ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: DESARROLLO DE UN ENTORNO INTELIGENTE EN EL CONTEXTO DEL HOGAR DIGITAL

AUTOR: Jorge Gironda Domínguez

TITULACIÓN: Grado en Ingeniería Telemática

TUTOR (o Director en su caso): Rubén de Diego Martínez

DEPARTAMENTO: Departamento de Ingeniería y Arquitecturas Telemáticas

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Eduardo Barrera López de Turiso

VOCAL: Rubén de Diego Martínez

SECRETARIO: Gregorio Rubio Cifuentes

Fecha de lectura:

Calificación:

El Secretario,

AGRADECIMIENTOS

Me gustaría mostrar mis agradecimientos a un conjunto de personas que han sido esenciales en la consecución de mis estudios, tanto por su apoyo moral como por el esfuerzo realizado para que pudiese estudiar esta titulación.

En primer lugar, por supuesto, agradecer a mis padres su gran esfuerzo, sin el cual me hubiese sido imposible cursar estos estudios. A mi madre, por ser mi principal valedora y un soporte irremplazable. A mi padre, por su punto de vista pragmático, una gran referencia a la hora de afrontar nuevos retos. Su confianza durante los buenos, pero sobre todo durante los malos momentos ha sido un respaldo inestimable, un gesto que nunca olvidaré.

A mi hermano, cuyos ánimos en los peores momentos me dieron fuerzas para levantarme y continuar; sin duda una de las personas que más han confiado en mis capacidades.

A Ana, mi pareja, amiga y principal apoyo durante todos los años dedicados a esta carrera. Por animarme y motivarme, por confiar incondicionalmente en mí, por compartir mis éxitos y mis fracasos y por hacer que todos estos años de duro trabajo hayan sido, a su vez, mis mejores años.

A mis compañeros y amigos de la universidad, que no solo han proporcionado una ayuda académica esencial en determinados momentos, sino que además han convertido muchos de esos largos días de laboratorios y biblioteca en momentos inolvidables que atesorar. Especial mención para mi grupo más cercano, con los que he afrontado mano a mano cada nuevo reto: Jorge (Busi), Iván, Patrycja y Dani.

Por último, agradecer a Rubén de Diego, mi tutor, sus ideas y recomendaciones para que este Proyecto Fin de Grado se haya convertido en una digna representación del conjunto de mis estudios.

RESUMEN

A lo largo de las últimas décadas el desarrollo de la tecnología en muy distintas áreas ha sido vertiginoso. Su propagación a todos los aspectos de nuestro día a día parece casi inevitable y la electrónica de consumo ha invadido nuestros hogares.

No obstante, parece que la domótica no ha alcanzado el grado de integración que cabía esperar hace apenas una década. Es cierto que los dispositivos autónomos y con un cierto grado de inteligencia están abriéndose paso de manera independiente, pero el hogar digital, como sistema capaz de abarcar y automatizar grandes conjuntos de elementos de una vivienda (gestión energética, seguridad, bienestar, etc.) no ha conseguido extenderse al hogar medio.

Esta falta de integración no se debe a la ausencia de tecnología, ni mucho menos, y numerosos son los estudios y proyectos surgidos en esta dirección. Sin embargo, no ha sido hasta hace unos pocos años que las instituciones y grandes compañías han comenzado a prestar verdadero interés en este campo. Parece que estamos a punto de experimentar un nuevo cambio en nuestra forma de vida, concretamente en la manera en la que interactuamos con nuestro hogar y las comodidades e información que este nos puede proporcionar.

En esa corriente se desarrolla este Proyecto Fin de Grado, con el objetivo de aportar un nuevo enfoque a la manera de integrar los diferentes dispositivos del hogar digital con la inteligencia artificial y, lo que es más importante, al modo en el que el usuario interactúa con su vivienda.

Más concretamente, se pretende desarrollar un sistema capaz de tomar decisiones acordes al contexto y a las preferencias del usuario. A través de la utilización de diferentes tecnologías se dotará al hogar digital de cierta autonomía a la hora de decidir qué acciones debe llevar a cabo sobre los dispositivos que contiene, todo ello mediante la interpretación de órdenes procedentes del usuario (expresadas de manera coloquial) y el estudio del contexto que envuelve al instante de ejecución.

Para la interacción entre el usuario y el hogar digital se desarrollará una aplicación móvil mediante la cual podrá expresar (de manera conversacional) las órdenes que quiera dar al sistema, el cual intervendrá en la conversación y llevará a cabo las acciones oportunas.

Para todo ello, el sistema hará principalmente uso de ontologías, análisis semántico, redes bayesianas, UPnP y Android. Se combinará información procedente del usuario, de los sensores y de fuentes externas para determinar, a través de las citadas tecnologías, cuál es la operación que debe realizarse para satisfacer las necesidades del usuario.

En definitiva, el objetivo final de este proyecto es diseñar e implementar un sistema innovador que se salga de la corriente actual de interacción mediante botones, menús y formularios a los que estamos tan acostumbrados, y que permita al usuario, en cierto modo, hablar con su vivienda y expresarle sus necesidades, haciendo a la tecnología un poco más transparente y cercana y aproximándonos un poco más a ese concepto de hogar inteligente que imaginábamos a finales del siglo XX.

ABSTRACT

Over the last decades the development of technology in very different areas has happened incredibly fast. Its propagation to all aspects of our daily activities seems to be inevitable and the electronic devices have invaded our homes.

Nevertheless, home automation has not reached the integration point that it was supposed to just a few decades ago. It is true that some autonomic and relatively intelligent devices are emerging, but the digital home as a system able to control a large set of elements from a house (energy management, security, welfare, etc.) is not present yet in the average home.

That lack of integration is not due to the absence of technology and, in fact, there are a lot of investigations and projects focused on this field. However, the institutions and big companies have not shown enough interest in home automation until just a few years ago. It seems that, finally, we are about to experiment another change in our lifestyle and how we interact with our home and the information and facilities it can provide.

This Final Degree Project is developed as part of this trend, with the goal of providing a new approach to the way the system could integrate the home devices with the artificial intelligence and, mainly, to the way the user interacts with his house.

More specifically, this project aims to develop a system able to make decisions, taking into account the context and the user preferences. Through the use of several technologies and approaches, the system will be able to decide which actions it should perform based on the order interpretation (expressed colloquially) and the context analysis.

A mobile application will be developed to enable the user-home interaction. The user will be able to express his orders colloquially though out a conversational mode, and the system will also participate in the conversation, performing the required actions.

For providing all this features, the system will mainly use ontologies, semantic analysis, Bayesian networks, UPnP and Android. Information from the user, the sensors and external sources will be combined to determine, through the use of these technologies, which is the operation that the system should perform to meet the needs of the user.

In short, the final goal of this project is to design and implement an innovative system, away from the current trend of buttons, menus and forms. In a way, the user will be able to talk to his home and express his needs, experiencing a technology closer to the people and getting a little closer to that concept of digital home that we imagined in the late twentieth century.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

B

BD: Base de Datos, 86, 87, 88, 89, 103, 109, 118
BN: Bayesian Network, 31

F

FOL: First-Order Logic, 31

G

GUI: Graphical User Interface, 3, 4, 43, 47, 49, 56, 57, 118, 120, 121, 129, 130, 131

H

HBN: Hybrid Bayesian Network, 31
HTTP: Hypertext Transfer Protocol, 21

I

IA: Inteligencia Artificial, 4, 14, 16, 18, 68, 70, 80, 81, 120, 146, 147, 171
ID: Influence Diagram, 31, 105, 106, 107, 113, 114, 123, 133, 157, 159, 160, 161, 162, 163, 164
IEEE: Institute of Electrical and Electronics Engineers, 21, 47
IP: Internet Protocol, 21, 22, 47, 54, 55

M

MEBN: Multy-Entity Bayesian Network, 31
MSBN: Multiply Sectioned Bayesian Network, 31

O

OKBC: Open Knowledge Base Connectivity, 18, 19
OOBN: Object-Oriented Bayesian Network, 31
OSI: Open System Interconnection, 54
OWL: Web Ontology Language, 17, 18, 19, 31, 75

P

PC: Personal Computer, 47

R

RDBMS: Relational DataBase Management System, 38
RDF: Resource Description Framework, 17, 18

S

SCAH: Semantic and Context Aware Home, 43, 44, 45, 169
SO: Sistema Operativo, 4, 25, 26, 34, 35, 36, 47, 52, 154, 167, 169

T

TCP: Transmission Control Protocol, 21, 54

U

UDP: User Datagram Protocol, 21, 54
UPnP: Universal Plug and Play, iii, v, 3, 4, 21, 22, 23, 24, 25, 26, 54, 55, 56, 57, 58, 60, 122, 124, 132, 135, 138, 139, 142, 167, 168

X

XML: eXtensible Markup Language, 17, 18, 21, 23, 24, 47, 96

Capítulo 1

Contextualización y descripción de la memoria

1.1 Objetivos del Proyecto Fin de Grado

El Proyecto Fin de Grado que se describe en esta memoria se centra en el diseño e implementación de un sistema inteligente capaz de controlar diferentes dispositivos en el ámbito domótico.

Más concretamente, se pretende desarrollar un sistema capacitado para tomar decisiones acordes al contexto y a las preferencias del usuario. Mediante el uso de diferentes tecnologías se va a dotar al hogar digital de cierta autonomía a la hora de decidir qué acciones debe llevar a cabo sobre los dispositivos que contiene, todo ello mediante la interpretación de órdenes procedentes del usuario y el estudio del contexto que envuelve el instante de ejecución.

El contexto al que se hace referencia, en lo que a este sistema se refiere, estará compuesto principalmente por el estado de la red de dispositivos (lo que implica el estado individual de cada uno de ellos), la situación climatológica, la ubicación del usuario, las preferencias y el momento de ejecución.

Como nexo entre el usuario y el sistema, se desarrollará una aplicación móvil que funcionará como interfaz gráfica de usuario (GUI) y permitirá el acceso a las funcionalidades que el sistema pueda proporcionar.

Para todo ello, el sistema hará principalmente uso de ontologías, redes bayesianas, UPnP y Android. Se combinará información procedente del usuario, de los sensores y de fuentes externas para determinar, a través de las citadas tecnologías, cuál es la operación que debe realizarse para satisfacer las necesidades del usuario.

A continuación se definen los objetivos concretos, así como los objetivos principales planteados para el desarrollo de este Proyecto Fin de Grado.

1.1.1 Objetivos concretos

Se listan a continuación los objetivos concretos que deben haber sido alcanzados al finalizar el Proyecto Fin de Grado que se presenta en esta memoria:

- Creación de un subsistema independiente capaz de tomar decisiones (selección y ejecución de acciones sobre dispositivos) a partir de la interpretación de las órdenes del usuario y el análisis de la situación contextual que envuelve al instante de ejecución.

  Este subsistema contendrá la inteligencia artificial del conjunto del sistema, y todo el peso del procesado de órdenes y toma de decisiones debe recaer sobre él.

  Para soportar esta funcionalidad, este subsistema deberá hacer uso de:

  - Ontologías: estructuración de conocimientos, tanto de la información de los dispositivos como de aquella referente a las órdenes provenientes del usuario.
  - Redes bayesianas: estudio probabilístico de los factores contextuales que puedan tener influencia en la toma de decisiones.

  - Configuración de usuario: capacidad de adaptar la toma de decisiones a las preferencias del usuario.
  - Cualquier otro mecanismo necesario para interpretar la orden que indique el usuario.

- Creación de una red simulada de dispositivos que independice el desarrollo de la IA de la tecnología subyacente. Este subsistema, que también será independiente, debe ser capaz de ejecutar acciones sobre los dispositivos y resultar transparente para el resto del sistema (como si de una red real de dispositivos se tratase).

  A lo largo de la memoria se hará referencia, además, a cómo podría ser trasladado el sistema a un entorno real como el presente en el hogar digital con el que cuenta la Escuela (red de dispositivos LonWorks), si bien esta opción no se implementará al disponer de la posibilidad de simular la red real.

- Creación de una aplicación para el SO Android capaz de llevar a cabo todas las funciones que el sistema pueda proporcionar al usuario. Esta aplicación debe hacer la función de GUI exclusivamente, intercambiando la información necesaria con el núcleo del sistema.

  La aplicación, desde el punto de vista del usuario, debe proporcionar cierta ilusión de inteligencia, de manera que dicho usuario interactúe con el sistema como si lo hiciese con un ente inteligente (de manera conversacional).

  Además, la aplicación debe ser capaz de reconocer órdenes habladas, así como de expresar, también de manera hablada, los mensajes que requiera comunicar al usuario.

- Implementación de una capa software que haga uso del conjunto de protocolos UPnP (*Universal Plug and Play*), con lo que se pretende proporcionar un medio de comunicación para el descubrimiento de dispositivos presentes en la red simulada.

  Esta tecnología será utilizada para el intercambio de información entre todos los subsistemas mencionados (incluida la aplicación Android).

1.1.2 Objetivos principales

El diseño e implementación de los elementos anteriormente listados implica la creación de un sistema completo y heterogéneo que abarca desde la propia red de dispositivos hasta la interfaz de usuario, pasando por el núcleo del sistema (que contendrá la IA).

No obstante, se define el desarrollo de la inteligencia artificial como objetivo principal de este Proyecto Fin de Grado, junto con el desarrollo de la aplicación que completará la experiencia del usuario (y que será el punto visible del sistema).

Cada uno de los subsistemas mencionados en el apartado anterior podrían ser objeto de detallados estudios y un gran nivel de desarrollo, pero se considera que el elemento que aporta realmente la componente de innovación a este proyecto es la creación de inteligencia artificial.

Son muchos y variados los estudios que se están llevando a cabo en diferentes partes del mundo e instituciones orientados al desarrollo de sistemas inteligentes, pero no son tantos los

orientados a entornos domóticos (en cuanto a sistemas conversacionales que interpreten órdenes directas del usuario se refiere) y a las incipientes redes de dispositivos del denominado hogar digital.

