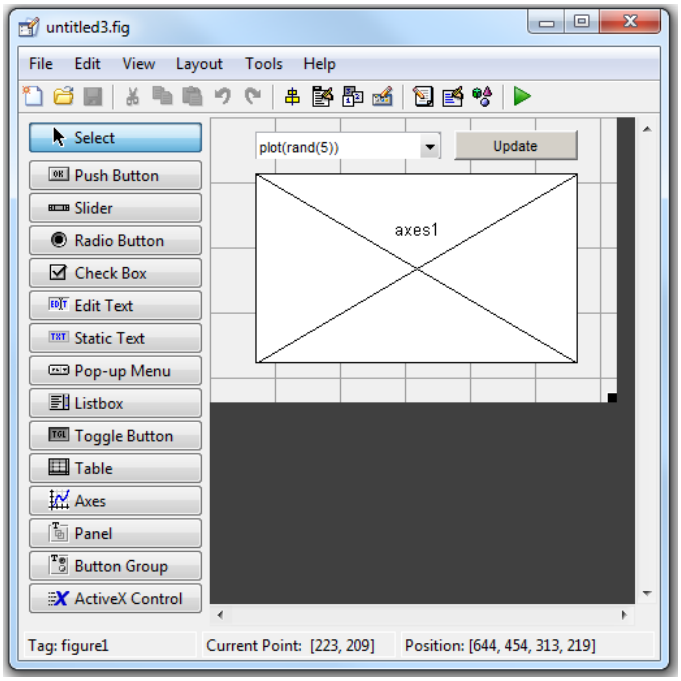


Figura 21. *Ejemplo GUI con menú desplegable y axes*

- **Modal Question Dialog**

  Muestra un cuadro de diálogo con dos botones, una axes y un texto estático. Dependiendo del botón pulsado se ejecutará una acción u otra.

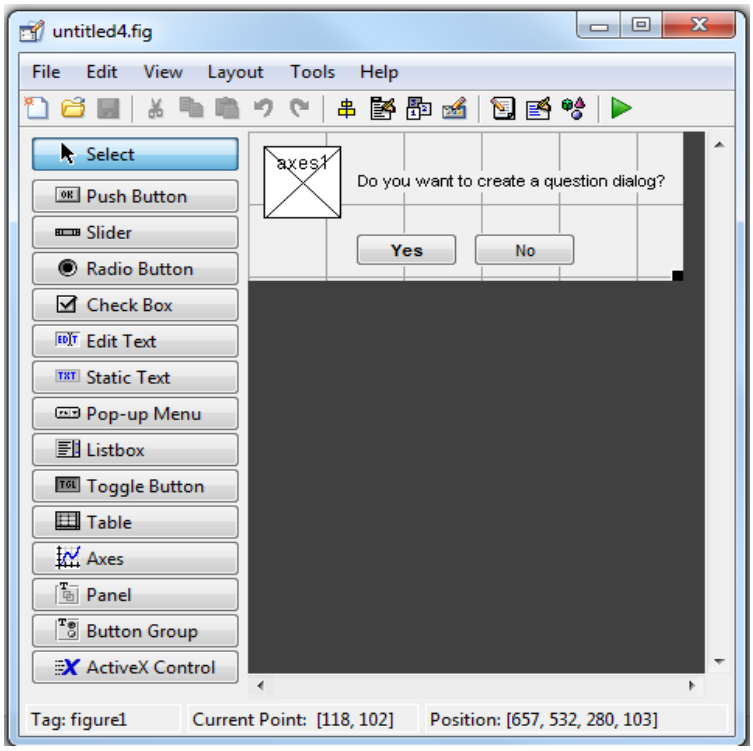


Figura 22. *Ejemplo GUI con texto estático, botones y axes*

En nuestro caso, se ha optado por la opción ***Blank GUI (default)*** ya que se va a desarrollar una GUI concreta adaptada a nuestras necesidades.
En primer lugar, seleccionamos la opción ***Blank GUI (default)*** y accedemos al formulario de desarrollo.

Una vez visualizado el formulario, cabe resaltar las partes más importantes del mismo:

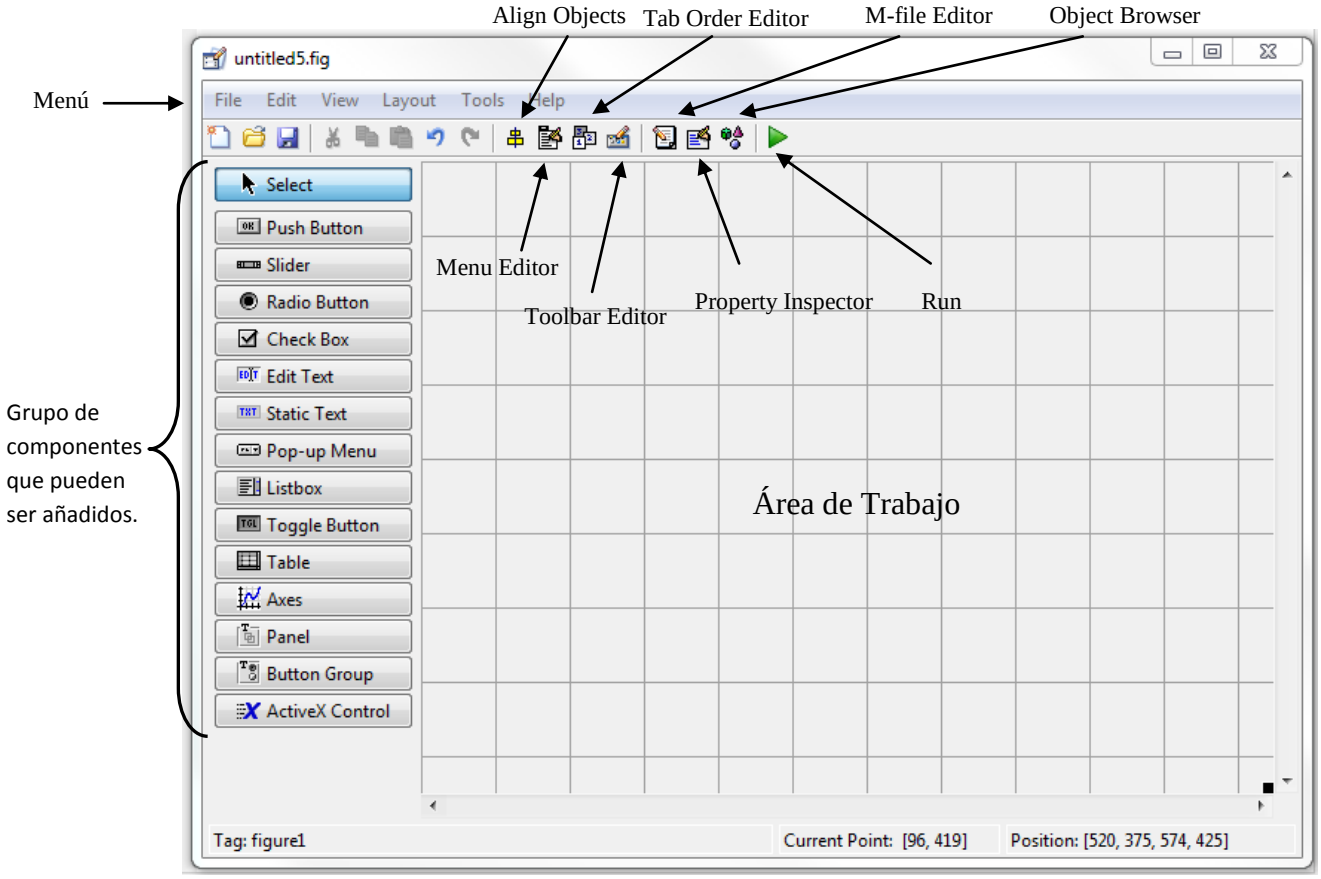


Figura 23. *Menús de edición en GUI Matlab*

- **Barra de Menús:** funciones básicas de Edición de GUIs.
- **Grupo de componentes:** muestra los uicontrols disponibles que podemos insertar en la GUI.
- **Barra de Herramientas:**
  - **Align Objects:** herramienta de alineación de objetos en un área de trabajo.
  - **Menu Editor:** herramienta para la edición del menú de nuestra GUI.
  - **Tab Order Editor:** herramienta que permite indicar el orden de cambio de cursor al pulsar la tabulación. Muy útil en grandes formularios.
  - **Toolbar Editor:** herramienta que permite configurar las herramientas a mostrar en la barra.
  - **M-file Editor:** herramienta para acceder al editor de funciones en lenguaje .m .
  - **Property Inspector:** herramienta que permite ver las propiedades del objeto seleccionado.
  - **Object Browser:** herramienta que permite localizar un objeto dentro de una GUI.
  - **Run:** herramienta para ejecutar la GUI.

4.3.3 Flujo de operación en una GUI

Cuando ejecutamos una GUI, el flujo de operación está controlado por las acciones que el usuario realiza sobre la interfaz gráfica. Este funcionamiento es debido a que, una vez interactuado sobre un objeto de la GUI, el objeto en cuestión ejecuta un script previamente desarrollado que le marca las pautas a seguir. Una vez ejecutado el script, se devuelve el control a la GUI para que el usuario pueda volver a interactuar con ella. Cabe destacar que, aunque el control pase a los scripts tras seleccionar un objeto, la ventana o *figure* no desaparece permaneciendo visible en todo momento[6].

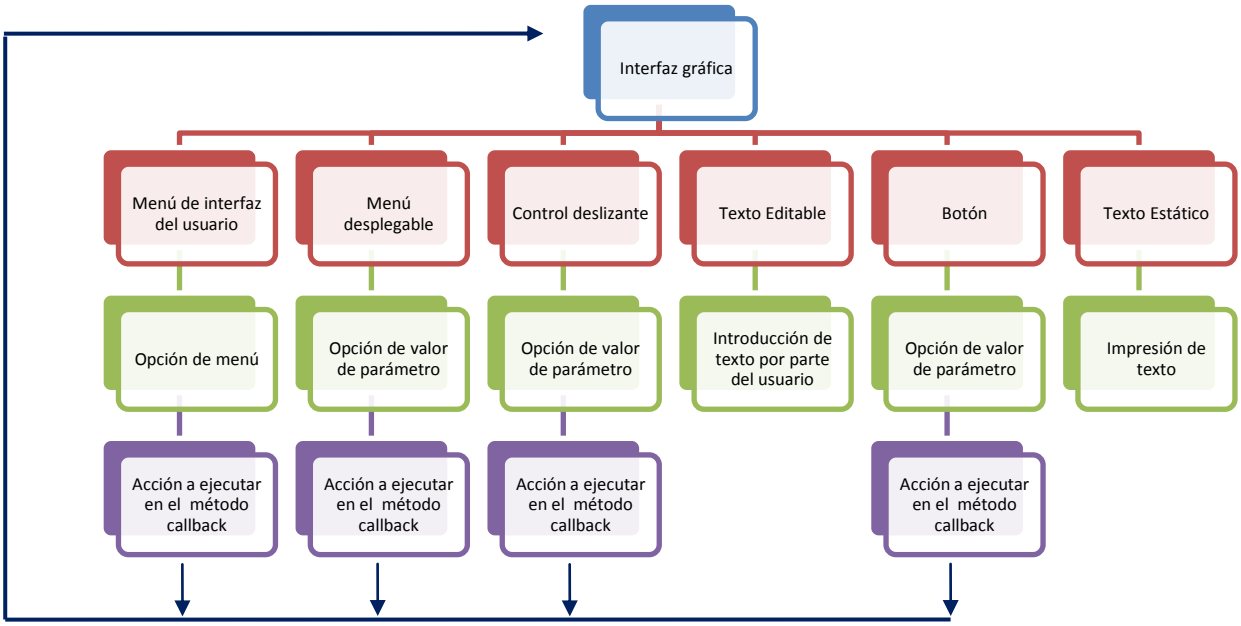


Figura 24. *Flujo de operación en una GUI*

Una vez creado un objeto, automáticamente se le asigna un identificador único con el cual poder localizarlo para poder realizar las modificaciones pertinentes. Qué ocurre si dispongo de varias ventanas y debo realizar modificaciones en la ventana activa?. Para ello Matlab otorga tres funciones que nos permiten localizar los objetos deseados para su posterior manipulación. Dichas funciones fueron explicadas anteriormente en la página 45.

4.3.4 Grupo de Componentes de una GUI

Una vez definido el tipo de interfaz que vamos a desarrollar, procedemos a la inserción de los diversos objetos que serán los encargados de obtener los resultados a través de la ejecución de los scripts programados.

[6] Esto puede ser modificado desde el script variando la propiedad *visible* de *On* a *Off*.

El grupo de componentes disponibles para la creación de la GUI son los siguientes:

- **Text:** El texto estático puede exhibir símbolos, mensajes o incluso valores numéricos en una GUI y puede colocarse en el lugar apropiado. Este control no tiene cadena de invocación.
- **Pop-up menu:** Menús desplegables que pueden adoptar cualquier posición dentro del área de trabajo.
- **Push Button:** Los botones son pequeños objetos de la pantalla generalmente acompañados con texto. Al presionar el botón con el ratón, se producirá una acción que será ejecutada por Matlab.
- **Checkbox:** Casillas diseñadas para realizar dos funciones únicamente: encendido o apagado.
- **Radio Button:** Cuando se usa un solo botón de radio, es igual que la casilla de verificación. Sin embargo, cuando se usan en grupo, estos son mutuamente exclusivos, es decir, si un botón de radio está encendido, los demás estarán apagados, mientras que las casillas de verificación son independientes entre sí.
- **Slider:** Objeto que permite modificar un parámetro de forma continua.
- **Edit Text:** El dispositivo de texto editable le permite al usuario introducir una cadena. Puede aceptar valores numéricos en forma de vector o matriz como una cadena mediante el mismo dispositivo. Objeto perfecto para introducir información concreta para el desarrollo del programa.
- **Axes:** Objeto encargado de definir la zona donde obtener información gráfica en forma de imágenes, superficies o gráficas. Abre un eje en un punto específico dentro de un panel.

4.3.5 Property Inspector

El *Property Inspector* es una herramienta incluida dentro de *GUIDE* utilizada para analizar y cambiar las propiedades de los diferentes objetos de los que se compone la GUI. La función del *Property Inspector* es controlar las características de todos los elementos que componen la ventana de la interfaz gráfica.

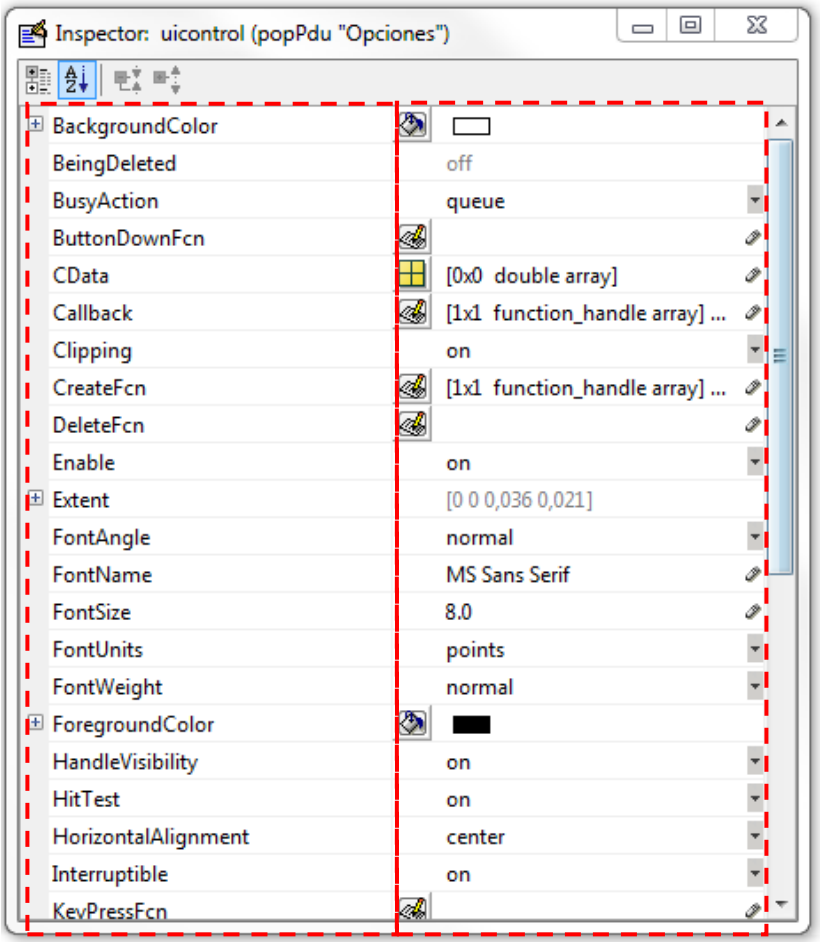


Figura 25. *Aspecto del Property Inspector*

4.3.6 Desarrollo de una interfaz gráfica de ejemplo paso a paso

Debido a la dificultad que puede entrañar el desarrollo de una GUI desde cero, se ha optado por realizar, a modo de ejemplo, una sencilla interfaz gráfica donde se explicará el funcionamiento de cada elemento y se mostrará la interactuación de dichos elementos entre sí.

Como ejemplo de interfaz gráfica, se ha decidido realizar una GUI dotada de *cinco pushbutton*, *seis statictext, un edittext y seis axes*. Dicha GUI tendrá como finalizada la aplicación de dos efectos visuales como *escala de grises y detección de bordes* sobre una imagen original. Para poder observar los efectos y poder compararlos con la imagen inicial, ésta será cargada en uno de los *axes* mientras los efectos serán cargados, cada uno¸ en *axes* independientes.

Una vez explicada de forma genérica la utilidad de la GUI, procederemos al desarrollo de la misma. Para ello, tal y como se indicó en el punto 4.3.2, ejecutaremos en el workspace la instrucción *guide* y seleccionaremos la opción *blank GUI (default)* obteniendo la figura 19.

A continuación debemos añadir los elementos anteriormente mencionados para crear la estructura básica de la GUI.

