Facultad de Ciencias

Desarrollo de una aplicación para el descubrimiento y supervisión de procesos de IDboxRT

(Development of an application for the discovery and monitoring of IDboxRT processes)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Emilio Blanco Villegas

Director: Rafael Duque Medina

Co-Director: Ángel Tezanos Ibáñez

Diciembre 2017

Índice

Índice de Figuras 3
Índice de tablas 4
Agradecimientos 5
Resumen 6
Abstract 7
1. Introducción 8
1.1. Definición de un protocolo de descubrimiento. 8
1.2. SCADA 9
1.3. IDbox 11
1.3.1. Arquitectura de IDbox 12
1.3.2. Los agentes de IDbox 13
1.4. Motivación 16
1.5. Objetivos 16
1.6. Estructura del documento 17
2. Herramientas y Metodología 18
2.1. Herramientas 18
2.2. Tecnologías 19
2.3. Metodología de desarrollo 20
3. Desarrollo 22
3.1. Captura y análisis de requisitos 22
3.2. Fase Previa de desarrollo: Investigación 23
3.3. Casos de uso 25
3.4. Diagrama de Clases 27
3.5. Diagramas de secuencia. 29
3.6. Interfaz de Usuario 31
4. Pruebas unitarias y de integración 34
5. Conclusiones y líneas futuras 36
6. Bibliografía 37

Índice de Figuras

Figura 1. Esquema de un servicio de descubrimiento 8
Figura 2. Estructura general de un SCADA [2] 9
Figura 3. Esquema básico de funcionamiento de IDbox 11
Figura 4. Arquitectura de IDbox [4] 12
Figura 5. Estructura del DataRecorder de IDbox [4] 13
Figura 6. Estructura del DataAgent de IDbox [4] 14
Figura 7. Estructura del DataNotification de IDbox [4] 15
Figura 8. Esquema de un modelo incremental iterativo [13] 20
Figura 9. Desarrollo del modelo incremental iterativo 21
Figura 10. Reutilización de puertos 24
Figura 11. Diagrama de casos de uso 25
Figura 12. Diagrama de Clases 28
Figura 13. Diagrama de Secuencia: Petición de descubrimiento 30
Figura 14. Diagrama de secuencia: Respuesta a un descubrimiento 30
Figura 15. Interfaz de usuario 31
Figura 16. Ejemplo con tres instancias en el mismo equipo 32
Figura 17. Ejemplo de interfaz con dos instancias en otro equipo y otras dos instancias en el equipo local 33
Figura 18. Pruebas fallidas 34
Figura 19. Pruebas satisfactorias 35

Índice de tablas

Tabla 1. Requisitos funcionales 22
Tabla 2. Requisitos no funcionales 23
Tabla 3. Caso de uso: Iniciar la aplicación 25
Tabla 4. Caso de uso: Descubrir Servicios 26
Tabla 5. Caso de uso: Detener escucha de Servicios 27

Agradecimientos

Con estas pocas líneas me gustaría expresar mi gratitud a todas las personas que durante el grado han aportado su granito de arena para hacerme evolucionar como persona y como informático.

Agradecer a mi familia y amigos el apoyo y el seguimiento del proyecto que han hecho durante estos meses y por aguantar mis enfados cuando algo no me salía.

Finalmente me gustaría dar las gracias a mi tutor Rafael Duque por prestarme su ayuda para estructura y revisar la memoria. Además, también, a Ángel Tezanos, que me ha sufrido durante todo el tiempo con mis preguntas a todas horas y mis inquietudes en el proyecto. Y al equipo de IDbox y CIC Consulting Informático de Cantabria por darme la oportunidad de trabajar en el proyecto y poner la confianza en mí para poder llevarle a cabo.

Resumen

Las tecnologías del mundo actual cada vez requieren de un control remoto más exhaustivo. El software de terceros que se utiliza en las diferentes empresas va evolucionando al control remoto en vez de que los encargados del propio proyecto vayan al lugar de instalación y una vez instalado, requiera de su supervisión. Asimismo, se requiere de un control continuo del estado de los dispositivos que tengan servicios en ejecución, o de sistemas en tiempo real.

Este proyecto tiene como objetivo la creación de una herramienta que permita saber en todo momento el estado de la instalación, de forma que no haga falta realizar el seguimiento de varias máquinas en las que puede estar instalado alguno de los nodos que componen IDbox, de forma que desde un solo lugar se pueda conocer el estado de cada máquina. En concreto se desarrollará una aplicación de escritorio que en todo momento dará información del estado de los nodos que estén instalados en las diferentes máquinas dentro de un mismo proyecto.

Palabras clave: SCADA, protocolo de descubrimiento, IDbox.

Abstract

Nowadays, technologies require from a more exhaustive remote control. Third-party software that is being used at the companies is evolving to a remote control, instead of being the managers of the Project itself the ones that must go to the place where the Company is located and install it there. Once it is installed, supervision will be required. Likewise, a continuous control of the devices' state that have services in execution or real-time systems, is essential.

This Project's objective is the creation of a tool that allows to know every time the installation status, so that, it will not be necessary to monitor many machines where any of the nodes that makes up IDbox may be installed. With this tool it will be required just one place where the staff can verify the status of each machine.

Specifically, a desktop application will be developed that will know the status of each node of IDbox every time, always in the same project.

Key words: SCADA, Discovery Protocol, IDbox.

1. Introducción

Para comenzar, deberemos tener unos conceptos básicos sobre lo que se va a tratar el proyecto. Por lo que conocemos del título del proyecto, se va a desarrollar un servicio de descubrimiento para IDbox. Para la realización de un servicio de descubrimiento, se ha tenido que implementar un protocolo de descubrimiento, por tanto, se indagarán en los conceptos de Protocolo de descubrimiento y sobre los componentes de IDbox a los que afecta. [1]

1.1. Definición de un protocolo de descubrimiento.

Un protocolo de descubrimiento es una herramienta que recopila información de dispositivos conectados a una misma red. Esto permite encontrar dispositivos en nuestro alcance de forma que podamos comunicarnos con ellos e intercambiar información. Este descubrimiento se puede hacer de forma "Activa", es decir, a petición de un usuario, que quiera realizar el descubrimiento en un momento dado, o de forma "pasiva", es decir, que periódicamente se estén realizando peticiones de descubrimiento por parte de los equipos.

Es similar a un medidor de audiencia de televisión. La cadena emite a todos los puntos de las antenas, y éstos las reciben. Si lo recibe mientras el usuario estaba viendo ese canal, la televisión responde a la cadena con una confirmación.

En la Figura 1 podemos observar cómo sería una representación de un equipo haciendo un descubrimiento de otros dos. Estos equipos no necesariamente tienen que responder (análogo a que una persona no tiene por qué estar viendo el canal de televisión).

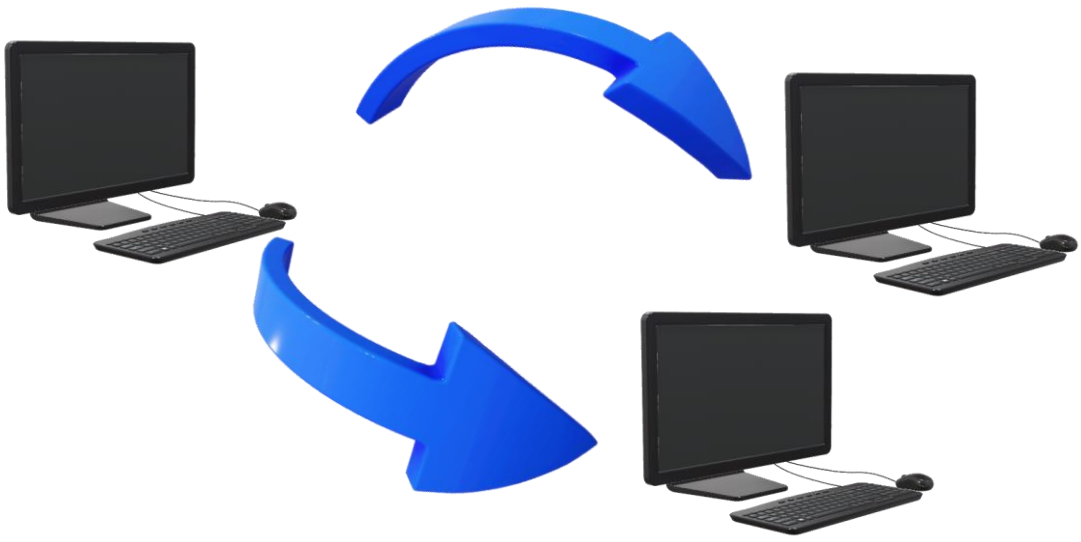

Figura 1. Esquema de un servicio de descubrimiento

1.2. SCADA

La palabra SCADA viene del acrónimo en inglés Supervisory Control And Data Acquisition o, traducido al castellano, Supervisión, Control y Adquisición de Datos. Se define a un Software de tipo SCADA como aquel sistema que recoja datos de otro sistema con el fin de controlar y optimizar ese sistema. El ejemplo más claro de uso de este tipo de software es en las centrales nucleares, donde se tiene que monitorizar muchos valores en tiempo real, y que, dependiendo de los datos recogidos por el software, tanto el propio software o un operario de la central, pueda actuar de acuerdo con la información mostrada. [2]

Últimamente se está utilizando mucho estos SCADAS para lo que se denomina Smart City, recogiendo datos alrededor de las ciudades, por ejemplo, para el control de aguas.