En este proyecto se pretende presentar un sistema diferente, que a partir de requisitos mínimos pueda llevar a cabo tan compleja tarea, intentando enfocar el diseño de una manera innovadora.

1.2 Estructuración de contenidos.

La memoria que aquí se presenta se ha dividido en 5 capítulos (siendo el primero de ellos el presente) y un anexo. El contenido de cada de una de estas secciones se describe brevemente a continuación:

II. **Capítulo 2. Estado del arte**

En este capítulo se pretende presentar los aspectos principales de cada una de las tecnologías o campos de estudio que han servido como base en el desarrollo de este proyecto.

La idea principal de esta sección es la de proporcionar al lector una pequeña base que le permita comprender las diferentes decisiones de diseño que se han tomado a lo largo del desarrollo e implementación del sistema.

III. **Capítulo 3. Especificaciones y análisis del sistema**

Este capítulo presenta de manera detallada todas las especificaciones, tanto funcionales como no funcionales, que se definen para el desarrollo del sistema. Estos requisitos satisfacen y amplían los objetivos descritos en la introducción, dando una primera visión de la estructura del sistema, así como de su funcionalidad final.

Además, este capítulo plantea algunos requisitos deseables, aunque no obligatorios, que se han pretendido alcanzar a lo largo del desarrollo en pro de la creación de un sistema con un cierto estándar de calidad.

IV. **Capítulo 4. Diseño e implementación**

Este apartado aglutina la mayor parte de la memoria y su objetivo es el de detallar los procesos de diseño que se han llevado a cabo, los razonamientos que los sustentan y el modelo final implementado para cada uno de los componentes del sistema.

Debido a que la gran mayoría de componentes tienen una estrecha relación de dependencia con otros elementos, se ha decidido plantear este capítulo de manera que en primer lugar se presenten los elementos de manera independiente y en detalle y, finalmente, se muestren en conjunto, definiéndose las correspondientes relaciones.

Por último, dentro de este capítulo, se proporciona un ejemplo de secuenciación de tareas que llevaría a cabo el sistema con el objetivo de que el lector pueda ubicar por última vez y en conjunto, el papel de cada uno de los elementos en el proceso global que lleva a cabo el sistema.

V. **Capítulo 5. Conclusiones y trabajos futuros**

Este apartado presenta, como último capítulo de la memoria, el conjunto de conclusiones que se han obtenido tras la finalización del Proyecto Fin de Grado, repasando los principales logros, así como los puntos débiles del sistema.

Adicionalmente, se definirán un conjunto de trabajos futuros que podrían mejorar los diferentes aspectos del sistema desarrollado, y se expondrán brevemente las medidas que se han tomado en el diseño para flexibilizarlo en vista a futuros desarrollos.

VI. **Anexo I**. **Manual de usuario**

Por último, cerrando la memoria, se proporciona un breve manual de usuario que tiene como objetivo especificar tanto los requisitos (en equipos y software) para poder hacer uso del sistema, como los pasos que deben seguirse en su instalación y ejecución.

Entre la información proporcionada se hace énfasis en la configuración de usuario (cómo hacer uso de los medios proporcionados para personalizar el sistema) y en el uso de la aplicación Android, explicando cada una de las funciones que soporta.

Además de los apartados mencionados, la memoria contiene de manera estructurada los índices correspondientes a figuras, tablas y ecuaciones (además del índice global de contenidos).

Existe también un índice de acrónimos que aglutina todas las referencias a los significados de los diferentes acrónimos utilizados a lo largo del documento.

Por su parte, todas las referencias bibliográficas se concentrarán en la parte final de la memoria, tras el Anexo I.

Capítulo 2

Estado del arte

En este capítulo se procurará dar una breve visión de los fundamentos de cada una de las tecnologías de las que se hará uso en el sistema con el objetivo de que el lector pueda comprender los diferentes capítulos que componen esta memoria.

La descripción de cada tecnología o campo de estudio no irá más allá de lo necesario para fundamentar el diseño e implementación de este proyecto y describir el marco de desarrollo. Para obtener una información más detallada, el lector puede acudir a las referencias bibliográficas que se proponen.

2.1 Domótica

El término domótica es ampliamente utilizado en la actualidad para hacer referencia a casi cualquier automatismo aplicado en un entorno doméstico.

No obstante, tal como se expone en un extendido libro sobre domótica e inmótica [1] , el término "domótica" proviene de la unión de la palabra "domo" (del latín *domus* que significa casa) y el sufijo "tica" (que proviene de la palabra "automática"). El término completo tiene como origen la palabra francesa *domotique*, definida por primera vez en la enciclopedia Larousse como *"el concepto de vivienda que integra todos los automatismos en materia de seguridad, gestión de la energía, comunicaciones, etc."*.

El CEDOM (Asociación Española de Domótica e Inmótica) [2] define la domótica de la siguiente manera: *"La domótica es el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, que aporta seguridad y confort, además de comunicación entre el usuario y el sistema."*.

De este breve repaso etimológico y de sus definiciones modernas, se puede deducir que el término "domótica" requiere no sólo de un dispositivo independiente con un cierto grado de automatización, sino de un sistema completo que sea capaz de gestionar gran cantidad de dispositivos para administrar diferentes conjuntos de servicios.

En lo que al estado de implantación se refiere, y tal como se ha introducido en el resumen de este documento, la domótica no ha alcanzado el grado de integración que se podía esperar unas pocas décadas atrás. La tecnología está sobradamente desarrollada y, de hecho, la población media de los países del primer mundo utiliza a diario dispositivos que, probablemente, superan con creces los requisitos tecnológicos que requeriría un hogar digital básico. Los motivos de su tardía expansión son muy diversos, pero se exponen a continuación aquellos que se consideran más relevantes:

- **Falta de participación de las instituciones en el proceso de actualización de infraestructuras**

  Para que un hogar digital pudiese llevar a cabo eficientemente algunas de las tareas que se le encomienda según el término "domótica" (por ejemplo, la gestión energética) el control autónomo e inteligente de los dispositivos del hogar es un paso necesario, pero sin una infraestructura eléctrica moderna, capaz de compartir información actualizada de la red, es posible que el factor diferencial que proporcione el hogar digital sea tan leve que afrontar la instalación y coste inicial de un sistema de

estas características no resulte rentable al usuario medio. Esta misma conclusión podría obtenerse de otros recursos como el agua o el gas.

Los gobiernos afirmaban durante la primera década del siglo que la domótica era el futuro de la gestión energética [3], opinión que aquí se comparte, y clamaban estar dando los primeros pasos mediante la automatización de sistemas de iluminación y calefacción en edificios públicos. Aunque, sin duda, es un paso importante, ese enfoque pertenece al ámbito de la inmótica y lo cierto es que, al margen de grandes estructuras y edificio públicos, la tecnología de automatización que se adapta al concepto de domótica no ha llegado a los hogares de una manera relevante.

- **Falta de interés por parte de las grandes compañías**

  En las últimas décadas y particularmente en lo que llevamos de siglo XXI, el desarrollo de la tecnología en diferentes áreas es, cuanto menos, sorprendente. Hemos visto evolucionar los teléfonos móviles básicos (primitivos, grandes y pesados) hasta los *smartphones* potentes, finos y ligeros que utilizamos hoy día. El concepto de televisión está cambiando, las redes de datos crecen y mejoran, la información de los usuarios es accesible desde la nube en cualquier lugar e incluso en algunas casas ya hay aspiradoras trabajando de manera autónoma.

  Sin duda, las grandes compañías de tecnología han trabajado y obtenido grandes resultados. Sin embargo, no parece que estos esfuerzos se hayan enfocado en el desarrollo de sistemas en el área de la dómotica. En esta coyuntura de evolución tecnológica constante resulta sorprendente el hecho de que las primeras bombillas regulables remotamente comercializadas a gran escala (como la bombilla Hue de Phillips) o los primeros termostatos con cierto grado de aprendizaje (como el desarrollado por Nest) hayan sido noticias sólo a partir de la segunda década del presente siglo. Aun apareciendo en el panorama, sus precios y limitadas funcionalidades no ayudan a que resulte una compra interesante en la mayoría de hogares.

- **Falta de concienciación y productos atractivos para el usuario**

  Si bien existen ciertas deficiencias en la impulsión de la domótica tal como se ha expuesto en los puntos anteriores, esto no implica que no existiesen opciones disponibles. Ha habido empresas, así como variados grupos de investigación, que han centrado sus esfuerzos en el desarrollo de redes de dispositivos más eficientes y flexibles, así como en la creación de estándares que impulsasen la industria e interoperabilidad; pero esta información apenas ha llegado a los usuarios y potenciales compradores.

  Por otra parte, parece ser que los productos finales que finalmente han llegado al usuario no lo han hecho como cabía esperar y no han resultado suficientemente atractivos para justificar el gasto.

Por suerte para el futuro de este campo, parece que la tendencia comienza a cambiar. Grandes compañías empiezan a desarrollar dispositivos y sistemas en este ámbito o bien están absorbiendo a otras más pequeñas que ya lo hacían. Un ejemplo significativo lo encontramos en la adquisición

por parte de Google de Nest y DropCam [4], dos compañías emergentes en el sector de la domótica. Las instituciones públicas empiezan a invertir en tecnologías relacionadas y aunque la domótica, por definición, sólo se podrá considerar realmente extendida cuando la población, de manera individual, comience a integrar estos sistemas en sus viviendas, el fomento de las infraestructuras y servicios telemáticos facilitarán su integración.

Como ejemplo de estas nuevas infraestructuras, existen gran cantidad de proyectos en el ámbito de la denominada Smart Grid [5] (red eléctrica inteligente) y la implantación de los nuevos Smart Meters, para la medida inteligente y actualizada del consumo y el precio de dicha electricidad. Esta evolución puede impulsar enormemente los sistemas domóticos de gestión energética. Por ejemplo, el sistema domótico podría hacer uso de algoritmos de optimización y aprendizaje que tengan en cuenta el precio de la electricidad a lo largo del día para gestionar eficientemente cuándo realizar tareas como cuando encender el lavavajillas o la lavadora.

España en concreto es uno de los países que más está impulsando proyectos de *Smart City* [6], que engloban muchas de las infraestructuras que pueden impulsar el crecimiento de la domótica en los próximos años, además de otros servicios de uso público que podrían reutilizar tecnologías aplicables al entorno domótico, lo que servirá de doble impulso para el sector.

En definitiva, teniendo presente los movimientos de los diferentes organismos públicos y empresas, así como la gran cantidad de investigación en este ámbito, se considera que la domótica, aunque tardía, está en auge y que los avances significativos llegarán en los próximos años, extendiéndose a los hogares de manera paulatina pero constante.

2.1.1 Redes de dispositivos

Un sistema domótico requiere de diferentes conjuntos de elementos para llevar a cabo sus tareas. Resulta obvio que el integrar diversos dispositivos y la información que cada uno de ellos proporciona, junto con la información externa que el sistema pueda requerir, debe hacer uso de algún software que, mediante los mecanismos oportunos, pueda tomar las decisiones más óptimas para mantener ese perfil de automatización y eficiencia que caracteriza al concepto de "domótica".

Sin embargo, lo que resulta aún más básico es que para obtener la información de los dispositivos debe existir una red que les permita compartir información, así como invocar remotamente las acciones que pretenda llevar a cabo el sistema.

Para este propósito existe un gran número de tecnologías, con sus correspondientes protocolos y topologías de red. En este apartado se presentará brevemente una de las más utilizadas en el ámbito domótico por su flexibilidad, amplia extensión y las diferentes herramientas proporcionadas por la empresa desarrolladora: LonWorks, desarrollado por Echelon.

Esta tecnología es a la que se hace mención en diferentes puntos de la memoria, si bien debe tenerse en cuenta que existen varias alternativas, cada una con sus pros y contras, que no se entrarán a explicar aquí por quedar fuera del ámbito de estudio de este proyecto. No obstante, se listan algunas de ellas (aquellas más orientadas a la domótica), proporcionando también algunas referencias.

- Zigbee [7]
    - Similar a Bluetooth, pero optimizado para domótica.
        - Conexión en malla con subredes de hasta 255 nodos (frente a los 8 nodos máximo de una subred Bluetooth).
    - Bajo consumo
    - Poca electrónica (bajo coste)
- KNX [8]
    - Primer sistema domótico aprobado como norma mundial
    - Protocolo abierto
    - Sistema robusto con gran capacidad de recuperación ante fallos
- 6lowPAN [9]
    - Soporte para conectividad mediante IPv6
    - Bajo consumo
    - Requiere de bajos recursos

2.1.1.1 LonWorks

LonWorks es el nombre que recibe una tecnología orientada al control e interconexión de dispositivos desarrollada por Echelon [10]. Esta tecnología, que ahora resulta un estándar abierto ampliamente extendido a lo largo de todo el mundo, se desarrolló con el objetivo de solucionar un problema tradicional de las redes de dispositivos: los nodos de dicha red debían estar diseñados para trabar juntos.

Tal como lo describe Echelon en su sitio web, el estándar LonWorks permite que los nodos trabajen juntos sin ningún acuerdo previo, incluso cuando provienen de diferentes compañías y desarrolladores. Esta tecnología proporciona a los dispositivos un modo de intercambiar información y cooperar entre sí. El protocolo LonWorks reúne los requisitos más estrictos que se definen en el control de redes de dispositivos.

Para llevar a cabo esta tarea, Echelon provee una serie de componentes como transceptores, nodos de control y herramientas software. La tecnología LonWorks no sólo se usa en entornos domóticos, sino que su uso se extienden a otras áreas mencionadas al inicio de este apartado: *Smart Grid, Smart Buildings, Smart Cities*, etc.

Con la creación de una asociación industrial con nombre *LonMark International*, Echelon ha conseguido que LonWorks esté cada vez más presente y la interoperabilidad haya mejorado en los dispositivos que los diferentes fabricantes sacan al mercado. A día de hoy ya existe una gran cantidad de dispositivos bajo el denominado estándar LON, listos para usar en este tipo de redes.