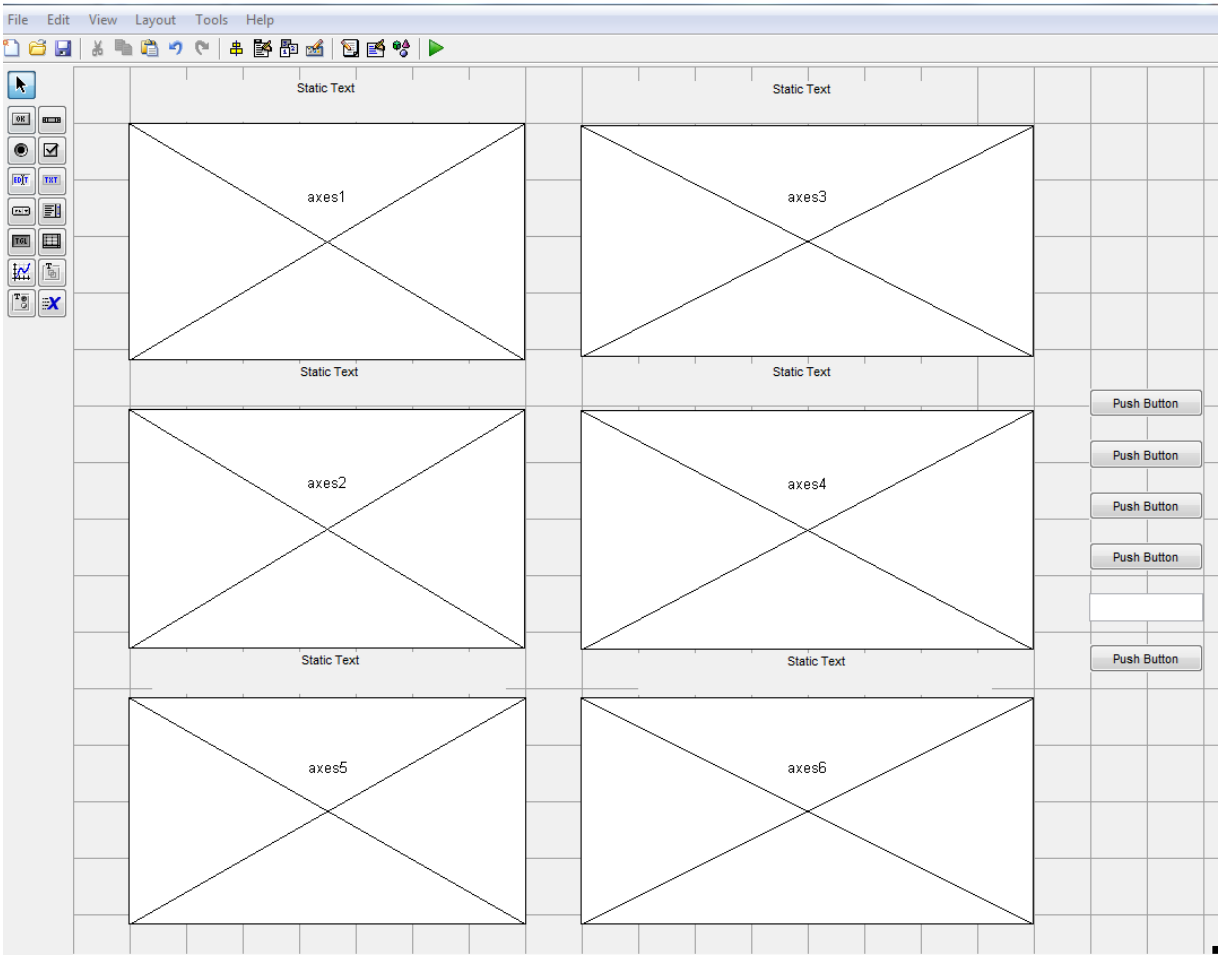


Figura 26. *Estructura de la GUI de ejemplo*

Una vez añadidos los elementos necesarios, debemos personalizarlos de forma que sea fácilmente entendible cual es la función de cada elemento. Para ello, debemos pulsar sobre un elemento y, una vez seleccionado, pulsar con el botón derecho del ratón y, del menú desplegable que aparece, seleccionamos la opción *Property Inspector*. Dicha opción permite la configuración de cada elemento por separado. Cabe destacar, nuevamente, que las opciones de personalización varían en función del objeto seleccionado.

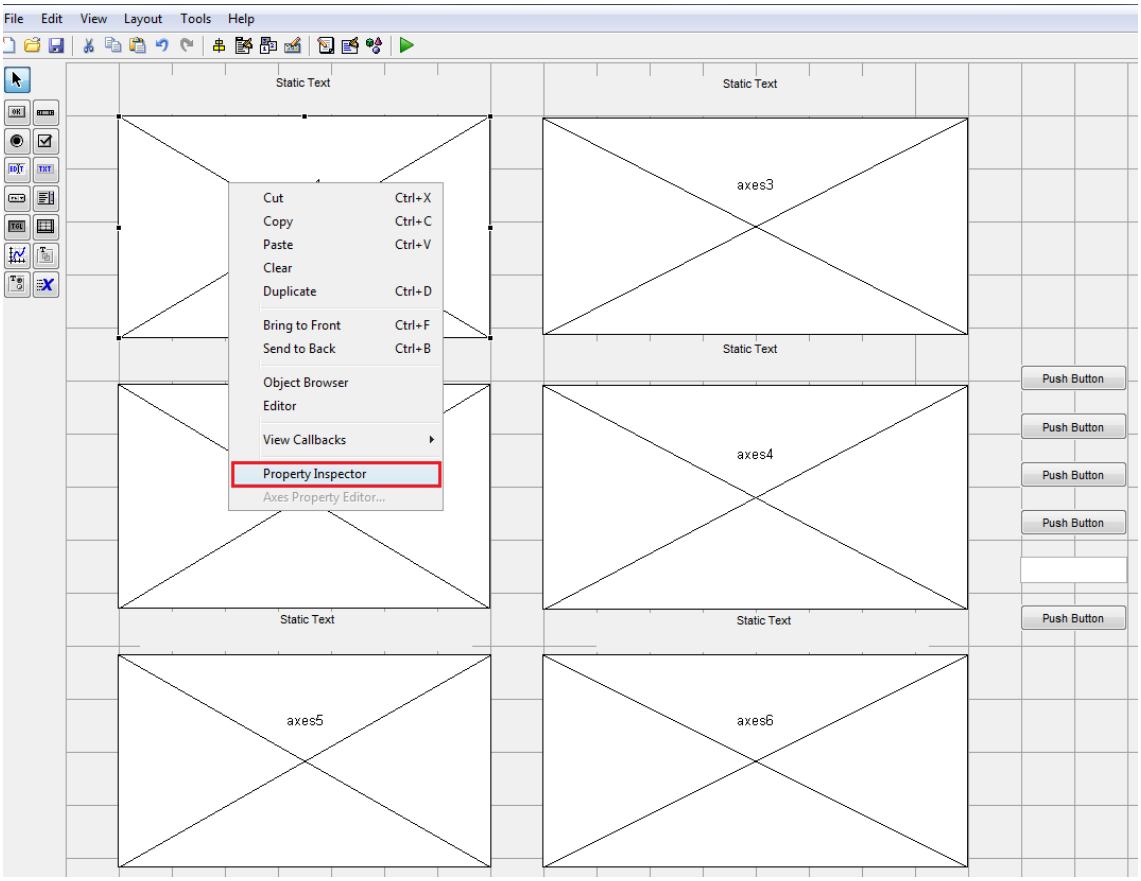


Figura 27. *Acceso al Property Inspector*

En nuestro caso, se ha procedido a la personalización de uno de los *axes* ubicados anteriormente. Procederemos a modificar la opción *Tag* encargada de identificar el elemento dentro de la GUI y la opción *Visible* la cual indica si nuestro objeto es visible o no según se indique en el menú desplegable.

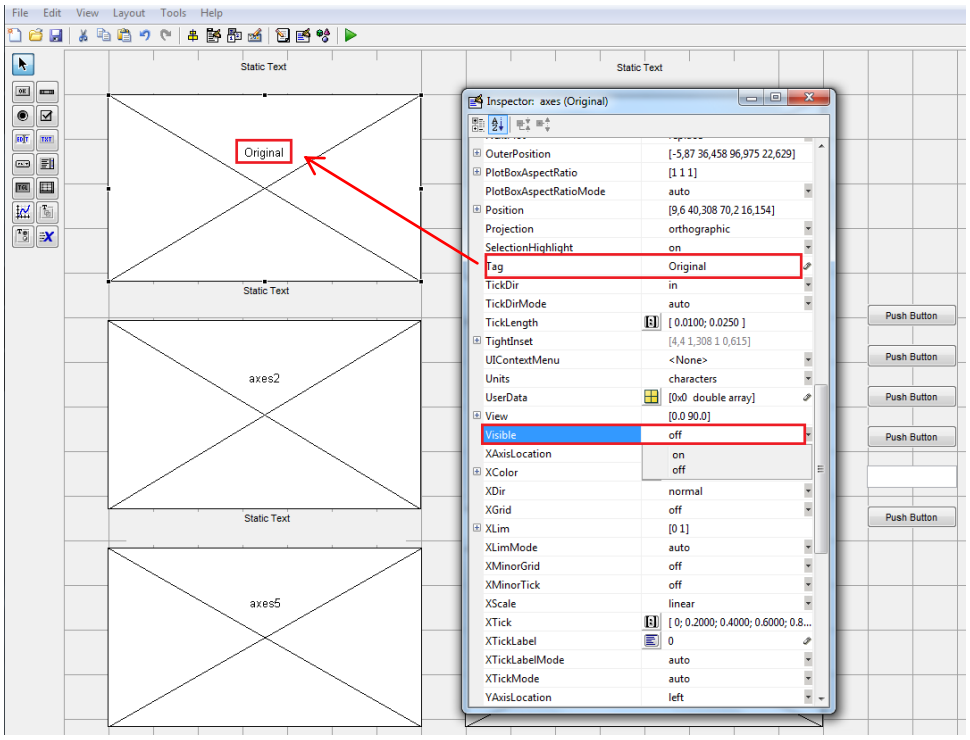


Figura 28. *Configuración de los axes según el Property Inspector.*

Para la personalización de los *pushbutton* optamos por modificar dos parámetros como son el *Tag* encargado de identificar el botón dentro de todos los elementos de la GUI y el parámetro *string* encargado de asignarle un texto al botón con el que el usuario final puede entender la funcionalidad de dicho botón.

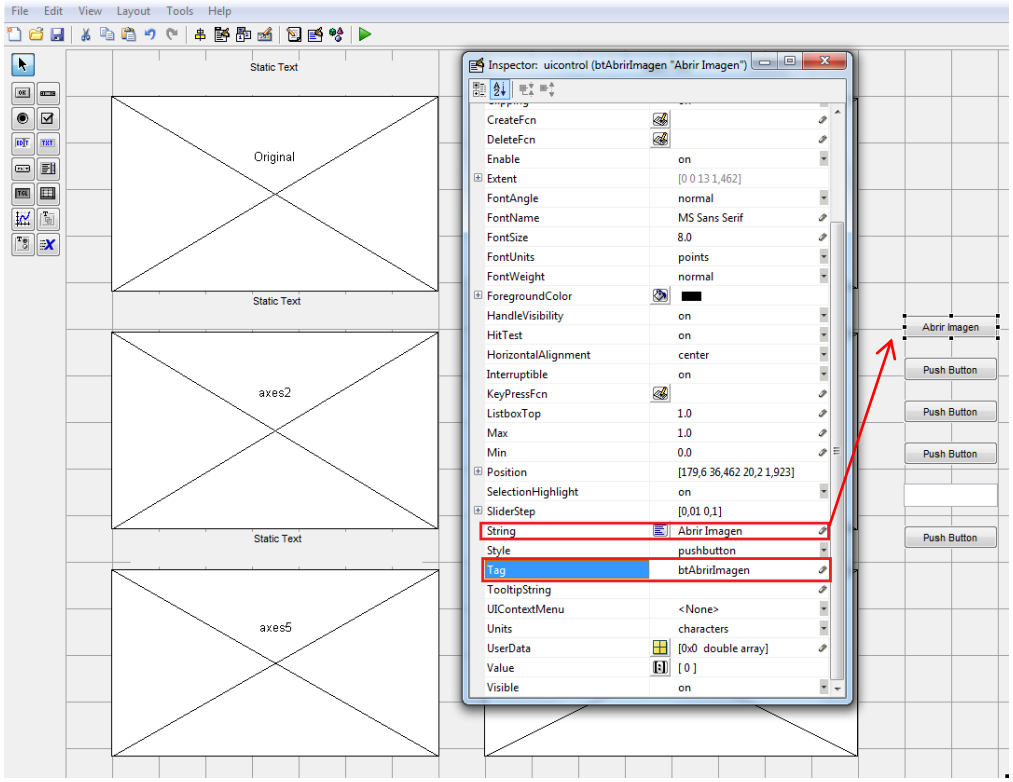


Figura 29. *Configuración de los pushbutton según el Property Inspector.*

Una vez editados todos los elementos presentes[7] en nuestra interfaz gráfica obtendríamos el siguiente diseño:

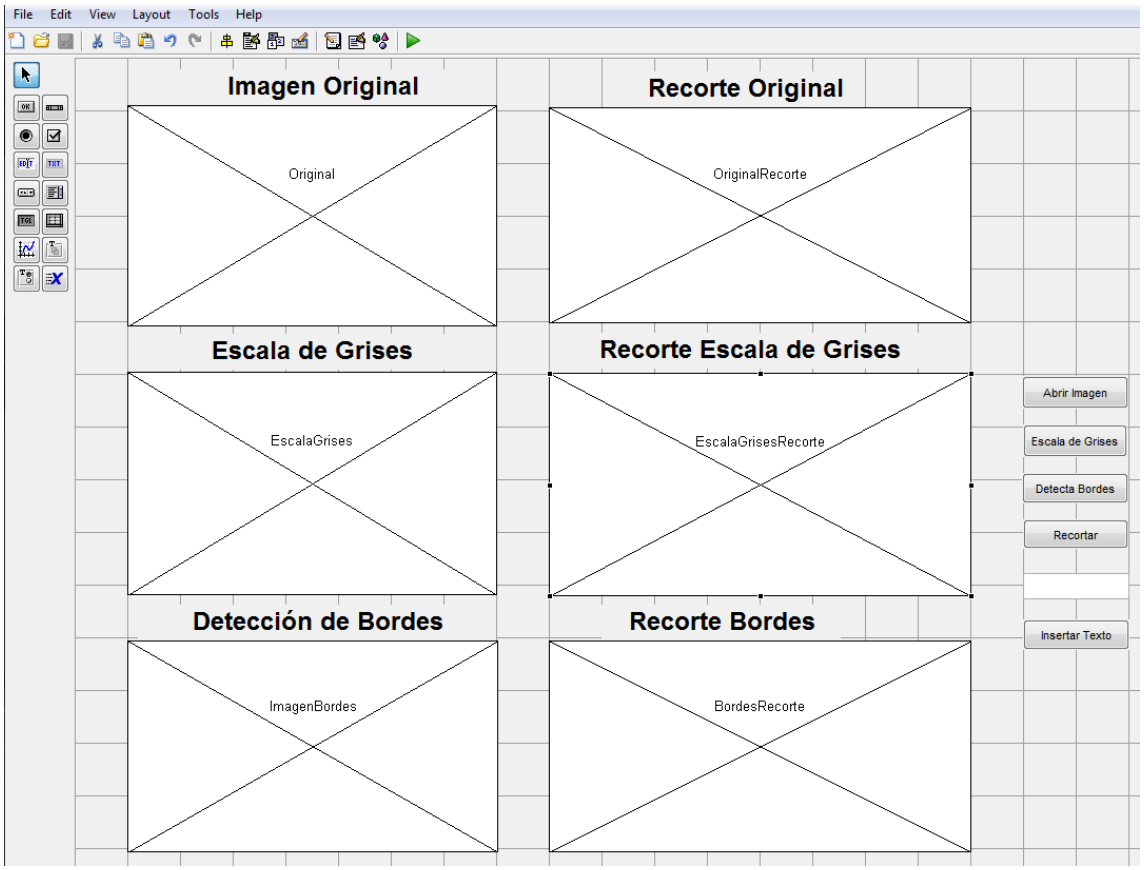


Figura 30. *Aspecto final de la interfaz gráfica diseñada.*

Como podemos observar, la interfaz gráfica está dotada de cinco botones cuyas funcionalidades son las siguientes:

- ***Abrir imagen:*** botón encargado de abrir una ventana de selección que nos permite navegar por nuestro sistema de archivos en busca de la imagen deseada.
- ***Escala de Grises:*** botón encargado de aplicar un efecto a la imagen original consistente en transformar la imagen RGB original en una imagen basada en escala de grises.
- ***Detecta Bordes:*** botón encargado de aplicar un efecto a la imagen original consistente en la detección de los bordes y visualizar en blanco y negro.
- ***Recortar:*** botón encargado de seleccionar un área de la imagen original que se quiera tratar en detalle.
- ***Insertar Texto:*** botón encargado de insertar mediante un click de ratón el texto introducido en el *edittext*.

[7] Una vez explicado de forma básica el manejo del *Property inspector* es trabajo del lector descubrir el nivel de personalización presente en cada elemento.

Una vez generada la parte visual debemos dotar de funcionalidad a cada elemento de la interfaz gráfica. Para dicha tarea, matlab posee varias funciones encargadas de ejecutar las subrutinas programadas en instante de tiempo diferentes. Dichas funciones son las siguientes:

- ***Nombre_GUI_OpeningFcn:*** función encargada de ejecutar el código alojado en ella una vez es llamada la GUI. Si queremos, por ejemplo, inicializar variables una vez se abre la GUI debemos insertar el código en esta zona.
- ***Nombre_GUI_OutputFcn:*** función encargada de visualizar en el workspace, por ejemplo, las condiciones iniciales de nuestra interfaz. Útil para conocer los valores iniciales de las variables.
- ***Nombre_Objeto_Callback:*** función encargada de dotar a los objetos de funcionalidad. Cuando un botón es pulsado se ejecutan las instrucciones ubicadas en su método *Callback*. Es la función más importante en el desarrollo de una interfaz gráfica ya que es la encargada de realizar la interactuación entre el usuario y el programa.

