
# TRABAJO DE FINAL DE CARRERA

**TÍTULO: Configuración de redes de sensores y pequeños dispositivos utilizando esquemas P2P para entornos de automatización doméstica.**

**AUTOR: Samuel Maseda Silva**

**DIRECTOR: Toni Oller Arcas**

**FECHA: 24 de junio de 2007**

**TÍTULO:** Configuración de redes de sensores y pequeños dispositivos utilizando esquemas P2P para entornos de automatización doméstica.

**AUTOR:** Samuel Maseda Silva

**DIRECTOR:** Toni Oller Arcas

**FECHA:** 24 de junio de 2007

**Resumen**

En este trabajo se describe las tareas realizadas para obtener una aplicación de gestión domótica basada en el control de presencia de los elementos de una casa. Se muestra el diseño, la arquitectura y la implementación del mismo.

El resultado de este trabajo permite detectar y representar gráficamente una serie de elementos de la casa sobre los que podremos actuar de manera remota a través de la red. Los elementos conectados se representan con procesos que realizan una llamada a nuestro servidor, quien lleva un control de los mismos, además de informar a los dispositivos de representación gráfica y gestión: los clientes.

La detección de elementos se realiza mediante el protocolo de presencia SIP, apoyándonos en un sistema de listado propio, basado en XML.

Tanto cliente como servidor han sido desarrollados bajo un entorno Java, siendo totalmente portables a prácticamente cualquier plataforma.

Este proyecto es la herramienta básica de visualización y actuación entre cliente y entorno domótico, quedando exento de este trabajo cualquier sistema de autogestión inteligente. A pesar de ello, se podría integrar en una estructura inteligente y con tareas programables formando, de esta manera, un gestor domótico aprovechado al cien por cien.

**Title:** Configuración de redes de sensores y pequeños dispositivos utilizando esquemas P2P para entornos de automatización doméstica.

**Author:** Samuel Maseda Silva

**Director:** Toni Oller Arcas

**Date:** June 24th, 2007

**Overview**

This project describes the tasks done to obtain a home automation application based on the presence control of the house elements. The project shows the design, architecture and implementation of the said application.

The result of this project detects and graphically represents a series of elements of the house on that we can act remotely over the net. The connected elements are represented by processes that call our server, that is in charge of control them and inform the visualization devices (the clients) about them, too.

Element detection is done through the SIP presence protocol, leaning on an own element listing system based on XML.

Client and server have been developed using Java, so they are completely portable systems, to almost any other platform.

This project is the basic visualization and actuation tool between client and home environment, and doesn't contain any kind of smart management system. Although that, it could be part of a smart structure that could perform programmable tasks, becoming a complete home automation device.

# ÍNDICE

# ÍNDICE DE FIGURAS Y TABLAS

# INTRODUCCIÓN

Hoy en día un hogar debe de ser, cuanto menos, confortable para el que lo habita. Después de una jornada de trabajo todo el mundo desearía poder llegar a casa y descansar. Despreocuparse de los quehaceres domésticos. Poder tener más tiempo para uno mismo.

Ésta es la máxima de una casa domótica, la autogestión de si misma apoyándose en la tecnología a nuestro alcance. En la medida de lo posible claro está.

Programar tareas tan cotidianas como el encendido y apagado de la calefacción, por poner un ejemplo, presenta una doble ventaja. No solamente puede resultar más cómodo para el habitante de la casa, quien se encontrará con la temperatura ideal al entrar, sino que además, un uso eficiente supone un ahorro en el consumo eléctrico, de agua, gas…

Por otro lado, las personas están en constante movimiento. Por ello, para poder supervisar o intervenir en la gestión de su hogar es imprescindible algo que hoy en día nos ofrecen la gran mayoría de tecnologías móviles: la posibilidad de un contacto remoto. Precisamente, la movilidad.

Con todo esto entendemos que los medios de que disponemos actualmente pueden cubrir por completo las principales necesidades que se nos presentan al pensar en un entorno de control domótico. Y si partimos de la base de que podemos interactuar con cualquier elemento de la casa, las posibilidades son infinitas.

Por ello, el objetivo de este trabajo es explorar y desarrollar un modo de interacción con los elementos que componen un hogar, centrándonos en la individualidad de cada uno de ellos.

El sistema, desarrollado en Java, consta, por un lado, de una interfície de visualización y gestión de los elementos de la casa, destinada a la interacción por parte del usuario. Por otro lado, el sistema cuenta también con un servidor que se encarga de gestionar y auto detectar cuándo un nuevo elemento es incorporado en el entorno doméstico e indicarlo a las correspondientes interfícies de control.

Los objetivos del proyecto, pues, son:

- Ofrecer un sistema de presencia de elementos en el entorno de la casa.
- Poder representar gráficamente la localización y tipología del elemento.
- Ofrecer posibilidad directa de actuación sobre cualquiera de los elementos representados, a través de la interfície de visualización.
- Una interfície en Java, portable y que solo requiere de un dispositivo compatible y una conexión de red para funcionar.

Este trabajo se divide en 6 partes:

En el primer capítulo se describen las funcionalidades de la aplicación con ayuda de un diagrama de casos de uso.

El segundo capítulo está destinado a dar unos conceptos básicos de cara a una mejor comprensión del resto del trabajo.

El tercer capítulo muestra el diseño y la arquitectura usada para desarrollar nuestro sistema, además de un apartado donde se muestra la interacción entre sus elementos

En el cuarto capítulo se explica la implementación de cada módulo de la arquitectura descrita en el escenario de pruebas usado.

En el quinto nos encontramos con la planificación del trabajo, los trabajos realizados y el tiempo empleado en cada uno de ellos.

Finalmente, el sexto capítulo expone las conclusiones tras la realización del TFC. Se compone de tres partes, los objetivos alcanzados, el impacto medioambiental y las conclusiones personales.

# CAPÍTULO 1. ESPECIFICACIÓN

En este proyecto se controla un entorno domótico completo. Parte de un servidor, encargado de detectar cada uno de los elementos que se conectan en ese entorno. Para ello, elementos y servidor se intercambian una serie de mensajes, basados en el protocolo SIP[1].

Al tratarse de una aplicación de SIP orientada a la domótica estamos usando una serie de mensajes de dicho protocolo destinados a este ámbito. Esta tipología de uso es conocida como “Home SIP”[2].

Así pues, cada vez que se conecte un elemento en nuestra casa domótica, dicho elemento enviará un mensaje al servidor para que éste tome nota de su presencia.

Para poder visualizar la información que posee el servidor de una manera intuitiva e inteligible para el usuario final, existe también la aplicación de gestión.

La aplicación de gestión puede estar instalada en un ordenador de sobremesa, portátil, o en cualquier entorno Java compatible. Desde esta aplicación el usuario se conectará al servidor y en su pantalla se mostrará un plano de la casa, con cada uno de los elementos conectados representados gráficamente en su posición actual. Además, el usuario tiene la capacidad de actuar sobre esos elementos. Puede, por ejemplo, seleccionar del listado de elementos que también verá, el elemento sobre el que desee actuar y aplicar el cambio convenientemente.

Para hacernos una idea del potencial de la interactividad, propongo un ejemplo: El dueño de la casa acaba de terminar su jornada laboral. Desde un ordenador del trabajo inicia la aplicación de gestión donde, entre otras cosas, observa la nevera. Quiere saber si de camino a casa debe parar para hacer la compra. Para ello selecciona la nevera en la lista de elementos y le pide información acerca de lo que hay en su interior. Sencillo y práctico.

A continuación veremos un diagrama de los posibles casos de uso por parte del usuario.