Una de las principales características del protocolo LonWorks es que provee servicios para las siete capas de la pila de protocolos OSI y que, además, el protocolo es abierto para su implementación por cualquier compañía. Se muestra en la siguiente figura esta distribución del

protocolo (conocido en sus orígenes como LonTalk) así como alguna de las principales características de cada una de las capas.

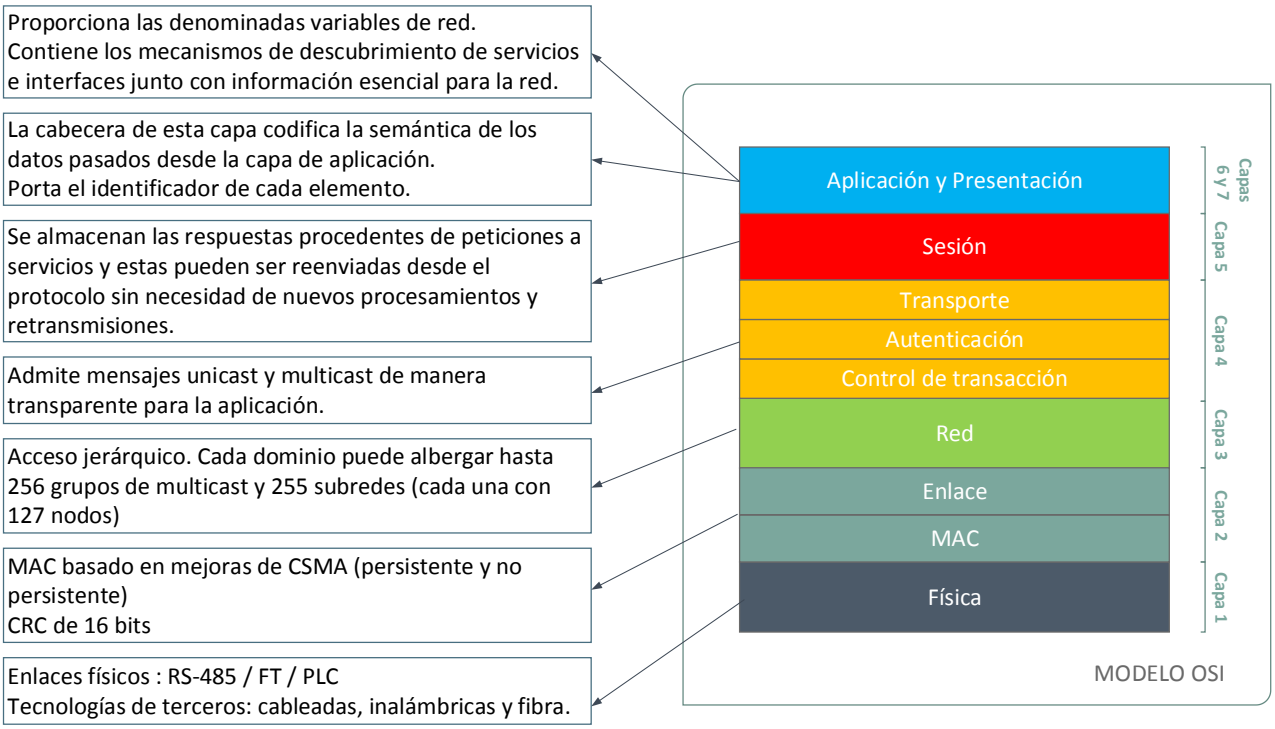


Figura 1. Protocolo LonWorks (LonTalk)

Otro de los atractivos de esta tecnología es la cantidad de herramientas de integración proporcionadas por Echelon, entre las que se pueden destacar:

- **OpenLNS Commissioning Tool**: software para la puesta en marcha, monitorización y mantenimiento de redes LonWorks de control. Esta herramienta cuenta con plug-ins que pueden ampliar sus funcionalidades.
- **LonScanner Protocol analyzer**: software para el análisis, caracterización y muestra de los paquetes de control de la red LonWorks.
- **LonMaker Integration Tool**: software para el diseño, puesta en marcha y mantenimiento de redes LonWorks de control de energía.

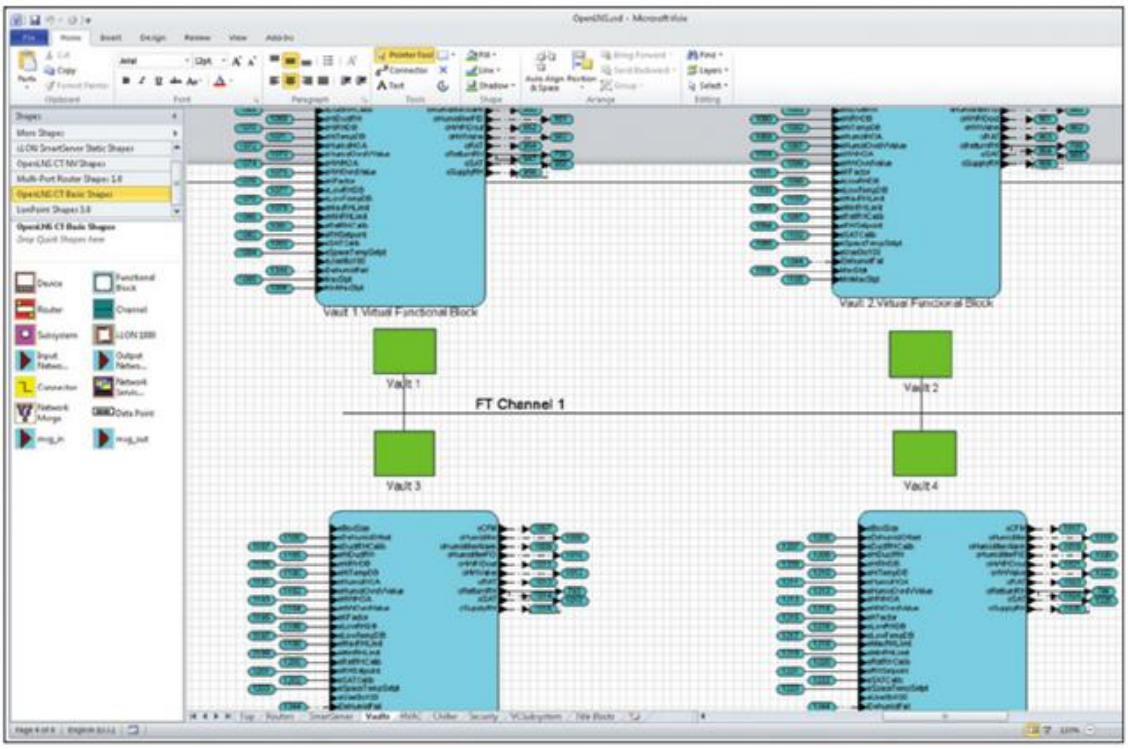


Figura 2. OpenLNS Commissioning Tool

En definitiva, LonWorks es una tecnología muy flexible y en amplio crecimiento, propagándose a cada vez más tipos de dispositivos y captando el interés de nuevos fabricantes. A la hora de plantear la implantación de una red en un entorno domótico, y aunque existen diferentes alternativas, LonWorks es una opción a tener muy en cuenta.

2.2 Inteligencia artificial

Cuando se intenta definir qué es la inteligencia artificial se pueden encontrar en la literatura tantas definiciones como artículos relacionados. Sin embargo, todos ellos apuntan a una serie de características comunes a este campo de estudio.

Para afrontar esta breve introducción se expondrán diferentes definiciones propuestas en un artículo de la página web de la Fundación General CSIC [11] sobre inteligencia artificial:

- La inteligencia artificial (IA) es una de las ramas de la informática, con fuertes raíces en otras áreas como la lógica y las ciencias cognitivas.
- El problema de la IA es construir una máquina capaz de comportarse de manera que si el mismo comportamiento lo realizara un ser humano, éste sería llamado inteligente.
- Rama que persigue la creación de **máquinas que actúen como personas**. El modelo a seguir para la validación de los programas corresponde al comportamiento humano.
- Rama que persigue la creación de **máquinas que razonen como las personas**. Lo importante es cómo se realiza el razonamiento (que debe ser similar a como lo harían

las personas) y no el resultado de este razonamiento. Este enfoque es compartido por la ciencia cognitiva.

- Rama que persigue la creación de **máquinas que razonan racionalmente**. En este caso la definición se focaliza también en el razonamiento, pero aquí se parte de la premisa de que existe una forma racional de razonar, valga la redundancia. La lógica permite la formalización del razonamiento y se utiliza para este objetivo.
- Rama que persigue la creación de **máquinas que actúan racionalmente**. En este enfoque, de nuevo, el objetivo son los resultados (es indiferente como se realiza el cálculo), pero los resultados deben ser evaluados de forma objetiva. Por ejemplo, este sería el enfoque de un programa de juego de ajedrez.

En este artículo, además, se realiza una nueva clasificación de la inteligencia artificial en función del objetivo de la investigación:

- **Inteligencia artificial débil**: se considera que los ordenadores sólo simularán el razonamiento, actuando como si realmente razonasen, pero internamente se cree que no se podrá construir ordenadores conscientes. En esta clasificación los ordenadores sólo simulan un proceso cognitivo, pero no lo llevan a cabo.
- **Inteligencia artificial fuerte**: en este caso se defiende que los ordenadores podrán tener una mente (con sus estados mentales) y que, por tanto, algún día será posible construir uno con todas las capacidades de la mente humana (razonar, imaginar, etc.).

Por último en lo que se refiere a esta breve introducción a la inteligencia artificial, se listan las cuatro características que, según este artículo del Instituto de Investigación en Inteligencia Artificial del CSIC, se consideran más importantes a la hora de implementar un sistema dentro de este campo:

1. **Resolución de problemas y búsqueda**: formalización del problema y búsqueda de soluciones.
2. **Representación del conocimiento y sistemas basados en el conocimiento**: es frecuente entre estos sistemas que requieran de la incorporación de ciertos dominios de conocimiento.
3. **Aprendizaje automático**: el rendimiento de un programa puede incrementarse si el programa aprende de la actividad realizada.
4. **Inteligencia artificial distribuida**: en sus primeros años la inteligencia artificial era monolítica, pero actualmente, con la existencia de los ordenadores multiprocesador e Internet, existe mayor interés en soluciones distribuidas.

   Como añadido a las cuatro características anteriormente mencionadas, se concluye el artículo nombrando algunas de las nuevas tendencias:

   a) El lenguaje natural

   b) La visión artificial

   c) La robótica

d) El reconocimiento del habla

Tal como se observará a lo largo de la memoria y más concretamente en el apartado 4.2, la IA desarrollada en este proyecto afronta cada uno de los cuatro puntos principales expuestos anteriormente, además del uso del lenguaje natural y el reconocimiento del habla.

2.2.1 Inteligencia artificial en la domótica

Es habitual encontrar en el ámbito de tecnología aplicada a una vivienda, el término "domótica" entremezclado con el de "inteligencia artificial".

Bien es cierto que soportando todas las decisiones *inteligentes* o *razonables* que debe tomar un sistema domótico debe existir una inteligencia artificial. Esta IA se encontraría dentro de la definición de *máquinas que actúan racionalmente* visto anteriormente, tomando decisiones óptimas desde un punto de vista objetivo.

No obstante, varias de las definiciones de IA apuntan a que el sistema debe comportarse como un ser humano y adquirir alguna de sus cualidades (razone o no como un humano de manera interna).

Respecto al primer enfoque es cierto que existen muchos dispositivos y sistemas emergentes con una inteligencia artificial latente que les permite aprender de las costumbres del usuario y tomar decisiones, *a priori*, óptimas. En esta línea existen ya sistemas de automatización de iluminación, temperatura, electricidad, etc., aunque como ya se ha mencionado en varias ocasiones, con poca integración en el hogar medio. Estos dispositivos, con una interfaz muy similar a otros sistemas más ordinarios, no satisfacen esta segunda clasificación orientada a la imitación del comportamiento humano.

La razón de esta lenta aceptación podría deberse en parte, por qué no, a la ausencia de sistemas comercializados con este enfoque de imitación humana. Existen en otras áreas como la telefonía móvil sistemas muy innovadores de inteligencia artificial como Siri [12] o Cortana [13] orientados a la asistencia en tareas cotidianas mediante una conversación coloquial. También existen sistemas conversacionales avanzados en Internet como es el caso del denominado "Eugene Goostman" [14], la primera máquina en superar el test de Turing, en el cual una máquina debe hacer creer a un número suficiente de interrogantes que se trata de un ser humano.

En el proyecto que aquí se desarrolla, uno de los principales objetivos definidos es la creación de una IA que procure encontrarse en un punto medio entre ambos enfoques, teniendo en cuenta la acción más eficiente pero, a su vez, proporcionando una interacción algo más humana de la que actualmente se encuentra presente en la mayoría de sistemas domóticos.

2.3 Ontologías

En este apartado, de nuevo, comenzaremos por la definición de ontología.

En un útil artículo desarrollado en la Universidad de Standford con nombre "Guía para crear tu primera ontología" [15] se define una ontología como "*descripción explicita y formal de conceptos en un dominio de discurso (clases, a veces llamadas conceptos), propiedades de cada concepto describiendo varias características y atributos del concepto (slots, a veces llamados roles*

o propiedades), y restricciones sobre los slots (facetas, algunas veces llamados restricciones de rol)."

Para dar una visión algo más concreta de cómo está compuesta una ontología, se describen a continuación los elementos más descriptivos de ésta, apoyándose en el modelo utilizado en el citado documento.

Una ontología, junto con un conjunto de individuos de clases (instancias) constituye lo que se denomina base de conocimiento. Las clases son el centro de la mayoría de las ontologías, las cuales describen conceptos de un dominio. En lo que se refiere a este proyecto, una clase definiría un concepto de un dispositivo, siendo el dominio de conocimiento la información de los dispositivos de la red. Una clase puede tener subclases que representan conceptos que son más específicos.