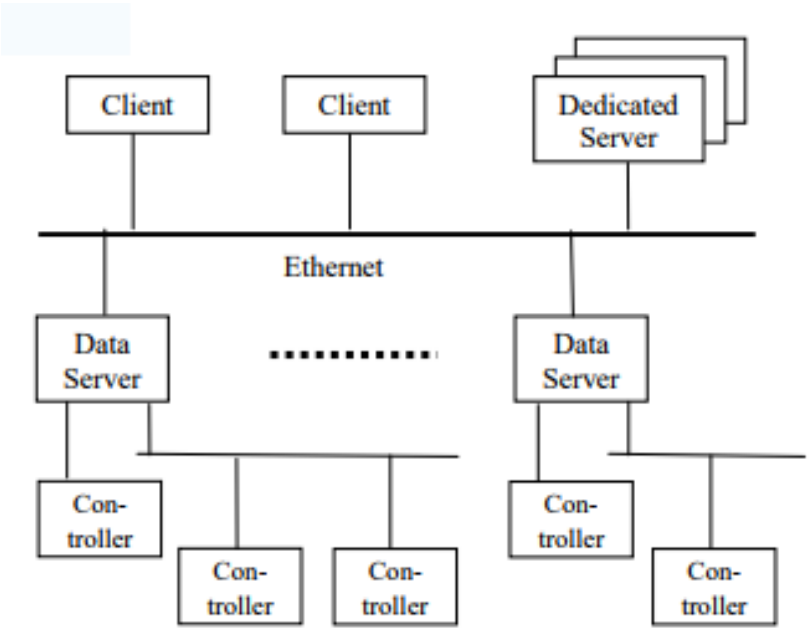

Figura 2. Estructura general de un SCADA [2]

En la Figura 2 podemos observar la estructura básica de un SCADA. Por un lado, tenemos un servidor dedicado, que será el que almacene la información recogida por los servidores de datos y se comunicará por Ethernet con éstos y con clientes que serán los que se dediquen a mostrar esa información.

Los controladores de la Figura 2 serán los sensores que detectan y recogen los datos que sean necesarios para su estudio. Estos por ejemplo pueden ser los niveles de temperatura de una central nuclear, o simplemente la temperatura que se recoge en cierto punto. Estos controladores son elementos hardware que tienen la capacidad de esa recogida de datos.

Los servidores de datos son los que tienen cierto control sobre estos controladores. Estos servidores pueden ser controladores lógicos programables (comúnmente conocidos como PLCs) que se utilizan para automatizar procesos electromecánicos como en cadenas de montaje, Ordenadores y todo tipo de estructuras habilitadas para ello. Una vez los datos pasan el control que requieran, se conectarán con el servidor dedicado para que almacene los datos recogidos.

Este servidor dedicado, debe ser capaz de guardar una cantidad de información muy grande, ya que el volumen de datos recogidos puede ser gigante. Estos datos almacenados deben estar disponibles para los clientes.

Estos clientes son los encargados de plasmar la información recopilada en el servidor dedicado, para que se pueda manipular y representar a gusto del usuario.

Estos sistemas automatizados están teniendo un impacto cada vez más importante en las empresas, dedicando mucho tiempo y dinero a la adquisición de estos datos, así como del software encargado de almacenar y representar estos datos. La importancia de los SCADAs en las compañías dedicadas al sector energético está creciendo a paso de gigante y las necesidades cada vez aumentan más.

1.3. IDbox.

IDbox es un sistema capaz de recoger, integrar, procesas y visualizar grandes volúmenes de información procedente de un conjunto de fuentes de naturaleza heterogénea (PLCs, dataloggers, sensores, historiales, computadores de proceso, ERPs, ESBs, ficheros, etc.) desarrollado por la empresa CIC Consulting Informático de Cantabria. Es decir, es un sistema SCADA.

IDbox integra toda la información necesaria para la operación y supervisión y la pone a disposición de todas las personas de la organización y sistemas externos, permitiendo realizar un análisis efectivo, tanto en situaciones normales de operación como ante disparos u otras alteraciones en la producción. [3]

El objetivo, en última instancia, es permitir mediante herramientas analíticas y gráficas la realización de cálculos y análisis de información (en tiempo real o histórico) y, de este modo, favorecer la creación de un escenario de ayuda, tanto en la toma de decisiones relativas a los procesos como en la propia operación.

IDbox está presente en muchas empresas del sector energético en España, y ampliándose poco a poco al resto de Europa. El sistema permite realizar cuadros de mando con los datos recogidos para poder analizar y operar conforme varían los datos. De esta forma, el operario será capaz de conocer en todo momento el estado del sistema. En la Figura 3 se puede observar cuál es este funcionamiento básico que se ha descrito de forma más esquemática y práctica.

Figura 3. Esquema básico de funcionamiento de IDbox

1.3.1. Arquitectura de IDbox

IDbox tiene una arquitectura bastante modular, dividida en varios bloques, que veremos en la siguiente Figura:

Figura 4. Arquitectura de IDbox [4]

El proyecto se centrará en el bloque "Agentes", que serán los que debemos localizar en la red. Aunque esto sea válido para cualquier equipo, lo que realmente interesa en IDbox son estos agentes.

Para entender la Figura 4, los conectores se comunican con los orígenes de información y se encargan de la adquisición de datos, la cual se plasma en señales. Por otro lado, tenemos los agentes, que distribuyen entre los servicios de almacenamiento y el motor de cálculo los datos recogidos por los conectores. Los servicios de almacenamiento se encargan de persistir los datos y finalmente, los servicios de análisis permiten la visualización de los datos en históricos o en tiempo real.

Para poder llegar a entender a qué nos enfrentamos, expondremos los diferentes agentes que componen IDbox y su funcionalidad dentro de éste.

1.3.2. Los agentes de IDbox

Como se ha propuesto en la sección anterior, se describirán a continuación los diferentes agentes que componen IDbox. Estos agentes actúan de forma crucial en el sistema, siendo la base de su correcto funcionamiento. Cada agente es independiente del resto, y juntos aglomeran las características más importantes que definen IDbox. [3]

- **Agente de Grabación** o DataRecorder

  El agente de grabación es el encargado de persistir los datos que provienen de los conectores. De esta forma, IDbox se vuelve independiente del origen de la información.

  Tiene la capacidad de recuperarse ante fallos, guardando datos en memoria para después poder persistirles una vez se haya recuperado.

  Posee receptores UDP que permite recibir los datos, así como la capacidad de leer ficheros. Con los datos recibidos, realiza un control de datos duplicados y procede a enviarlos a base de datos y en disco con colas circulares.
  En la Figura 5 podemos observar un esquema de la estructura de un DataRecorder de IDbox.

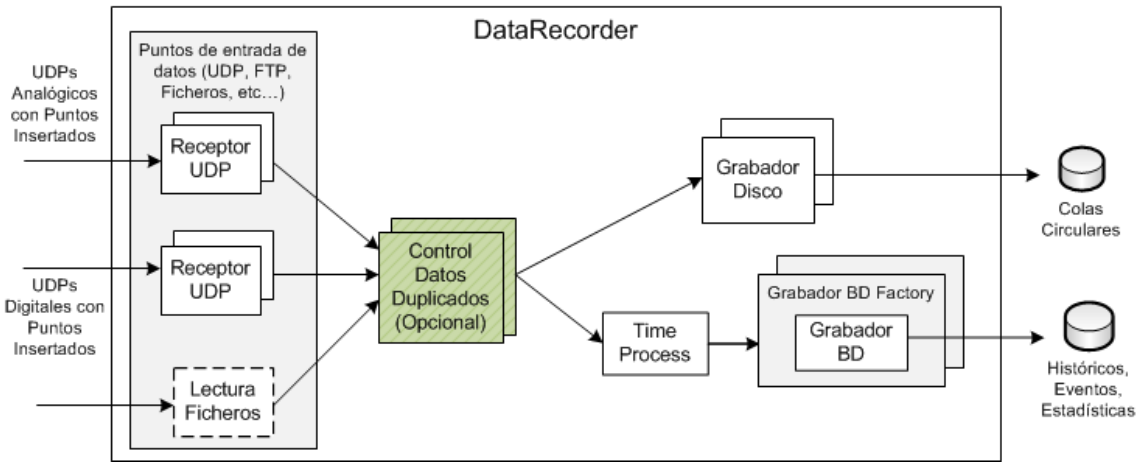

Figura 5. Estructura del DataRecorder de IDbox [4]

- **Agente de Consulta** o DataAgent

  Uno de los principales objetivos del DataAgent son centralizar el suministro de información, de forma que sea cual sea la plataforma desde la que se pidan los datos, todas esas consultas se realicen en este servicio. Cuenta con una cadena de proveedores, de forma que siempre se realice de la forma más rápida. Hoy en día, la cadena de proveedores la podemos ver con la Figura 6.