[1] Session Initiation Protocol. Consultar los ACRÓNIMOS.
[2] Proyecto de control domótico mediante SIP. Consultar el apartado 2.2

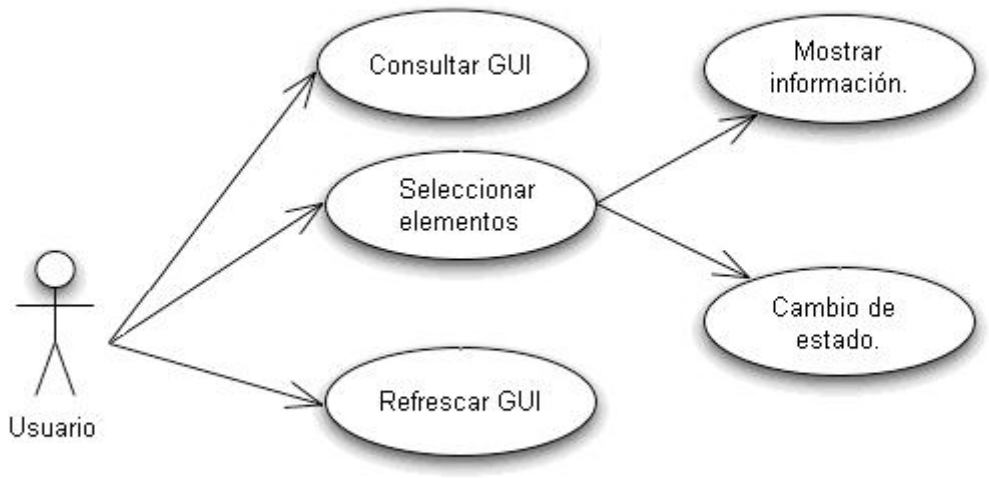


**Fig. 2.1 Casos de uso del usuario.**

**Tabla 1.1 Descripción de los casos de uso del usuario.**

| | |
|---|---|
| **Nombre** | Consultar GUI[1]. |
| **Descripción** | Revisar los elementos del entorno domótico. |
| **Actor** | Usuario. |
| **Flujo normal** | El usuario observa el plano, listado de elementos y consola para conocer la situación de los elementos. |

| | |
|---|---|
| **Nombre** | Mostrar información. |
| **Descripción** | Obtener información detallada de un elemento. |
| **Actor** | Usuario. |
| **Flujo normal** | 1. El usuario selecciona del listado con un clic el elemento del cual desea obtener información.<br>2. Pulsa el botón “Estado”. |
| **Entrada** | Elemento Seleccionado. |
| **Salida** | Nombre, tipo, coordenadas, y estado de ese elemento. |

| | |
|---|---|
| **Nombre** | Cambio de estado. |
| **Descripción** | Cambiar el estado de un elemento. |

[1] Graphic User Interface. Consultar los ACRÓNIMOS.

| | |
|---|---|
| **Actor** | Usuario. |
| **Flujo normal** | 1. El usuario selecciona del listado con un clic el elemento del cual desea obtener información.<br>2. Pulsa el botón “Acción”. |
| **Entrada** | Elemento Seleccionado. |
| **Salida** | La visualización del elemento sobre el que se ha actuado cambia, para representar el estado en que se encuentra. |

| | |
|---|---|
| **Nombre** | Refrescar GUI. |
| **Descripción** | Actualización de la interfaz gráfica forzada por el usuario. |
| **Actor** | Usuario. |
| **Flujo normal** | El usuario pulsa el botón “Actualizar”. |
| **Entrada** | Instrucción mediante botón. |
| **Salida** | Borrado y nueva representación de todos los elementos sobre mapa y listado, en base a la última notificación recibida. |

# CAPÍTULO 2. CONCEPTOS BÁSICOS

Antes de comenzar a adentrarnos en la arquitectura del proyecto, es necesario introducir brevemente una serie de conceptos previos para sentar las bases de posteriores explicaciones.

## 2.1 SIP

El Session Initiation Protocol (SIP) o Protocolo de Inicio de Sesión es un protocolo que permite la creación, modificación y finalización de sesiones multimedia entre varios participantes.

Creado por el IETF[1] MMUSIC[2] Working Group, es uno de los protocolos de señalización para VoIP[3] junto con H.323[4]. En SIP pueden intervenir elementos multimedia tales como vídeo, voz, mensajería instantánea, juegos online…

Para lograr tales objetivos, SIP se basa en el intercambio mensajes entre los elementos que quieren formar parte de una sesión multimedia (UA[5]), y el servidor que los gestiona. Algunos de estos mensajes son:

- **REGISTER**: El UA lo envía al proxy SIP para registrarse en su listado de usuarios.
- **SUBSCRIBE**: El UA lo envía al proxy SIP con el fin de suscribirse a eventos. El proxy reenvía este mensaje al Agente de Presencia (PA[6]), que se encarga de gestionar este tipo de peticiones. Cada vez que uno de esos eventos suceda, se le notificará al UA suscrito.
- **NOTIFY**: Mensaje destinado a notificar un evento a uno o varios UA. Es generado por el PA.
- **DO**: Mensaje petición de acción de un UA sobre otro UA.
- **OK**: Reconocimiento positivo de cualquiera de los mensajes anteriores.

## 2.2 HOME SIP

Home SIP es un proyecto en el que se pretende controlar un entorno domótico mediante el uso del protocolo SIP.

[1] **I**nternet **E**ngineering **T**ask **F**orce. Consultar los ACRÓNIMOS.
[2] Multiparty Multimedia Session Control. Consultar los ACRÓNIMOS.
[3] Voice over IP. Consultar los ACRÓNIMOS.
[4] Recomendación ITU-T para proveer comunicaciones audiovisuales sobre red.
[5] User Agent. Consultar los ACRÓNIMOS.
[6] Presence Agent. Consultar los ACRÓNIMOS.

El uso de SIP en domótica presenta una serie de ventajas que han impulsado la proliferación del estudio de su aplicación doméstica. Algunas de estas ventajas son:

- No es necesario conocer la IP[1] del UA al que queremos acceder para contactar con él.

- Los dispositivos que se encuentren tras el NAT[2], gateway o firewall no pueden ser directamente direccionables. Así, sus direcciones pueden ser IP privadas.

- Provee seguridad: autenticación o autorización y privacidad al acceder a los dispositivos. Especialmente útil para acceder a ellos desde Internet.

- Las peticiones que podemos llegar a realizar a los dispositivos pueden ser síncronas (*streaming* de video) y asíncronas (petición de encendido/apagado de un elemento).

- Los contenidos del mensaje se basan en tipos MIME[3], por lo que el contenido es muy flexible.

De esta manera, un protocolo multimedia actualmente usado en telefonía IP puede también reconvertirse en un completo conjunto de herramientas perfectamente válidas para el control domótico, como iremos viendo en este trabajo.

[1] Internet Protocol. Consultar los ACRÓNIMOS.
[2] Network Address Translation. Consultar los ACRÓNIMOS.
[3] Multipurpose Internet Mail Extensions. Consultar los ACRÓNIMOS.

# CAPÍTULO 3. ARQUITECTURA

En este apartado entraremos en el funcionamiento del trabajo con un mayor detalle. Para ello comenzaremos comentando un diagrama general con los elementos que componen nuestro entorno domótico.