Los slots describen propiedades de clases e instancias (las instancias son los datos que se guardan en la ontología, respetando la estructura definida por clases, slots y facetas). Los slots de un dispositivo, siguiendo el ejemplo, podrían ser aquellos que representen el estado, la descripción o las acciones. Por su parte, las facetas definen restricciones sobre los slots, como cardinalidad y tipo (clases, primitivos, valores predefinidos y listas). En la misma línea, una faceta de un slot de estado podría limitar el número de estados (máximo y mínimo) a uno, forzando a que este sea la instancia de otra clase que represente dicho estado.

En los últimos años el desarrollo de ontologías ha estado ampliándose a distintos ámbitos y han llegado a ser comunes en el World-Wide Web. Las ontologías Web van desde grandes taxonomías que categorizan sitios Web tales como Yahoo! A categorizaciones de productos para vender y sus características tales como Amazon.com.

Las ontologías, por supuesto, han sido de un gran interés en el desarrollo de la inteligencia artificial y es por ello que en este proyecto se hará uso de ellas y serán el eje en la manipulación de información y conocimientos del sistema.

2.3.1 OWL

Tal como se introducía previamente, el uso de ontologías se ha extendido al World-Wide Web y en este marco se desarrolló OWL (del inglés *Web Ontology Language*), que se ha convertido en un estándar Web (W3C, World-Wide Web Consortium). OWL es una de los lenguajes más extendidos y con más documentación disponible, por lo que en lo que a esta memoria corresponde se resumirán únicamente las características más importantes. Más información sobre OWL puede ser obtenida en su documento de referencia [16].

OWL es un lenguaje de marcado semántico para publicar y compartir ontologías en el World-Wide Web. OWL se ha desarrollado como un vocabulario de extensión de RDF (del inglés *Resource Description Framework)* y está escrito en XML. OWL, junto con RDF y otros componentes hacen posible el proyecto de web semántica [17].

Este lenguaje para describir ontologías se desarrolló para proporcionar un modo común de procesar el contenido de una Web, en lugar de únicamente mostrarlo. En este sentido, OWL se diseñó para ser leído por computadoras, no por humanos.

RDF y OWL son en gran parte lo mismo, pero OWL es un lenguaje más robusto con más facilidad de interpretación (desde sistemas informáticos) y un vocabulario y sintaxis más extensa. Mediante el uso de XML, la información de la ontología puede ser fácilmente intercambiada entre máquinas que utilizan diferentes sistemas operativos y/o lenguajes de aplicación.

OWL está formado por tres sublenguajes:

- OWL Lite: jerarquía de clasificación simple, válida para ontologías sencillas.
- OWL DL: clasificación más compleja y flexible, pero con ciertas restricciones.
- OWL Full: clasificación más compleja, pero con menos restricciones predefinidas y con más libertad de diseño.

En la siguiente figura se puede observar como cada uno de estos lenguajes está contenido en el siguiente, de manera que cada uno de ellos amplía la funcionalidad del anterior.

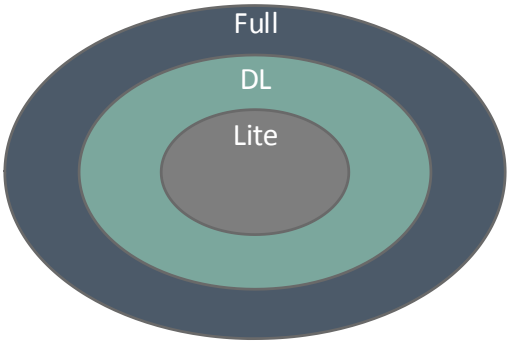


Figura 3. Sublenguajes OWL

Dado que OWL está orientado a la publicación y compartición de información a través de la Web, enfoque que se sale del objetivo de este proyecto, no será éste el lenguaje ontológico utilizado, pero se ha considerado importante su mención para que el lector pueda apreciar como las ontologías están jugando un papel importante en la estructuración de la información en la Web y en los sistemas de IA como el que aquí se pretende desarrollar.

2.3.2 Ontologías basadas en Marcos (*Frame-based ontologies*)

Los modelos basados en marcos (*Frame-based models*) utilizan marcos y sus propiedades como primitivas en el modelado. El concepto de marco corresponde al concepto de clase visto anteriormente. Cada atributo (slot) es aplicable sólo al marco en el que está definido. Las restricciones de valor (facetas) se pueden aplicar a cada atributo por separado.

Una característica importante de las ontologías basadas en marcos es la posibilidad de herencia entre marcos. Esta relación de herencia permite que ciertos marcos hereden los atributos y restricciones de la clase superior o padre. La base de conocimiento en sí está formada por las instancias de esos marcos (objetos o conceptos).

Un ejemplo de uso del modelo basado en marcos es OKBC (Open Knowledge Base Connectivity), que define una API para el acceso a sistemas de representación de conocimientos.

OKBC se basa en marcos, slots, facetas, instancias. Al igual que ocurre en otros modelos, las ontologías basadas en marcos admiten el paradigma de diseño orientado a objetos, lo que puede resultar muy útil cuando se pretende que dicha ontología trabaje con lenguajes de programación también orientados a objetos.

En el artículo previamente mencionado [15] se define un ejemplo sencillo que puede resultar muy representativo sobre un dominio de conocimiento orientado a los tipos y características de los vinos:

Una clase de vinos representa todos los vinos *Bordeaux*. Se podría dividir la clase de todos los vinos en vino rojo, blanco y rosado. Para describir las propiedades podemos tener como referencia el vino *Chteau Lafite Rothschild Pauillac*, que se sabe que es producido en el establecimiento vinícola *Chteau Lafite Tothschild* y de cuerpo *intenso*. En este ejemplo existen dos slots que describen el vino: el *cuerpo* y el *productor*.

Si se definen estos slot en la clase vino (de la que heredan vino rojo, blanco y rosado), todas las instancias de vino que se añadan a la base de conocimientos poseerán estos dos slot y sus restricciones. La siguiente figura representa el modelo de la ontología de este ejemplo junto con la instancia correspondiente (en ella se supone que Productor es una clase independiente con atributos, los cuales no se especifican por simplicidad).

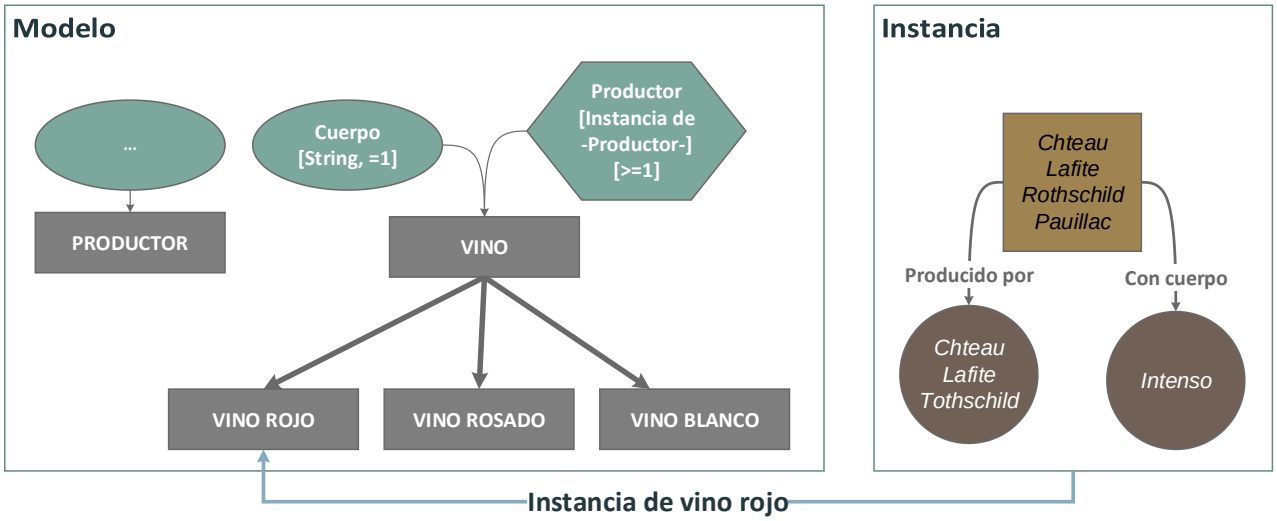


Figura 4. Ejemplo de ontología basada en marcos

2.3.2.1 Protégé-Frames

Protégé es un editor de ontologías open source gratuito que proporciona un framework de desarrollo para la definición de nuevas ontologías. Existen dos variantes de este editor: Protégé-Frames [18] y Protégé-OWL [19]. En este proyecto se ha hecho uso de ontologías basadas en marcos, por lo que solo se expone el correspondiente editor, si bien ambos tienen bastantes similitudes.

Este software permite definir de manera gráfica las clases, atributos y slots de forma muy intuitiva. Además, permite guardar formularios y *queries* para introducir y recuperar datos que se

ajusten al modelo diseñado, de manera que puedan reutilizarse en la entrada y salida de información desde la ontología.

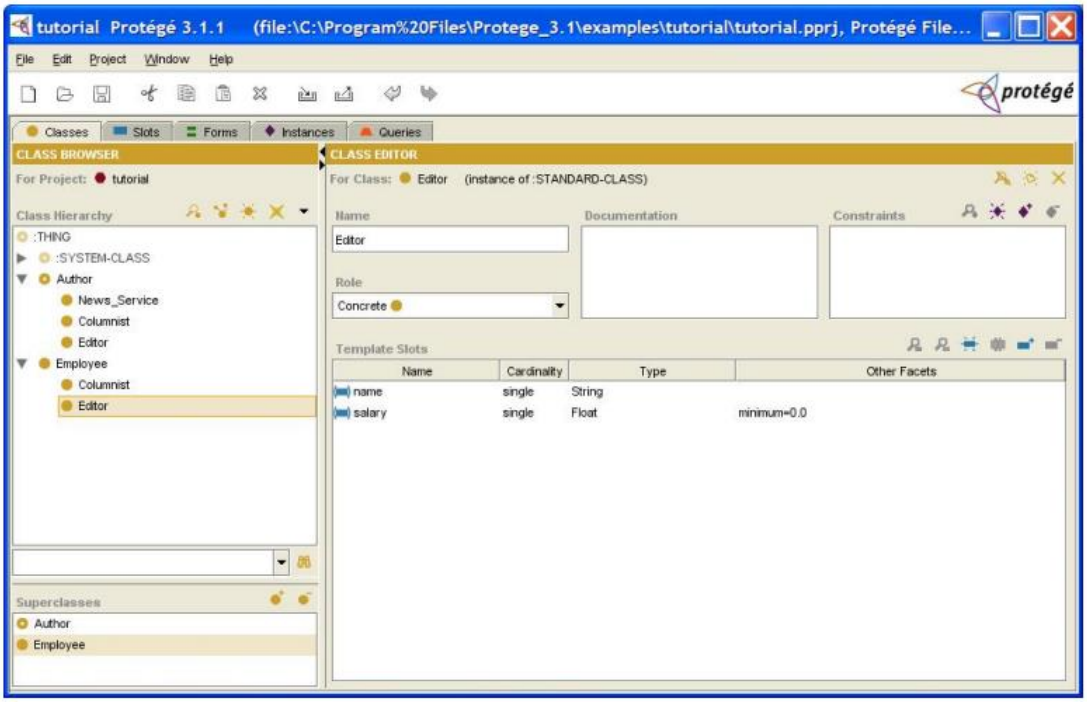


Figura 5. Captura de pantalla, visión general (Protégé-Frames)

Con este editor es muy rápido y sencillo añadir clases y subclases, así como sus correspondientes slots. Los slots de clases inferiores pueden sobrescribir slot de clases superiores y, si es necesario, una clase puede reubicarse en la jerarquía incluso una vez diseñada.

Entre otras funcionalidades, destaca el hecho de poder configurar clases como *concretas* o *abstractas*. Las clases *abstractas* son útiles a la hora definir conceptos generales y sus atributos, pero la ontología no permitirá instancias de esa clase, sino que habrá que añadir las instancias en subclases de tipo *concretas* (que heredarán las características).

Con este conjunto de herramientas reducidas (pero potentes en conjunto) se puede definir ontologías con restricciones muy concretas y completamente personalizadas para la base de conocimiento que se quiere utilizar en el sistema.

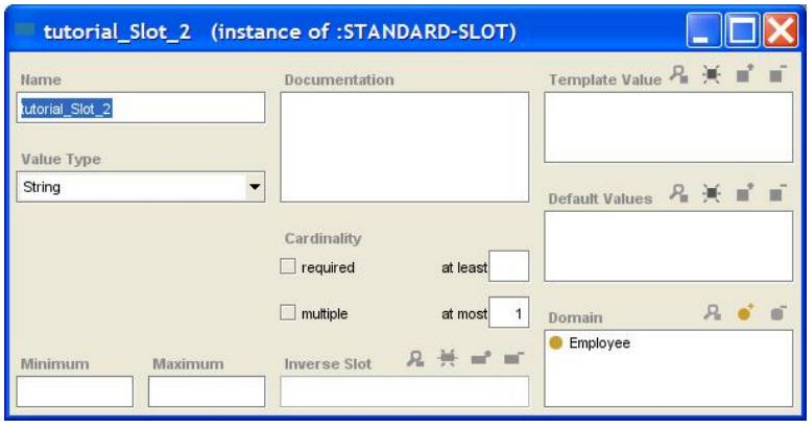


Figura 6. Creación de slot (Protégé-Frames)

La gran ventaja de este editor *open source* escrito en Java es que es fácilmente manipulable desde otro programa escrito en Java. El modelo puede ser creado gráficamente, asegurando que todo esté bien diseñado e incluso probar la introducción de instancias.

En definitiva, la definición de ontologías mediante este editor es rápida, sencilla y eficiente, razón por la cual se ha utilizado en el desarrollo de este proyecto tal como se verá a lo largo de la presente memoria.