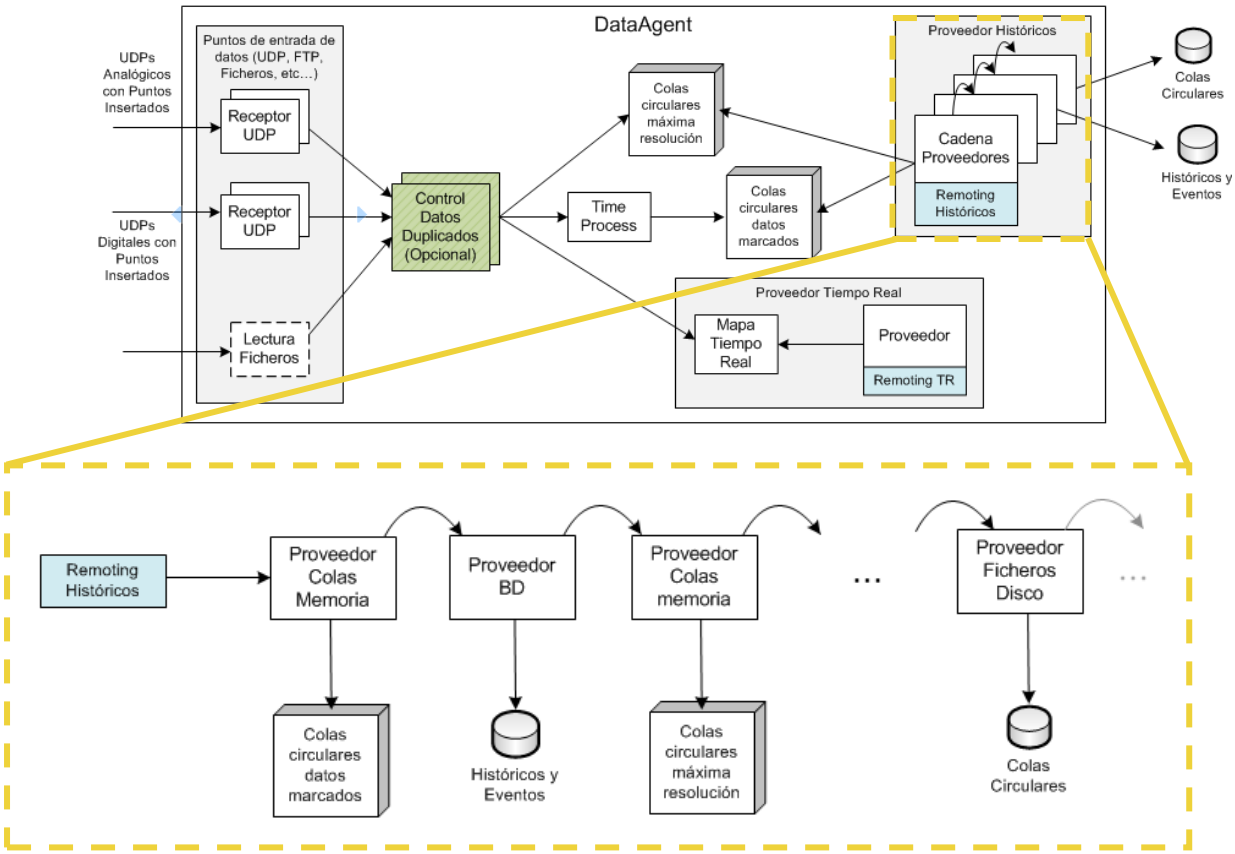

Figura 6. Estructura del DataAgent de IDbox [4]

- **Agente de notificaciones o DataNotification**

  El agente de notificaciones es el encargado de controlar las alarmas de las señales. Las alarmas se definen dentro de un motor de reglas (o RuleManager), que emitirán un aviso cuando las reglas se cumplan. Por ejemplo, se puede configurar reglas para que se disparen si una señal alcanza cierto valor, como la temperatura de una pieza.

  Estas alarmas además pueden ir sujetas a notificaciones, que se envían cuando la alarma se dispara. Siguiendo con el ejemplo anterior, una vez que la alarma salte, se puede configurar que nos llegue un correo, o una notificación push al móvil, un SMS o cualquier sistema de notificaciones que haya disponible. Con la Figura 7 se puede ver la estructura interna del componente.

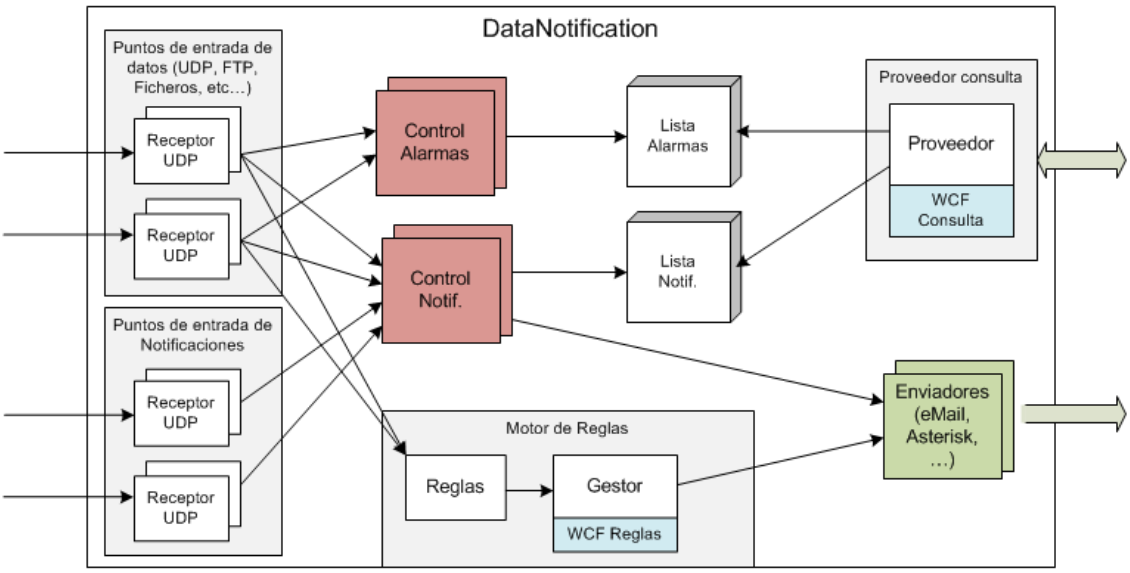

Figura 7. Estructura del DataNotification de IDbox [4]

- **Agente de Operación** o DataOperation

  El agente de operación se encarga de hacer el envío de escrituras en los PLC. De esta forma, permite gestionar el control de las señales de operación, dando una respuesta ante fallos.

  Una vez conocemos los agentes que forman IDbox, sabemos qué debemos localizar en nuestra red desde el software a desarrollar, y que características nos pueden llegar a proporcionar cada uno de ellos.

1.4. Motivación

Los sistemas SCADA hoy en día, como ya se ha descrito en secciones anteriores, están tomando una importancia y relevancia grande a lo largo de los últimos años, y cada vez más. Es por eso por lo que era necesario una herramienta que se encargara de monitorizar en tiempo real el estado de las máquinas en las que se encuentran los componentes encargados de almacenar, recoger y visualizar los datos.

Desde IDbox surge la necesidad de conocer el estado de las máquinas en las que esté el sistema en todo momento, por lo que emerge este proyecto, satisfaciendo así esa necesidad de monitorizar el estado de máquinas (no sólo los agentes descritos) de los clientes en los que esté instalado IDbox.

1.5. Objetivos

El objetivo de este proyecto fin de grado es la implementación de una solución software que permita la monitorización del estado de los agentes y máquinas que componen IDbox.

Se dividen estos objetivos en:

- **Análisis de los agentes de IDbox:** Para poder conocer qué datos nos pueden ser beneficiosos y que nos faciliten información de cada agente, debemos estudiar sus posibilidades. Esto también se puede incluir en cada máquina ajena a los agentes que puede requerir una monitorización también de su estado.
- **Propuesta de diseño y arquitectura de la aplicación de escritorio:** Para que sea más fácil el seguimiento de esta información que, a priori, puede parecer angustioso, se propone el diseño e implementación de una aplicación de escritorio minimizable que permita conocer el estado de monitorización en todo momento de las máquinas que se deseen.
- **Estudio e investigación de las tecnologías disponibles para realizar el proyecto:** Si se desea realizar un protocolo de descubrimiento, debemos conocer las tecnologías que hay actualmente en el entorno software que cumplan con los requisitos de IDbox y que permita el desarrollo del servicio.
- **Desarrollo de un protocolo de descubrimiento:** Como se ha explicado en el apartado introductorio, se pide el desarrollo de un protocolo de descubrimiento que permita conocer la localización de las máquinas en las que el software esté instalado.

- **Desarrollo de la aplicación de escritorio minimizable:** Una vez el protocolo está funcionando, hay que desarrollar la aplicación de escritorio que permita este seguimiento.
- **Realización de pruebas:** Como colofón, todo el desarrollo realizado puede tener errores o *bugs*. Eso es inevitable en cualquier software, por lo que para minimizar su impacto y mejorar la posible detección de *bugs* se realizan diferentes pruebas para comprobar el funcionamiento.

1.6. Estructura del documento

El objetivo de este documento es explicar detalladamente el proyecto, así como su planteamiento y desarrollo. Para ello, el documento se compone de 5 capítulos, estructurados de la siguiente forma:

- Capítulo 1: Introducción a los conceptos básicos. Se han explicado qué son los sistemas SCADA, un protocolo de descubrimiento, la plataforma IDbox y así como la motivación y objetivos del proyecto.
- Capítulo 2: Herramientas. Se expondrán las herramientas utilizadas para el desarrollo y despliegue del proyecto.
- Capítulo 3: Desarrollo. Se explicará con detalle el proceso de desarrollo del software presentado. En los subapartados del mismo, se presentarán los requisitos y diagramas que definen las funcionalidades del trabajo. Por último, también se expondrá las pruebas satisfactorias y las fallidas.
- Capítulo 4: Pruebas unitarias y de integración. En este capítulo se recoge el testing unitario y de integración realizado a la aplicación y sus resultados.
- Capítulo 5: Conclusiones y Líneas futuras. Se recogen las conclusiones tras la finalización del proyecto y algunas ideas de lo que puede extenderse el proyecto a un futuro desarrollo y expansión del mismo.
- Capítulo 6: Bibliografía. En este anexo se detallarán todas las fuentes consultadas en el documento.

2. Herramientas y Metodología

En este apartado se detallan las herramientas, tecnologías y metodologías con las que ha sido posible realizar el desarrollo del software.

2.1. Herramientas

Para comenzar a entender el desarrollo de este proyecto, se deben conocer las herramientas utilizadas para su finalización. Estas herramientas han sido indispensables para la realización del trabajo y han sido estudiadas previamente para conocer las capacidades de cada una. Las herramientas utilizadas durante las fases del proyecto son las siguientes:

- **Visual Studio 2015:** El entorno de desarrollo integrado (IDE) de Microsoft nos da el framework .Net que nos permitirá programar el protocolo, así como la interfaz de usuario con las tecnologías que se describirán en el siguiente apartado. [5]

- **Wireshark:** Herramienta de análisis de la red que permite capturar e interactuar con el tráfico de la red de un computador. La interfaz de usuario facilita la visualización de tramas por la red, describiendo contenido y direcciones de origen y destino. [6]

- **MagicDraw:** Herramienta CASE para realización de modelos del estándar UML. Se utiliza para aumentar la productividad facilitando el diseño de sistemas para organizarlos en diagramas UML. En este caso, se ha utilizado para diseñar los diagramas de casos de uso, secuencia y de clases. [7]

2.2. Tecnologías

En este apartado se describirán las tecnologías utilizadas. La mayoría de ellas están integradas con las herramientas previamente descritas, pero es necesario hacer una introducción de cada una para conocerlas un poco más.