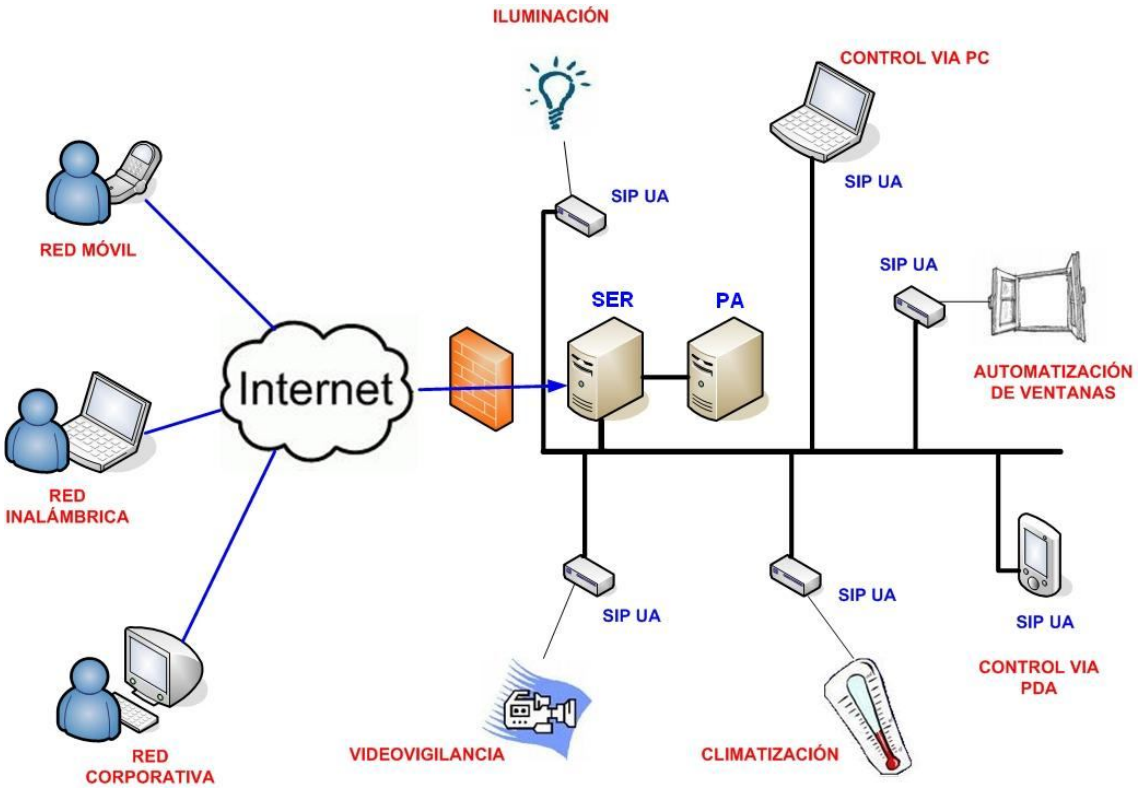


**Fig. 3.1 Arquitectura general del sistema.**

En este esquema podemos observar diversos elementos, con dos cosas en común. La primera es que todos ellos se apoyan en el uso de un Agente de Usuario (UA) que les permite la comunicación mediante el protocolo SIP. La segunda es que todos ellos se comunican directamente con el proxy SIP. Por tanto, el UA realiza la función de "intérprete" para que cada uno de los elementos pueda formar parte de este sistema centralizado.

Así, cada UA posee un módulo SIP que puede generar y recibir mensajes sobre TCP[1] para comunicarse con otras UA. Este módulo SIP puede ser también denominado Pila SIP.

[1] Transmisión Control Protocol. Consultar los ACRÓNIMOS.

## 3.1 Categorización de elementos por tipo

Ahora que hemos tenido una visión general, focalizaremos nuestra atención en las individualidades de cada uno de los componentes principales del entorno de la figura. Los podemos diferenciar empleando cuatro categorías: dispositivos, Herramientas de monitorización y el servidor junto con el Agente de Presencia.

### 3.1.1 Dispositivos

En este grupo se encuentran encuadrados los elementos sobre los cuales se puede ejercer una acción voluntaria por parte del usuario, con el fin de conseguir un resultado. Una bombilla es un ejemplo sencillo, pero también lo sería un sensor o una alarma los cuales están totalmente sujetos a su programación o gestión por parte del usuario.

Esta petición de acción se lleva a cabo empleando el mensaje de tipo DO.

Además, cada uno de ellos se registra en el SER y se suscribe en el PA, como veremos más adelante.

### 3.1.2 Herramientas de monitorización

Este grupo está formado básicamente por los dispositivos que constan de la interfaz gráfica y de gestión. Entran pues, ordenadores, PDAs y prácticamente cualquier soporte con conexión de red y compatible con Java. Serían, por decirlo de manera sencilla, los actuadores en nuestro sistema, ya que poseen la capacidad de gestionar a los dispositivos, cosa que no pasa a la inversa.

Podríamos representarlos de la siguiente manera:

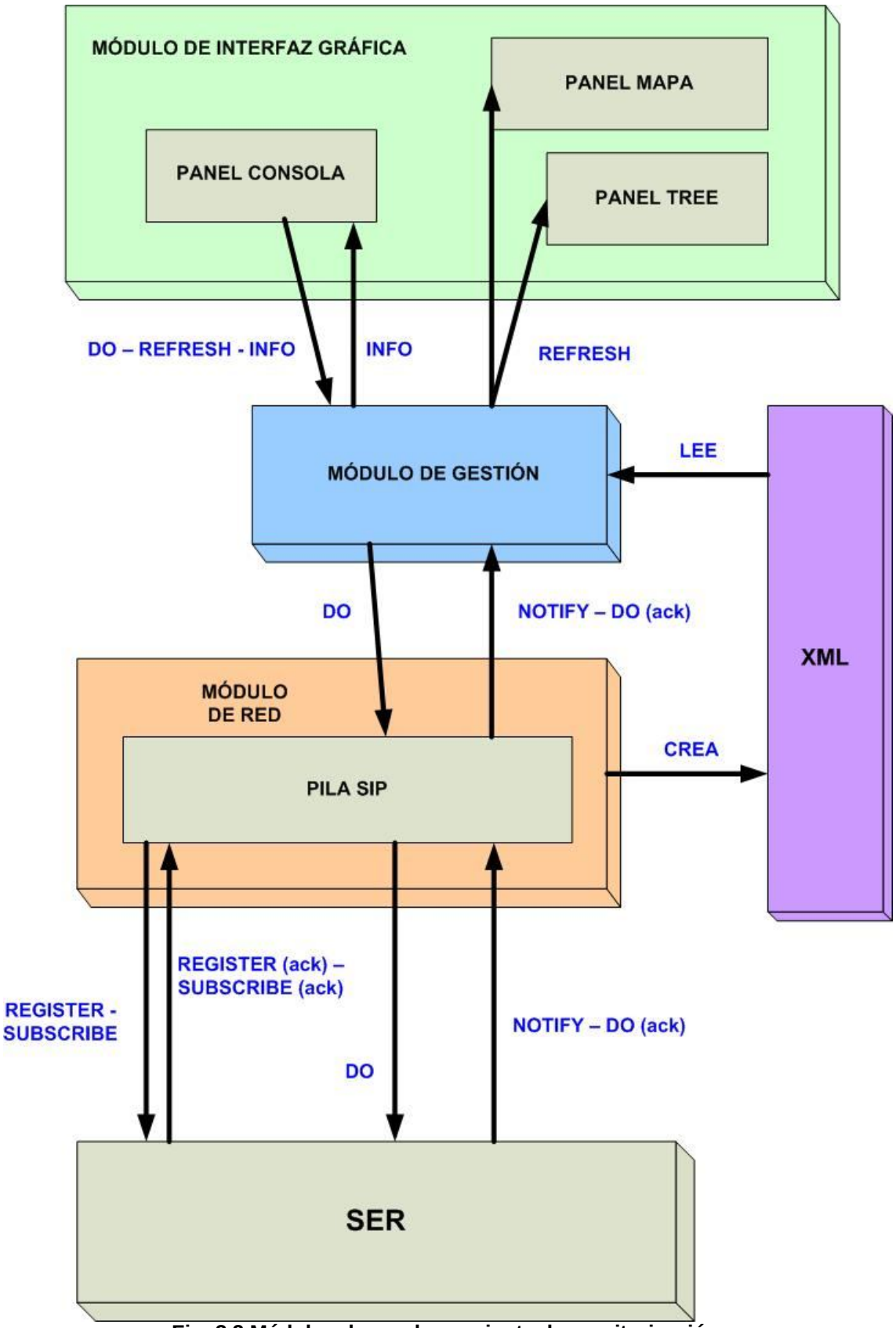


**Fig. 3.2 Módulos de una herramienta de monitorización.**

Así, lo mejor será que expliquemos cada uno de los tres módulos que conforman este tipo de elementos separadamente.