Una vez presentados los tipos de funciones disponibles procederemos a dotar de funcionalidad a los botones de nuestra interfaz. Para ello adjuntaremos el código comentado para el correcto entendimiento del mismo.Botón

- Botón ***Abrir Imagen***

Dicho botón permite visualizar la imagen seleccionada en el exes identificado como “Original”

```
% --- Executes on button press in btAbrirImagen.
function btAbrirImagen_Callback(hObject, eventdata, handles)
% hObject    handle to btAbrirImagen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Declaramos variable global para que sea accesible por todos los
métodos
% y por todos las ventanas que desarrollemos
global imagen;

[FileName Path]=uigetfile({'*.jpg;*.bmp'},'Abrir Imagen');
% comprueba si el FileName está vacío
if isequal(FileName,0)
    return
else
    % leo la imagen
    imagen=imread(strcat(Path,FileName));

    % la instrucción set permite modificar las opciones de los objetos
    %   en   este   caso   hago   visible   el   objeto   "Original"
y"textImagenOriginal"
    set(handles.Original,'Visible','on');
    set(handles.textImagenOriginal,'Visible','On');

    % la instrucción axes permite asignar cuál es el axes activo
    % donde visualizaré la información necesaria
    axes(handles.Original);

    % visualizo la variable "imagen" en el axes "Original"
    imshow(imagen);
end

% instrucción para almacenar y actualizar todas las variables
% para que otro método recoja las modificaciones.
guidata(hObject,handles)
```

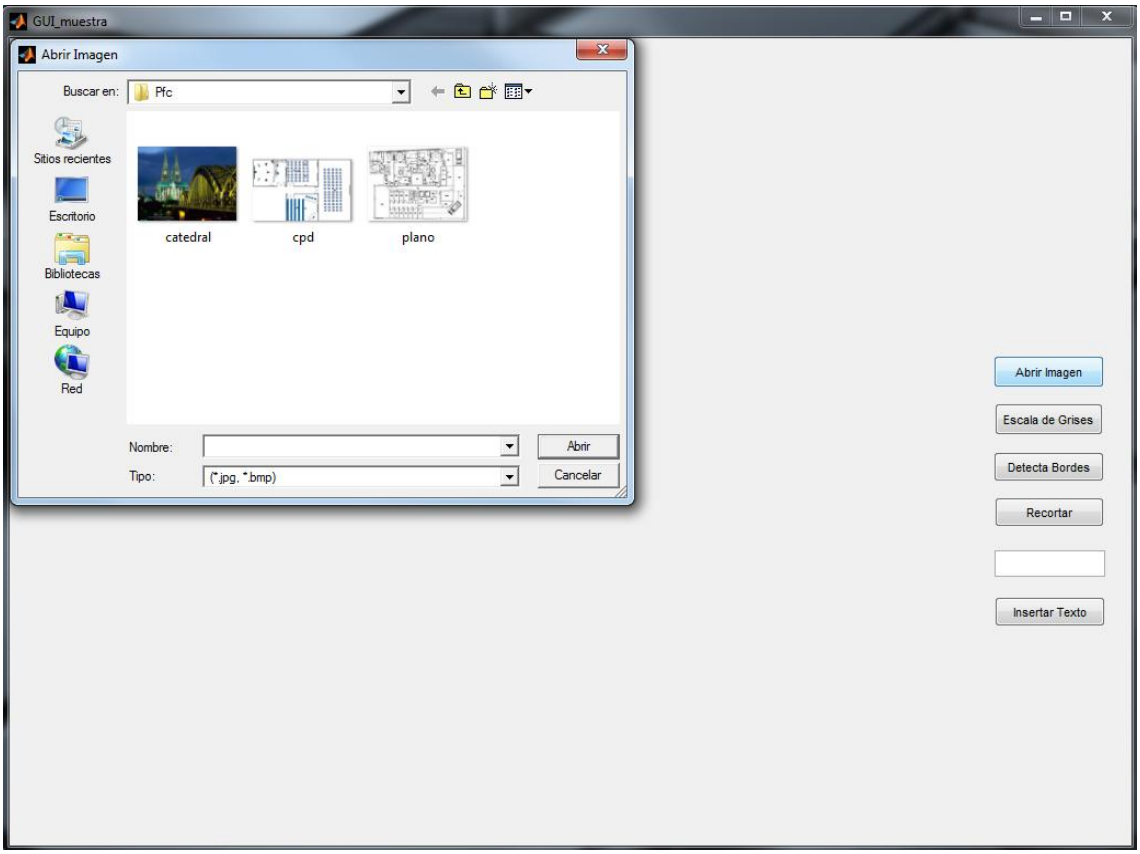


Figura 31. *Selección de la imagen a mostrar.*

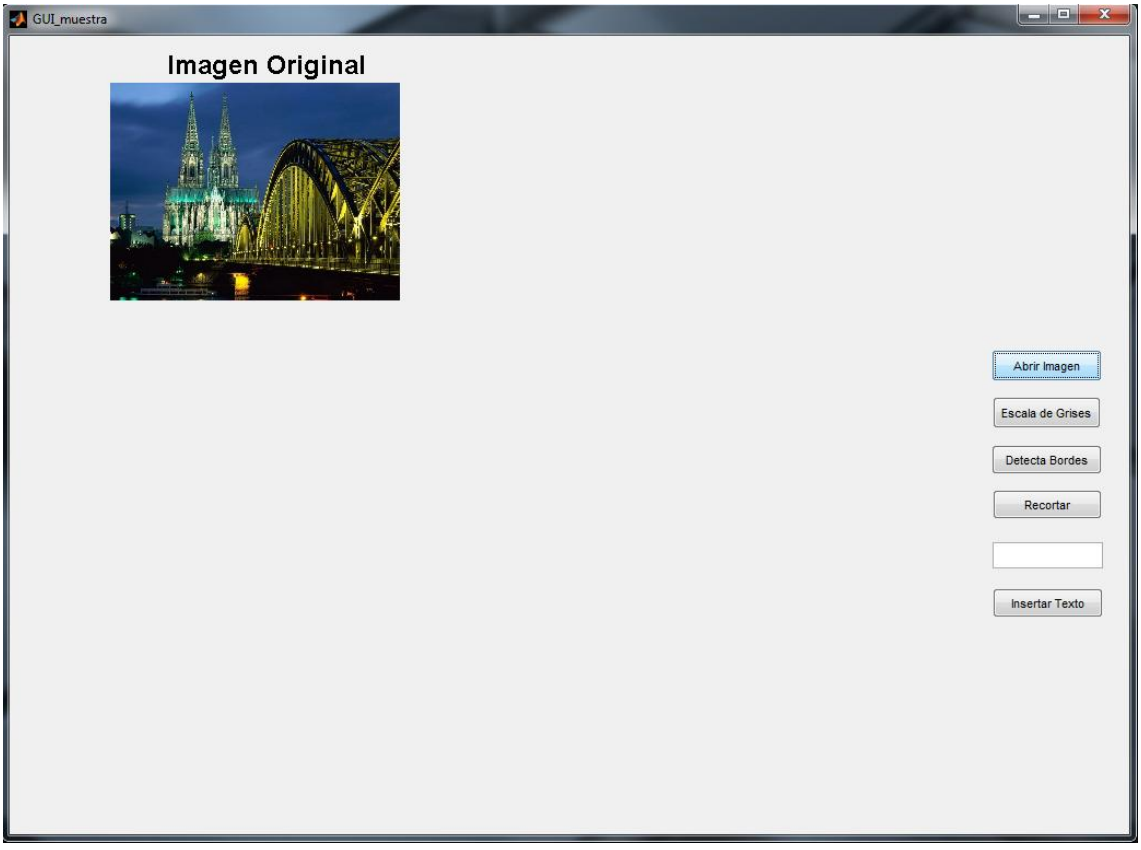


Figura 32. *Visualización de la imagen seleccionada.*

- Botón ***Escala de Grises***

Dicho botón permite convertir la imagen seleccionada anteriormente en una imagen en escala de grises. Para ello modificamos la variable global "imagen" y la convertimos en escala de grises.

```
% --- Executes on button press in btEscalaGrises.
function btEscalaGrises_Callback(hObject, eventdata, handles)
% hObject    handle to btEscalaGrises (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% necesario declarar todas las variables globales a utilizar.
%  si no declaramos global imagen no podremos acceder a la imagen
original
global imagen;
global imagen_bn;

% conversión de la imagen original a escala de grises
imagen_bn=rgb2gray(imagen);

% permitimos la visualización del axes "EscalaGrises"
% y el statictext ·textEscaladeGrises
 set(handles.EscalaGrises,'Visible','on');
set(handles.textEscaladeGrises,'Visible','On');
 % activamos el axes "EscalaGrises" como receptor de lo siguiente a
% visualizar
axes(handles.EscalaGrises);
% mostramos la imagen procesada en el axes "EscalaGrises"
imshow(imagen_bn);
 guidata(hObject,handles)
```

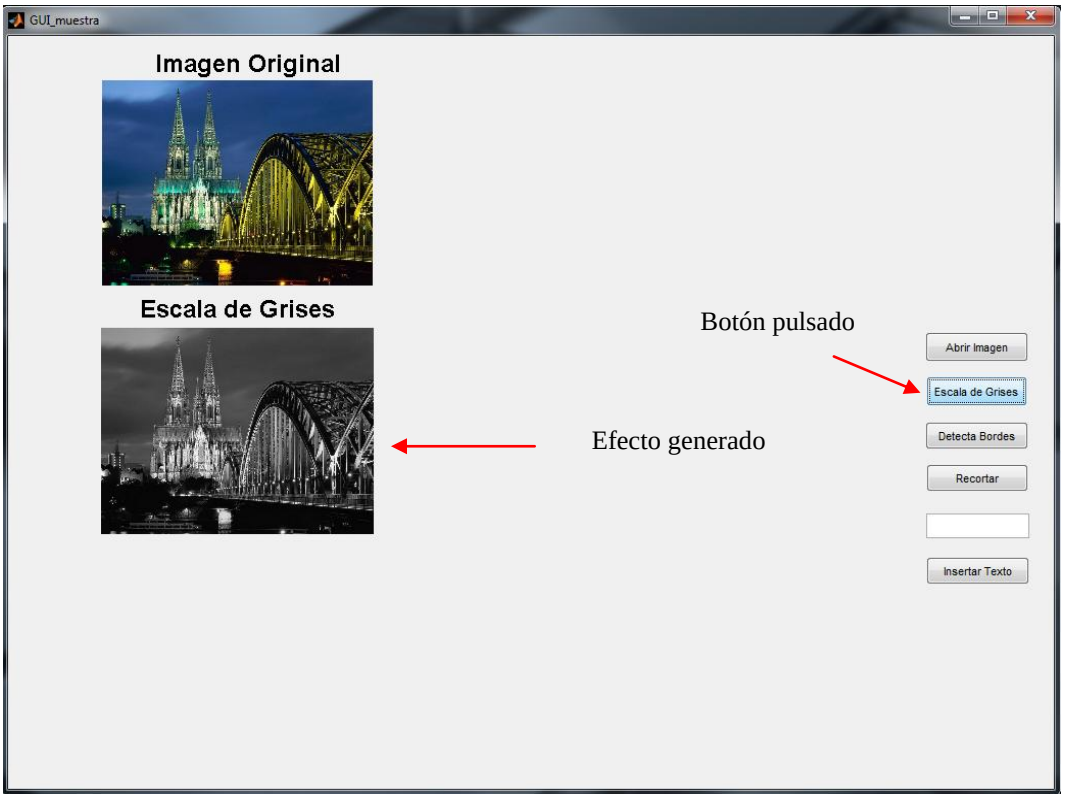


Figura 33. *Visualización de la imagen seleccionada en escala de grises.*

- Botón ***Detecta Bordes***

Botón encargado de aplicar el efecto de detección de bordes de una imagen.

```
% --- Executes on button press in btBordes.
function btBordes_Callback(hObject, eventdata, handles)
% hObject    handle to btBordes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% declaramos global imagen_bn porque el efecto lo debemos aplicar
% sobre una imagen en escala de grises
global imagen_bn;

% calculamos los bordes de la imagen mediante la función edge.
rgb_sobel=edge(imagen_bn,'log');

% activamos la visualización del axes "ImagenBordes" y el statictext
set(handles.ImagenBordes,'Visible','on');
set(handles.textBordes,'Visible','on');


% activamos el axes contenedor del efecto de detección de bordes
axes(handles.ImagenBordes);
% visualizamos el efecto.
imshow(rgb_sobel);
```

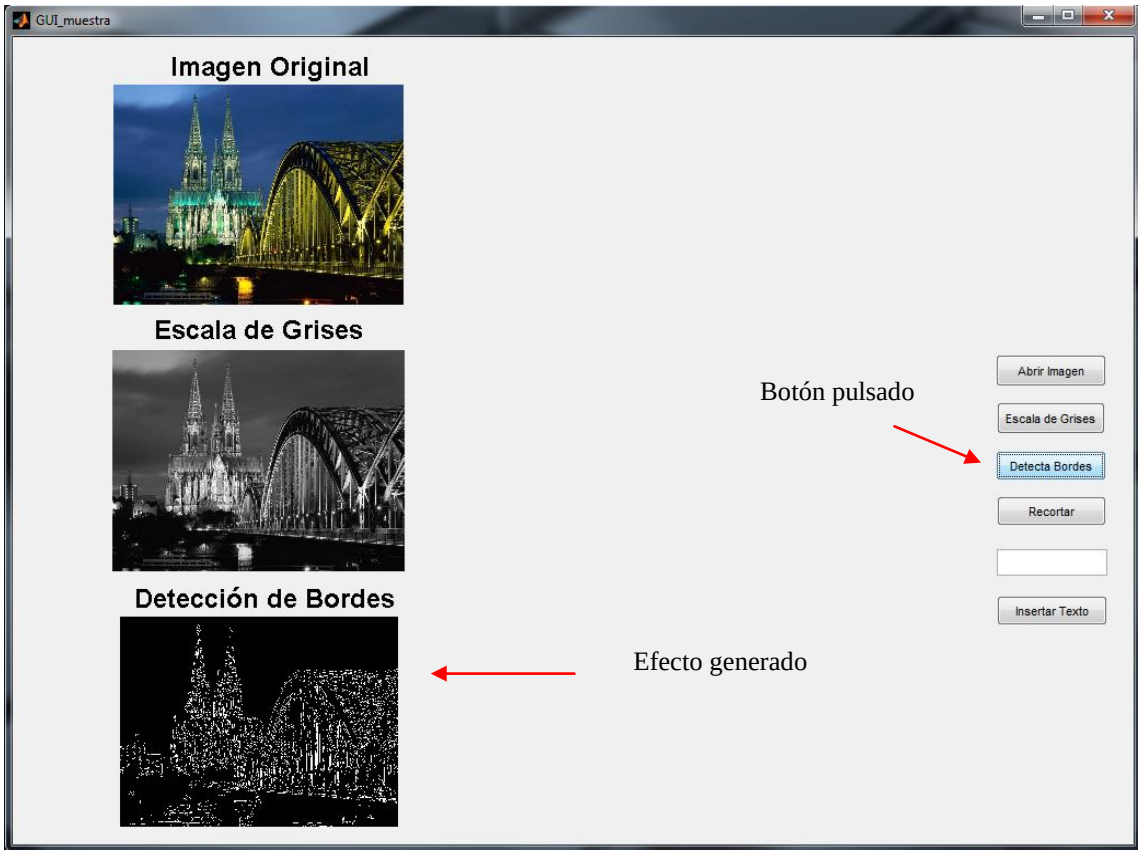


Figura 34. *Visualización de la imagen seleccionada con efecto de bordes.*

- Botón ***Recortar***

Dicho botón tiene como finalidad poder recortar, de forma interactiva, una sección de la imagen original y mostrar, en diferentes axes, dicha sección y sus efectos. Para ello debemos ir activando los axes necesarios para poder mostrar las imágenes necesarias.

```
% --- Executes on button press in btRecortar.
function btRecortar_Callback(hObject, eventdata, handles)
% hObject    handle to btRecortar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% activamos el axes del cual podremos seleccionar un área
% en nuestro caso será de la imagen original
axes(handles.Original);

% la función imcrop permite seleccionar una sección de la imagen
% de modo interactivo mediante el uso del ratón. También se pueden
% indicar las coordenadas pero, para nuestro finalidad, nos da más
% versatilidad la opción interactiva.

rec=imcrop;

% permitimos la visualización del axes donde mostrar el resultado
set(handles.OriginalRecorte,'Visible','on');
set(handles.textRecorteOriginal,'Visible','On');
% activamos el axes donde mostrar el resultado
axes(handles.OriginalRecorte);
imshow(rec);

% permitimos la visualización del axes donde mostrar el resultado
% del efecto EscalaGrises de la imagen recortada
set(handles.EscalaGrisesRecorte,'Visible','on');
set(handles.textRecorteProcesado,'Visible','On');
% activamos el axes donde mostrar el resultado escala de grises
recortado
axes(handles.EscalaGrisesRecorte);
% convertimos a escala de grises
bn=rgb2gray(rec);
% visualizamos el resultado
imshow(bn);

% permitimos la visualización del axes donde mostrar el resultado
% del efecto Bordes de la imagen recortada
set(handles.BordesRecorte,'Visible','on');
set(handles.textBordesRecorte,'Visible','On');
% activamos el axes donde mostrar el resultado escala de grises
recortado
axes(handles.BordesRecorte);
% convertimos y visualizamos la imagen recortada con el efecto bordes
imshow(edge(bn,'log'));
```