- **Ethernet:** Es la tecnología que se utiliza más en redes de áreas locales (LAN) y redes de área extensa (WAN). Comúnmente se asocia esta tecnología a los cables de red que se tienen en casa que conectan el router con el dispositivo que requiera de una conexión.

  Esta tecnología nos permite realizar envíos de mensajes entre dispositivos mientras estén asociados a una misma red de Ethernet. Por ejemplo, dos ordenadores conectados a un router estarán en la misma red, permitiendo enviarse mensajes entre ellos. Esta tecnología es el pilar base del proyecto, que nos permitirá enviar tramas por la red para poder saber qué equipos están conectados a la misma red que nosotros. [8]

- **C#:** Microsoft nos brinda con un lenguaje de programación orientado a objetos. Este lenguaje deriva de C y C++ y tiene una sintaxis similar a la de Java. C# forma parte del Framework de Microsoft .Net que engloba una gran cantidad de lenguajes de programación, como se ha descrito en la sección de herramientas donde se describía Visual Studio. Con este lenguaje se ha escrito el protocolo. [9]

- **Windows Forms:** Esta tecnología, también incluida en Visual Studio, permite realizar de forma muy sencilla aplicaciones de escritorio ejecutables en Windows. Con ella, se ha desarrollado una interfaz de usuario simple e intuitiva.

2.3. Metodología de desarrollo

Una de las bases de desarrollo de aplicaciones es sin duda, una buena planificación y cómo se va a estructurar dentro de esa planificación.

Las metodologías permiten a los desarrolladores estructurar su trabajo de forma que sea rápida su acción. En este caso, la metodología aplicada para el proyecto ha sido la metodología incremental iterativa.

El modelo incremental se basa en un procedimiento lineal produciendo "incrementos" de software susceptibles a entregarse. Es decir, el proceso de desarrollo del software se basa en entregas, de más básicas a más completas del software hasta encontrar la solución final. Con cada entrega, se incrementa la funcionalidad de la entrega anterior, y se repite el proceso. Como se puede observar en la Figura 8, en cada entrega se realiza un incremento. [10]

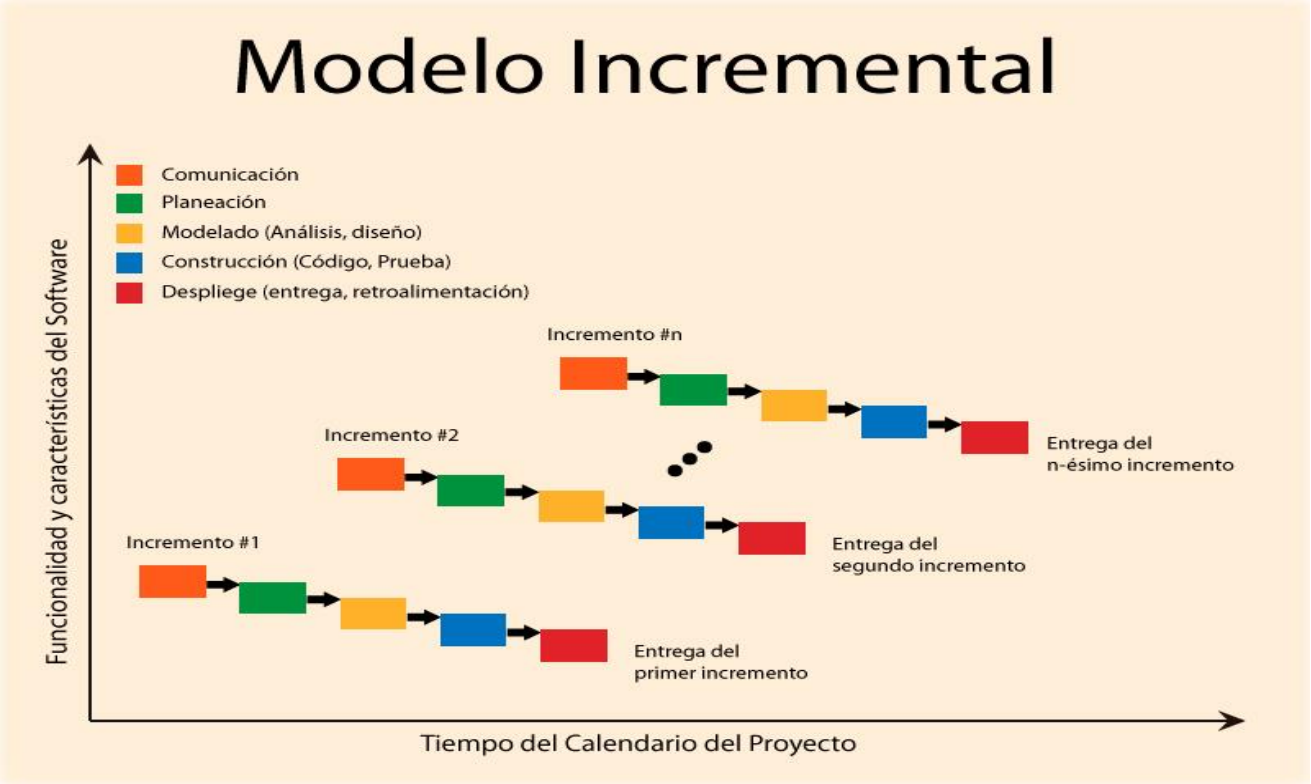

Figura 8. Esquema de un modelo incremental iterativo [13]

En nuestro caso, hemos hecho 3 iteraciones. Estas iteraciones vienen representadas en la siguiente Figura:

Figura 9. Desarrollo del modelo incremental iterativo

Con la primera iteración se ha desarrollado un protocolo de descubrimiento que busca y localiza en la red local en la que se esté ejecutando, otras instancias del software desarrollado. Por tanto, tendremos nuestra aplicación ejecutándose en cada agente descrito en el apartado de los agentes.

En la segunda iteración, se ha hecho un análisis de las opciones que nos puede devolver cada agente. Como mínimo, deberán proporcionar su IP y puerto y el nombre del agente. Dentro de cada uno, es libre a implementación de cada uno, las características que puede llegar a proporcionar, como el estado de la CPU, del Disco Duro, de la Memoria RAM, etc.

Por último, en la última iteración se ha desarrollado una aplicación de escritorio minimizable que se minimiza en la bandeja de herramientas y se está ejecutando continuamente. Se puede conocer a través de ella el estado de los equipos descubiertos.

3. Desarrollo

En este capítulo recogeremos el proceso de desarrollo del proyecto. Éste se dividirá entre la propuesta de proyecto, la captura y análisis de requisitos, diagramas de casos de uso y, finalmente, los diagramas de clase y secuencia.

3.1. Captura y análisis de requisitos

A continuación, contemplaremos las diferentes tablas de requisitos que se han conseguido recopilar analizando las necesidades del sistema.

En primer lugar, numeraremos los requisitos funcionales, que definen el correcto comportamiento del sistema. Se especifica cómo el sistema debe interaccionar a las órdenes del usuario y cómo debe ir actualizando su estado conforme estas acciones.

Tabla 1. Requisitos funcionales

Referencia	Descripción
RF00	El protocolo debe permitir comprobar el número de elementos en ejecución
RF01	El protocolo debe permitir el envío de información extra extensible
RF02	El protocolo debe soportar trabajar en grupos diferentes sobre la misma red física
RF03	El sistema debe discernir el descubrimiento de sí mismo de otros elementos
RF04	Cada elemento del sistema debe responder a peticiones de descubrimiento
RF05	El sistema debe estar operativo constantemente
RF06	El puerto de solicitudes debe ser compartido por todos los elementos
RF07	Varios elementos deben poder convivir en el mismo puerto de comunicación

Por otro lado, tendremos los requisitos no funcionales, que describen las propiedades del sistema.

Tabla 2. Requisitos no funcionales

Referencia	Descripción
RNF00	Simplicidad de uso y extensibilidad
RNF01	Bajo consumo de recursos (CPU, Memoria y Red)
RNF02	El procesado de una solicitud no debe bloquear o impedir la gestión de otra en cada elemento.

3.2. Fase Previa de desarrollo: Investigación

La parte más tediosa del proyecto proviene de un tiempo de investigación. Se ha tenido que estudiar con detalle la arquitectura interna de IDbox, reconociendo sus agentes y necesidades para poder satisfacer los requisitos que se expondrán en el siguiente apartado.

Uno de los pilares fundamentales para IDbox es la rapidez de respuesta, así como la disponibilidad de los recursos, ya que se requiere el uso de Big Data junto con gráficas e imágenes de análisis.

Sabiendo esto, se debe realizar un pequeño trabajo de estudio sobre qué se puede ofrecer para garantizar esta disponibilidad para que un usuario pueda realizar un descubrimiento por la red en cualquier momento, incluso cuando el propio programa puede estar realizando uno ya.

Uno de los requisitos primeros que se realizaron, era que todo el sistema debía usar el mismo puerto.

Por norma general, si un proceso está utilizando un puerto en la red, éste se bloquea para que nadie pueda utilizar ese puerto, entonces esto dificultaría realizar una trama de Broadcast para comprobar qué servicios hay levantados.

El Broadcast es una forma de transferir mensajes con información a todas las IPs de una red a la vez. Teniendo en cuenta la afirmación anterior de los puertos, si nuestra aplicación estuviera en ejecución y no tuviéramos activo esta reutilización de puerto, deberíamos tener una instancia en puertos diferentes, y esto no es abordable para IDbox.

Tras varios días de investigación por la red, encontré una librería llamado YABE de las siglas Yet Another BacNet.

YABE es un protocolo de comunicación de datos para crear y controlar redes. Un protocolo de comunicación de datos es un conjunto de reglas que intercambian los datos con otro nodo de la red. [11]

En esta librería se utiliza un protocolo UDP para realizar un descubrimiento en la red y que, además, utilizan el mismo puerto siempre.