#### *3.1.2.1 Módulo de Red*

Contiene la pila SIP. Al iniciarse, se registra en el SER y se suscribe a eventos de presencia en el PA. Es el único tipo de elemento que puede generar mensajes SIP de tipo DO.

Al recibir los mensajes de tipo NOTIFY, elabora un archivo de tipo XML con los datos de los elementos activos. Si observamos el campo de datos de estos mensajes, podremos ver una cadena de caracteres en formato XML que, al ser procesada por la pila SIP de una herramienta de monitorización, da como resultado un archivo de extensión XML con los datos de cada uno de los elementos presentes.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  - <clientes>
    - <element>
        <name>GUI</name>
            <type>0</type>
            <coord_x>10</coord_x>
            <coord_y>23</coord_y>
      </element>
      - <element>
            <name>Camara#1</name>
           <type>1</type>
            <coord_x>10</coord_x>
            <coord_y>19</coord_y>
      </element>
      ...
  </clientes>
```

#### *3.1.2.2 Módulo de interfaz gráfica*

Posee un mapa de localización de dispositivos, que los representa mediante iconos en una rejilla en las coordenadas que corresponda. Posee también un listado de los elementos activos en el escenario y por último una consola de información con tres botones: Acción (mensaje DO al dispositivo seleccionado de la lista), Estado (información sobre el elemento seleccionado de la lista) y Refrescar (actualiza la interfaz gráfica).

#### *3.1.2.3 Módulo de gestión*

Su cometido básico es doble. Por un lado debe arrancar los dos módulos anteriores, y por otro interconectarlos.

El módulo de gestión trabaja arrancando la interfaz gráfica y la de red, que crea una pila SIP que se registra en el SER y se suscribe a eventos en el Agente de Presencia. Así, cada vez que llegue un mensaje de tipo NOTIFY y el módulo de red lo convierte en un fichero XML, el gestor se encarga de leer ese archivo, convertirlo en una lista de usuarios para posteriormente indicar a la GUI que la represente gráficamente.

También trabaja a la inversa, desde la interfaz gráfica hacia la de red. Esto se produce cuando indicamos que queremos realizar una acción sobre un elemento. La GUI detecta el evento generado por la pulsación de un botón, e indica al gestor el evento a realizar. El gestor, por su parte, acude a la pila SIP que ha generado para llamar a la función correspondiente a la acción solicitada.

### 3.1.3 Proxy

El proxy es intérprete, árbitro y gestor. Intérprete porque comprende el protocolo SIP y todos los mensajes que se usan. Gestor porque toma nota de cada uno de los elementos que aparecen en el escenario, llevando el control de cada uno de ellos. Y árbitro porque pone en contacto elementos aunque estos no conozcan la IP del destinatario de su mensaje. Es el caso de las acciones sobre elementos.

Cuando encendemos o apagamos una bombilla, la herramienta de monitorización manda un mensaje al dispositivo para que actúe de una determinada manera. El mensaje de la herramienta de monitorización ha de pasar por el proxy, quien conoce los datos del dispositivo y re direcciona la información.