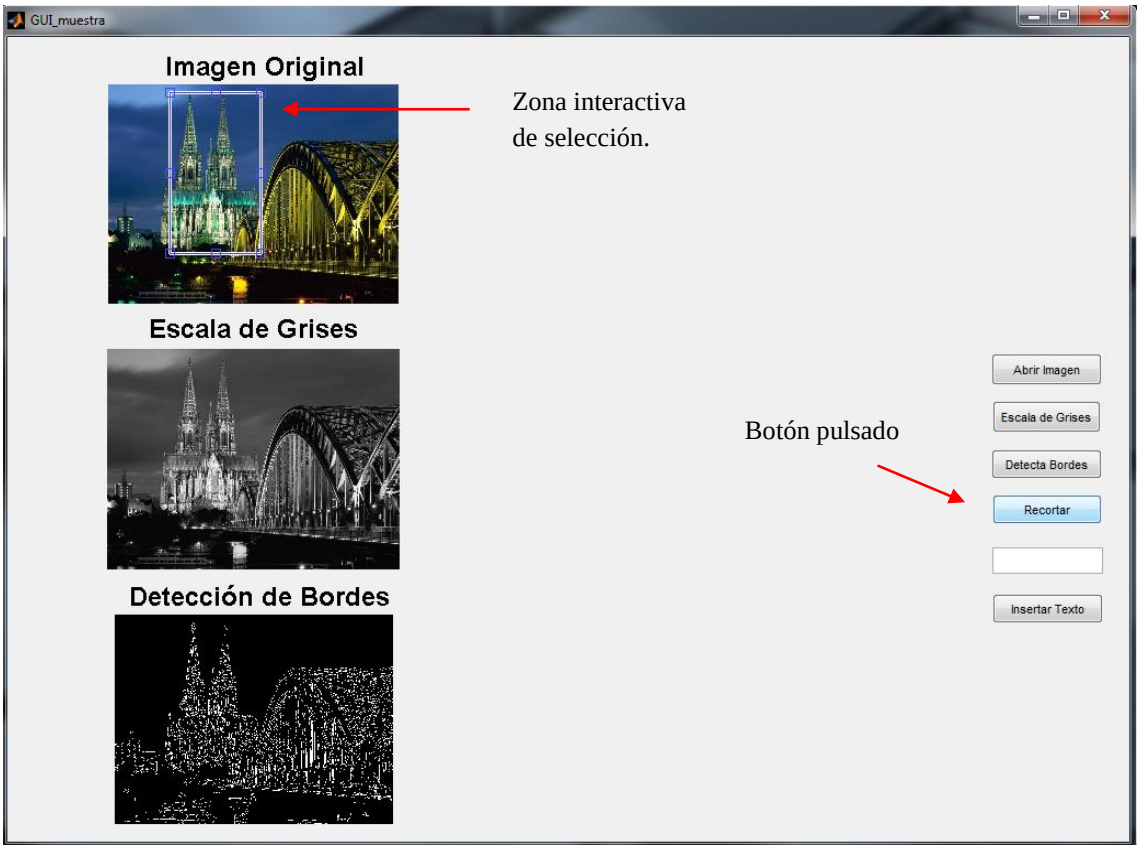


Figura 35. *Selección de la región interactiva..*

Una vez seleccionada la zona deseada se ejecuta el resto de la rutina mostrando lo siguiente:

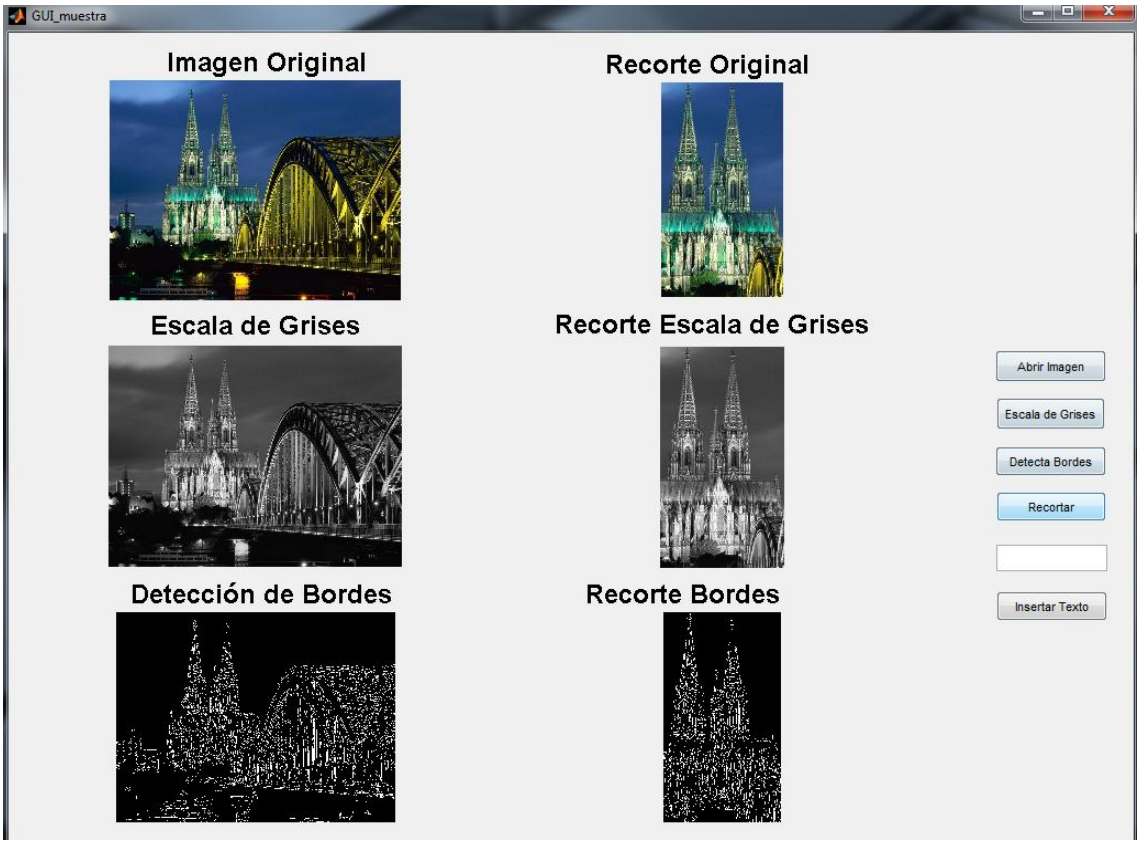


Figura 36. *Visualización de los efectos en la zona seleccionada.*

- Botón ***Insertar Texto***

Botón encargado de insertar en la imagen deseada el texto ubicado en el edittext. La posición del texto será la seleccionada, de forma interactiva, mediante un click de ratón sobre la imagen deseada.

```
% --- Executes on button press in btTexto.
function btTexto_Callback(hObject, eventdata, handles)
% hObject    handle to btTexto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% la función get toma valores o cadena del obejto indicado.
% en nuestro caso, toma la cadena introducida en el edittext "etexto"
% y la almacena en la variable "texto"
texto=get(handles.etexto,'String');
% comprobamos si la variable "texto" está vacía.
% en caso de estar vacía, mostramos un texto de advertencia
if isempty(texto)
    errordlg('Error. Debe introducir un texto','Mensaje de Error');
% si no se encuentra vacía, procedemos a insertar el texto en la
imagen
else
    % la función gtext insertar el texto ubicado en su primer
parámetro
    % en la posición indicada por un click de ratón.
    gtext(texto,'Color','w');
end
```

Como se ha indicado en el código, si no introducimos una cadena de texto en el *edittext* nos mostrará un mensaje de error. Dicho mensaje recuerda la obligación de introducir un texto.

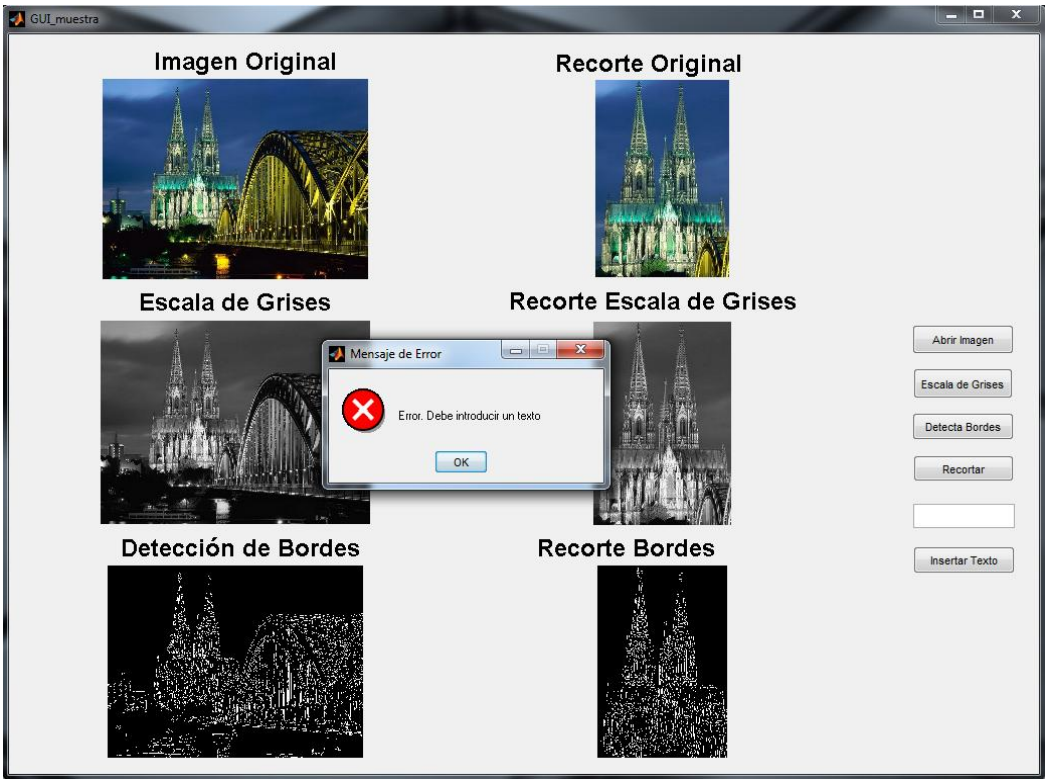


Figura 37. *Mensaje de error al pulsar* ***Insertar Texto****.*

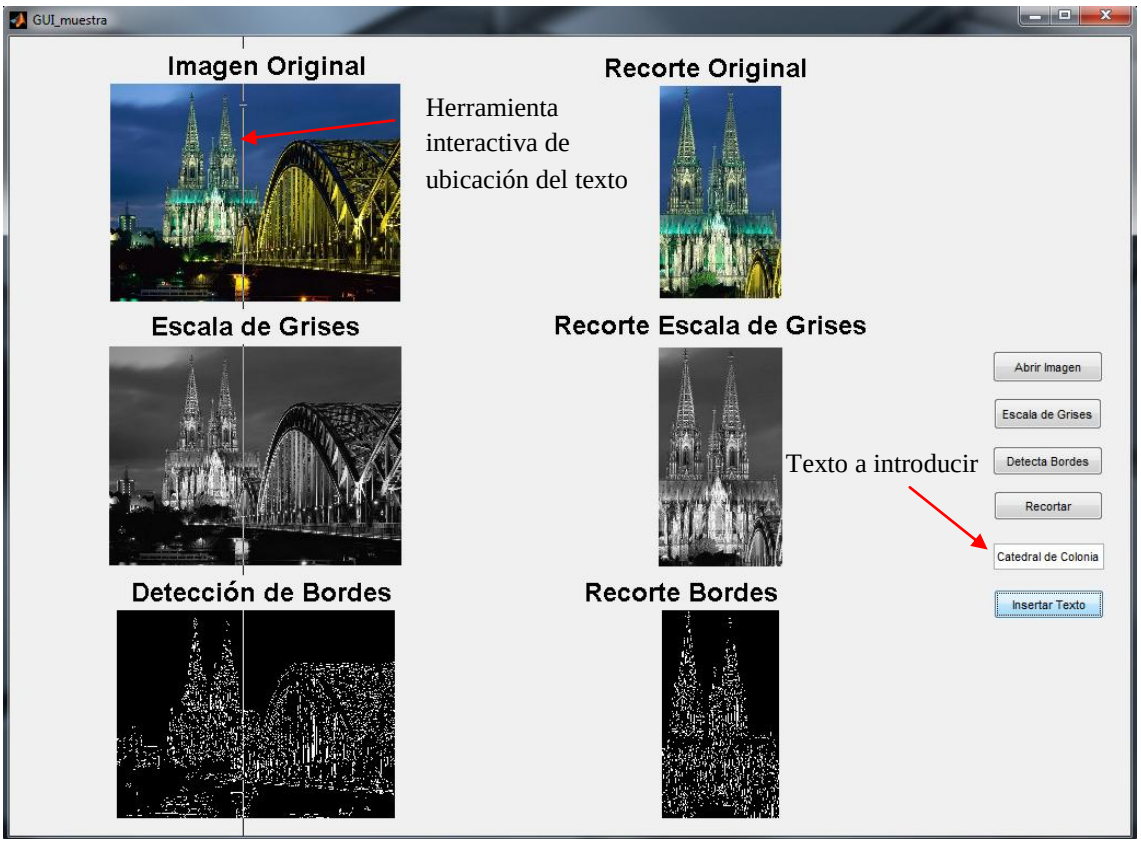


Figura 38. *Ubicación del texto introducido.*

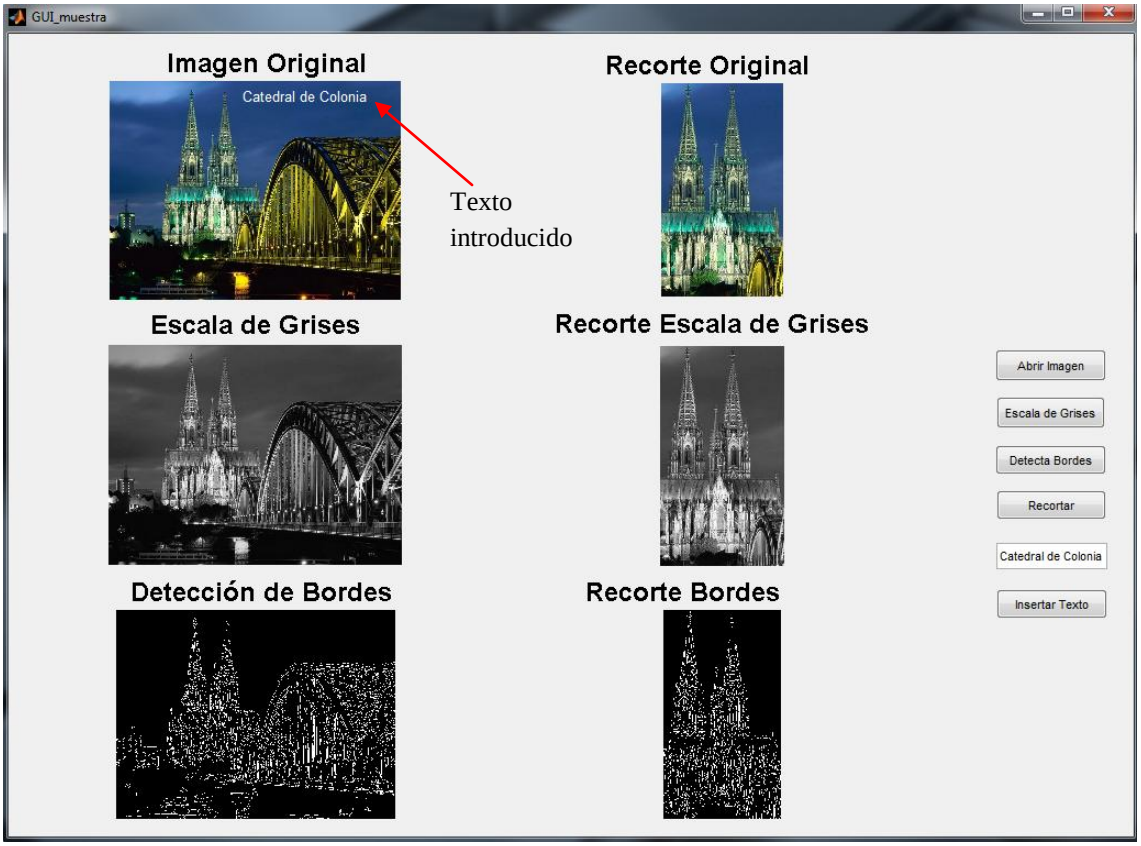


Figura 39. *Visualización del texto introducido en la posición seleccionada.*

4.3.7 Desarrollo de la interfaz gráfica propia del proyecto

Una vez explicados los conceptos básicos para entender cómo desarrollar una interfaz gráfica en Matlab procedemos al diseño de la herramienta encargada de visualizar los resultados de la monitorización de la red de sensores anteriormente expuesta.

Debido a la necesidad de mostrar información relativa a dos tipos de dispositivos diferentes, información de PDU (temperaturas y humedades) e información del SAI (tensiones de entrada, potencias de salida, modo de funcionamiento…), se ha optado por dividir la GUI en dos zonas de trabajo, una por cada dispositivo. Este diseño permite trabajar, de forma independiente, sobre cada dispositivo por lo que ofrece mayor versatilidad a la hora de tratar los datos recogidos.

Como vista general, el planteamiento seguido en el desarrollo de la interfaz gráfica ha sido la de optar por una herramienta de trabajo basado en gráficas de situación donde se muestra el estado de los dispositivos. Si dentro de estas gráficas se detectara algún problema, tendríamos la posibilidad de exportar dichos datos para hacer un tratamiento más exhaustivo de la situación.

4.3.7.1 Entorno de trabajo de las PDUs

Como se indicó anteriormente, las PDUs son dispositivos encargados de recolectar información sobre los consumos eléctricos por tomas además de temperaturas y humedades del ambiente. Junto a esto, como cada rack contiene dos sensores de temperaturas y humedades distribuidos en la zona baja y alta del rack, debemos ser capaces de procesar dicha información de la forma más clara posible para poder garantizar un entendimiento de la situación térmica del armario. Esto implica la generación de múltiples gráficas dentro de un mismo axes de la interfaz gráfica.

Debido a este principio, se ha optado por la inserción de un gran *axes* llamado *graficaPdu* el cual será el encargado de mostrar las gráficas necesarias. Además, la GUI contará con un menú desplegable *(pop-up menú)* llamado *popPdu* que será el encargado de albergar los diferentes dispositivos para poder ser seleccionados. Junto a esto, dispondremos de dos botones encargados de seleccionar las fechas de inicio y fin para seleccionar el rango de datos deseado.

Una vez tenemos los parámetros deseados (dispositivo, fecha de inicio y fecha de fin) procederemos a la obtención y representación de los datos. Para tal fin se dispone de un botón con el texto *Plot* identificado como *btPlotPdu.* Dicho botón recogerá los valores de fecha inicio y fecha fin y, mediante una llamada a la base de datos creada anteriormente, procederá a leer y procesar dichos valores con el fin de representarlos al usuario.