2.4 UPnP

UPnP es una arquitectura que ofrece conectividad *peer-to-peer* (punto a punto) entre PC, electrodomésticos inteligentes y dispositivos inalámbricos. La arquitectura UPnP es abierta y distribuida, aprovechando TCP/IP y la Web para posibilitar la conexión, control y transferencia de datos entre dispositivos en redes tales como las propias del hogar, la oficina y cualquier otro lugar similar.

UPnP Forum [20], en el sitio web de UPnP, proporciona la siguiente descripción sobre esta tecnología. Entre las características de UPnP se destacan las siguientes:

- **Independencia de red**: la tecnología UPnP puede utilizarse sobre cualquier tecnología de red, inclusyendo Wi-Fi, coaxial, línea telefónica, línea eléctrica, IEEE 1394 y Ethernet.
- **Independencia de plataforma**: los fabricantes pueden utilizar cualquier sistema operativo y cualquier lenguaje de programación para crear productos UPnP.
- **Tecnología basada en Internet**: la arquitectura UPnP se ha construido sobre IP, TCP, UDP, HTTP y XML, entre otras.
- **Control de Interfaz de dispositivos**: la arquitectura UPnP permite a los fabricantes controlar la interfaz que mostrará el dispositivo a través del navegador.
- **Control programable**: la arquitectura UPnP permite a aplicaciones convencionales controlar los dispositivos.

- **Extensible**: los fabricantes pueden proporcionar a cada producto UPnP servicios adicionales situados en capas por encima de la arquitectura básica de dispositivos.

La tecnología UPnP ha sido optimizada para su uso en redes domésticas, en pequeñas empresas y edificios comerciales, permitiendo la comunicación entre dos dispositivos cualesquiera de la red. Una de las principales ventajas de esta tecnología es que soporta el descubrimiento automático de dispositivos sin configuración, siempre y cuando estos puedan:

- Unirse a la red dinámicamente
- Obtener una dirección IP
- Anunciar su nombre
- Exponer sus capacidades bajo demanda
- Aprender de la presencia de capacidades de otros dispositivos
- Abandonar la red automáticamente sin dejar ningún estado no deseado sin la correspondiente información.

Entre los escenarios más habituales que contempla el UPnP Forum para esta tecnología se encuentran dos grupos: entretenimiento digital y dispositivos inteligentes.

Para el primero de estos grupos, una serie de ejemplos habituales son:

1 Videojuegos basados en Internet

2 Visionado en la TV de fotografías y videos provenientes de otros dispositivos

3 Compartición de música a través de los diferentes dispositivos de una casa

Respecto al segundo grupo se afirma que la bajada de precios y el aumento de la seguridad en las distintas redes han impulsado la creación de redes inteligentes aplicadas a dispositivos electrónicos ordinarios. Entre los principales usos en redes inteligentes se destacan tres:

- Control remoto de dispositivos
- Compartición de audio, imágenes y video entre dispositivos
- Compartición de información entre dispositivos y la World-Wide Web

Tomando como referencia un documento estandarizado sobre la arquitectura UPnP de dispositivos [21], la pila de protocolos de los que hace uso esta tecnología quedaría tal como se representa en la Figura 7:

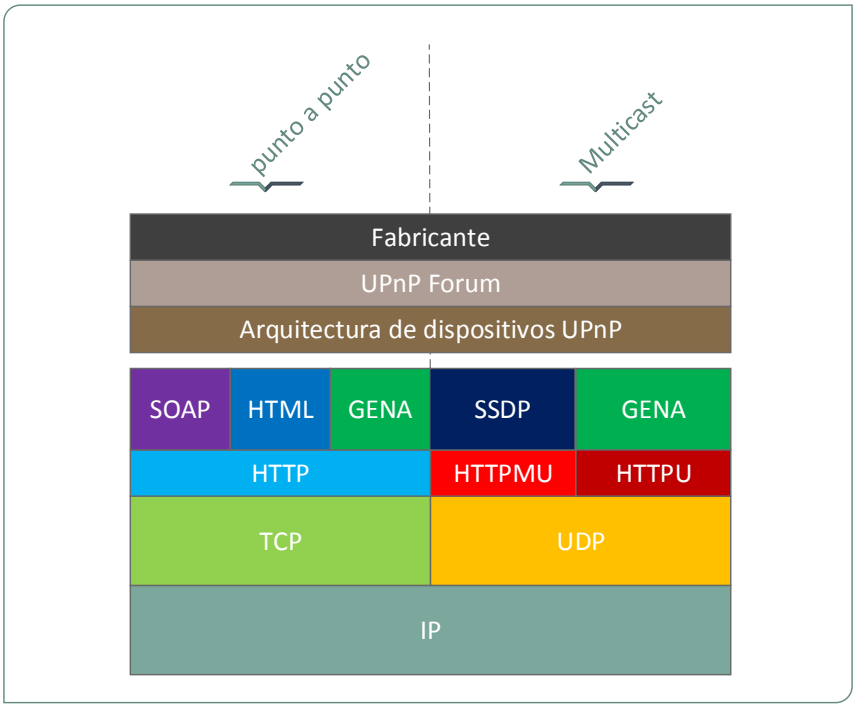


Figura 7. Pila de protocolos UPnP

2.4.1 Componentes de una red UPnP

Una red UPnP está formada por tres componentes básicos: dispositivos, servicios y puntos de control. En el documento de la descripción de la arquitectura de dispositivos que proporciona UPnP Forum [21] se describen estos tres componentes de la siguiente manera[1]:

- **Dispositivos**

  Un dispositivo UPnP es un contenedor de servicios y dispositivos.

  Por ejemplo, una videograbadora puede consistir en un servicio de transporte de cinta, un servicio de afinación y un servicio de reloj. Un dispositivo integrado que incluye TV/videograbadora constaría no sólo de servicios, sino también de un dispositivo integrado.

  Diferentes categorías de dispositivos UPnP se asociarán con diferentes conjuntos de servicios y dispositivos integrados.

  Toda esta información se captura en un documento de descripción de dispositivo XML que el propio dispositivo debe alojar. Además del conjunto de servicios, la descripción del dispositivo también enumera las propiedades asociadas a este.

- **Servicios**

  La unidad de control más pequeña en una red UPnP es un servicio. Un servicio expone acciones y modela su estado con variables de estado.

[1] Se muestra exclusivamente un resumen de los puntos más importantes extraídos de la definición completa dada en el documento referenciado.

Por ejemplo, un servicio de reloj se puede modelar para tener una variable de estado, hora actual, que define el estado del reloj, y dos acciones, definir hora y obtener hora, que le permiten controlar el servicio.

Esta información es parte de una descripción de servicio XML estandarizada por el UPnP Forum. Dentro del documento de descripción del dispositivo se incluye una dirección Web para estas descripciones de servicio (pudiendo un dispositivo incluir varios servicios).

Un servicio consiste en una tabla de estado, un servidor de control y un servidor de eventos. La tabla de estado modela el estado del servicio a través de las variables de estado y las actualiza cuando este cambia. El servidor de control recibe solicitudes de acción, las lleva a cabo, actualiza la tabla de estado y devuelve respuestas.

El servidor de eventos publica eventos para subscriptores interesados en cualquier momento en el que dicho estado cambie.

- **Puntos de control**

  Un punto de control es un controlador capaz de descubrir y controlar a otros dispositivos. Después del descubrimiento, un punto de control puede:

  - Recuperar la descripción del dispositivo y obtener una lista de servicios asociados
  - Recuperar las descripciones de servicio para los servicios de interés
  - Invocar acciones para controlar el servicio
  - Subscribirse a eventos para un determinado servicio. En este caso, siempre que un estado cambie, el servidor de eventos enviará un evento al punto de control.

A continuación se muestra una figura extraída en un documento descriptivo proporcionado en el sitio Web de Microsoft para la implementación de UPnP en Windows XP [22], que representa los distintos dispositivos así como la relación entre ellos.

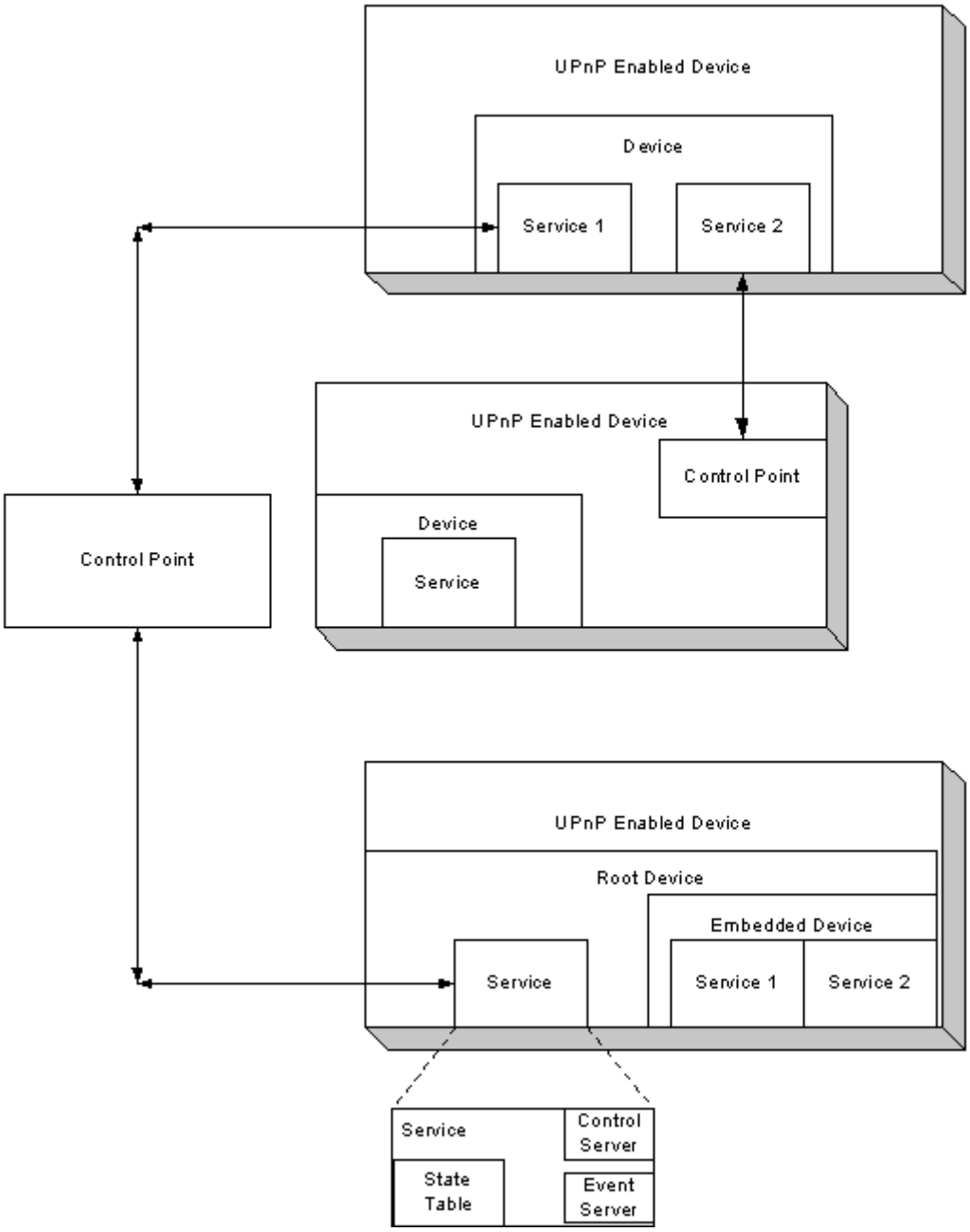


Figura 8. Componentes de una red UPnP

2.4.2 Cling

A la hora de desarrollar una red UPnP desde cero, es decir, en un sistema que no contiene componentes que implementasen esta tecnología previamente, existen muchas y variadas opciones de software libre que podemos utilizar y que ya implementan la pila de protocolos requerida.

Este enfoque, por supuesto, facilita la implantación de UPnP y proporciona unas ciertas garantías de funcionamiento al tratarse de software bien testado y ampliamente extendido. No obstante, cuando se desea que este software esté implementado en un lenguaje concreto (Java, en este proyecto) y sea compatible con el SO Android, las opciones se reducen.

Cling [23] se adapta a estos requisitos y proporciona una sencilla y completa API que puede ser utilizada tanto en programas Java como en el SO Android (que recordemos que sus aplicaciones también son construidas, al menos en parte, mediante Java).

Cling proporciona diferentes versiones del software con sus correspondientes funciones:

- **Cling Core**

  Una librería Java embebida que implementa la versión 1.0 de la arquitectura de dispositivos UPnP del UPnP Forum. Esta librería permite exponer servicios mediante una interfaz UPnP o construir puntos de control. Además, Cling Core puede ser integrado en una aplicación Android.

- **Cling Support**

  Conjunto de clases opcionales que sirven de extensión a Cling Core y se orientan al trabajo con servidores multimedia.

- **Cling Workbench**

  Una aplicación de escritorio que muestra la información de dispositivos UPnP como si de un navegador se tratase, permitiendo la interacción con ellos.

- **Cling MediaRenderer**

  Servidor standalone UPnP MediaRenderer, basado en GStreamer (*framework* multimedia multiplataforma libre).

De entre todas las versiones software proporcionadas por Cling, aquella de utilidad para este proyecto corresponde a Cling Core. El sitio web de Cling contiene un manual detallado y ejemplificado [24] para la utilización de estas librerías tanto en un programa Java ordinario como en una aplicación Android (con sus particularidades en lo referente a servicios corriendo en segundo plano).

2.5 Motores de búsqueda

Tal como se describe en [25], un motor de búsqueda, también conocido como buscador, es un sistema informático que busca archivos almacenados en servidores Web. Los buscadores de Internet varían unos de otros, dado que algunos buscan únicamente el contenido en la Web y otros procesan además información procedente de otras fuentes como podría ser aquella alojada en servidores FTP.