Los sockets son un concepto abstracto que representa un nodo en una red, por el que se puede enviar y recibir paquetes de la red. Es la representación de un cable de red Ethernet, que interconecta dos nodos en una red (como el router y un pc con interfaz de red).

Con esa definición en la mano, en .Net hay una clase que se llama Socket, y otra que se llama UdpClient. Estas representan un socket y un nodo con socket respectivamente.

Una vez se ha revisado cómo implementa YABE el protocolo, descubrimos que hay una propiedad de los sockets en .Net que se llama ReuseAddress. Esta propiedad es un booleano, por lo que le ponemos a true. Además, en UdpClient tiene otra que se llama ExclusiveAddressUse que también es un booleano y entonces podremos utilizar el mismo puerto una y otra vez.

```
public void Start()
{
    Client.ExclusiveAddressUse = false;
    Client.Client.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReuseAddress, true);
}
```

Figura 10. Reutilización de puertos

En la Figura 10 podemos ver una sección de nuestro código, donde Client es el UdpClient, cuya propiedad Client es un socket, como se ha descrito antes. Al socket le ponemos la opción ReuseAddress a true.

3.3. Casos de uso

A partir de esta captura de requisitos, comprobamos que debe haber una persona encargada de iniciar la aplicación, y que esa persona pueda consultar el estado de los dispositivos instalados a petición. A este usuario le llamaremos usuario Administrador, y será el encargado de ejecutar los casos de uso que veremos en la figura 11, así como en las tablas 3, 4 y 5.

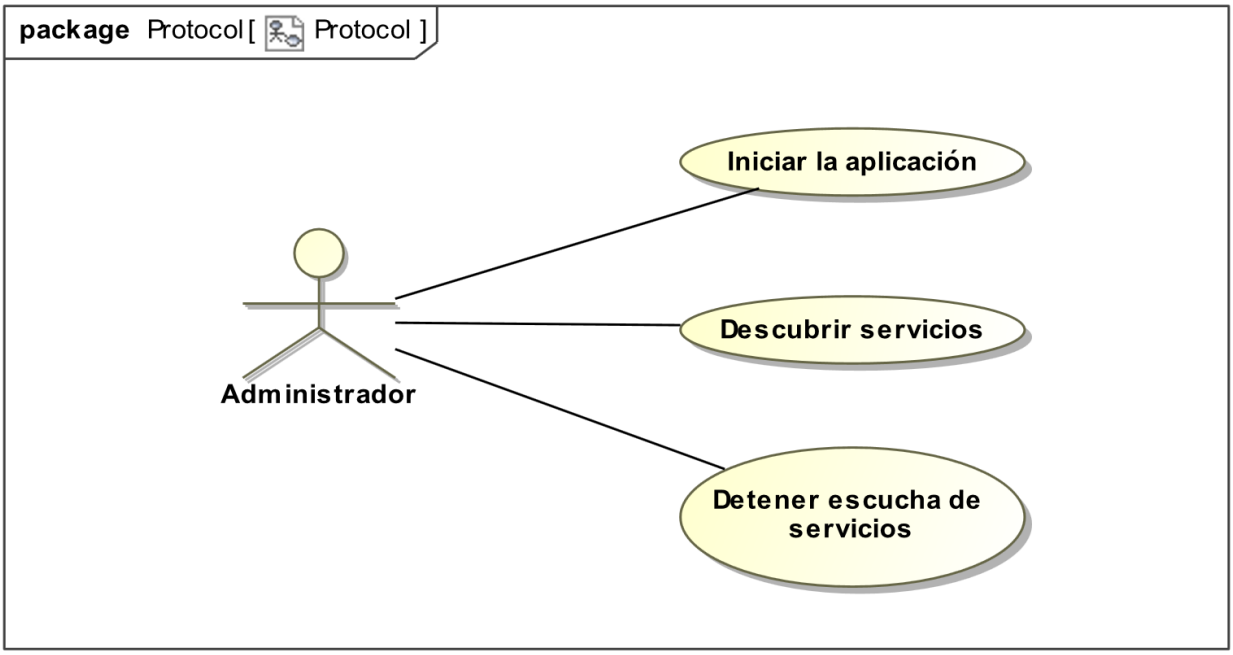

Figura 11. Diagrama de casos de uso

Para entender el entorno de cada caso de uso, se va a explicar cada uno con detalles:

Tabla 3. Caso de uso: Iniciar la aplicación

Caso de uso: Iniciar la aplicación
Precondiciones: • La aplicación se encuentra instalada en el dispositivo • Se dispone de una red local
Postcondiciones: • Aparece la interfaz de usuario con la consola de descubrimiento
Escenario principal de éxito 1. El administrador inicia la aplicación 2. La aplicación inicia el proceso de escucha en la red 3. El administrador accede a la aplicación

Extensiones **Punto 2 del escenario principal:** • El proceso ha generado un error • La aplicación va al punto 3 del escenario principal • En la consola aparecerá que ha habido un error • El administrador deberá mirar el log de la aplicación para saber más sobre el error.

Tabla 4. Caso de uso: Descubrir Servicios

Caso de uso: Descubrir Servicios
Precondiciones: • Se dispone de una red local • La aplicación se encuentra en ejecución • La aplicación se encuentra en ejecución en los dispositivos a descubrir
Postcondiciones: • Después de los segundos configurados, en la consola aparecerán los dispositivos descubiertos
Escenario principal de éxito 1. El administrador selecciona la opción “Descubrimiento” 2. La aplicación inicia un proceso de descubrimiento 3. El temporizador de la aplicación llega a 0 4. La aplicación muestra en la consola los equipos que ha descubierto
Extensiones **Punto 2 del escenario principal:** • El proceso ha generado un error • En la consola aparecerá que ha habido un error • El administrador deberá mirar el log de la aplicación para saber más sobre el error. • Se regresa al punto 3 del escenario principal

Tabla 5. Caso de uso: Detener escucha de Servicios

Caso de uso: Detener escucha de Servicios
Precondiciones: • Se dispone de una red local • La aplicación se encuentra en ejecución
Postcondiciones: • El dispositivo deja de escuchar en la red local
Escenario principal de éxito 1. El administrador selecciona la opción “Suprimir Respuestas” 2. La aplicación fuerza un cierre de la escucha de solicitudes en la red 3. La aplicación muestra en la consola que se ha detenido la escucha
Extensiones **Punto 2 del escenario principal:** • El proceso ha generado un error • En la consola aparecerá que ha habido un error • El administrador deberá mirar el log de la aplicación para saber más sobre el error.

3.4. Diagrama de Clases

Para entender cómo se va a distribuir el protocolo, graficaremos el programa con un diagrama de clases UML. En él podremos observar cómo todas las implementaciones extienden de una interfaz genérica, para poder extrapolar su comportamiento y que tenga el mismo comportamiento independientemente de los tipos de datos que nos lleguen. Así, vemos que tenemos una interfaz “*IData*” que representa el bloque de datos que nos puede enviar la máquina. Como se ha repetido a lo largo del documento, un ejemplo de esto puede ser el estado de la máquina: Nivel de uso de CPU, porcentaje de almacenamiento en disco duro, porcentaje de RAM en utilización, etc. Pero se puede enviar cualquier dato que pueda parecer interesante. La única restricción que tiene esta interfaz es que los datos enviados sean serializables. Es decir, que pueda conformarse el estado de un objeto instanciado en una secuencia de bytes que puedan viajar a través de la red o de las clases, y una vez llegan al destino, se pueda desconformar, creando una copia exacta del estado guardado en el momento de la serialización. [12]

El diagrama de clases, que corresponde a la Figura 12, se encuentra en la siguiente página en formato apaisado para una mejor comprensión del lector.