### 3.1.4 Agente de Presencia

El Agente de Presencia (PA) trabaja conjuntamente con el proxy SIP, dotándolo de una funcionalidad esencial para el desarrollo de este trabajo: el control de presencia.

Para ello, el PA ofrece un servicio de suscripción a eventos. Enviará notificaciones con la lista de los elementos suscritos, generadas directamente por el PA, que trabaja apoyándose en el proxy SIP tanto para enviar como para recibir mensajes. No implementa, pues, los mismos mensajes que el resto y debe ser accesible en todo momento.

Cuando una herramienta de monitorización se suscribe al evento “presence”, es decir, cuando se produce una nueva suscripción, el PA recupera los datos

del elemento suscrito, los añade a una lista y envía los mensajes de tipo NOTIFY correspondientes con información en una cadena de bytes en formato XML.

## 3.2 Funcionamiento de los elementos

Ahora que hemos conocido los diferentes elementos que componen un escenario, expliquemos cómo funcionan.

La manera de trabajar de los elementos la podemos clasificar en tres apartados: cómo el proxy descubre un elemento, cómo la interfaz gráfica representa cada evento sucedido y por último, cómo el usuario actúa sobre los elementos.

### 3.2.1 Descubrimiento de un nuevo elemento

Obviamente nuestro proxy, así como cada uno de los elementos, comprenden los mensajes básicos de SIP que usaremos. Así, cuando un nuevo elemento es conectado, envía un mensaje de registro al servidor para indicarle que está presente en la casa.

Además, el elemento se incluye automáticamente en una lista de elementos activos de la casa. Así se indica a nuestro proxy, mediante un mensaje de suscripción a eventos, que debe notificar de los cambios que se produzcan en el mismo. Esto servirá para la posterior representación gráfica del elemento. Lo veremos más adelante.

Para la suscripción a eventos, nuestro proxy cuenta con un PA, destinado a gestionar el listado de todos los elementos activos suscritos. Delega, pues, esa tarea al proceso del PA.

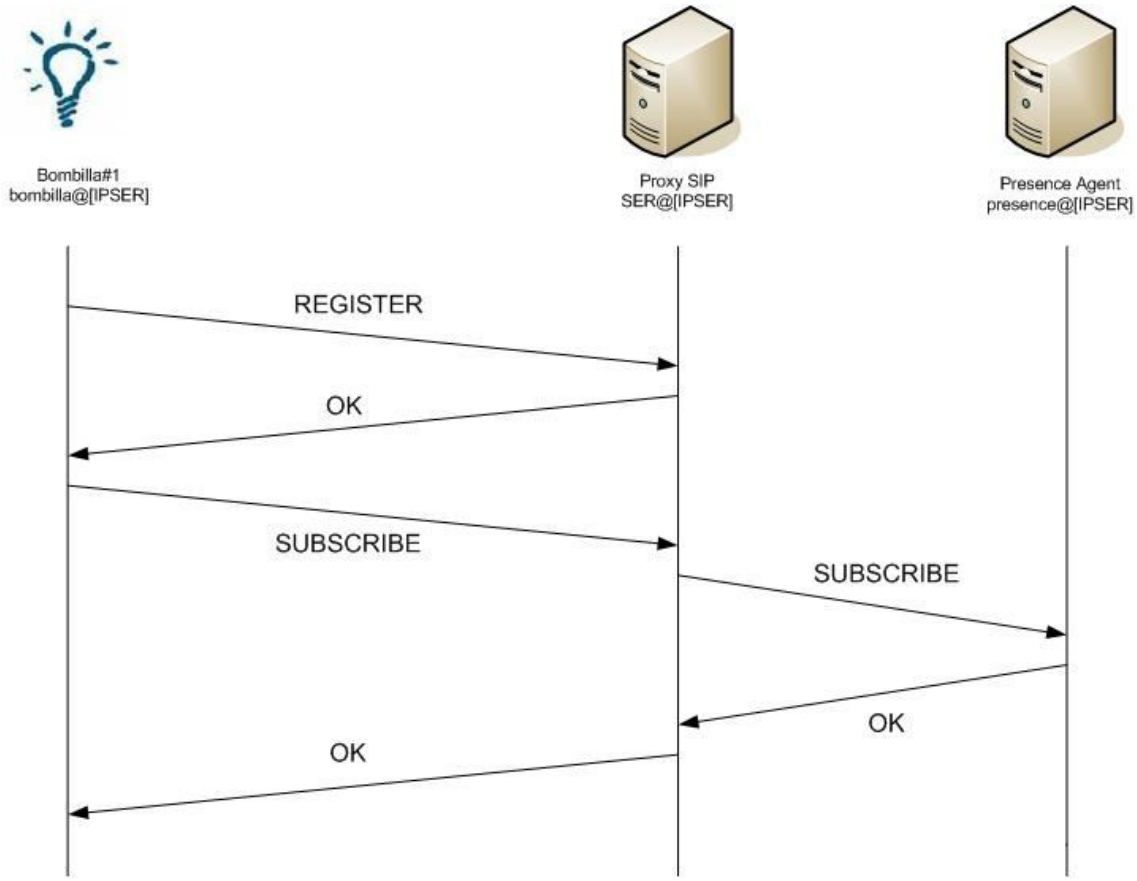


**Fig. 3.3 Intercambio de mensajes de suscripción y registro**

### 3.2.2 Representación gráfica y eventos

Como hemos podido observar en el esquema anterior, una subscripción a eventos implica un mensaje de notificación. Ese mensaje se envía a todos los usuarios suscritos.

Por tanto, si creamos un elemento que se centre en escuchar los eventos, podremos, fácilmente elaborar un listado actualizado con los elementos activos. Y si este elemento lo complementamos con una interfaz gráfica, podremos usarlo para representar visualmente los eventos sucedidos.

El ejemplo lo vemos en el siguiente esquema, que muestra lo que sucede al incorporar un elemento de representación gráfica en un entorno compuesto por una bombilla ya registrada y subscrita:

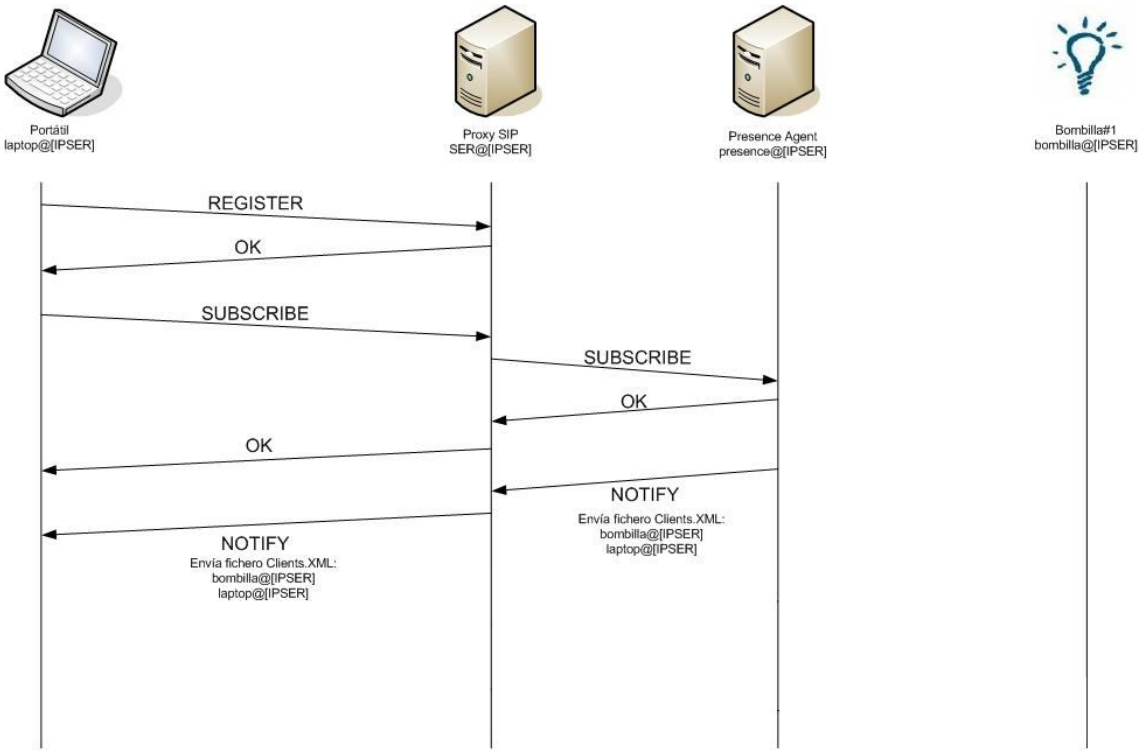


**Fig. 3.4 Envío de mensajes de notificación.**

De esta manera, extrayendo los elementos activos del campo de datos del mensaje de notificación, podemos elaborar y dibujar un listado sobre un plano. La información enviada en el mensaje de notificación, por consiguiente, ha de contener el "nombre" del elemento, el tipo, y sus coordenadas.

Así, la interfaz gráfica carga el icono correspondiente al tipo de elemento que sea, lo sitúa en el lugar de la casa que corresponda y lo añade al listado de elementos activos para que posteriormente el usuario pueda interactuar con él, si así lo desea.