En los motores de búsqueda se accede a la información mediante el uso de palabras clave. Estos buscadores son en su base grandes bases de datos que incorporan automáticamente páginas web mediante robots que recorren Internet y sus recursos de manera reiterativa, actualizando o añadiendo información. Cuando se realiza una búsqueda en estos sistemas, el resultado es lo que se denomina “página de resultados”, que consiste en una lista de direcciones web (quizá con una breve información asociada) que están relacionadas con las palabras clave utilizadas.

Se sobreentiende que el concepto de motor de búsqueda es ampliamente conocido y no requiere de mayor explicación, pues grandes compañías a nivel global, como Google, surgieron de este tipo de servicios.

El objetivo de este apartado es introducir al lector a alguna de las herramientas no tan conocidas de los motores de búsqueda que permiten filtrar de una manera más precisa la información devuelta, lo cual se tratará en el siguiente apartado.

2.5.1 Filtros de búsqueda

Los buscadores son herramientas extremadamente útiles a la hora de encontrar información en Internet. No obstante, la gran mayoría de usuarios no utilizan estas extendidas herramientas con su máximo potencial. Existen comandos especiales en estos buscadores que permiten aplicar filtros en los resultados y que resultan realmente sencillos de utilizar.

A continuación se listan algunos de estos comandos, si bien se puede encontrar un conjunto más amplio en esta sencilla página web dedicada a ello [26]. En esta lista se hará referencia a tres de los motores de búsqueda más extendidos, lo que no implica que estos comandos sean válidos en otros buscadores.

Tabla 1. Comandos para motores de búsqueda

Comando	Motor de búsqueda	Función
-	Google, Yahoo, Bing	Excluye el término de búsqueda
$	Google, Yahoo, Bing	Extensión de palabras
|	Bing	OR: al menos uno de los términos
~	Google	Sinónimo
“ ”	Google, Bing	Búsqueda exacta de términos (en el mismo orden)
+	Google, Yahoo, Bing	Incluye todos los términos
( )	Google, Bing	Combinación de comandos
*****	Google, Bing	Comodín

2.6 Redes Bayesianas

Las redes bayesianas, también conocidas como redes de creencias o redes de Bayes, son grafos acíclicos dirigidos que representan un conjunto de variables aleatorias y sus dependencias condicionales.

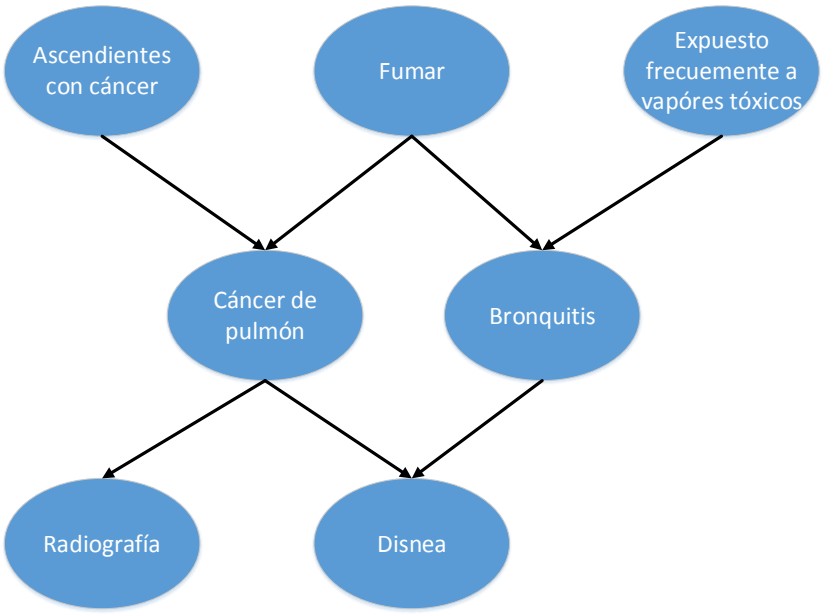


Figura 9. Ejemplo de red bayesiana

En [27], C. Bielza y P.Larrañaga explican en detalle cuáles son las ventajas de este tipo de redes, así como sus fundamentos matemáticos. Entre dichas ventajas se destacan las siguientes:

- Representación explícita de conocimientos inciertos
    - Representación gráfica e intuitiva
- Permite tratar la incertidumbre en sistemas que pretenden razonar y tomar decisiones
- Se fundamenta en teorías probabilísticas, proporcionando una semántica clara y un sólido fundamento teórico
- Puede manejar muchas variables
- Cuenta actualmente con un gran desarrollo en distintas áreas

Las redes bayesianas ilustran visualmente las cantidades conocidas y desconocidas de un evento a través de variables de asignación, así como sus dependencias condicionales. De ellas puede extraerse la probabilidad de que la variable se produzca tomando como base si se producen o no otras.

Estas características hacen de las redes bayesianas unas firmes candidatas a la hora de seleccionar herramientas lógicas y matemáticas que puedan dar soporte a diferentes sistemas en el área de la Inteligencia Artificial y la predicción de eventos.

Las redes bayesianas son muy útiles cuando se quiere predecir las relaciones probables y es por ello que su uso se ha extendido en sistemas médicos, donde se pretende determinar la probabilidad de una cierta enfermedad en base a los síntomas presentes. Este modelo probabilístico también se utiliza en los buscadores a la hora de predecir qué resultados se pretenden obtener en función de las palabras clave (véase apartado 2.5) o en software de reconocimiento de imágenes y habla.

Las redes bayesianas pueden utilizarse también para mecanismos de aprendizaje, en la que la estructura de la red (variables, relaciones de dependencia, etc.) varíe en función de cuan exacta fue la salida obtenida en inferencias anteriores.

Como evolución de los grafos estáticos, existen también las denominadas redes bayesianas dinámicas, las cuales añaden un nuevo parámetro, el tiempo. En este modelo, lo que ocurra en el instante de tiempo t, tiene influencia en lo que ocurre en el instante de tiempo t+1 y, por extensión, se establecen conexiones entre nodos pertenecientes a diferentes redes temporales (t, t+1) consecutivas.

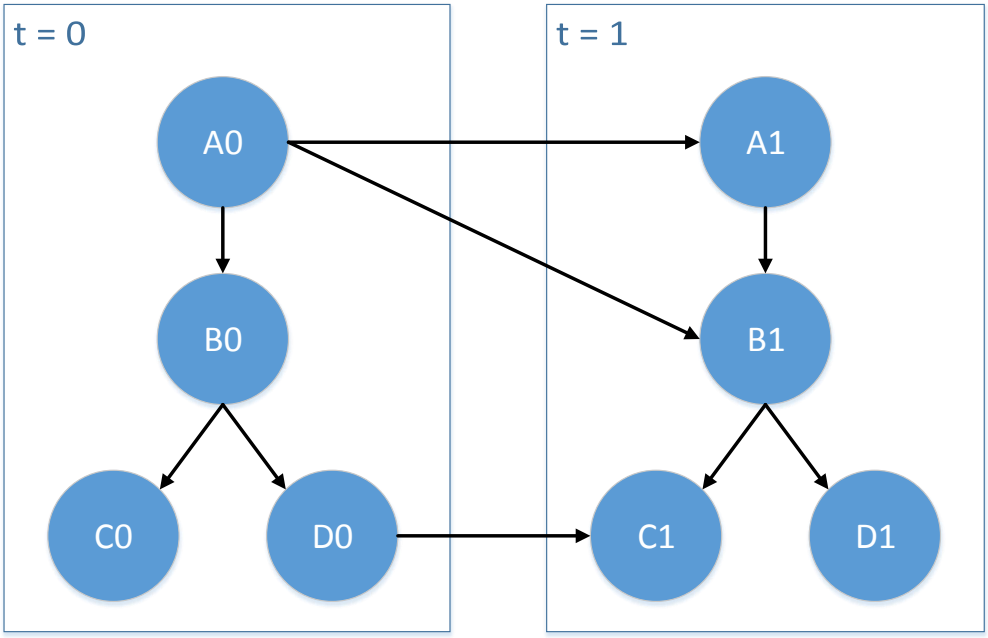


Figura 10. Redes bayesianas dinámicas

2.6.1 Fundamento matemático

A la hora de representar un dominio a través de una red bayesiana, en muy extrañas ocasiones existe independencia entre los diferentes eventos. Por fortuna, a la hora de inducir la probabilidad de un cierto evento, existirán algunas independencias condicionales, las cuales son explotadas en este modelo probabilístico.

Desde un punto de vista matemático, la base se halla en la probabilidad condicional:

$$P(x|y) = p \qquad (1)$$

Si $\boldsymbol{y}$ sucede con certeza y todo lo demás es irrelevante para $\boldsymbol{x}$, entonces la probabilidad de $\boldsymbol{x}$ es $\boldsymbol{p}$.

La regla de la cadena para dos variables nos dice que:

$$\boldsymbol{P(x,y) = P(x|y)P(y)} \qquad (2)$$

De la misma manera, la regla de la cadena para tres variables queda tal como sigue:

$$P(x, y, z) = P(x|y, z)P(y, z) = P(x|y, z)P(y|z)P(z)$$

$$\boldsymbol{P(x, y|z) = P(x|y, z)P(y|z)} \quad (3)$$

Finalmente, se representa la regla de la cadena para ***n*** variables:

$$P(X_1, X_2, \dots, X_n) = P(X_1|X_2, \dots, X_n)P(X_2|X_3, \dots, X_n) \cdots P(X_{n-1}|X_n)P(X_n) \quad (4)$$

Por su parte, la regla de la probabilidad total permite calcular P(A) a partir de P(A, B) de la siguiente manera:

$$P(A) = P(A, B = b_1) + P(A, B = b_2) + \cdots + P(A, B = b_m) = \sum_{i}^{m} P(A, B = b_i)$$

$$= \sum_{B} P(A, B) = \sum_{B} P(A|B)P(B) \quad (5)$$

Mediante estas bases, Thomas Bayes, nacido en 1702, trabajó en el establecimiento de una base matemática para inferir probabilidades basadas en la frecuencia en la que un evento ha ocurrido en el pasado. Uno de los resultados de este trabajo es la conocida como regla de bayes, que permite actualizar la creencia en un evento dado una nueva evidencia (un nuevo evento observado):

$$P(h|e) = \frac{P(e|h)P(h)}{P(e)} \quad (6)$$

Donde:

- P(h) → probabilidad previa de la hipótesis
- P(e) → probabilidad previa de la evidencia
- P(h|e) → probabilidad de la hipótesis a posteriori
- P(e|h) → probabilidad de la evidencia dada la hipótesis

Este modelo matemático permite inferir nuevas probabilidades para ciertas hipótesis a partir de nuevos eventos conocidos o, como se le denominan comúnmente, evidencias, y sirve como base de las descritas redes bayesianas.

2.6.2 UnBBayes

UnBBayes [28] es un software de código abierto desarrollado entre los años 2001 y 2010 para el modelado, aprendizaje y razonamiento basado en redes probabilísticas. Su desarrollo ha surgido como colaboración de dos universidades: University of Brasilia (UnB) y George Mason University (GMU). UnBBayes fue construido originalmente mediante el lenguaje de programación Delphi, pero más tarde se migró a Java, lenguaje con el que ahora se distribuye.

Este software no fue diseñado únicamente para la manipulación de redes bayesianas simples, sino que proporciona soporte para una gran variedad de modelos basados en:

- Redes probabilísticas
    - Redes bayesianas (*Bayesian Network,* BN)
        - Algoritmo *Junction Tree*
    - Diagramas de influencia (*Influence Diagram,* ID)
    - Redes bayesianas múltiplemente seccionadas (*Multiply Sectioned Bayesian Network,* MSBN)
    - Redes bayesianas híbridas (*Hybrid Bayesian Network,* HBN)
    - Redes bayesianas orientadas a objetos (*Object-Oriented Bayesian Network,* OOBN)
- Redes probabilísticas con lógica de primer orden (*First-Order Logic* –FOL-*Probabilistic Network*)
    - Redes bayesianas multientidad (*Multy-Entity Bayesian Network,* MEBN)
    - Lenguaje de ontologías probabilísticas (*Probabilistic Ontology Language,* PR-OWL)
- Redes bayesianas de aprendizaje

Tal como veremos en el siguiente apartado, cuando se utiliza UnBBayes para modelar una red probabilística no se define el tipo de red que se quiere diseñar de entre las expuestas anteriormente, sino que es tarea del usuario hacer uso de los diferentes nodos para modelar una red que se adapte a sus necesidades.