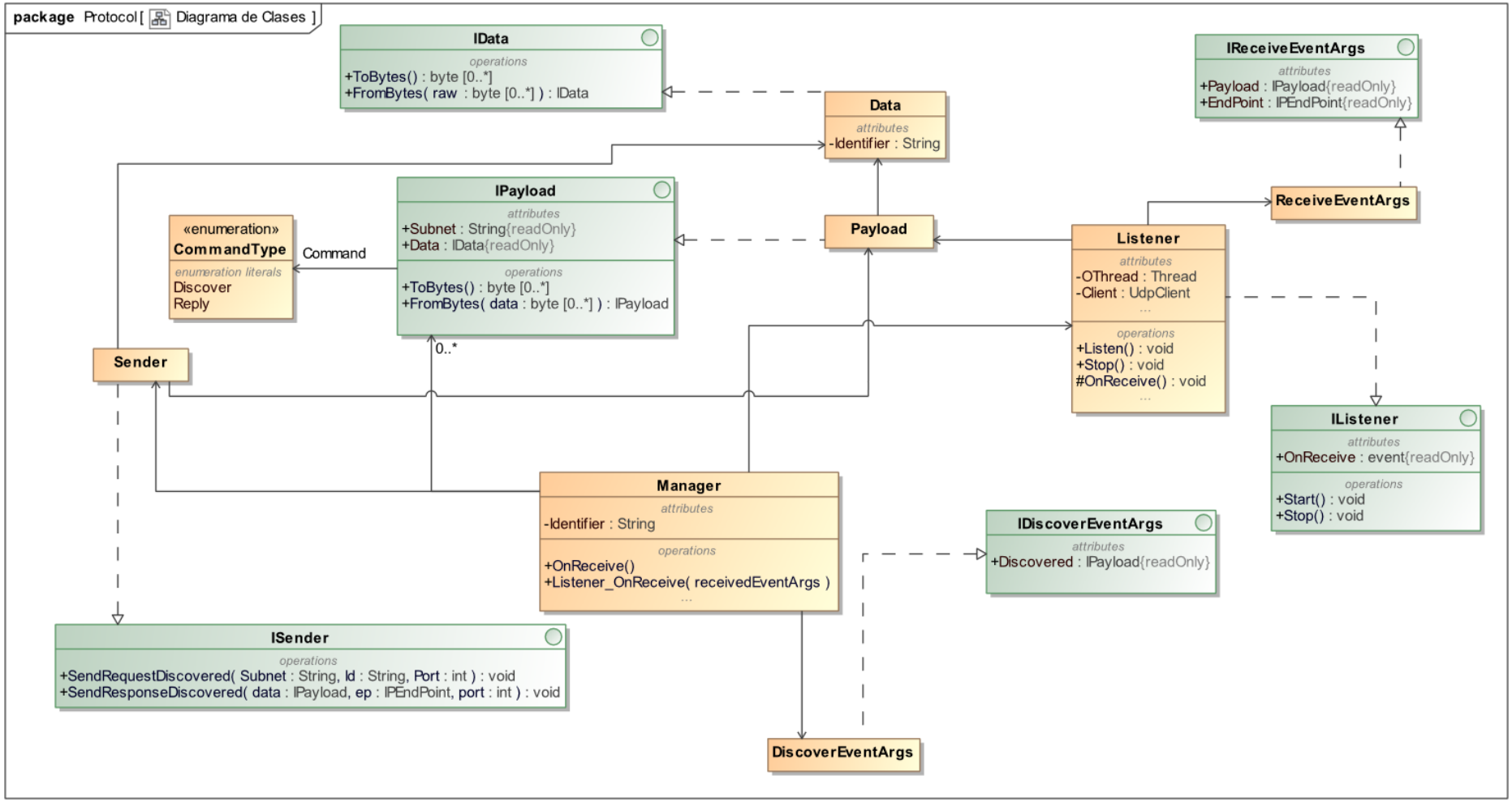

Figura 12. Diagrama de Clases

3.5. Diagramas de secuencia.

Para poder entender cómo va a funcionar el protocolo de descubrimiento, se expondrán diferentes diagramas de secuencia que muestran algunos de los flujos de ejecución que se darán a lo largo del tiempo en el programa. Para empezar, veremos qué ocurre cuando iniciamos el programa:

Caso Listener. Inicio de aplicación.

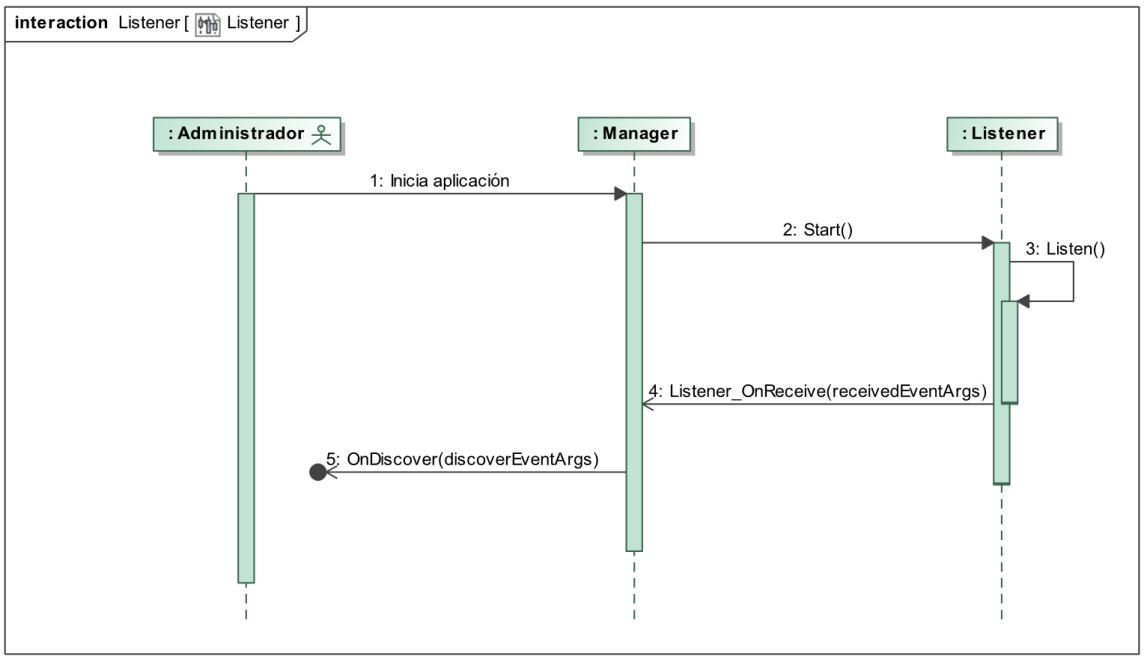

Figura 13. Diagrama de secuencia: Inicio de aplicación

Como se puede observar, en el momento en el que (normalmente) el administrador inicia el programa, se realizará una llamada de inicio al Manager, que es nuestro proceso demonio que facilitará el manejo de la clase encargada de escuchar la red y de la clase encargada de enviar tramas por la misma. También se encargará de lanzar un evento de .Net cada tantos segundos como se haya indicado en la configuración que nos mostrará en la interfaz los equipos que se hayan descubierto.

El proceso de escucha se inicia en un hilo aparte, ya que es una llamada bloqueante. Si no se iniciara en un nuevo hilo, el programa se quedaría en espera y no respondería. En cuanto la clase Listener detecta que le llega una trama nueva, se lanza un proceso que deserializará la información recibida y se encargará de enviársela al manager, para que pueda incluirlo en la lista para enviarlo a la interfaz.

Caso Sender. Petición de descubrimiento

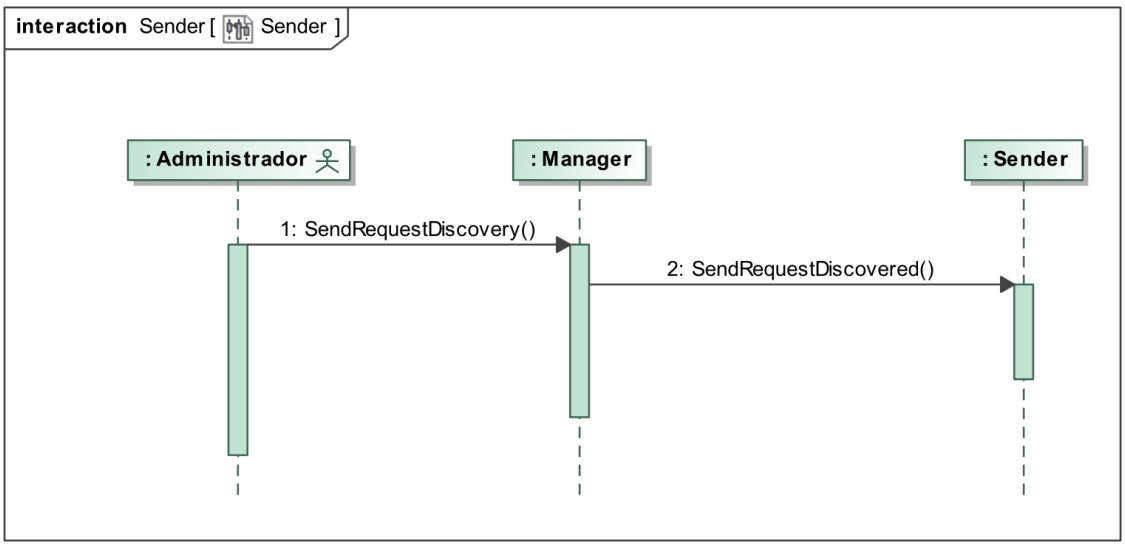

Figura 13. Diagrama de Secuencia: Petición de descubrimiento

En este caso se muestra el primer caso del Sender. El usuario o simplemente, según la configuración establecida, cada X segundos, se realizará una petición de descubrimiento en la red. De esta forma, se realizará una llamada al Sender, que se encarga de enviar la trama de broadcast. Una vez se envía, si hay algún programa en escucha, se realizaría similar al primer diagrama del caso Listener.

Caso Sender. Respuesta a un descubrimiento

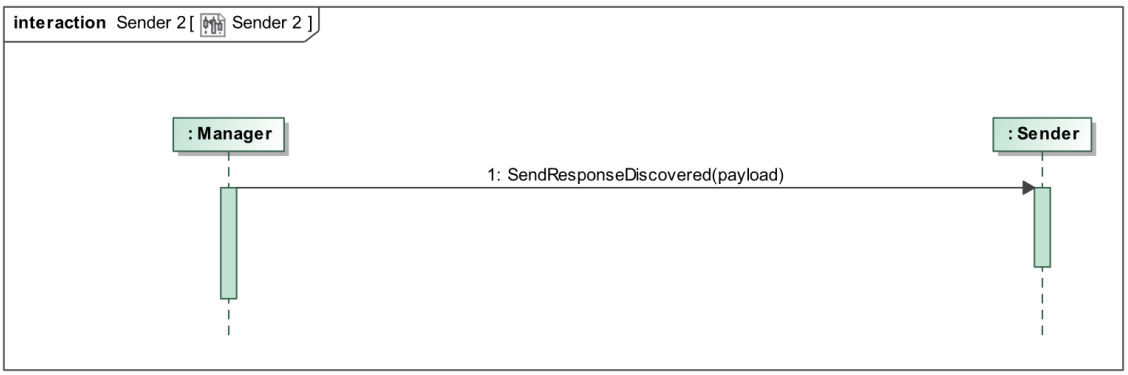

Figura 14. Diagrama de secuencia: Respuesta a un descubrimiento

Otro caso en el que se ve al Sender en acción es en el momento en el que el proceso demonio detecta que el Listener ha recibido una trama. En ese momento, se comprueba si lo que se ha recibido es una trama de petición de descubrimiento o si ha sido una respuesta a un descubrimiento. En el caso de que sea una petición, se hace una llamada al Sender para que envíe una respuesta a la aplicación que ha solicitado el descubrimiento.

3.6. Interfaz de Usuario

Hasta ahora no se conocía la interfaz final que ha quedado tras la realización de la aplicación de escritorio. Aún queda algunas pequeñas fases de diseño de la misma, pero la esencia de la simplicidad y la información está en ella.