La distribución de los objetos se puede observar en la siguiente gráfica:

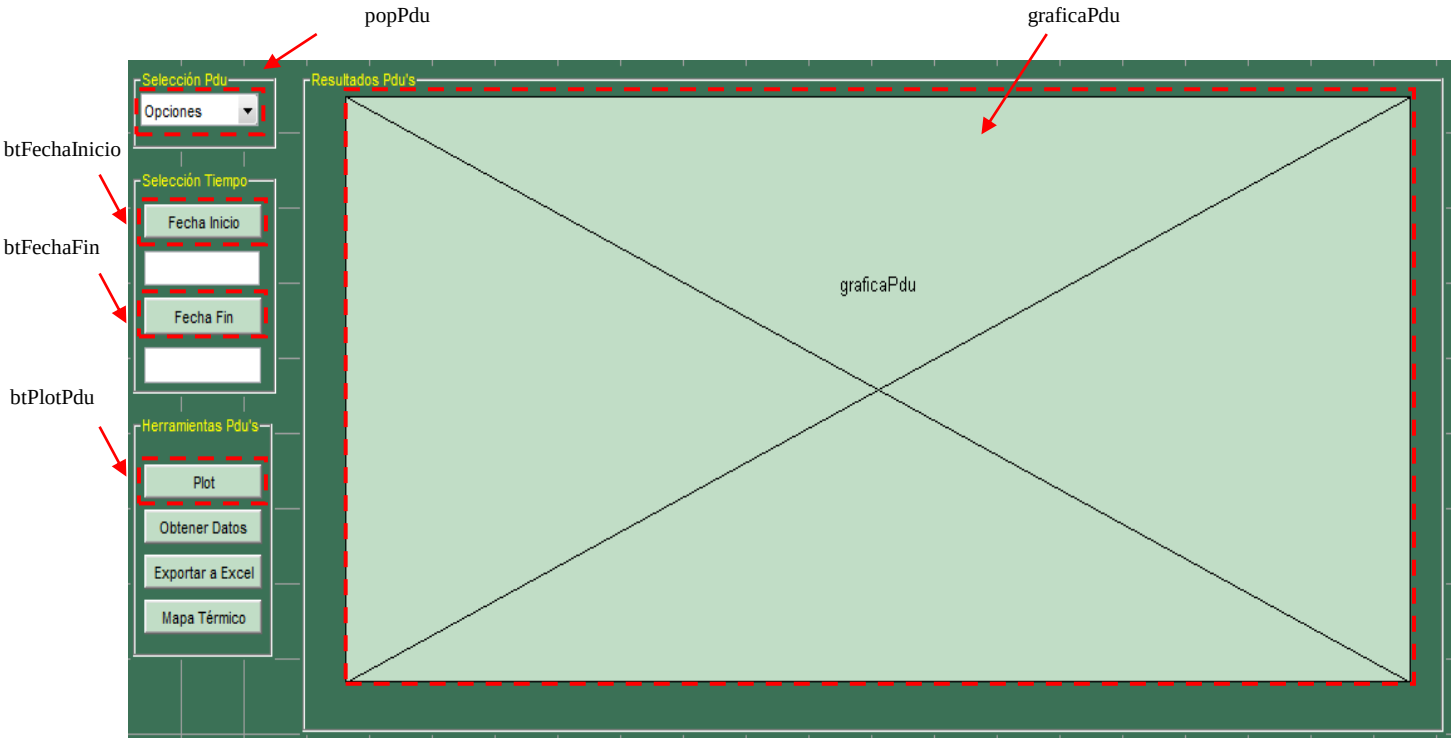


Figura 40. *Diseño de la Herramienta de las PDUs*

Tras la ejecución obtendríamos lo siguiente:

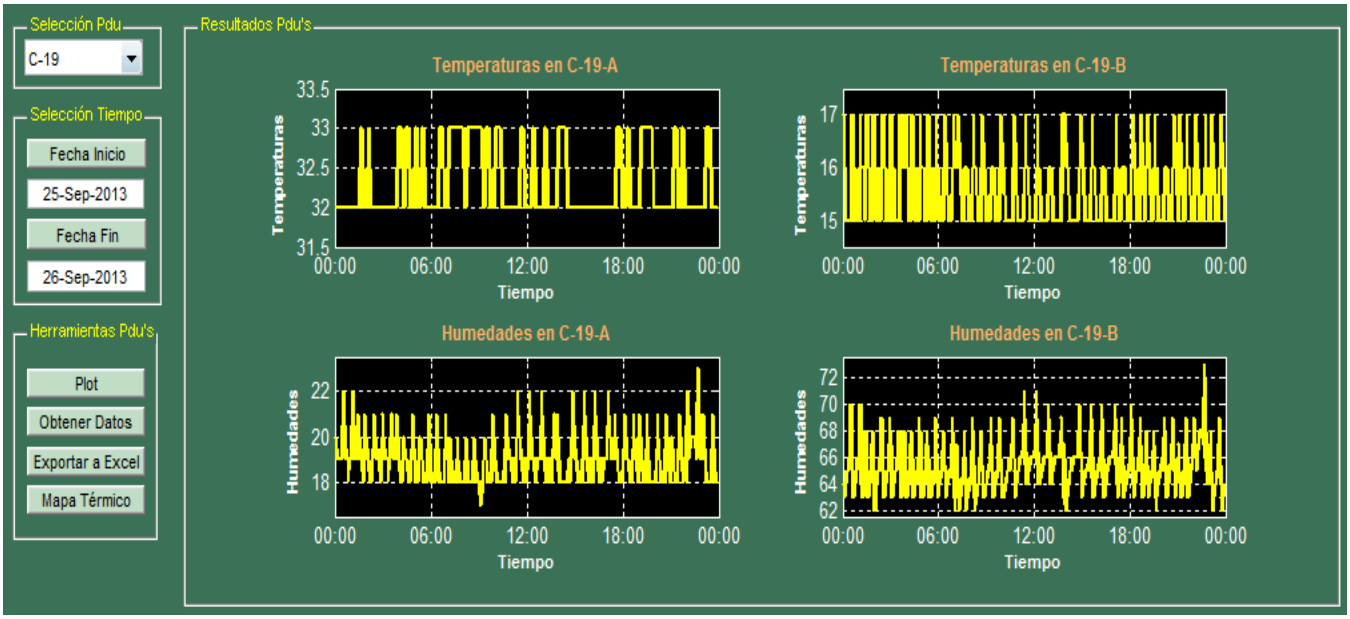


Figura 41. *Resultado de la ejecución de la herramienta Pdu*

Una vez hemos obtenido las gráficas de la Pdu seleccionada, disponemos de un par de botones que nos permiten realizar diversas tareas sobre esas gráficas. El botón *Obtener Datos* nos permitiría indicar un valor de datos a obtener de la gráfica (por ejemplo 5) y, tras 5 pulsación sobre la gráfica, obtendríamos las coordenadas X e Y de los 5 valores seleccionados mostrados en una tabla.

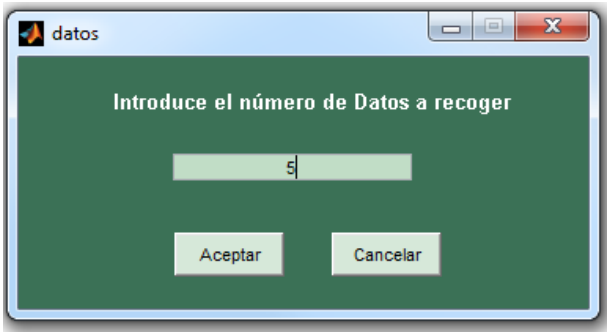


Figura 42. *Indicación de los puntos a obtener*

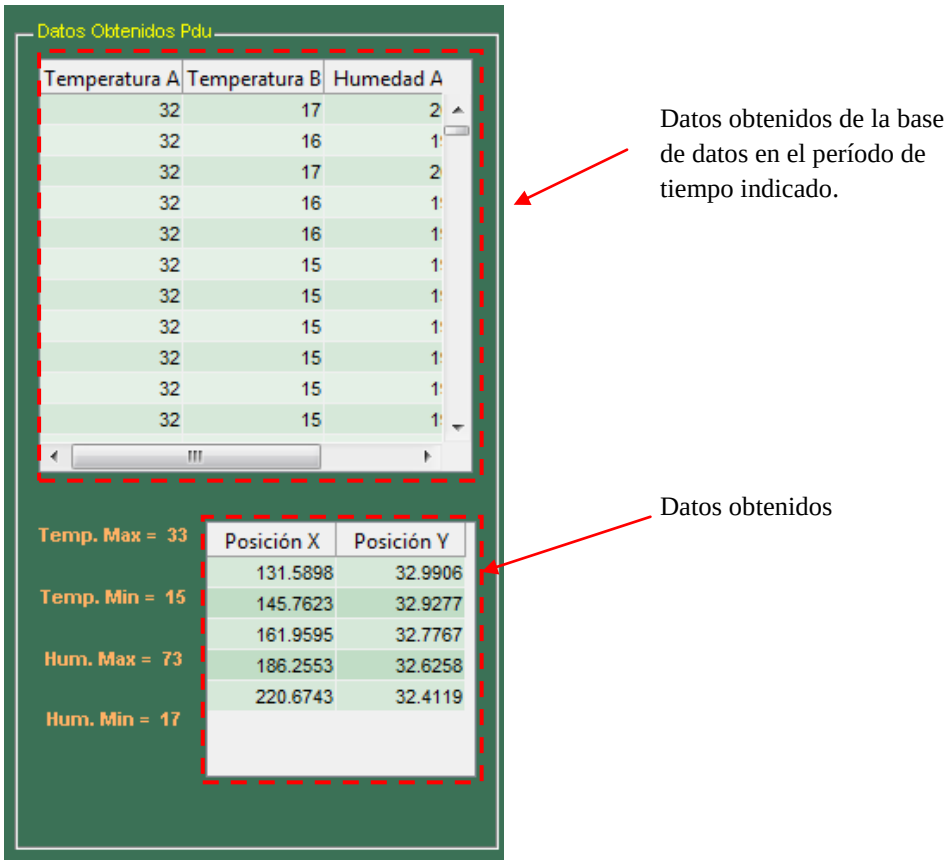


Figura 43. *Resultados obtenidos*

4.3.7.2 Entorno de trabajo de los SAIs o UPS

Como se indicó anteriormente, los SAIs o UPS son dispositivos que garantizan, durante un tiempo determinado, el suministro eléctrico a los dispositivos conectados a ellos. En caso de fallo eléctrico y fallo de la fuente secundaria (grupo electrógeno), pasarían a ser la última fuente de energía disponible por los equipos alojados en el CPD. Este hecho provoca que se cataloguen como elementos críticos ya que, si hubiese un fallo eléctrico y nadie se diese cuenta de dicho fallo, se podrían consumir las baterías de los SAIs provocando un apagado general de los equipos del CPD con la consiguiente caída de servicios. Debido a este hecho, se ha optado por la monitorización de las tensiones de entrada y las potencias de salida de los diferentes Sais para poder ver las posibles fluctuaciones derivadas de un servicio eléctrico inestable.

Para el desarrollo de la herramienta de monitorización de los Sais se ha optado por seguir el mismo diseño que en la herramienta de Pdus. Esto es, un gran axes identificado como *graficaSai* encargado de la generación de la gráficas necesarias, un menú desplegable identificado como *popSai* encargado de alojar los diferentes Sai existentes, dos botones encargados de seleccionar las fechas de inicio y fin para seleccionar el rango de datos deseado y un botón identificado como *btExcelSai* encargado de transferir los datos a una hoja de cálculo de Excel para el estudio en profundidad en caso de detección de avería.

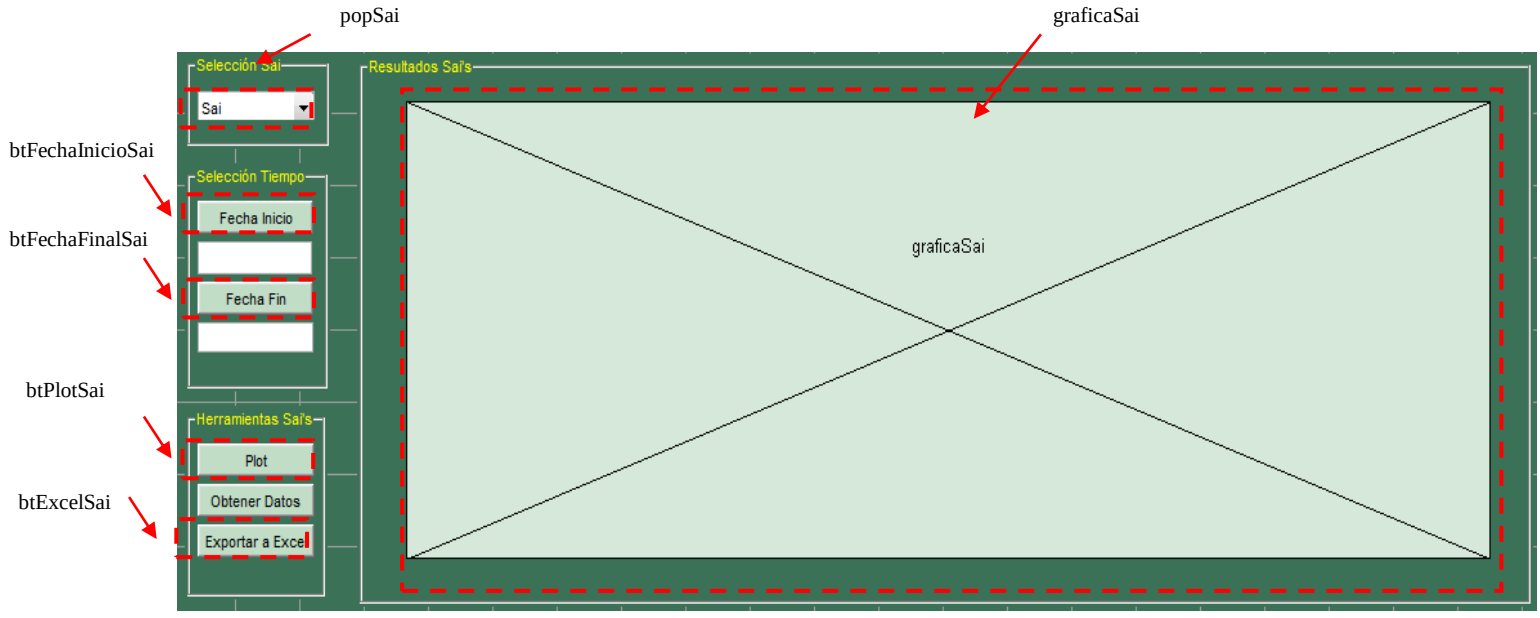


Figura 44. *Diseño de la Herramienta de los Sais o Ups*

Tras la ejecución se obtendría lo siguiente:

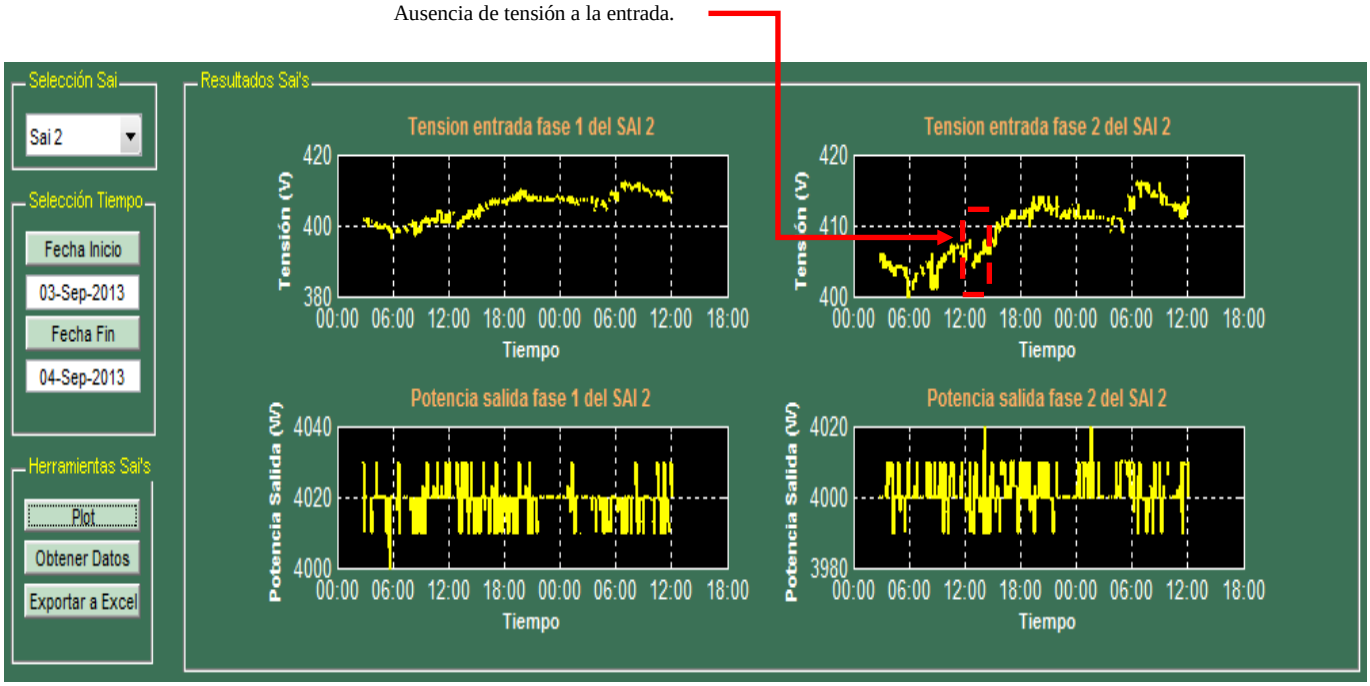


Figura 45. *Resultado de la ejecución de la herramienta de Sai.*

Una vez generadas las gráficas podríamos obtener valores concretos pulsando el botón *Obtener Datos* cuyo funcionamiento es idéntico al explicado en la herramienta de Pdus.

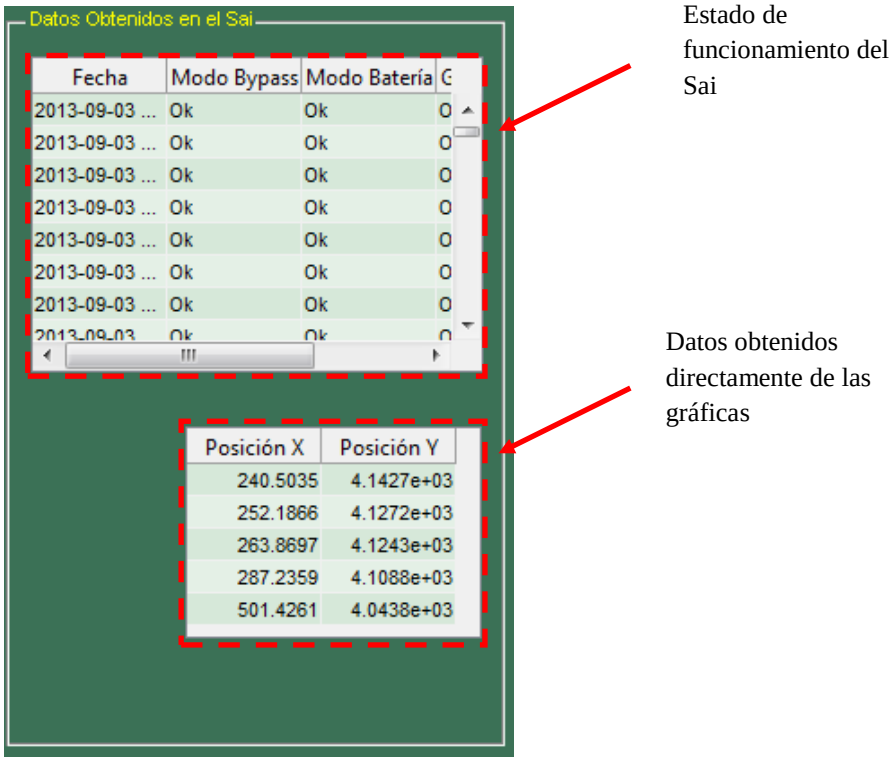


Figura 46. *Resultados obtenidos Sais*

Qué ocurriría si detectásemos múltiples cortes?. Desde esta herramienta no sería cómodo analizarlo por lo que podríamos exportar los datos a una hoja de Excel y tratarlos desde ahí. Para ello pulsaríamos sobre el botón *Exportar a Excel* lo que mostraría un cuadro de diálogo donde se nos solicita el nombre de la hoja de Excel donde colocar los datos y, posteriormente, el nombre del fichero donde guardar la información.