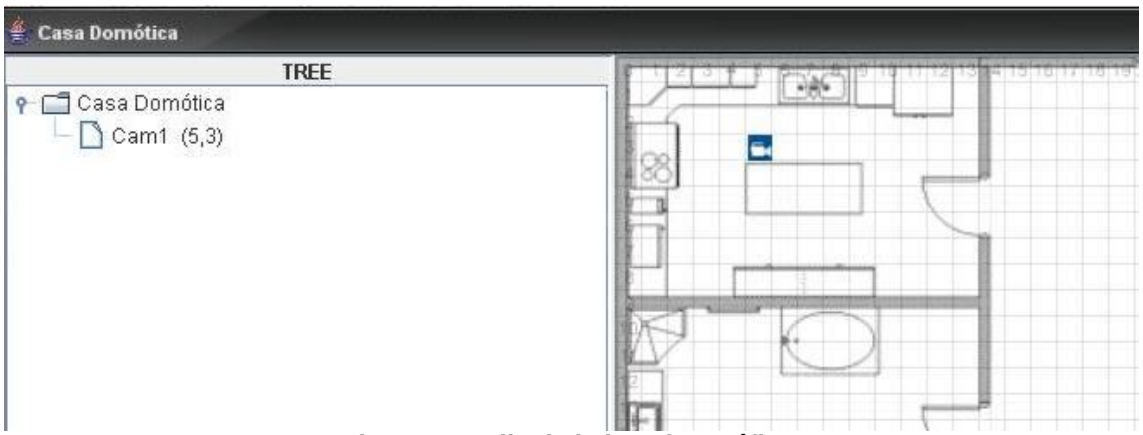


**Fig. 3.5 Detalle de la interfaz gráfica.**

### 3.2.3 Actuación sobre los elementos

Una vez sabemos los elementos que están presentes, podemos actuar sobre cada uno de ellos, sin excepción. La diferencia radica en que cada tipo de elementos podrá hacer una serie de funciones concretas. Las bombillas se encenderán y apagarán, el climatizador nos permitirá regular su temperatura, los relojes nos mostrarán la hora y nos dejaran programar alarmas, etc.

Para ello, nos serviremos del mensaje SIP de acción sobre los diferentes elementos. El mensaje es el mismo para todos los diferentes tipos de elementos que podamos encontrar. La diferenciación de la acción entre elementos se da en la clase que procesa los mensajes DO de cada dispositivo.

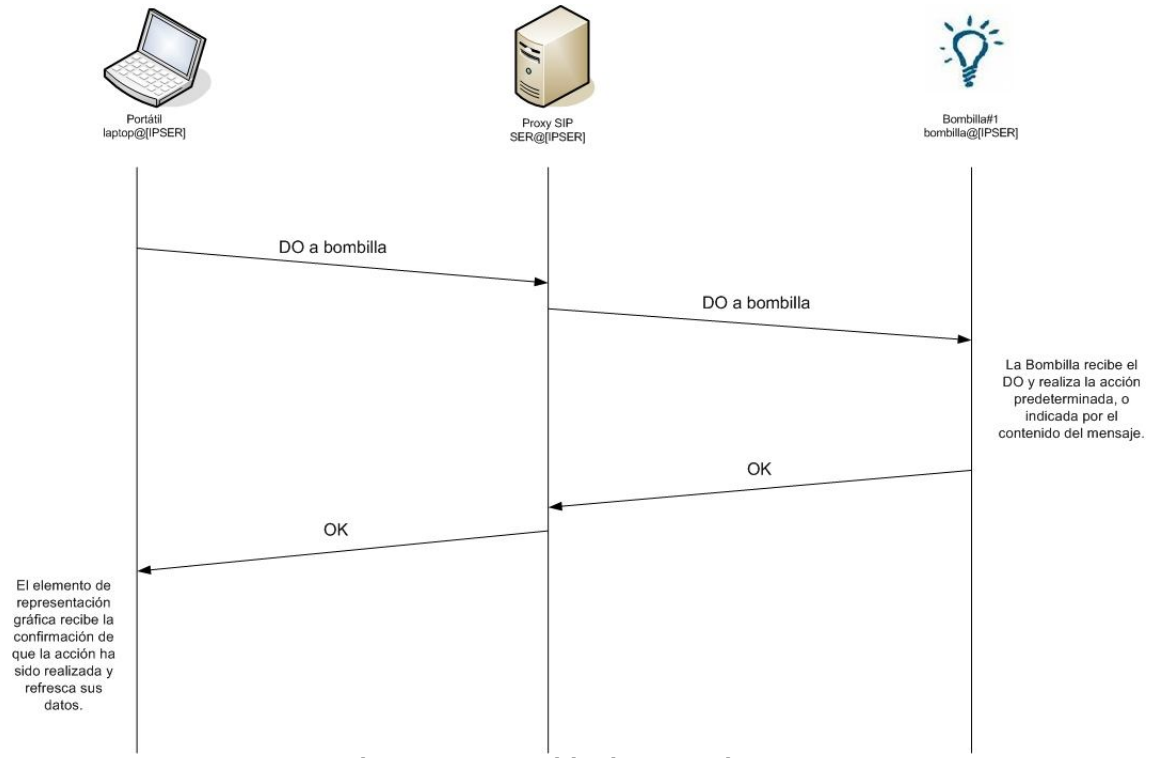


**Fig. 3.6 Intercambio de mensajes DO.**

# CAPÍTULO 4. IMPLEMENTACIÓN

En el primer apartado de este capitulo se describe la implementación de cada uno de los módulos de los cuales se compone la aplicación. Sigue el mismo orden que el apartado anterior.

En el segundo se describen las herramientas utilizadas para el desarrollo de todas las tareas que han compuesto el trabajo.

## 4.1 Implementación de los elementos

La implementación de los elementos que conforman el proyecto se basa en Agentes de Usuario (UA).

Los UA han sido implementados empleando Java, siguiendo las especificaciones definidas por SIP y con la API JAIN SIP, desarrollada por NIST. Cada UA tiene, independientemente de la función que cumpla, las siguientes clases:

- **SipClient**: Interfície que define los métodos principales para la creación del UA.

- **SipClientImp**: Clase que implementa a SipClient. La usamos para guardar el estado de las comunicaciones y indicar las acciones a realizar desde el gestor de elementos.

- **SipActor**: Clase con los métodos necesarios para crear todos los mensajes SIP que usamos: register, notify, do, ack…

- **MySip**: Extiende de SipActor y implementa la interfície SipListener, por lo que puede procesar peticiones y respuestas en el entorno de SIP. Se apoya en los métodos del propio SipActor para elaborar mensajes.

Pero cada elemento tiene su función, por lo que existen también una serie de elementos diferenciadores en base al tipo de elemento. Los vemos a continuación.

### 4.1.1 Implementación de dispositivos

Los dispositivos son básicamente las cuatro clases comentadas anteriormente. La diferencia principal entre ellos, en cuanto a implementación, radica en la llamada desde la función que procesa las acciones a la clase **Manager**, que posee la capacidad de realizar la acción en función del tipo de elemento con el que estemos trabajando. Como ya hemos comentado, es el único tipo de elemento que en la clase **MySip** tiene una función capaz de procesar mensajes SIP de tipo DO.

Por tanto, para crear un dispositivo solo necesitamos indicar al arrancar el proceso, mediante el paso de parámetros, el nombre, tipo de elemento de que se trata y sus coordenadas. Las clases identificarán esos datos y actuaran como el elemento que se le haya indicado que debe ser.