Para describir esta interfaz, a la vez vamos a ir ejecutando un ejemplo realizado en mi propia casa utilizando la red local para el protocolo. En este caso, tendré en ejecución tres instancias de la aplicación. A modo de ejemplo de aplicación en IDbox, tendremos en uno la Web, que se encargará de hacer una petición de descubrimiento para conocer el estado de los agentes. Desde la Web, como se describirá más tarde, habrá una sección en la que se podrá monitorizar el estado de los agentes o máquinas en la que esté en ejecución la aplicación.

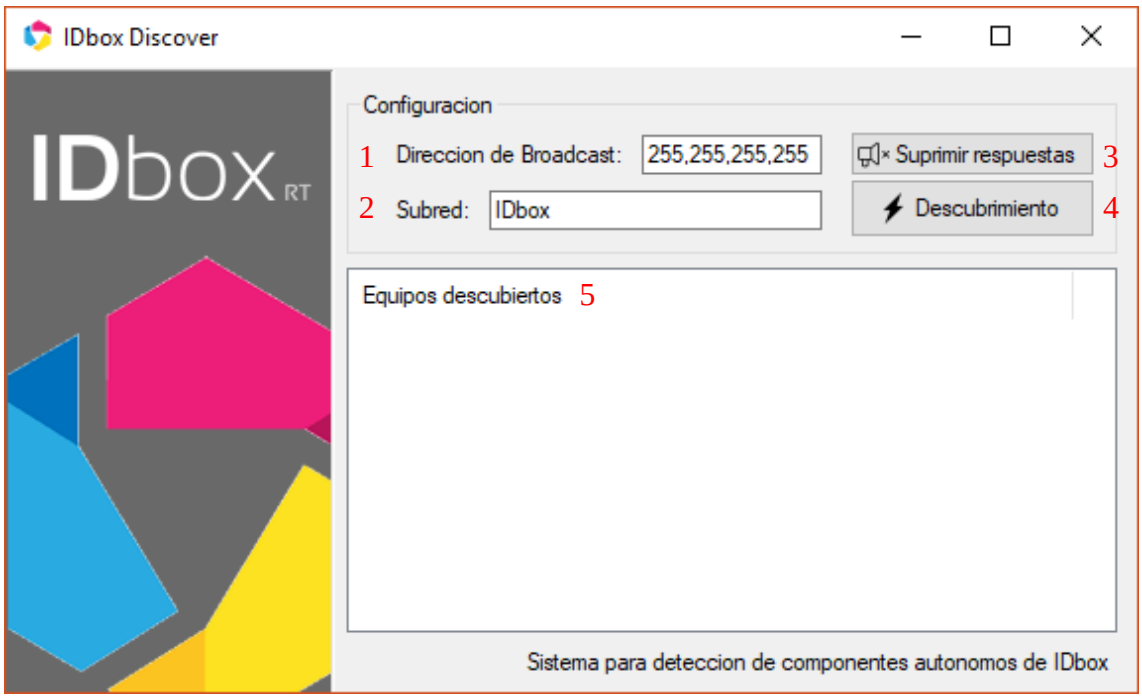

Figura 15. Interfaz de usuario

1. **Campo de dirección:** En este campo se introducirá la dirección de broadcast a la que se quiera realizar el descubrimiento.
2. **Subred:** Este campo sirve para identificar subredes por nombre, a modo de posible agrupación de equipos.
3. **Botón suprimir respuestas:** Este botón realiza una parada de la escucha de futuras peticiones. Si estaba parado y se selecciona el botón, se reanudará la escucha.
4. **Botón descubrimiento:** Al seleccionar este botón, se lanzará una petición de descubrimiento a la dirección de broadcast y subred establecidas.
5. **Consola de equipos descubiertos:** Aquí aparecerán descritos los equipos descubiertos, así como los datos que hayan enviado.

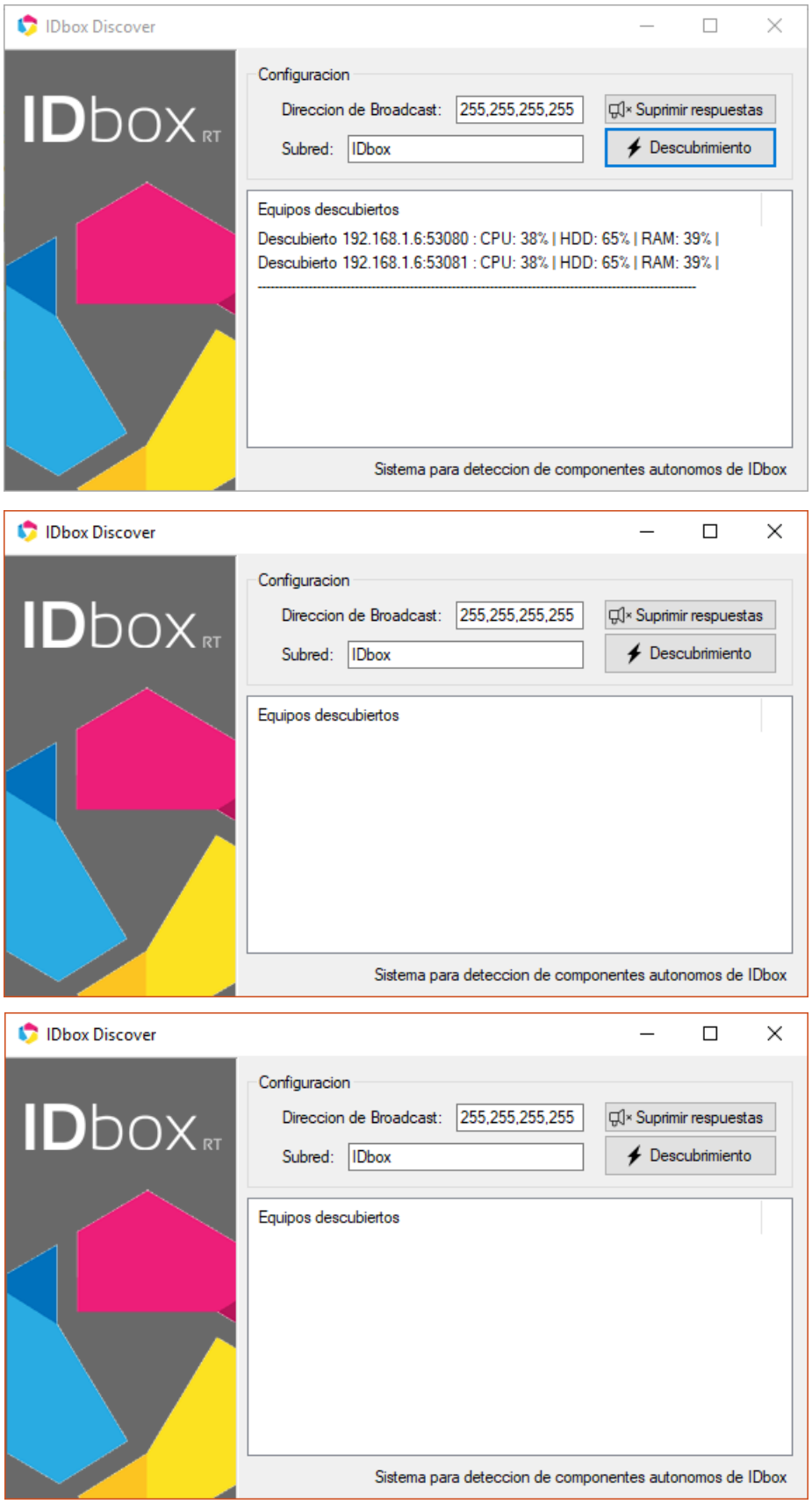

Figura 16. Ejemplo con tres instancias en el mismo equipo

En la Figura 16, podemos comprobar que, habiendo tres instancias de la aplicación, si ejecutamos una petición de descubrimiento (que corresponde al programa superior) veremos que después de los 5 segundos que tengo configurado de tiempo de descubrimiento, en la consola nos aparecen los programas descubiertos.

Ahora vamos a probar cómo se comporta si la comunicación se realiza entre diferentes computadores.

Estableceré dos instancias de la aplicación en mi ordenador portátil personal, y otras dos en mi equipo de sobremesa. Desde el equipo de sobremesa, seleccionaré la opción Descubrimiento, y por la consola de descubrimiento descrita en la interfaz, deberá aparecer un bloque de tres IPs.

Estas IPs deben ser diferentes, ya que corresponden a equipos diferentes.

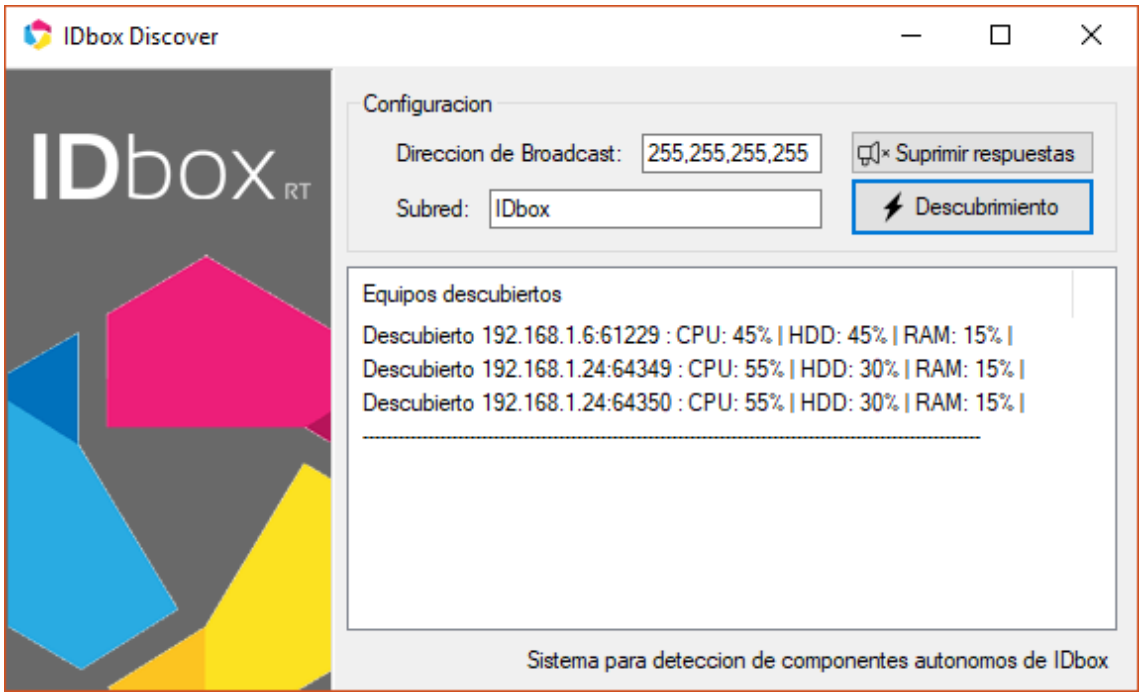

Figura 17. Ejemplo de interfaz con dos instancias en otro equipo y otras dos instancias en el equipo local

Efectivamente, se puede comprobar que aparecen dos IPs:

1. 192.168.1.6 corresponde a la instancia que se establece en el equipo de sobremesa
2. 192.168.1.24 corresponde a la instancia del equipo portátil.