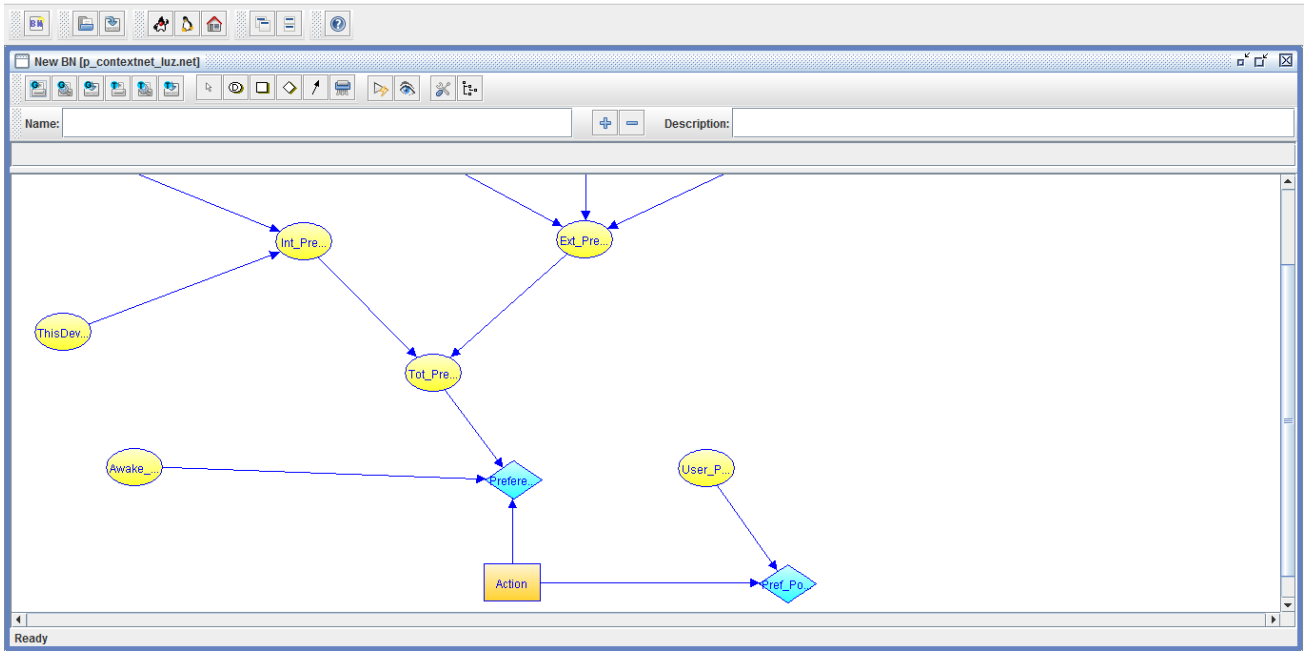


Figura 11. Captura de la ventana de edición de UnBBayes (versión 4.10.4)

2.6.2.1 Tipos de Nodos

A la hora de crear una red probabilística mediante UnBBayes, una de las cosas que caracterizan a este software y más concretamente a su editor gráfico es su simplicidad. Más allá de ordenar las diferentes ventanas de edición, el software sólo permite la introducción de tres tipos de nodos, la conexión entre ellos y la compilación del conjunto de la red mediante el algoritmo *Junction Tree*. Estos tres nodos se describen a continuación:

- **Nodo de variables probabilísticas** (elipse)

  Este tipo de nodos contiene un conjunto de variables que representan la probabilidad de un cierto estado respecto al estado de los nodos precedentes.

  Como se puede intuir, la suma de las probabilidades de todas las variables (se pueden definir tantas como se quiera) debe sumar 1 para cada columna (para cada combinación de eventos previos). Este tipo de nodos componen generalmente la mayor parte de los elementos de una red.

  La tabla que contiene el conjunto de probabilidades para los distintos estados en función de los estados de los nodos precedentes recibe el nombre de tabla de potencias.

ThisDev_ON	YES			NO		
OtherDev_...	several	one	none	several	one	none
YES	1	0,9	0,7	0,9	0,7	0
NO	0	0,1	0,3	0,1	0,3	1

Figura 12. Ejemplo de tabla de potencias (UnBBayes)

- **Nodo de variables de utilidad** (rombo)

  Los nodos de esta clase permiten definir una única variable, cuyo valor dependerá de los valores que toman las variables de los nodos previos, tal como ocurría con los nodos de variables probabilísticas.

  La particularidad de este tipo de nodos es que los valores que se cargan en las variables no son probabilidades, sino un conjunto de valores numéricos. Estos valores no tienen ninguna limitación e incluso pueden ser negativos y decimales.

  La representación que se le dé a dichos valores depende exclusivamente del diseñador de la red. Es habitual que estos valores representen unidades monetarias en modelos que pretenden predecir las ganancias o pérdidas probables de una inversión, por ejemplo.

User_Posit...	Same room			Other room		
Action	INCREASE	REDUCE	NOTHING	INCREASE	REDUCE	NOTHING
Utility	75	75	75	25	25	25

Figura 13. Ejemplo de valores en un nodo de variables de utilidad (UnBBayes)

- **Nodo de variables de decisión** (rectángulo)

  Por último, el editor permite seleccionar los denominados nodos de variables de decisión, cuya finalidad es la de extraer conclusiones de la red a partir del estado de los diferentes valores de los nodos precedentes.

  A la hora de definir un nodo de esta clase, el desarrollado únicamente puede añadir las variables, pero no habrá valores asociados. Los valores que adquirirá cada variable provendrán de los nodos precedentes una vez que se ejecute el algoritmo *Junction Tree*.

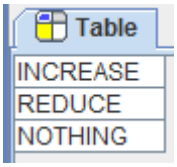


Figura 14. Ejemplo de variables en un nodo de variables de decisión (UnBBayes)

2.6.2.2 Algoritmo Junction Tree

El algoritmo *Junction Tree* (árbol de unión, en español) tiene como objetivo la simplificación de una red de nodos cualquiera. En la teoría de grafos, una descomposición del árbol (*tree descomposition* [29]) tal como la que lleva a cabo el algoritmo *Junction Tree* consiste en mapear el grafo original para crear un grafo equivalente reducido, de manera que pueda acelerarse el procesado de dicho grafo.

En el área del *machine learning*¸ este mecanismo es denominado también *Junction Tree*, razón por la cual el algoritmo de simplificación del árbol de nodos que utiliza UnBBayes ha adquirido este nombre.

Este mapeo de nodos juega un rol importante en tareas como la inferencia probabilística, la optimización de consultas y la descomposición de matrices.

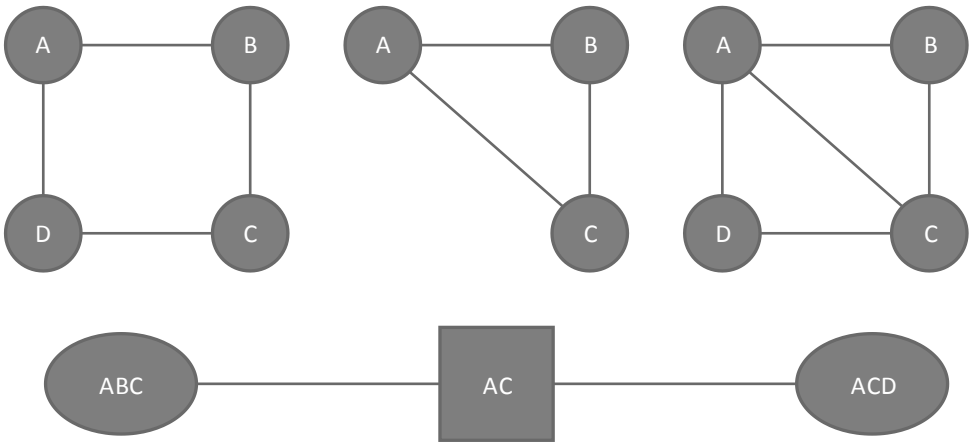


Figura 15. Descomposición de grafo

Para la descomposición de un grafo tal como el que muestra la Figura 15 en la parte superior izquierda, el algoritmo elimina un nodo (nodo *D*) y añade una dependencia directa con el nodo inverso. Por simetría, la eliminación del nodo *B* obtendría un resultado equivalente. De esta manera, se determinan dos nodos en los extremos formados por los conjuntos resultantes (elipses), junto con un nodo separador que contiene la intersección de los nodos extremos (cuadrado).

No es intención de este apartado entrar en detalle en los fundamentos matemáticos que sustentan la descomposición del grafo, si bien gran parte de ellos se hallan en las mismas bases probabilísticas definidas en el apartado 2.6.1. No obstante, si resulta de especial interés para el lector, puede encontrarse una descripción detallada de este algoritmo en una presentación sobre el mismo desarrollada por la Universidad de Massachusetts [30].

2.7 Android

Android, según clama su página web [31], es el sistema operativo para móviles más famoso del mundo, con más de 1000 millones de dispositivos (principalmente *smartphones* y *tablets*) utilizando este SO. Aunque el sistema fue desarrollado por Android Inc, una compañía de software ubicada en California que trabajó en el desarrollo de este SO basado en Linux y orientado a dispositivos móviles, terminó siendo adquirida por Google en 2005, convirtiéndose desde entonces en una referencia mundial y a la que hoy día pocos sistemas operativos móviles pueden hacer frente.

A lo largo de los últimos años Android se ha extendido a otros tipos de dispositivos como las denominadas *smartTVs* o a los *weareables* (dispositivos que se "visten") como los *smartwatches*. Estos nombres hacen referencia a televisiones y relojes inteligentes, respectivamente. En esta corriente precisamente está trabajando actualmente Google, y Android está siendo adaptada a esta

clase de dispositivos *weareables* con funciones más acotadas. Con seguridad se oirá hablar mucho de su nueva versión del SO: Android Wear.

No termina ahí su expansión, pues se comienza a ver en el mercado coches que integran Android para manipular mapas y archivos multimedia, e incluso nuevos ordenadores de prestaciones limitadas que encuentran en Android un SO ideal por sus bajos requisitos en cuanto a hardware se refiere pero con capacidad suficiente para navegar por internet y soportar aplicaciones suficientemente complejas para cubrir las necesidades de la mayor parte de los usuarios.

Android se caracteriza por su interfaz cuidada pero altamente personalizable, su continua remodelación y actualización (hasta ahora se han dado varias actualizaciones al año en la que se cuida mucho de dar soporte a las tecnologías más punteras, de manera que los fabricantes puedan incorporarlas lo antes posible a sus dispositivos) y su flexibilidad a la hora de adaptarse a diferentes dispositivos, con sus correspondientes resoluciones y modos de interacción.

Figura 16. Android en diferentes dispositivos

Además de su extendida presencia en estos dispositivos tan utilizados en el día a día y a su política de permitir su utilización por cualquier fabricante, uno de los motivos principales de su rápida aceptación (incluso desbancando a iOS, el sistema operativo de Apple, tras varios años como referencia del sector) es su gran repositorio de aplicaciones: Google Play.

Las aplicaciones desarrolladas para el SO Android que el usuario puede descargar e instalar de una manera extremadamente sencilla desde Google Play no tienen por qué provenir de grandes compañías de software, sino que el mercado ha quedado abierto a que pequeñas empresas o incluso desarrolladores independientes puedan compartir y, si se desea, vender sus aplicaciones a través de este repositorio.

Existe además un SDK gratuito que Google ofrece a los desarrolladores de manera que la creación de aplicaciones (comúnmente denotadas como *apps*) resulte un proceso sencillo, fomentando el rápido crecimiento del repositorio y ofreciendo constantemente al usuario un nuevo abanico de aplicaciones. Es evidente que Google, con esta estrategia que ya hizo líder a Apple en este sector de los dispositivos móviles, encuentra en el desarrollador a su principal aliado a la hora de mantener a Android en lo más alto.

Además de este SDK, existe una cantidad enorme de APIs desarrolladas para este SO (tanto por Google como por terceros) que permiten al desarrollador implementar en su aplicación de manera sencilla prestaciones de gran complejidad, como reconocimiento del habla, ubicación GPS sobre mapas y mensajería PUSH (notificaciones remotas), entre muchas otras.

Tal como se verá en el apartado 4.9, algunos de los recursos proporcionados por Android serán de vital importancia a la hora de implementar la interfaz gráfica de usuario desarrollada para el sistema que se describe en esta memoria. En esta línea se presenta el siguiente apartado, cuyo objetivo es el de describir brevemente algunos de los elementos principales que el desarrollador debe manejar a la hora de crear una aplicación para este SO.

2.7.1 Elementos principales de las aplicaciones Android

A la hora de crear una nueva aplicación Android existen algunos elementos que serán de uso requerido, sin importar la complejidad de la aplicación. Estos elementos esenciales serán los que describirán brevemente en este apartado. No obstante, toda la información que se muestra de manera resumida en este apartado puede consultarse, junto con mucha otra y de manera mucho más detallada, en la página oficial de desarrolladores Android [32] y sus elementos fundamentales se encuentran resumidos en su sección *Application Fundamentals* [33].

- **Activities (actividades)**

  Una actividad representa una sola pantalla de la interfaz de usuario. Por ejemplo, una aplicación podría tener una actividad para representar la lista de emails y otra para leer dichos emails. Aunque las actividades trabajan juntas para proporcionar una experiencia coherente al usuario, cada una es independiente del resto. Como tal, una aplicación diferente podría arrancar una actividad (como la de abrir la lista de emails) siempre y cuando la aplicación de correo electrónico lo permita.

- **Services (servicios)**

  Un servicio es un componente que corre en segundo plano para llevar a cabo operaciones de larga duración o ejecutar tareas para procesos remotos. Un servicio no proporciona interfaz de usuario. Por ejemplo, un servicio podría ser la reproducción de música en segundo plano mientras el usuario navega por otra aplicación. Otro componente, como una actividad, puede lanzar un servicio y enlazarse a él para mantener una cierta interacción.

- **Content providers (proveedores de contenido)**

  Un proveedor de contenido administra un conjunto de información compartida de una aplicación. La información puede ser almacenada en un fichero, en una base de datos

SQLite, en la Web o en cualquier otro recurso accesible desde la app que permite el almacenamiento persistente. A través de este proveedor, otras aplicaciones puede solicitar información o incluso modificarla (siempre que la aplicación dueña de los datos lo permita). Por ejemplo, el propio sistema Android proporciona un proveedor que administra la información de contactos.

- **Broadcast receivers (receptores de difusión)**

  Un receptor de difusiones es un componente que responde a avisos procedentes de cualquier parte del sistema. Muchos de estos receptores los proporciona el propio sistema (por ejemplo, la difusión del aviso de que la pantalla se ha apagado o de que la batería está baja). Aunque estos receptores de difusión no muestran ninguna interfaz de usuario, pueden crear notificaciones en la barra de estado (*status bar*). De manera general, estos componentes sirven de enlace con otros, de manera que su trabajo es mínimo.

Estos componentes (con excepción del proveedor de contenido) se activan a través de un mensaje asíncrono denominado **intent**. Los *intent* enlazan unos componentes con otros en tiempo de ejecución (como si de una invocación se tratase), pertenezca el componente lanzado a la misma app o a otra diferente.

No obstante, antes de que el sistema Android pueda arrancar un componente de una aplicación, el sistema debe saber que el componente existe a través de la lectura del denominado **Manifest File** (archivo de "manifiesto"). Las aplicaciones deben declarar todos sus componentes en este fichero, el cual debe ubicarse en la raíz del directorio de la aplicación.