```
public static void main(String[] args) {
        Red r = new Red();

        […]
        int posY = Integer.parseInt(args[3]);
        r.iniciaCliente(nombre, tipo, posX, posY);

    }
```

### 4.1.2 Implementación de herramientas de monitorización

En este caso, además de las clases que conforman el UA, contamos también con una interfaz gráfica y una API que actúa como puente o enlace entre ambos módulos. Veamos que clases contiene cada uno de ellos.

#### *4.1.2.1 Módulo de Red*

Contiene las clases anteriormente descritas y contiene una función en la clase **SipActor** que le permite generar mensajes DO. Además, al recibir mensajes de tipo NOTIFY, recurrirá a la clase **CreaXML** para, tras leer el campo de datos de este mensaje, crear el fichero XML mediante SAX[1].

#### *4.1.2.2 Módulo de Interfaz Gráfica*

Consta de cuatro clases principales, basadas en SWING[2]:

- **PanelMapa** que es un plano sobre el cual de representan los iconos que representan a los elementos.
- **PanelTree** que es la lista de elementos activos.

[1] Simple Api for XML. Consultar los ACRÓNIMOS.
[2] Biblioteca gráfica para Java.

- **PanelConsola** que contiene una consola de información y los botones de *Acción, Estado* y *Refrescar* (mensajes DO, INFO y REFRESH, respectivamente).

- **Ventana**, que es el JFrame que agrupa a las clases anteriores en una ventana.

#### *4.1.2.3 Módulo de Gestión*

Para realizar su labor cuenta con las siguientes clases:

- **Manager**: Es la clase principal. Puede acceder a las clases de los otros dos módulos y es instanciado por ellas para que puedan acceder a algunas de sus funciones.

- **Elemento**: Clase que a partir de parámetros básicos como un nombre, unas coordenadas y el tipo, generan un nuevo tipo de datos Elemento, que es con el que trabajan las listas del gestor.

- **ListaElementos**: Contiene las funciones de añadir y quitar elementos en un listado de tipo Vector.

- **LeeXML**: Interpreta el archivo XML para que posteriormente, con la ayuda de las clases anteriores, genere un elemento que será añadido al listado. Todo ello a partir de la información contenida en el fichero. Emplea un parser SAX.

- **XMLHandler**: Clase requerida por LeeXML que contiene funciones para parsear el archivo XML de manera personalizada para nuestro caso.

### 4.1.3 Implementación del SER

Para poder establecer la comunicación entre usuarios se ha usado el Proxy SIP IPTEL. Este es un servidor SIP de código libre, gratuito, que acepta el envío de los mensajes usados para este proyecto. Esta disponible para sistemas Unix/Linux.

En este caso, usamos IPTEL como proxy registrar, servidor de localización, acción y de re direccionamiento. Incorpora un fichero de configuración en el cual podemos definir una serie de reglas en función de las funcionalidades que queramos darle. Para este trabajo, se ha tenido que añadir una regla para que acepte los mensajes de tipo DO. Por el contrario, para el desarrollo de este proyecto no se implementa la autenticación.

### 4.1.4 Implementación del Agente de Presencia (PA)

Como hemos comentado anteriormente, también para el Agente de Presencia la librería JAIN SIP permite que trabaje como un UA. Aún y así, este UA es diferente, ya que su cometido es diferente que el del resto.

El PA está formado por las mismas clases que un dispositivo, aunque la implementación cambia, de manera que al recibir un mensaje de tipo SUBSCRIBE, el PA comprueba las cabeceras y almacena los datos del elemento suscrito para poder generar posteriormente los mensajes de notificación necesarios.

## 4.2 Herramientas de implementación

Para poder realizar la implementación de los módulos previamente descritos hemos hecho uso de una serie de herramientas, que son las siguientes:

- **Java Runtime Environment**: Entorno de ejecución Java desarrollado por Sun, en su versión 6 (Update 1). Gratuito. Disponible en http://www.java.com/es/download/manual.jsp

- **Eclipse SDK**: Versión 3.2.1 del entorno de desarrollo gratuito. Permite el desarrollo, compilación y ejecución de proyectos en distintos lenguajes, entre los que se encuentra Java. http://www.eclipse.org

- **JAIN SIP API**: API[1] para la implementación de módulos SIP. https://jain-sip.dev.java.net/

- **LOG4J**: LOG for Java. API de *logging* de Apache. http://logging.apache.org/log4j/docs/manual.html

- **Microsoft Office Word 2007**: Editor de texto desarrollado por Microsoft. http://www.microsoft.com/latam/office/preview/programs/word/overview.mspx

- **Microsoft VISIO 2007**: Editor gráfico destinado a la creación de esquemas y diagramas. http://office.microsoft.com/es-es/visio/HA101656403082.aspx

[1] Application Programming Interface. Consultar los ACRÓNIMOS.

# CAPÍTULO 5. PLANIFICACIÓN

Este apartado muestra la división de tareas tal y como se planificó para este TFC, así como el desarrollo temporal que han tenido. Cada una de las actividades que conforman este trabajo, han sido divididas de la siguiente manera:

**Estudio previo**. Esta tarea engloba el estudio de diferentes tecnologías y la implementación de pequeños ejemplos para desarrollar en el proyecto.

**Diseño**. Esta tarea especifica el diseño de la implementación en el proyecto como puede ser la creación de gráficos UML, diseño de arquitecturas o navegación del usuario por la aplicación.

**Testeo**. Esta se refiere a toda explotación y pruebas realizadas en la aplicación para la comprobación de un correcto funcionamiento.

**Documentación.** Es la redacción de la memoria y otros documentos realizados.

## 5.1 Tiempo de dedicación

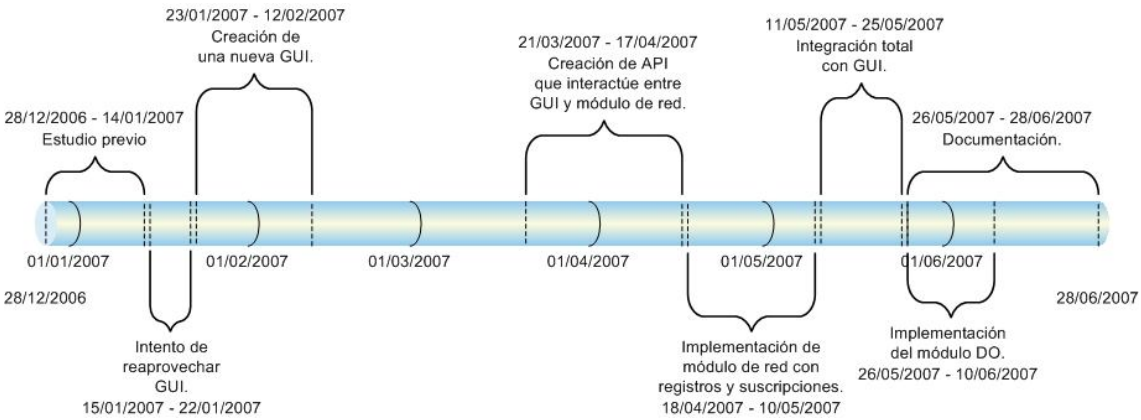


**Fig. 5.1 Tiempo de dedicación.**

Tal y como vemos en el esquema, y pese a existir un período de inactividad en el desarrollo del proyecto, el tiempo de trabajo total es el recomendado de un TFC.

Otro punto a destacar del gráfico es que he podido elaborar un planning organizado, en el cual elaborar la siguiente tarea implica el haber terminado la anterior. Cabe decir también que, aparte de los clásicos puntos en los que uno se suele quedar encallado, no he tenido demasiados problemas al integrar los prototipos o módulos que conforman el trabajo.

El tiempo empleado es menor del recomendado (alrededor de 500 horas) debido, entre otras cosas, a que me he visto forzado a elaborar la memoria en mucho menos tiempo del que me hubiese gustado.

## 5.2 Tareas realizadas

La siguiente tabla muestra las tareas realizadas a lo largo de todo el trabajo, clasificadas por tipo y con el tiempo de dedicación de cada una.

**Tabla 5.1 Tareas Realizadas**

| Tarea | Descripción | Tiempo invertido |
|---|---|---|
| Estudio previo | Estudio inicial de los objetivos, planificación, recursos y conocimientos a adquirir. | 20 horas |
| Implementación | Intento de reaprovechamiento de la interfaz gráfica de un proyecto anterior. Sin éxito debido a la cantidad de referencias cruzadas del código. | 23 horas |
| Estudio previo | Búsqueda de documentación de SWING. | 12 horas |
| Implementación | Creación de una nueva interfaz gráfica. | 54 horas |
| Estudio previo | Búsqueda de información de almacenamiento de datos en listas. | 4 horas |
| Implementación | Creación del módulo de gestión principal del proyecto, que actúa como API entre GUI y módulo de red. | 72 horas |
| Estudio previo | Búsqueda de documentación del intercambio de mensajes SIP. | 24 horas |
| Diseño | Diseño de esquemas de unión de todos los elementos del proyecto. | 12 horas. |
| Implementación | Creación de un escenario de intercambio de mensajes de registro y suscripción a eventos de presencia. | 38 horas |
| Estudio previo | Búsqueda de documentación sobre XML. | 10 horas. |
| Implementación | Unión de la API de gestión con el escenario de red basándose en el intercambio de XML. | 67 horas. |
| Implementación | Acabar de integrar con la GUI propiamente dicha. | 23 horas. |
| Estudio previo | Búsqueda de información sobre el mensaje DO. | 12 horas. |
| Implementación | Implementación del modulo DO. | 22 horas. |
| Implementación | Último ajustes en el SER. | 4 horas |
| Testeo | Testeo final. | 20 horas |
| Documentación | Realización de la memoria. | 53 horas |
| **TOTAL** | | **472** |

# CAPÍTULO 6. CONCLUSIONES

## 6.1 Objetivos alcanzados

Ya antes de comenzar este trabajo era de prever que supondría un reto. El hecho de que la espina dorsal del trabajo sea un protocolo tan reciente como SIP ya daba a suponer que existiría poca documentación al respecto. Ello reducía los recursos con los que contar a la hora de empezar a trabajar.

El objetivo principal era el de crear un entorno de gestión domótica basado en el reconocimiento y la interacción con sus elementos basándose en el protocolo SIP.