4. Pruebas unitarias y de integración

Una vez se finaliza el desarrollo, se procede a realizar tests de tipo unitario tanto en los Listener como en los Sender. De esta forma, comprobaremos la funcionalidad requerida de cada uno. Aunque estas pruebas se describan como pruebas unitarias, se incluyen también como pruebas de integración, ya que son componentes diferentes, pero uno no puede funcionar sin el otro.

En una primera ejecución de las pruebas, se ve que, aunque el Sender envía correctamente, el Listener no recibe. Tras un trabajo de investigación, encontré que el error era que, utilizando el mismo puerto para todas las aplicaciones, era necesario que el envío de tramas sea por Multicast. El envío multicast se denomina al modo de envío en el que se envía la información a un grupo de IPs destino, dentro de un grupo de destino.
Como esto no interesaba, pues no siempre conoceremos las IP de destino, ambos envíos del Sender se realizarán en modo Broadcast.

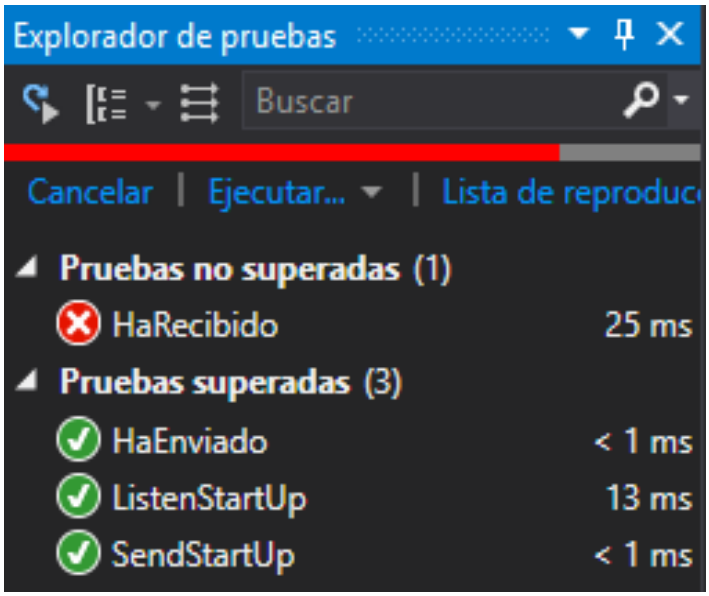

Figura 18. Pruebas fallidas

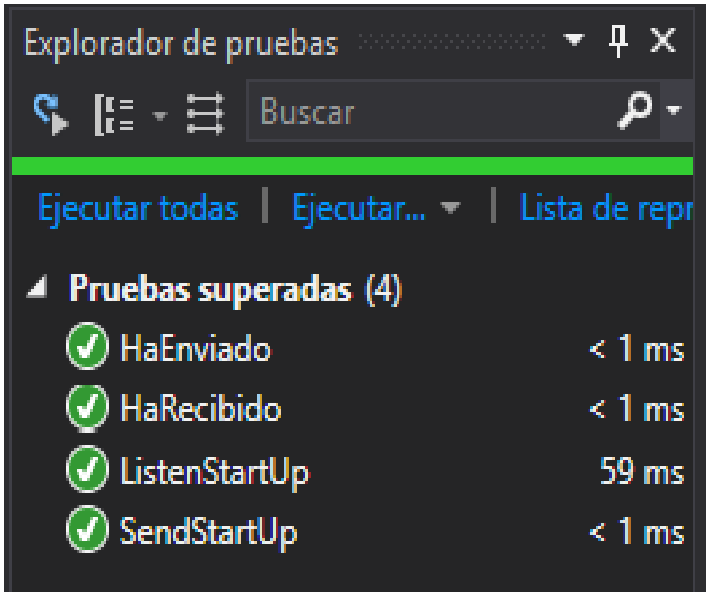

Figura 19. Pruebas satisfactorias

Con ese cambio de envío de tramas, comprobamos que superamos los test con éxito. Por tanto, concluimos que la aplicación, responde ante una petición de descubrimiento, y es capaz de recibir dichas peticiones.

En resumen, un Listener se instancia cada vez que se inicia la aplicación, y se queda escuchando por un puerto hasta que reciba una petición, o se le mande la orden de detener esa escucha. Seguido, un Sender realizará un descubrimiento en una trama Broadcast para que todos los equipos de la red reciban dicha trama.

El Listener instanciado escucha esa petición, y envía una respuesta a la dirección de la que recibió la petición de descubrimiento. El Listener que se instanció en la aplicación que inició el descubrimiento, escucha esta petición, y lo adjunta a una colección que, posteriormente, gracias a un evento de .Net, se escribirá en la interfaz para saber que se ha descubierto, dando todos los datos proporcionados desde el dispositivo descubierto, como se comprobó en la interfaz.

5. Conclusiones y líneas futuras

A pesar de que el proyecto no ha requerido de una codificación exhausta y compleja, el trabajo de investigación subyacente ha sido una de las partes más importantes del proyecto.

Antes de tener que escribir código, se ha tenido que estudiar la arquitectura de IDbox, su funcionamiento, características, estructuración… y unos cuantos elementos más que serían irrelevantes.

Al final, lo que ha determinado parte de la complejidad del proyecto ha sido la investigación, ya que era complicado encontrar bien como utilizar el protocolo y aprender a utilizar los UdpClient y Sockets de .Net.

El resultado del proyecto ha sido satisfactorio, aunque aún no se ha implementado en IDbox, esperando a futuras pruebas de rendimiento, y posibles vistas a futuro de la utilidad de esta herramienta.

Al principio, me esperaba que el proyecto iba a ser sencillo, ya que seguramente habría protocolos por internet dedicados a esto mismo, pero me encontré con que no era así y que, aunque era poco código, era muy importante que su funcionamiento sea exacto a los requerimientos.

En líneas generales, estoy bastante contento con el proyecto, aunque me ha llevado más tiempo del que esperaba. Los resultados son buenos y la empresa está contenta con esta aplicación, aunque solo es el principio de ella.

La aplicación ahora mismo está en *"Stand by"* para sacar partido a las implicaciones a futuro que puede tener el proyecto.

Actualmente, sigo en CIC Consulting Informático de Cantabria y aunque ahora mismo no esté desarrollando para esta aplicación, es solo cuestión de tiempo que se comience un desarrollo para ampliar el abanico de posibilidades que ofrece, como, por ejemplo, las siguientes fases pensadas sería el despliegue de nuevos servicios de IDbox a través de este servicio, o recibir solicitudes de configuración, siendo capaz de enviar comandos de IDbox a los servicios implicados en esas solicitudes.
Además, como he comentado en el apartado de la interfaz de usuario, ésta no está finalizada, y queda integrarla con una interfaz similar a la del resto de aplicaciones y herramientas de IDbox.

6. Bibliografía

[1] E. GUTTMAN, C. PERKINS y J. VEIZADES, «Service location protocol,» IEEE, Mountain View, 1997. [Último acceso: Agosto 2017]

[2] A. DANEELS y W. SALTER, «What is SCADA?,» 1999. [Último acceso: Agosto 2017]

[3] IDbox Team, «Manual de usuario de IDbox,» Santander, 2016. [Último acceso: Noviembre 2017]

[4] IDbox Team, «Presentación de IDbox,» Santander, 2017. [Último acceso: Noviembre 2017]

[5] Wikipedia, «Visual Studio,» [En línea]. Available: https://es.wikipedia.org/wiki/Microsoft_Visual_Studio. [Último acceso: Noviembre 2017].

[6] Wireshark, «Frequently Asked Questions,» [En línea]. Available: https://www.wireshark.org/faq.html. [Último acceso: Octubre 2017].

[7] Wikipedia, «MagicDraw,» Wikipedia, [En línea]. Available: https://es.wikipedia.org/wiki/MagicDraw_UML. [Último acceso: Noviembre 2017].

[8] Wikipedia, «Ethernet,» [En línea]. Available: https://es.wikipedia.org/wiki/Ethernet. [Último acceso: Noviembre 2017].

[9] Wikipedia, «C Sharp,» [En línea]. Available: https://es.wikipedia.org/wiki/C_Sharp. [Último acceso: Noviembre 2017].

[10] R. S. Pressman y J. M. Troya, Ingeniería del Software, 1988. [Último acceso: Octubre 2017]

[11] H. M. NewMan, «BacNet.org,» Octubre 1998. [En línea]. Available: http://www.bacnet.org/FAQ/HPAC-3-97.html. [Último acceso: Octubre 2017].

[12] P. OBERMEYER y J. Hawkins, Object Serialization in the .Net Framework, 2001. [Último acceso: Noviembre 2017]

[13] «Modelo incremental Iterativo,» [En línea]. Available: http://isw-udistrital.blogspot.com.es/2012/09/ingenieria-de-software--i.html. [Último acceso: Octubre 2017]