Figura 47. *Exportación de datos a Excel*

Una vez exportados los datos, podemos acceder a ellos como una hoja normal y tratarlos como sea necesario.

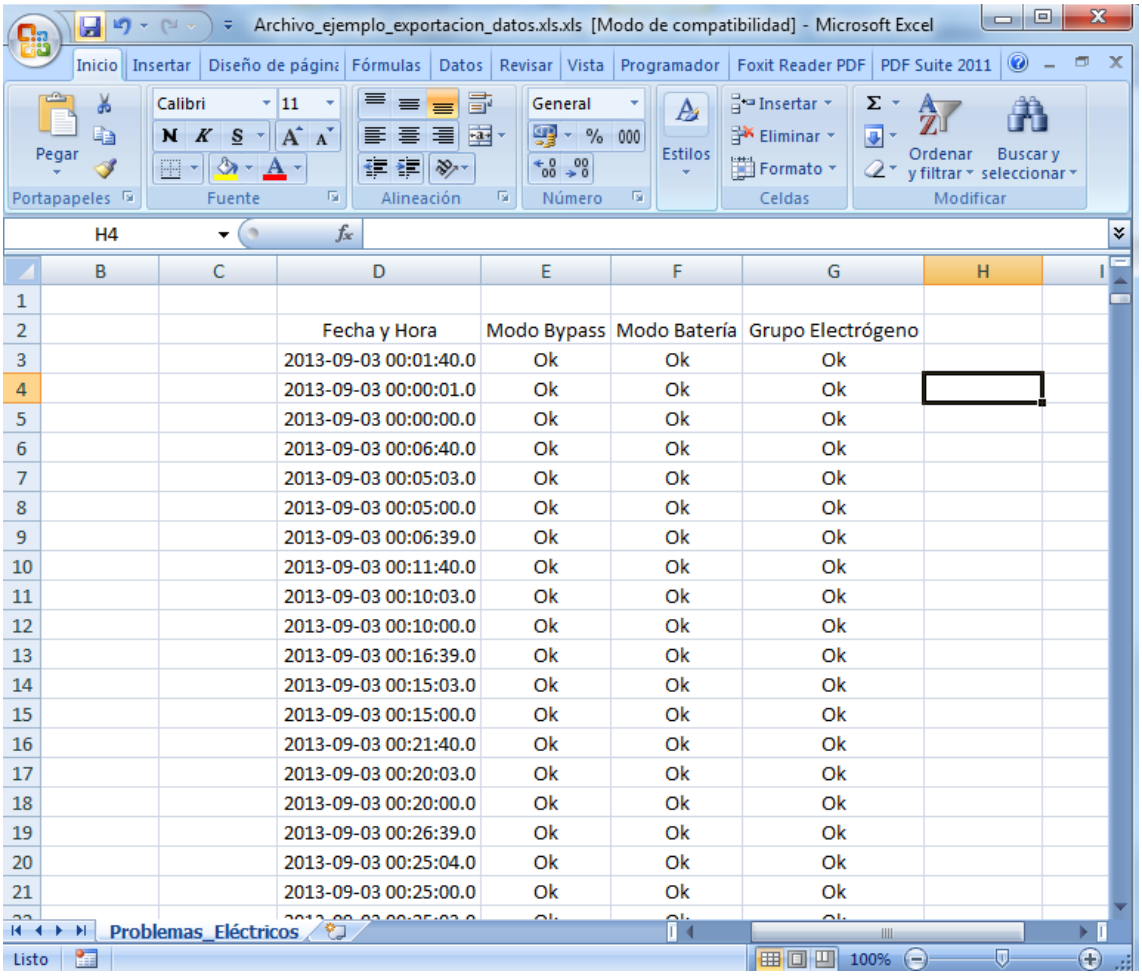


Figura 48. *Archivo generado tras la exportación de datos a Excel*

4.3.7.3 Desarrollo del mapa térmico

La finalidad máxima de este proyecto residía en la generación de un mapa térmico de las instalaciones de un centro de procesado de datos. Debido a la seguridad existente en estos centros, es imposible mostrar el plano del CPD por lo que se realizará esta parte con un plano de un CPD ficticio. En primer lugar, debemos identificar las áreas de interés. Para ello debemos obtener las coordenadas de dichas regiones. Esta operación la realizaremos con la función *imcrop* como se utilizó en la GUI de muestra para seleccionar una región a tratar en concreto. La única diferencia radica en las opciones devueltas ya que le indicaremos que nos devuelva las coordenadas de la región seleccionada. Para ello utilizaremos la función *imcrop* de la siguiente manera: *[I,rect] = imcrop*. En la variable *rect* guardamos las coordenadas del rectángulo seleccionado. Esta acción la repetiremos tantas veces como regiones deseemos incluir en el mapa térmico.

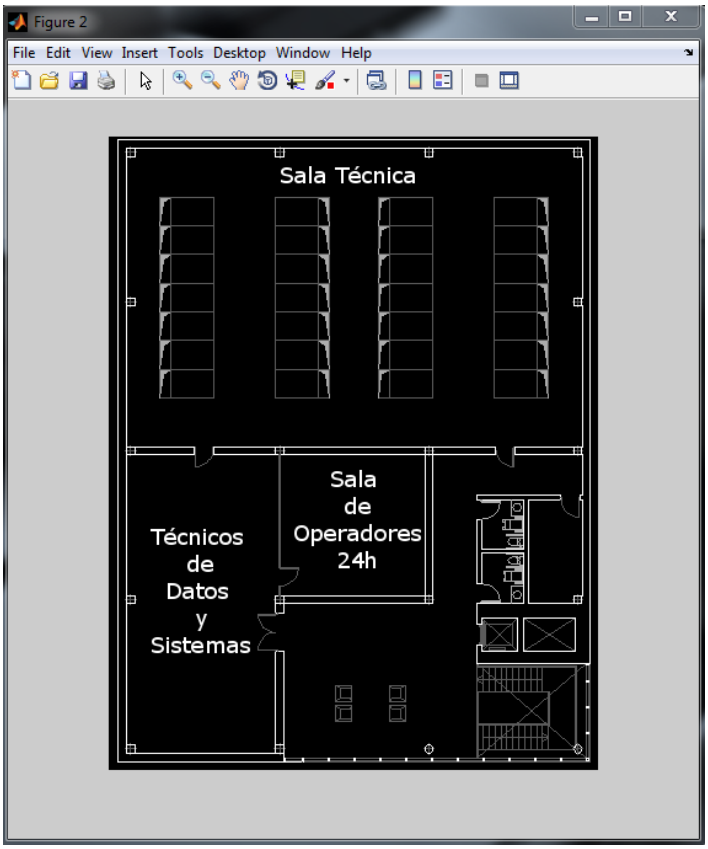


Figura 49. *Mapa Original*

En nuestro ejemplo analizaremos la situación térmica de 28 zonas correspondientes a los 28 Racks o Bastidores ubicados en el interior de nuestro CPD . Una vez definidas todas las zonas de evaluación, uniremos todas las coordenadas en una matriz Nx4 siendo N el número de zonas a evaluar. Dicha matriz será guardada mediante la instrucción *save* para poder ser cargada al inicio de la aplicación mediante la función *load* y tener definidas las zonas de evaluación.

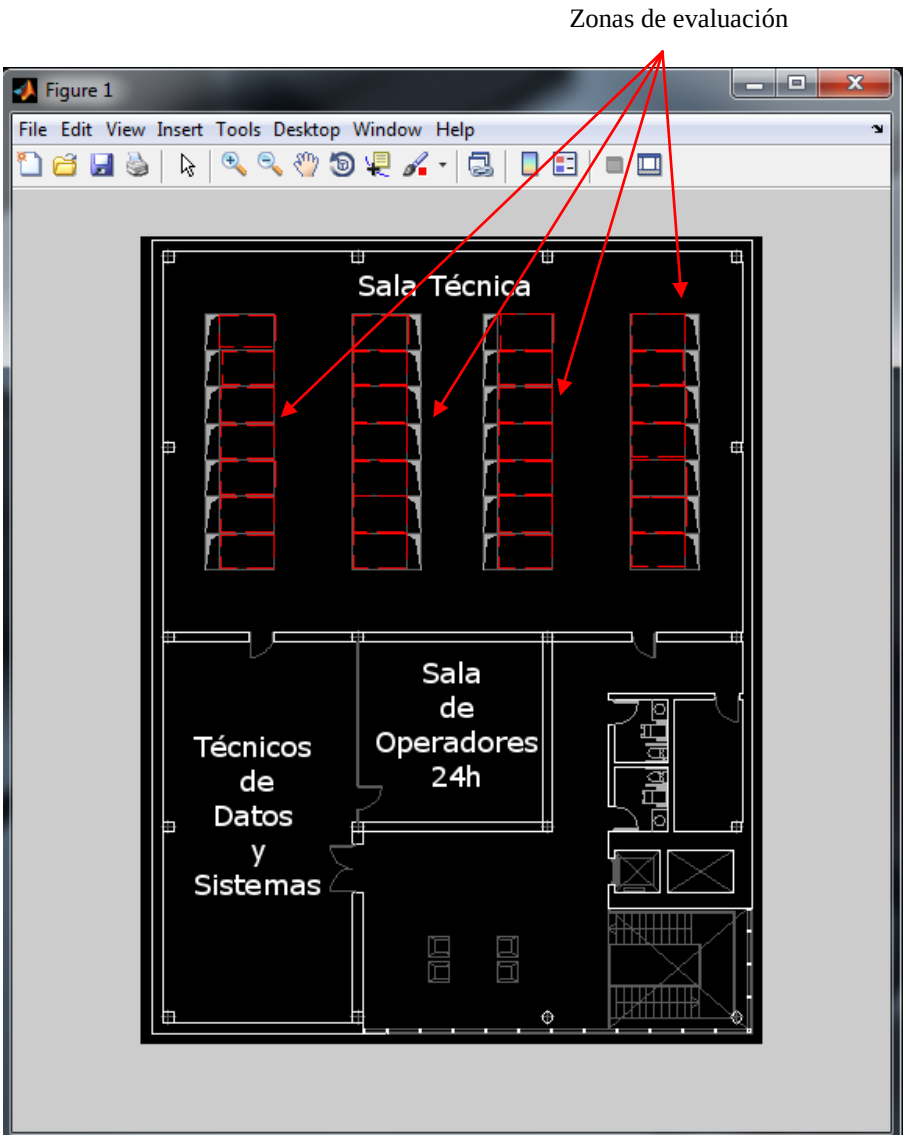


Figura 50. *Zonas de estudio.*

Llegados a este punto estaríamos listos para ejecutar el análisis térmico de las zonas. Para ello ejecutamos la función *mapa_termico* para realizar el cálculo de forma automática. Dicha función tiene el siguiente aspecto:

```
function mapa_termico

% cargamos la variable donde se ubican las coordenadas
load coordenadas;
% leemos y mostramos el plano a evaluar
I=imread('cpd.png');
imshow(I);
hold on;
title (datestr(now));
% obtenemos las coordenadas de las diferentes zonas
Rect=coordenadas;
% calculamos las dimensiones de cada zona de evaluación
% imcrop devuelve las coordenadas como [XMIN YMIN WIDTH HEIGHT]

% a y b valores máximos y mínimos de temperaturas

 [A,B]=meshgrid(a,b);

for i=1:length(Rect)
    inicio_x=Rect(i,1);
    fin_x=inicio_x+Rect(i,3);
    inicio_y=Rect(i,2);
    fin_y=inicio_y+Rect(i,4);

    x=linspace(inicio_x+2,fin_x-2,21);
    y=linspace(inicio_y+2,fin_y-2,21);
    [X,Y]=meshgrid(x,y);

    %función de la temperatura


    pcolor(X,Y,Z);
    shading('interp')

end
```

Una vez ejecutada la función *mapa_termico* obtendríamos lo siguiente:

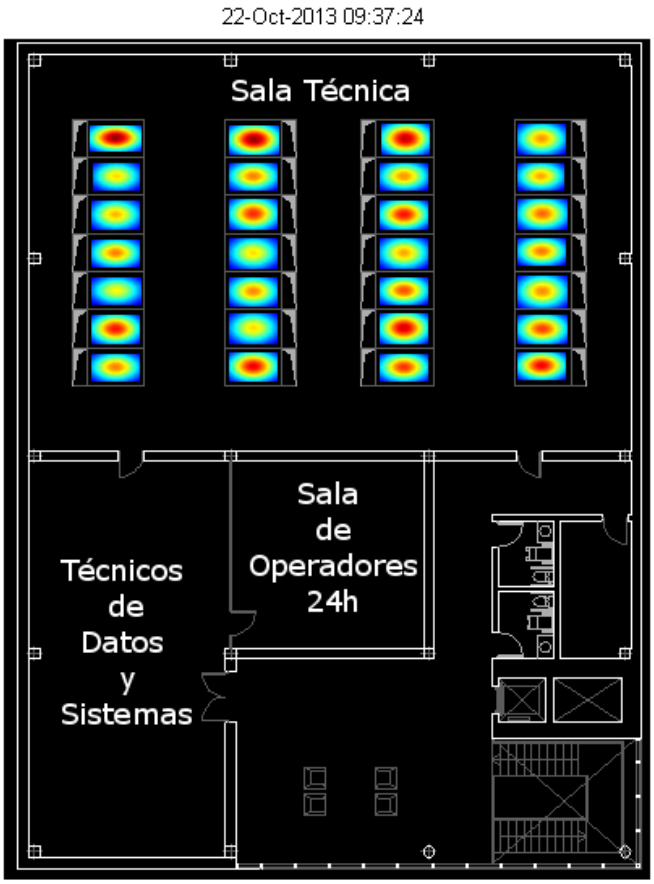


Figura 51. *Estado final del mapa térmico.*

En el mapa resultado se puede observar los diferentes focos de calor en función de los niveles de temperaturas registrados en su interior. Este hecho proporcionaría una fotografía térmica del estado del CPD en un instante de tiempo concreto lo que facilitaría la comparativa y la relación directa entre incremento de temperatura e incremento de consumo energético.

A modo de ejemplo, mostraremos varias capturas para poder visualizar la diferencia térmica en cada instante de tiempo. Esto implica poder realizar un estudio de cómo varía la temperatura en función del instante temporal relacionándolo con la actividad de negocio de cada cliente alojado en el CPD.

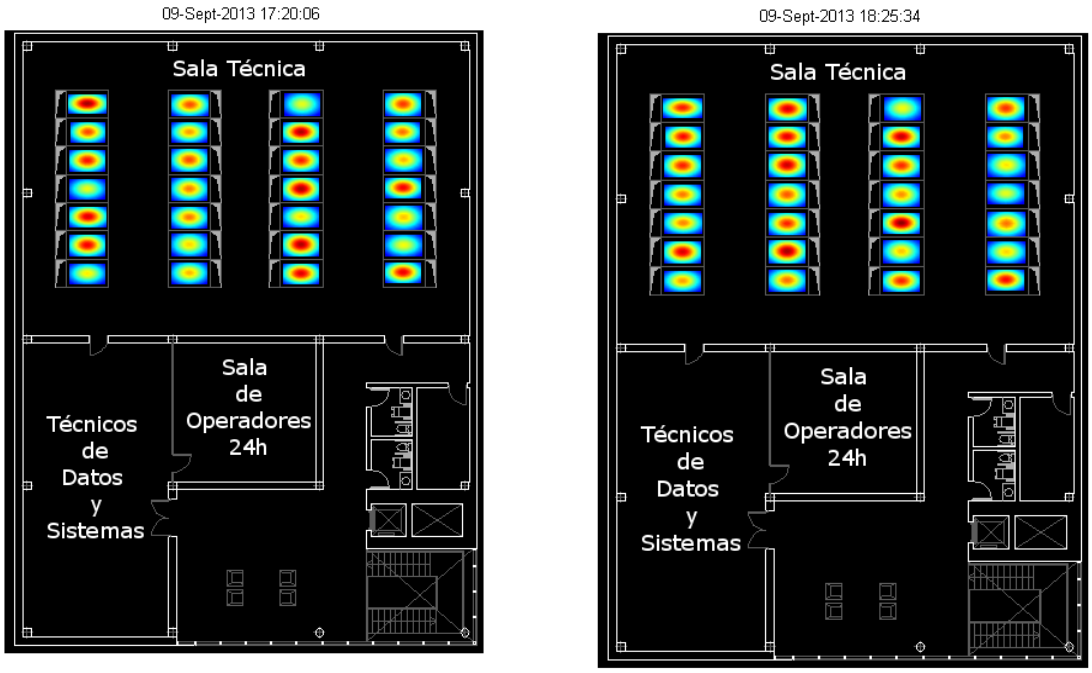


Figura 52. *Comparativa térmica en función del instante temporal*

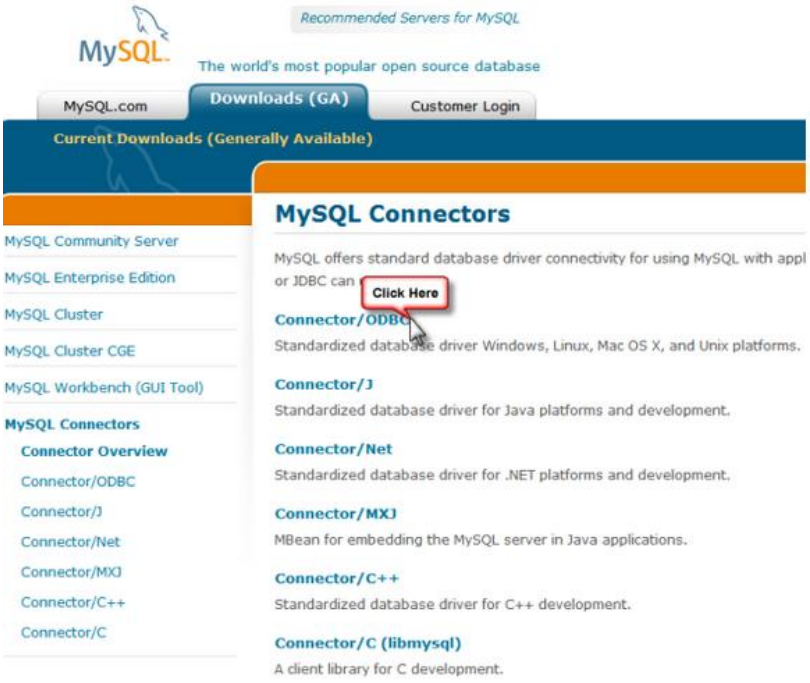
4.4 Conexión entre la aplicación y la base de datos

Uno de los mayores problemas planteados ha sido la conexión de la aplicación con la base de datos de donde obtener los valores. Para resolver dicho problema, Matlab contiene unas herramientas capaces de facilitar dicha tarea.