Para llevarlo a cabo, las herramientas empleadas han sido el entorno de desarrollo Eclipse y la tecnología Java. Las herramientas, pues, no han tenido coste alguno al ser totalmente gratuitas.

Como además de las funcionalidades básicas de SIP hemos necesitado otras adicionales, como el control de presencia, hemos hecho uso de la API JAIN SIP, que nos ha permitido integrar todas las funcionalidades requeridas.

Otra de las funcionalidades que se ha alcanzado es la de la acción directa sobre el elemento final, basándonos en otros módulos que también se han realizado con Java.

## 6.2 Impacto medioambiental

La gestión informática de las tareas puede convertirlas en algo totalmente automatizado y desasistido por parte del usuario. Pero lo que es más importante, aunque no salte tanto a la vista: puede convertirlas en tareas optimizadas al máximo en lo que al uso de energía se refiere.

Si a este proyecto le añadimos un entorno inteligente de autogestión de cada uno de los elementos, es evidente que la máxima de sólo tener encendidos los aparatos que estamos utilizando se puede cumplir al cien por cien. Si las luces de una estancia vacía se apagan automáticamente, el climatizador se autorregula antes de nuestra llegada a casa, trabajando en conjunto con los sistemas de apertura y cierre de ventanas y persianas, conseguiremos unos resultados óptimos con un coste energético significativamente inferior.

Aún es muy pronto para poder dar datos concretos sobre la eficiencia energética que supone un entorno basado en SIP, pero nos podemos hacer una ligera idea tomando como base una casa domótica estándar, donde se reduce el consumo energético entre un 30 y un 50%.

La conclusión pues, es lógica: cuanta menos energía consumamos, menos gases debidos a su generación y/o consumo emitiremos a la atmósfera. Por tanto, este proyecto es una idea a seguir desde el punto de vista del ahorro energético y la conservación medioambiental.

## 6.3 Conclusiones personales

La realización de este trabajo me ha aportado nuevas experiencias en la elaboración de proyectos. Al ser uno de los proyectos de mayor envergadura que he tenido que realizar, la planificación se torna el punto clave en el desarrollo del mismo. Estar preparado para todo tipo de imprevistos es algo más que recomendable.

Desde el punto de vista formativo, también me ha aportado una mayor profundidad en el conocimiento de la tecnología Java y una buena primera impresión de las posibilidades reales y la capacidad de adaptación de SIP a todo tipo de entornos.

Pero lo que más destaco en la realización de este TFC es la sensación que he tenido durante todos estos meses de estar realizando un trabajo que desborda utilidades prácticas. Ha estimulado mi imaginación a la hora de buscar el modo de aplicar las funcionalidades requeridas. He visto claramente su utilidad en la vida cotidiana de cualquiera de nosotros en cada paso que realizaba y eso me ha motivado mucho a seguir avanzando.

## 6.4 Trabajos futuros

Esta aplicación podría ser la base de algunos futuros trabajos a mi parecer, interesantes.

Un primer aspecto a trabajar que se ha quedado en el tintero en este trabajo por una inesperada falta de tiempo, sería el de ahondar más en la capacidad de actuación sobre los elementos. Así podríamos obtener ventajas realmente vistosas de la aplicación. Por ejemplo, un sistema de actuación más complejo en el que los elementos pudiesen hacer más acciones que un cambio de estado. Una sugerencia sería que el mensaje DO sirviese para interrogar a los elementos sobre las acciones que pueden realizar y, una vez hecho esto, decidir cómo actuar. Combinando esto con un sistema de indicación de eventos programable, tendría un resultado final más espectacular de cara al público.

Otro aspecto a trabajar, como ya he comentado en otros apartados, sería el de integrarlo con un sistema inteligente que permitiese que un agente informático llevara a cabo una serie de tareas programadas y programables por parte del usuario. Desde combinar eventos como el encendido de la calefacción y el cierre de ventanas a cualquier cosa que se nos pueda pasar por la imaginación.

# BIBLIOGRAFÍA

[1] **SIP:** RFC 3261. Session Initiation Protocol, documentación de referencia. (En línea). [Última Consulta: 15 de abril de 2007]. URL: http://www.ietf.org/rfc/rfc3261.txt

[2] **SIP Specific Event Notification:** RFC 3265. Extensión de SIP que define la manera como los nodos SIP demandan notificaciones a nodos remotos para saber el estado de ciertos eventos. (En línea). [Última Consulta: 21 de abril de 2007]. URL: http://www.ietf.org/rfc/rfc3265.txt

[3] **JAIN SIP:** Web oficial de JAIN SIP, información de la librería, código fuente, descarga de ficheros binarios. (En línea). [Última Consulta:3 de abril de 2007]. URL: https://jain-sip.dev.java.net/

[4] **Log4j:** Introducción a la API de *logging* de Apache. Tutorial, ejemplos... (En línea). [Última Consulta: 19 de mayo de 2007]. URL: http://logging.apache.org/log4j/docs/manual.html

[5] **Home SIP:** Proyecto que tiene como objetivo controlar un entorno domótico mediante el uso del protocolo SIP. (En línea). [Última Consulta: 19 de junio de 2007]. URL: http://www.enseirb.fr/cosynux/HomeSIP/

[6] **IPTEL:** Proxy SIP de código abierto. (En línea). [Última Consulta: 18 de mayo de 2007]. URL: http://www.iptel.org/

[7] **BREKEKE:** Proxy SIP de código abierto, usado para pruebas paralelas al uso de IPTEL. (En línea). [Última Consulta: 10 de mayo de 2007]. URL: http://www.brekeke.com/products/products_sip_2.php

[8] **WIKIPEDIA:** *Wikipedia, la enciclopedia libre*. Enciclopedia web. (En línea) [Última Consulta: 23 de junio de 2007]. URL: http://en.wikipedia.org/wiki/Main_Page

# ACRÓNIMOS

**API**

**A**pplication **P**rogramming **I**nterface. És un conjunto de especificaciones para la comunicación entre componentes de software.

**GUI**

Graphic User Interface. Entorno gráfico mediante el cual un usuario interacciona con una aplicación.

**IETF**

**I**nternet **E**ngineering **T**ask **F**orce. Organización abierta que desarrolla y promueve estándares de Internet.

**IP**

**I**nternet **P**rotocol. Es un protocolo de la capa de red que se usa para comunicar datos a través de una red de paquetes. IP aporta un servicio de direcciones únicas y globales para la comunicación entre equipos.

**MIME**

Multipurpose Internet Mail Extensions. Convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos de forma transparente al usuario. RFC 2045-2049.

**MMUSIC**

Multiparty Multimedia Session Control. Grupo de trabajo de IETF especializado en el desarrollo de sesiones de control multimedia.

**NAT**

**N**etwork **A**ddress **T**ranslation. Dispositivo encargado de reescribir las direcciones origen y/o destino de los paquetes IP que pasan a través de un router o firewall a fin de convertirlas en direcciones IP privadas.

**PA**

**P**resence **A**gent. Elemento SIP encargado de gestionar la suscripción y notificación de eventos de los User Agent SIP.

**RFC**

**R**equest **F**or **C**omments. Son documentos con una serie de memorandums que incluyen noves investigaciones, innovaciones y metodologías aplicables a tecnologías de Internet.

**SAX**

Simple Api for XML. API de Java para la creación/edición de archivos XML.

**SIP**

**S**ession **I**nitiation **P**rotocol. Es un protocolo de la capa de aplicación (señalización) para crear, modificar y acabar sesiones con uno o más participantes. Estas sesiones incluyen llamadas para Internet, distribución multimedia i conferencias multimedia.

**TCP**

**T**ransmission **C**ontrol **P**rotocol. Es un protocolo de la capa de transporte que se utiliza para establecer conexiones entre equipos de la misma red. Estos equipos pueden transmitir flujos de datos mediante sockets.

**UA**

**U**ser **A**gent. En VoIP, una aplicación cliente en un sistema SIP se llama SIP user agent. Un UA es la combinación de un UAC y un UAS. El SIP user agent permite llamadas peer-to-peer sirviéndose de un protocolo cliente-servidor.

**UAC**

**U**ser **A**gent **C**lient. En SIP el user agent cliente inicia una SIP request que envía hacia al UAS.

**UAS**

**U**ser **A**gent **S**erver. En SIP el user agent server acepta las SIP *requests* de un UAC y genera una respuesta *accept*, *reject* o *redirect* del usuario.

**VoIP**

**V**oice **o**ver **IP**. Es el encaminamiento de conversaciones de voz sobre Internet o sobre cualquier otra red basada en IP.

**XML**

e**X**tensible **M**arkup **L**anguage. Lenguaje de marcado ampliable o extensible. Desarrollado por el World Wide Web Consortium (W3C).