En primer lugar, debemos crear un conector ODBC para permitir la conexión entre nuestra aplicación y la base de datos alojada en nuestra máquina virtual.

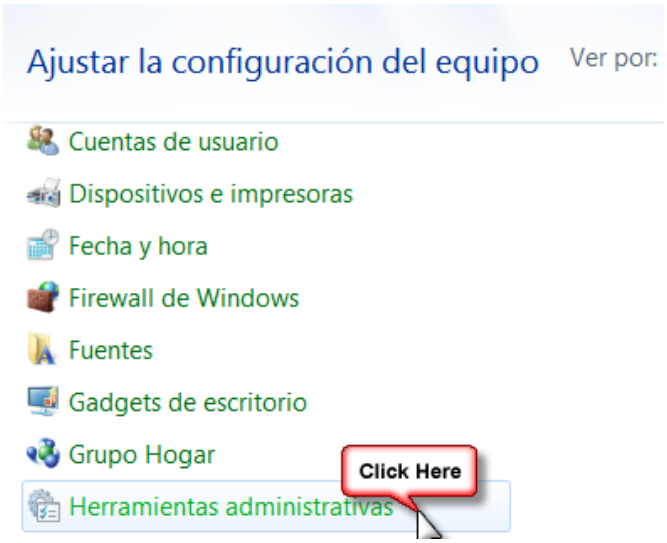
Para ello debemos seguir los siguientes pasos:

- Instalar el conector acorde a la base de datos que tengamos. En nuestro caso necesitamos un conector MySQL. Podemos obtenerlo del siguiente enlace: http://www.mysql.com/downloads/connector/

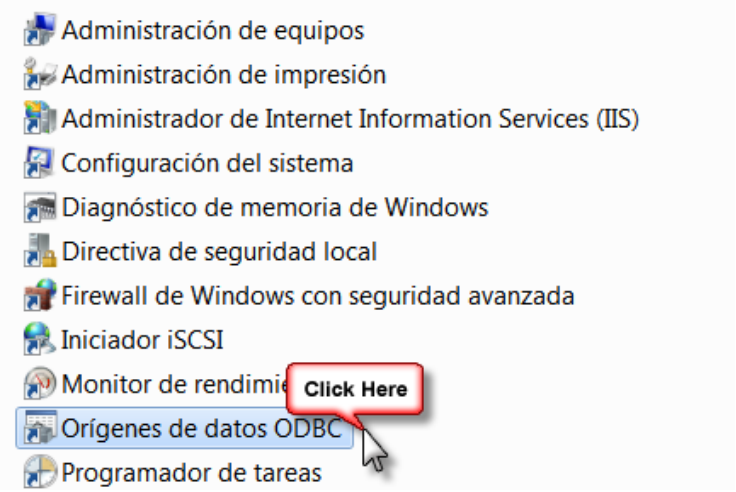


- Instalar el ejecutable descargado anteriormente. Nótese que dicho ejecutable debe ir acorde a nuestro sistema de 32 o 64 bits.

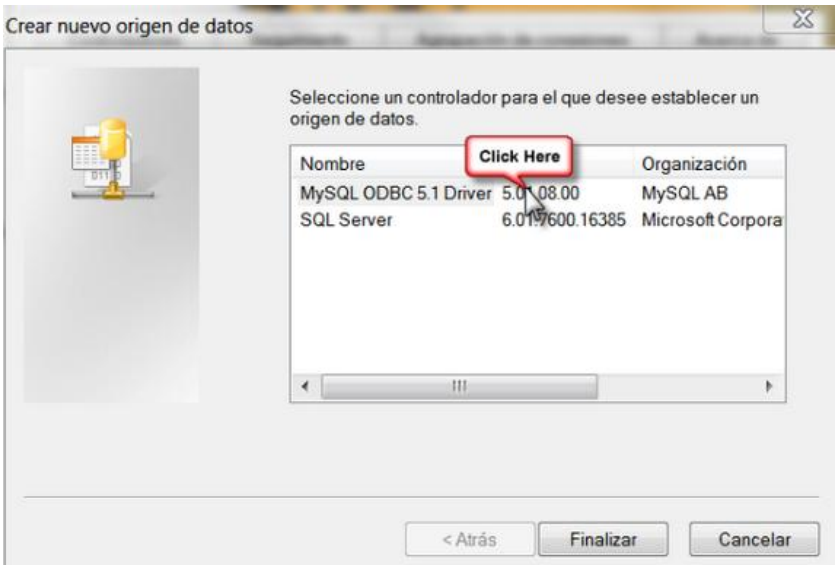
- Una vez finalizada la instalación dispondremos de nuestro conector en los repositorios de Windows. Para hallarlo debemos ir a Panel de Control + Herramientas Administrativas.



- Dentro de Herramientas Administrativas debemos encontrar la opción *Orígenes de datos ODBC.*



- Creamos un nuevo origen de datos seleccionando el conector instalado previamente.



- Configuramos los parámetros de acceso a nuestra base de datos.

Puerto de conexión a la base de datos. Por defecto, 3306 para MySQL.

- Pulsamos Test. Si aparece nuestra base de datos, el proceso ha finalizado correctamente.

Desde este momento nuestra base de datos sería accesible desde matlab. Para ello, debemos ejecutar unas instrucciones para realizar la conexión entre matlab y la base de datos.

Un ejemplo de conexión sería la siguiente función:

```
%-------------------------------------------------------------------------

function[temperaturas,humedad]=conexion_bd_pdu(conector,user,password,
elemento,fecha_inicio,fecha_final)

% conector = identificador del conector ODBC
% user = usuario de la base de datos
% password = contraseña de la base de datos
% elemento = pdu
% fecha_inicio= fecha inicial del análisis
% fecha_final = fecha final del análisis

bp=waitbar(0,'Obteniendo datos. Por favor, espera ...');

I=0.1; %contador para la barra de progreso

%creo la conexion a la base de datos
conn=database(conector,user,password);
I=0.2;
waitbar(I,bp);

%ejecuto la instruccion para obtener los datos

elemento=[char(39) elemento char(39)];
fecha_inicio=[char(39) fecha_inicio char(39)];
fecha_final=[char(39) fecha_final char(39)];

%isntrucción MySQL para obtener los datos buscados

texto_temp=['select pdu_seleccionada_temperatura from pdu p INNER JOIN
pdu_seleccionada ps ON p.pdu_id=ps.pdu_id AND p.pdu= ' elemento ' AND
ps.last_update BETWEEN DATE(' fecha_inicio ') AND DATE(' fecha_final
')'];

texto_hum=['select pdu_seleccionada_humedad from pdu p INNER JOIN
pdu_seleccionada ps ON p.pdu_id=ps.pdu_id AND p.pdu= ' elemento ' AND
ps.last_update BETWEEN DATE(' fecha_inicio ') AND DATE(' fecha_final
')'];

%ejecuto la conexión con la instrucción necesaria para obtener los
datos buscados

temp=exec(conn,texto_temp);

I=0.6;

waitbar(I,bp);

hum=exec(conn,texto_hum);

I=0.75;
waitbar(I,bp);
```

```
I=0.9;
waitbar(I,bp);

%indico cómo quiero que sme muestre los valores devueltos
setdbprefs('DataReturnFormat','cellarray');
%obtengo el número concreto de valores que quiero
temperaturas=fetch(temp);
humedad=fetch(hum);
% fechas=fetch(fechas);

waitbar(I,bp);
%convierto el formato devuelto a formato manejable de los valores
obtenidos
%(temperaturas.data)
temperaturas=cell2mat(temperaturas.data);
humedad=cell2mat(humedad.data);
% fechas=cell2mat(fechas.data);

waitbar(I,bp);
close(bp)
close(conn);


%------------------------------------------------------------------------
```

Tras la ejecución de dicha función, tendríamos disponibles los datos *temperaturas y humedades* obtenidos de la base de datos de los días seleccionados. A continuación, podríamos crear las gráficas con esos datos o tratarlos según nuestras necesidades.

5. CONCLUSIONES Y LÍNEAS FUTURAS

El objetivo de este trabajo era el desarrollo de una red de monitorización y la adquisición de los datos necesarios para, una vez obtenidos y tratados, poder encontrar la forma de reducir el impacto medioambiental que generan los CPDs debido a su alto consumo energético y, por ende, la reducción del impacto económico que supondría la optimización de dicho consumo.

Para llegar a este punto, dicho trabajo fue dividido en cuatro partes claramente diferenciadas:

1. Estudio de las instalaciones y necesidades para desarrollar la red de monitorización.
2. Desarrollo y despliegue de la red de monitorización encargada de la adquisición de datos.
3. Implementación de una base de datos capaz de alojar los valores obtenidos de la monitorización. Junto a esto, desarrollo de la tarea programada para automatizar el trabajo de volcado de datos a la base de datos.
4. Desarrollo de la interfaz gráfica que nos permita visualizar las variables tanto térmicas como el modo de funcionamiento del Sai así como el mapa térmico de la sala técnica.

A lo largo de este trabajo se ha ido mostrado el correcto funcionamiento de cada parte de forma independiente. Por lo tanto, podemos concluir que se ha cubierto las necesidades planteadas en dicho trabajo satisfactoriamente.

Al tratarse de una versión inicial, podemos concluir que el sistema funciona correctamente. Una vez comprobada la capacidad que ofrece la monitorización del Centro de Procesado de Datos, se puede optar por aumentar las variables de monitorización con el fin de perfeccionar el grado de correlación entre la temperatura y el consumo energético. Sería interesante tener en cuenta el funcionamiento de los equipos TI dentro de los racks ya que son los mayores generadores de calor. Debido a esto sería necesario obtener el valor de las siguientes variables:

- Frecuencia de trabajo de los equipos TI.
- Número de cpus de los equipos TI.
- Carga de trabajo en función del tiempo de los equipos TI.

Junto a estas nuevas variables, sería necesario tener una referencia térmica del exterior con la que observar también el grado de dependencia con el aire exterior que recogerían las climatizadoras.

La inclusión de nuevas variables implicaría aumentar el número de registros en la base de datos por lo que se debería optar por una base de datos Oracle la cual permite manejar tablas con mayor número de registros.

Una primera línea futura de trabajo sería la depuración de la representación gráfica donde se incluya, simultáneamente, temperaturas y humedades. Este hecho provocaría obtener la relación directa entre dichas variables ya que, mediante las variaciones de la temperatura, podemos realizar pequeñas correcciones de la humedad para mantenerla dentro del umbral óptimo.. Este hecho facilitaría la comparativa entre períodos de igual duración ubicados en diferentes épocas a lo largo del año.

Una segunda línea futura de trabajo sería el desarrollo de una interfaz 3D donde poder ver el estado real de cada rack y localizar los equipos que mayor temperatura disipan en función de la carga de trabajo que soportan. Una vez localizados los equipos se podría plantear cambiar la distribución de los dispositivos dentro del rack para poder optimizar el flujo de aire frío inyectado en los bastidores debido a un impacto directo sobre los equipos cuyo caudal de aire caliente generado es mayor.

Por último, una tercera línea futura de trabajo sería el desarrollo de un algoritmo de cálculo computacional capaz de tratar todos los datos en busca de una expresión que fuese capaz de aproximar el

impacto económico que tendría el consumo energético en la empresa en los meses venideros. Este hecho produciría una mejoría en la gestión del capital de la empresa.

ANEXO I. FUNCIONES

Obtención de Datos de las Pdu

```
Function[temperaturas,humedad,fechas]=conexion_bd_pdu(conector,user,password,elemento,fecha_inicio,fecha_final)
%[temperaturas,humedad]=conexion_bd_pdu(conector,user,password,elemento,fecha_inicio,fecha_final)

bp=waitbar(0,'Obteniendo datos. Por favor, espera ...');
I=0.1; %contador para la barra de progreso
%creo la conexion a la base de datos
conn=database(conector,user,password);
I=0.2;
waitbar(I,bp);

%ejecuto la instruccion para obtener los datos
elemento=[char(39) elemento char(39)]; %con char(39) introduzco comillas simples
fecha_inicio=[char(39) fecha_inicio char(39)]; % '2013-08-20'
fecha_final=[char(39) fecha_final char(39)];

texto_temp=['select pdu_seleccionada_temperatura from pdu p INNER JOIN pdu_seleccionada ps ON p.pdu_id=ps.pdu_id AND p.pdu= ' elemento ' AND ps.last_update BETWEEN DATE(' fecha_inicio ') AND DATE(' fecha_final ')'];

texto_hum=['select pdu_seleccionada_humedad from pdu p INNER JOIN pdu_seleccionada ps ON p.pdu_id=ps.pdu_id AND p.pdu= ' elemento ' AND ps.last_update BETWEEN DATE(' fecha_inicio ') AND DATE(' fecha_final ')'];

texto_fecha=['select ps.last_update from pdu p INNER JOIN pdu_seleccionada ps ON p.pdu_id=ps.pdu_id AND p.pdu= ' elemento ' AND ps.last_update BETWEEN DATE(' fecha_inicio ') AND DATE(' fecha_final ')'];

temp=exec(conn,texto_temp);
I=0.6;
waitbar(I,bp);
hum=exec(conn,texto_hum);
I=0.75;
waitbar(I,bp);
fechas=exec(conn,texto_fecha);
I=0.9;
waitbar(I,bp);
%indico cómo quiero que sme muestre los valores devueltos
setdbprefs('DataReturnFormat','cellarray');
%obtengo el número concreto de valores que quiero
temperaturas=fetch(temp);
humedad=fetch(hum);
 fechas=fetch(fechas);
waitbar(I,bp);
%convierto el formato devuelto a formato manejable
temperaturas=cell2mat(temperaturas.data);
humedad=cell2mat(humedad.data);
fechas=cell2mat(fechas.data);
waitbar(I,bp);
% Cierro la conexión
close(bp)
close(conn);
```

Obtención de Datos del Sai

```
function[potencias_salidas,tensiones_entradas,fechas_sai]=conexion_bd_sai(conector,user,password,elem
ento,fecha_inicio,fecha_final)

bp=waitbar(0,'Obteniendo datos. Por favor, espera ...');
I=0.1; %contador para la barra de progreso
%creo la conexion a la base de datos
conn=database(conector,user,password);
I=0.2;
waitbar(I,bp);
%ejecuto la instruccion para obtener los datos

elemento=[char(39) elemento char(39)]; %con char(39) introduzco comillas simples
fecha_inicio=[char(39) fecha_inicio char(39)]; % '2013-09-20' por ejemplo
fecha_final=[char(39) fecha_final char(39)];

texto_potencia_salida=['select ss.potencia_salida1,ss.potencia_salida2,ss.potencia_salida3 from sai s
INNER JOIN sai_seleccionado ss ON s.sai_id=ss.sai_id AND s.sai= ' elemento ' AND ss.last_update
between DATE(' fecha_inicio ') AND DATE (' fecha_final ')'];
texto_tensiones_entrada=['select ss.tension_entrada1,ss.tension_entrada2,ss.tension_entrada3 from sai s
INNER JOIN sai_seleccionado ss ON s.sai_id=ss.sai_id AND s.sai = ' elemento ' AND ss.last_update
between DATE(' fecha_inicio ') AND DATE (' fecha_final ')'];
texto_fechas_sai=['select ss.last_update from sai s INNER JOIN sai_seleccionado ss ON
s.sai_id=ss.sai_id AND s.sai = ' elemento ' between DATE(' fecha_inicio ') AND DATE (' fecha_final ')'];

potencias_salidas=exec(conn,texto_potencia_salida);
fechas_sai=exec(conn,texto_fechas_sai);

I=0.6;
waitbar(I,bp);
tensiones_entrada=exec(conn,texto_tensiones_entrada);

I=0.8;
waitbar(I,bp);
%indico cómo quiero que me muestre los valores devueltos
setdbprefs({'DataReturnFormat';'NullStringRead'},{'cellarray','NULL'});

%obtengo el número concreto de valores que quiero
potencias_salidas=fetch(potencias_salidas,400);
tensiones_entrada=fetch(tensiones_entrada,400);
fechas_sai=fetch(fechas_sai,400);
waitbar(I,bp);

%convierto el formato devuelto a formato manejable de los valores obtenidos
%(temperaturas.data)
 potencias_salidas=cell2mat(potencias_salidas.data);
tensiones_entradas=cell2mat(tensiones_entrada.data);
fechas_sai=cell2mat(fechas_sai.data);

waitbar(I,bp);close(bp);
close(conn);
```

Dibuja Funciones

```
function dibuja_funciones(temperaturas,humedad,num_graf,pdu)
% Función que adecúa la cantidad de gráficas en el axes seleccionado en función de los valores monitorizados.

if num_graf==1

subplot(2,1,1)
    plot(1:length(temperaturas),temperaturas,'Color','y','LineWidth',2);
   axis([1 length(temperaturas) min(temperaturas)-0.5 max(temperaturas)+0.5]);
   grid on;
   texto=['Temperaturas en ' pdu];
   title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
   xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
   ylabel('Temperaturas','fontsize',8,'fontweight','bold','color','white');
   set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
   set(gca,'XColor','w','YColor','w');


   subplot(2,1,2)
   plot(1:length(humedad),humedad,'y','Linewidth',2);
   axis([1 length(humedad) min(humedad)-0.5 max(humedad)+0.5 ])
   grid on;
   texto=['Humedades en ' pdu(1,:)];
   title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
   xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
   ylabel('Humedades','fontsize',8,'fontweight','bold','color','white');
   set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
   set(gca,'XColor','w','YColor','w');

elseif num_graf == 2

   %----- Hago las gráficas de las temperaturas de ambas pdus -----
   subplot(2,2,1)
   plot(1:length(temperaturas(:,1)),temperaturas(:,1),'Color','y','LineWidth',2);
   axis([1 max(length(temperaturas(:,1))) min(temperaturas(:,1))-0.5 max(temperaturas(:,1))+0.5]);
   grid on;
   texto=['Temperaturas en ' pdu(1,:)];
   title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
   xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
   ylabel('Temperaturas','fontsize',8,'fontweight','bold','color','white');
   set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
   set(gca,'XColor','w','YColor','w');

   subplot(2,2,2)
   plot(1:length(temperaturas(:,2)),temperaturas(:,2),'Color','y','LineWidth',2);
   axis([1 max(length(temperaturas(:,2))) min(temperaturas(:,2))-0.5 max(temperaturas(:,2))+0.5]);

   grid on;
   texto=['Temperaturas en ' pdu(2,:)];
   title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
   xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
   ylabel('Temperaturas','fontsize',8,'fontweight','bold','color','white');
   set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
   set(gca,'XColor','w','YColor','w');

   % ----- Hago las gráficas de las humedades de ambas pdus ----
   subplot(2,2,3)
   plot(1:length(humedad(:,1)),humedad(:,1),'y','Linewidth',2);
```

```
    axis([1 length(humedad(:,1)) min(humedad(:,1))-0.5 max(humedad(:,1))+0.5 ])
    grid on;
    texto=['Humedades en ' pdu(1,:)];
    title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
    xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
    ylabel('Humedades','fontsize',8,'fontweight','bold','color','white');
    set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
    set(gca,'XColor','w','YColor','w');

    subplot(2,2,4)
    plot(1:length(humedad(:,2)),humedad(:,2),'y','Linewidth',2);
    axis([1 length(humedad(:,2)) min(humedad(:,2))-0.5 max(humedad(:,2))+0.5 ])
    grid on;
    texto=['Humedades en ' pdu(2,:)];
    title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
    xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
    ylabel('Humedades','fontsize',8,'fontweight','bold','color','white');

    set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
    set(gca,'XColor','w','YColor','w');

    %-------------------------------------------------------------

elseif num_graf == 3

     %----- Hago las gráficas de las temperaturas de ambas pdus -----
    subplot(2,3,1)
    plot(1:length(temperaturas(:,1)),temperaturas(:,1),'Color','y','LineWidth',2);
    axis([1 max(length(temperaturas(:,1))) min(temperaturas(:,1))-0.5 max(temperaturas(:,1))+0.5]);
    grid on;
    texto=['Temperaturas en ' pdu(1,:)];
    title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
    xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
    ylabel('Temperaturas','fontsize',8,'fontweight','bold','color','white');
    set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
    set(gca,'XColor','w','YColor','w');

    subplot(2,3,2)

    plot(1:length(temperaturas(:,2)),temperaturas(:,2),'Color','y','LineWidth',2);
    axis([1 max(length(temperaturas(:,2))) min(temperaturas(:,2))-0.5 max(temperaturas(:,2))+0.5]);
    grid on;
    texto=['Temperaturas en ' pdu(2,:)];
    title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
    xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
    ylabel('Temperaturas','fontsize',8,'fontweight','bold','color','white');
    set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
    set(gca,'XColor','w','YColor','w');

    subplot(2,3,3)
    plot(1:length(temperaturas(:,3)),temperaturas(:,3),'Color','y','LineWidth',2);
    axis([1 max(length(temperaturas(:,3))) min(temperaturas(:,3))-0.5 max(temperaturas(:,3))+0.5]);

    grid on;
    texto=['Temperaturas en ' pdu(3,:)];
    title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
    xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
    ylabel('Temperaturas','fontsize',8,'fontweight','bold','color','white');
    set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
    set(gca,'XColor','w','YColor','w');
```

```
    % ----- Hago las gráficas de las humedades de ambas pdus ----
    subplot(2,3,4)
    plot(1:length(humedad(:,1)),humedad(:,1),'y','Linewidth',2);
    axis([1 length(humedad(:,1)) min(humedad(:,1))-0.5 max(humedad(:,1))+0.5 ])
    grid on;
    texto=['Humedades en ' pdu(1,:)];
    title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
    xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
    ylabel('Humedades','fontsize',8,'fontweight','bold','color','white');
    set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
    set(gca,'XColor','w','YColor','w');

    subplot(2,3,5)
    plot(1:length(humedad(:,2)),humedad(:,2),'y','Linewidth',2);
    axis([1 length(humedad(:,2)) min(humedad(:,2))-0.5 max(humedad(:,2))+0.5 ])
    grid on;
    texto=['Humedades en ' pdu(2,:)];
    title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
    xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
    ylabel('Humedades','fontsize',8,'fontweight','bold','color','white');
    set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
    set(gca,'XColor','w','YColor','w');

    subplot(2,3,6)
    plot(1:length(humedad(:,3)),humedad(:,3),'y','Linewidth',2);
    axis([1 length(humedad(:,3)) min(humedad(:,3))-0.5 max(humedad(:,3))+0.5 ])
    grid on;
    texto=['Humedades en ' pdu(3,:)];
    title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
    xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
    ylabel('Humedades','fontsize',8,'fontweight','bold','color','white');
    set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
    set(gca,'XColor','w','YColor','w');

    %------------------------------------------------------------
End
```

Dibuja Gráficas Sai

```
function dibuja_graficas_sai(potencias,tensiones,fechas_sai,elemento,sai)

%--- Declaración de las variables necesarias -------
tension1=tensiones(:,1);
tension2=tensiones(:,2);
fechas=datenum(fechas_sai);

potencias1=potencias(:,1);
potencias2=potencias(:,2);

len1p=length(potencias1);
len2p=length(potencias2);

%--- Definición de la 1º gráfica--------

subplot(2,2,1)

plot(fechas,tension1,'y','LineWidth',2);
datetick('x',15)
grid on;

texto=['Tension entrada fase 1 del ' sai];
title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
ylabel('Tensión (V)','fontsize',8,'fontweight','bold','color','white');
set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
set(gca,'XColor','w','YColor','w');

%set(gca,'XTick',fechas);
%datetick('x',15);

subplot(2,2,2)

plot(fechas,tension2,'y','LineWidth',2);
datetick('x',15)
%axis([1 len2t min(tension2)-5 max(tension2)+10]);
grid on;

texto=['Tension entrada fase 2 del ' sai];
title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
ylabel('Tensión (V)','fontsize',8,'fontweight','bold','color','white');
set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
set(gca,'XColor','w','YColor','w');

subplot(2,2,3)

plot(fechas,potencias1,'y','LineWidth',2);
datetick('x',15)
%axis([1 len1p min(potencias1)-5 max(potencias1)+10]);
grid on;

texto=['Potencia salida fase 1 del ' sai];
```

```
title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
ylabel('Potencia Salida (W)','fontsize',8,'fontweight','bold','color','white');
set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
set(gca,'XColor','w','YColor','w');


subplot(2,2,4)

plot(fechas,potencias2,'y','LineWidth',2);
datetick('x',15)
%axis([1 len2p min(potencias2)-5 max(potencias2)+10]);
grid on;

texto=['Potencia salida fase 2 del ' sai];
title(texto,'fontsize',9,'color',[1.0 0.69 0.392],'fontweight','bold');
xlabel('Tiempo','fontsize',8,'fontweight','bold','color','white');
ylabel('Potencia Salida (W)','fontsize',8,'fontweight','bold','color','white');
set(gca,'Color',[0 0 0]); % cambiar color fondo grafica
set(gca,'XColor','w','YColor','w');
```

Encuentra Problemas Sai

```
function[Bypass,Bateria,GrupoElectrogeno,Last_Update]=encuentra_problemas_sai(conector,user,passw
ord,elemento,fecha_inicio,fecha_final)

conn=database(conector,user,password);

elemento=[char(39) elemento char(39)]; %con char(39) introduzco comillas simples
fecha_inicio=[char(39) fecha_inicio char(39)]; % '2013-08-20' por ejemplo
fecha_final=[char(39) fecha_final char(39)];

Bypass=['select ss.modo_bypass from sai s INNER JOIN sai_seleccionado ss ON s.sai_id=ss.sai_id AND
s.sai= ' elemento ' AND ss.last_update between DATE(' fecha_inicio ') AND DATE (' fecha_final ')'];
Bateria=['select ss.modo_bateria from sai s INNER JOIN sai_seleccionado ss ON s.sai_id=ss.sai_id AND
s.sai = ' elemento ' AND ss.last_update between DATE(' fecha_inicio ') AND DATE (' fecha_final ')'];
GrupoElectrogeno=['select ss.g_electrogeno from sai s INNER JOIN sai_seleccionado ss ON
s.sai_id=ss.sai_id AND s.sai = ' elemento ' AND ss.last_update between DATE(' fecha_inicio ') AND
DATE (' fecha_final ')'];
Last_Update=['select ss.last_update from sai s INNER JOIN sai_seleccionado ss ON s.sai_id=ss.sai_id
AND s.sai = ' elemento ' AND ss.last_update between DATE(' fecha_inicio ') AND DATE (' fecha_final
')'];

Bypass=exec(conn,Bypass);
Bateria=exec(conn,Bateria);
GrupoElectrogeno=exec(conn,GrupoElectrogeno);
Last_Update=exec(conn,Last_Update);

setdbprefs({'DataReturnFormat';'NullStringRead'},{'cellarray','Ok'});

Bypass=fetch(Bypass);
Bateria=fetch(Bateria);
GrupoElectrogeno=fetch(GrupoElectrogeno);
Last_Update=fetch(Last_Update);

%para volcar datos al uitable deben ser cellarray

Bypass=(Bypass.data);
Bateria=(Bateria.data);
GrupoElectrogeno=(GrupoElectrogeno.data);
Last_Update=(Last_Update.data);

close(conn);
```

ANEXO II. ÍNDICE DE FIGURAS

ANEXO III. GLOSARIO

Acceso Biométrico. Acceso a instalaciones mediante controles de reconocimiento de humanos únicos tales como huellas dactilares, escáner de retina, patrones faciales…

Agua Nebulizada. Sistema de extinción de fuego basado en la creación de micro gotas que producen el enfriamiento del fuego, el desplazamiento del oxígeno y la atenuación del calor radiante.

AM. Sistema de transmisión analógica basado en la variación de la amplitud de la señal transmitida.

ASK. Sistema de transmisión digital basado en la variación de la amplitud de la señal transmitida.

Cable Coaxial. Cable utilizado para el transporte de señales eléctricas de alta frecuencia.

CentOS. *Community Enterprise Operative System.* Bifurcación de la distribución Linux *Red Hat Enterprise* compilado por voluntarios a partir del código fuente liberado por *Red Hat.*

Cifrado. Método informático destinado a aumentar la seguridad de un archivo mediante la codificación del contenido.

CPD. *Centro de Procesado de Datos*. Localización de los recursos necesarios para el procesamiento de la información de una empresa.

CRON Nombre del programa Linux que permite ejecutar de manera automática scripts o o comandos en un período de tiempo concreto.

C++. Lenguaje de programación multiparadigma donde permite la programación estructurada o la programación orientada a objetos.
Daemons. Proceso informático no interactivo que se ejecuta en segundo plano en equipos Linux. En Windows son llamados *servicios* o *programa residente* en MS-DOS.

Fibra Óptica. Medio de transmisión por el que se propagan pulsos de luz qu representan los datos a transmitir.

FIREWALL. Dispositivo de una red de datos destinado a bloquear accesos no autorizados.

FM. *Frecuencia Modulada*. Modulación angular analógica que permite la transmisión de información a través de una onda portadora variando su frecuencia.

FSK. *Modulación por desplazamiento de frecuencia.* Modulación digital que permite la transmisión de la información gracias al uso de dos frecuencias asociadas a cada nivel de la señal digital.

FTP. *File Transfer Protocol.* Protocolo de transferencia de archivos basado en la arquitectura cliente-servidor sobre una red TCP.
GUI. *Graphical User Interface.* . Programa informático destinado a la representación de la información de forma que el usuario puede interpretarla.

HONEYPOT. Conjunto de computadores o software destinado a la simulación de ser un sistema vulnerable con el fin de atraer atacantes para poder analizar y corregir los fallos de seguridad de nuestro sistema.

HTTP. *HyperText Transfer Protocol.* Protocolo de transferencia de información utilizado en la *World Wide Web (www).*

IMAP. *Internet Message Access Protocol.* Protocolo de aplicación destinado al acceso de mensajes almacenados en servidores de Internet.

JAVA. Lenguaje de programación orientado a objetos. Las aplicaciones son compiladas en bytecodes lo que permite la ejecución en cualquier plataforma gracias a la ejecución sobre una máquina virtual java.

MATLAB. *Matrix Laboratory*. Software informático de cálculo númerico con entorno de desarrollo integrado y lenguaje de scripts propio. Optimizado para el tratamiento de grandes cantidades de datos así como de matrices.

MIB. *Management Information Base*. Información organizada jerárquicamente para facilitar el acceso a los datos de los dispositivos.

MPLS. *Multiprotocol Label Switching*. Protocolo de cada red destiando al transporte de datos de alta velocidad y voz digital en una única conexión.

Monomodo. Tipo de fibra óptica por la que únicamente se transmite un modo de luz.

Multimodo. Tipo de fibra óptica por la que puede ser transmitida la luz por más de un camino o modo.

MRTG. Software de libre distribución cuya finalizada es realizar consultas a los agentes SNMP con el fin de recoger los datos necesarios.

MySQL. Sistema de gestión de base de datos multiplataforma, multihilo y multiusuario.

ORACLE.. Sistema de gestión de base de datos multiplataforma, multihilo y multiusuario.

PDU. *Power Distribution Unit.* Dispositivo destinado al suministro eléctrico de los diferentes equipos conectados a él.

PHP. Lenguaje de programación utilizado en el lado del servidor destinado para el desarrollo web dinámico.

PM. Modulación angular analógica basada en la variación de la fase de la señal portadora en función de la señal modulante.

POP3. Protocolo de red destinado a obtener los mensajes de correo ubicados en servidores remotos.

PSK. *Modulación por desplazamiento de fase.* Modulación angular digital basada en la variación de la fase de la señal portadora entre un número de valores discretos.

RACK.. Soporte metálico utilizado para alojar en su interior todos los equipos electrónicos, informáticos y de telecomunicaciones ubicados en un CPD.

RAM. *Random Access memory* Memoria utilizada para cargar aquellas instrucciones que debe ejecutar el procesador y otras unidades de cómputo.

OUTER. Dispositivo destinado a encaminar los datos a través de las redes correctas. Trabaja sobre la capa 3 del modelo OSI.

SAI. *Sistema de Alimentación Ininterrumpida*. Dispositivo dotado de baterías capaz de suministrar energía eléctrica en caso de fallo de la fuente principal.

SMTP. Protocolo de capa de red basado en el intercambio de mensajes entre los múltiples dispositivos capaces de mandar / recibir correos electrónicos.

SNMP. *Simple Network Management Protocol.* Protocolo de capa de aplicación destinado a facilitar el intercambio de información entre el administrador y los agentes de la red.

SSH. Protocolo de acceso a máquinas remotas. Permite tomar el control completo de la computadora. La información a través de este protocolo viaja cifrada en comparación al protocolo Telnet.

SWITCH. Dispositivo lógico encargado de la interconexión de redes de computadoras.

TCP. *Transmission Control Protocol.* Protocolo de la capa de transporte encargado de garantizar la entrega de los datos en su destino sin errores y en el mismo orden de la transmisión.

TELNET. Protocolo de red destinado al acceso remoto de máquinas. No recomendable su uso ya que la información viaja por la red como texto plano frente a la encriptación de SSH.

UDP. Protocolo de nivel de transporte basado en el intercambio de datagramas. No es necesario haber establecido una conexión previa ya que los datagramas llevan la información encapsulada incluyendo el destino.

Virtualización. Proceso por el cual, mediante software concreto, se crea una versión virtual de los recursos de nuestra red. Dentro de dichos recursos se puede localizar el sistema operativo, el hardware de la máquina y los recursos de red.

VPN. *Virtual Private Network.* Tecnología de red basada en la creación de una conexión punto a punto virtual con el fin de realizar una comunicación segura sobre una red pública. Permite crear una extensión de la redes locales sobre redes públicas.

ANEXO IV. BIBLIOGRAFÍA

Referencias bibliográficas

- Dubois,Paul. *La Biblia de MySQL*. 3º Ed. Madrid: Anaya Multimedia, 2009.
- Barragán Guerrero, Diego. *Manual de Interfaz Gráfica de Usuario en Matlab.*
- Mathworks. *Creating Graphical User Interfaces.* R2013a.
- Mathworks. *Programming Fundamentals with Matlab*. R2013a.
- Tobi Oetiker's MRTG - The Multi Router Traffic Grapher, Acceso 26 de Abril de 2013 en: http://oss.oetiker.ch/mrtg/
- EATON Powering Businees Worlwide, Acceso el 30 de Abril de 2013 en: http://powerquality.eaton.com/support/software-drivers/downloads/epdu-firmware.asp
- Ares Gomes, Jose Enrique. *Climatización en los Centros de Procesado de Datos.* Acceso el 15 de Abril de 2013 en: http://www.rediris.es/difusion/publicaciones/boletin/76/enfoque2.pdf
- Cliatec, Acceso el 15 de Abril de 2013 en: http://www.cliatec.com/soluciones/climatizacion-de-cpd.php
- Socored, Acceso el 15 de Abril de 2013 en: http://www.socored.es/data-center-climatizacion.php
- Blog.aodbc.es, Acceso el 20 de Abril de 2013 en: http://blog.aodbc.es/2012/07/10/clasificacion-tier-en-el-datacenter-el-estandar-ansitia-942/
- UpTimeInstitute, Acceso múltiple en : http://uptimeinstitute.com/
- Blogs.salleurl.edu, *Seguridad y Auditoría en un CPD.* Acceso el 20 de Abril de 2013 en: http://blogs.salleurl.edu/datacenter-cpds-new-generation/2013/04/29/seguridad-y-auditoria-en-un-cpd/