Además, el *Manifest File* lleva a cabo algunas otras tareas:

- Identifica los permisos de usuario que la aplicación requiere (acceso a Internet, acceso a la información de contactos, etc.)
- Declara el nivel mínimo de API requerido por la app (API Level)
- Declara el software y hardware que será utilizado por la aplicación (cámara, bluetooth, servicios, etc.)
- Declara las librerías adicionales que deben ser enlazadas a esta aplicación, más allá del propio Framework de Android (por ejemplo, la librería de *Google Maps*)

Son muchos los elementos adicionales que las nuevas versiones de Android han traído consigo, como los *Fragments* (similares a las actividades, pero que pueden correr en una sección de la pantalla, sin afectar a la actividad principal) o las *Action Bars* (que aglutinan iconos e incluso las opciones antiguamente ligadas al botón físico de menú), pero aquellas consideradas como pilares son las mencionadas previamente.

2.8 Bases de datos

En el libro *Introducción a los Sistemas de Bases de Datos* [34], en el que se exponen los fundamentos del campo de las Bases de Datos en su conjunto, se proporciona la siguiente definición de "Base de Datos":

"Un sistema de base de datos es básicamente un sistema computarizado para guardar registros; es decir, es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones. La información en cuestión puede ser cualquier cosa que sea de importancia para el individuo u organización; en otras palabras, todo lo que sea necesario para auxiliarle en el proceso general de su administración".

Tal como se extrae de esta definición, la finalidad de una base de datos es la de almacenar información que pueda ser recuperada. No obstante, los modelos de Base de Datos más extendidos permiten además realizar operaciones aritméticas y lógicas sobre los datos, así como procesar de diferentes maneras las tablas que los almacenan (unificando datos, por ejemplo). Las bases de datos actuales no son un sistema cuya finalidad es únicamente la de almacenar dichos datos, sino que a través de las herramientas que proporcionan se han convertido en un elemento importante en el procesado de la información de los diversos sistemas informáticos.

Existen diferentes bases de datos, cada una con sus correspondientes ventajas y desventajas, pero las más utilizadas son aquellas basadas en SQL. Una página web dedicada en exclusiva a las bases de datos [35] proporciona en su web anualmente la lista de bases de datos más utilizadas. En su última lista aparecen en las cuatro primeras posiciones bases de datos SQL en el siguiente orden: Oracle, MySQL, Microsoft SQL Server y PostgreSQL. La base de datos NoSQL que más alto se encuentra en la lista corresponde a MongoDB, en la quinta posición.

En lo que a este proyecto se refiere, la base de datos de interés es SQLite, que ocupa el octavo puesto de este ranking. El resto de bases de datos no serán expuestas, si bien el lector encontrará gran cantidad de información tanto en la literatura relacionada [34] como en los propios sitios web de cada una de ellas.

2.8.1 SQLite

La mayoría de las grandes bases de datos requieren de un RDBMS (Sistema de Gestión de Base de Datos Relacional) que se hace cargo de gestionar la estructura física de los datos en disco y su acceso posterior. Estos sistemas de gestión, aunque pueden resultar muy útiles, requieren de procesos independientes y complican el acceso a la base de datos por parte del usuario. Para ello, los RDBMS son capaces a su vez de gestionar un modelo de vistas para ocultar la información innecesaria y limitar el acceso a otros usuarios, si se requiere, pero todo ello aumenta el peso de la BD y los requisitos de procesamiento.

SQLite se diferencia de la mayoría de base de datos precisamente en que evita el uso de un proceso servidor e implementa una base de datos SQL sencilla y con bajos requerimientos.

Se expone a continuación fragmentos de la descripción que SQLite proporciona en su sitio Web [36] y que resume de una manera concisa y precisa las principales características de esta base de datos.

"SQLite es una librería que implementa un motor de base de datos SQL transaccional, sin servidor, auto contenido y sin necesidad de configuración. El código de SQLite es de dominio público y puede ser utilizado para cualquier propósito, público o privado […]".

"[…] SQLite es un motor de base de datos SQL embebido. A diferencia de la mayoría del resto de bases de datos SQL, SQLite no tiene un proceso servidor independiente. SQLite lee y escribe directamente sobre el fichero de disco, de manera que una base de datos SQL completa (múltiples tablas, índices, triggers –disparadores- y vistas) está contenida en un único fichero. Además, el formato del fichero de la base de datos puede ser copiado libremente entre sistemas de 32 y 64 bits y entre arquitecturas little-endian y big-endian […]".

"[…] SQLite es una librería compacta, con un peso que oscila los 500KiB con todas las características habilitadas (depende de las configuraciones durante la compilación) […]".

"[…] SQLite es testada cuidadosamente y tiene reputación de ser muy fiable […]".

Por todas estas características, entre otras que pueden encontrarse en el sitio Web mencionado [36], SQLite goza de gran popularidad y es la alternativa que Android recomienda a la hora de almacenar datos de usuario desde sus aplicaciones.

En lo que respecta a su manipulación desde Java, esta base de datos puede ser accedida mediante el driver SQLite JDBC, una librería ensamblada en un único fichero JAR (Java Archive) para acceder y crear base de datos SQLite.

Capítulo 3

Especificaciones y análisis del sistema

3.1 Especificaciones del sistema.

Antes de poder realizar un análisis fidedigno y detallado del sistema, se deben poner sobre la mesa aquellas especificaciones que acotarán su diseño, procurando identificar todos los rasgos relevantes del sistema.

Para ello, se dividen las especificaciones atendiendo a los requisitos, funcionales y no funcionales, y a las limitaciones propias del proyecto. Por último, se definen en el contexto de estas limitaciones una serie de especificaciones complementarias deseables, pero no requeridas.

3.1.1 Requisitos del sistema.

Previo a su diseño e implementación, se han definido una serie de requisitos en el sistema que deben ser respetados y cuya funcionalidad, en su caso, debe alcanzarse.

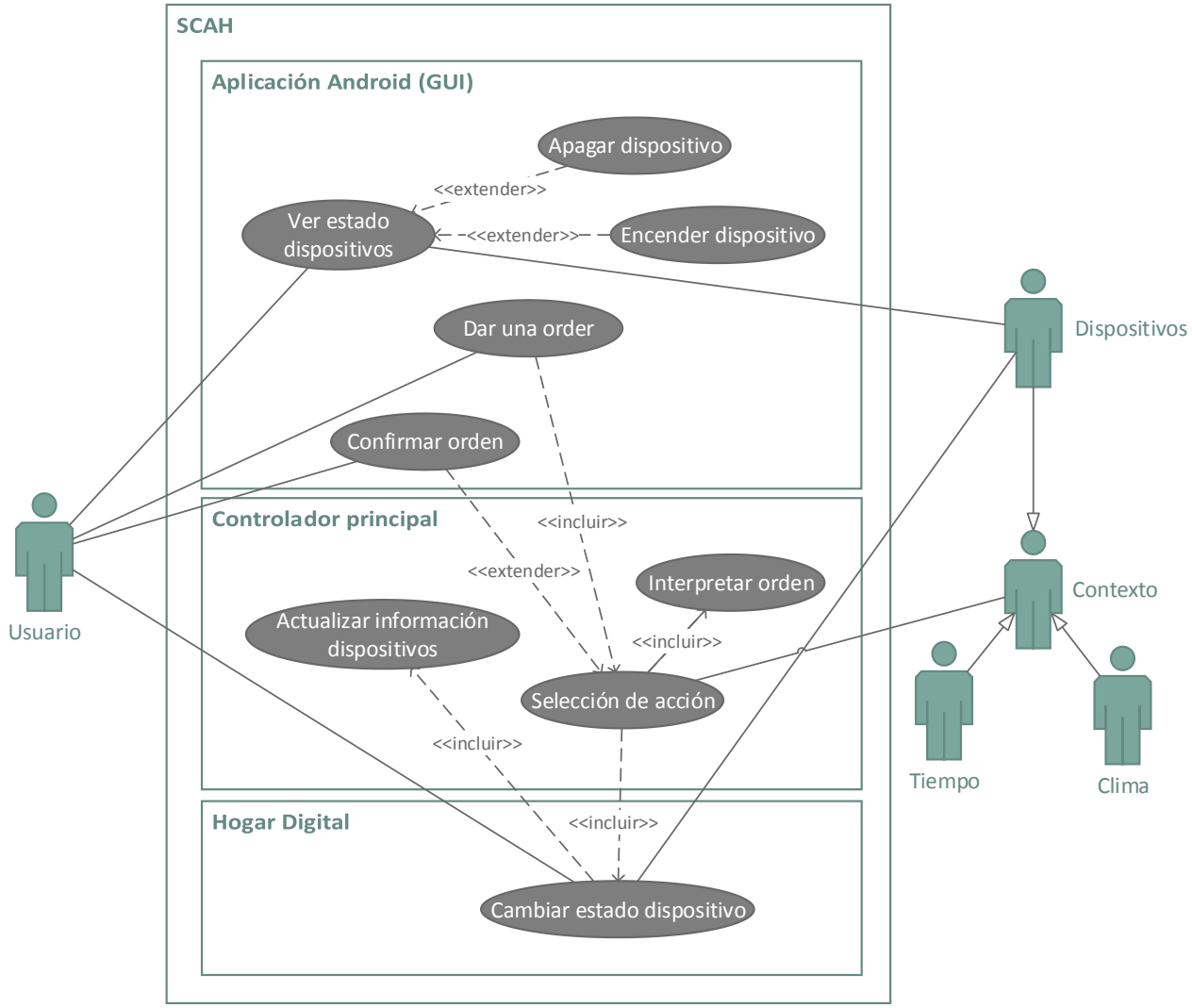


Figura 17. Diagrama de casos de uso

3.1.1.1 Requisitos funcionales.

Entre el conjunto de requisitos, se comentan aquí aquellos que hacen referencia a los servicios que debe proporcionar el sistema al usuario, describiendo su comportamiento desde un punto de vista plenamente funcional y a muy alto nivel.

La Figura 17 muestra el *diagrama de casos de uso* definido para el sistema. En este diagrama se muestran los agentes que participan en dichos usos además de la relación entre ellos y entre los propios casos.

A primera vista, se pueden diferenciar claramente tres subsistemas (*Aplicación*, *Controlador principal* y *Hogar digital*), los cuales se encuentran contenidos en el sistema *SCAH*, que engloba el conjunto de funcionalidades. Estos subsistemas reflejan una cierta independencia entre ellos en función del uso y funcionalidad que deben proporcionar:

- ***Aplicación Android (GUI)***: este subsistema será el encargado de la interacción con el usuario, ya sea mediante el despliegue de información o mediante la recepción de órdenes.
- ***Controlador principal***: será el subsistema encargado de llevar el peso de las decisiones. Una vez una orden, ya sea directa[1] o indirecta[2], se reciba en la *Aplicación Android*, este subsistema será el encargado de interpretarla e inferir una conclusión acorde tanto a dicha orden como al contexto.
- ***Hogar digital***: este subsistema engloba aquellas acciones ejecutadas sobre los dispositivos. Se deben considerar dos direcciones en la propagación de la información del estado de los dispositivos, pues la acción puede ser el resultado del procesamiento de una orden en el subsistema *Controlador principal*, el resultado de una acción directa proveniente del subsistema *Aplicación Android* o incluso una ejecución física de la acción (p.e. el usuario pulsa el interruptor que enciende una lámpara).

Por su parte, se reflejan con claridad un conjunto de actores (entidades externas al sistema) más allá del usuario, representando tanto a la red de dispositivos como el contexto. Todos estos actores influyen en el modo de funcionar del sistema y particularmente en el conjunto de conclusiones a las que debe llegar tras el análisis de una orden indirecta. Se detallan a continuación el concepto encerrado tras cada uno de estos actores:

- ***Usuario***: representa al usuario y sus asociaciones, con el conjunto correspondiente de casos de uso.
- ***Dispositivos***: este actor representa al conjunto de dispositivos y más concretamente al estado que dichos dispositivos tienen en el momento de interactuar con el sistema. El hecho de que el estado actual (entiéndase "actual" como el momento en el que se procesa una orden) de los dispositivos influya en la toma de decisiones lo relaciona mediante una generalización con el actor *Contexto*.

[1] En el ámbito de este proyecto, se entiende orden *directa* como aquella que no requiere de interpretación, sino que en sí misma identifica la acción a realizar (p.e. cuando el usuario *clica* un botón de la aplicación para encender una lámpara).
[2] En el ámbito de este proyecto, se entiende orden indirecta como aquella que requiere interpretación. Este tipo de órdenes representan las palabras exactas que dice el usuario de una manera coloquial (p.e. "hay muy poca luz", teniendo que interpretar que hay que encender una lámpara).

- ***Contexto***: este actor representa una generalización de un conjunto de actores, entre los que se encuentran el actor *Tiempo* y el actor *Clima* (además del mencionado actor *Dispositivos*). La finalidad de este actor es la de representar al conjunto de circunstancias contextuales (en el momento de procesar una acción) que influyen en la toma de decisiones por parte del sistema *SCAH*, y más concretamente del subsistema *Controlador principal*.
- ***Tiempo***: es el actor que representa la influencia en el sistema de la hora actual, pero también la influencia de la franja horaria (diurna o nocturna), de la estación (primavera, verano, etc.) o incluso de otras circunstancias relacionadas con el momento en el que se procesa una orden (p.e. en un horario en el que comúnmente el usuario se acuesta). El tiempo tiene una relación de generalización con el actor *Contexto*.
- ***Clima***: este actor representa la influencia del estado climatológico en la toma de decisiones. Entre estos estados se encuentra el estado actual del cielo (p.e. un cielo despejado) o la temperatura exterior (p.e. 27ºC). De la misma manera que los actores *Tiempo* y *Dispositivos*, este actor tiene relación de generalización con el actor *Contexto*.

Por último, en cuanto al diagrama se refiere, quedan reflejados una serie de casos de uso básicos que representan, a alto nivel, la funcionalidad que debe proporcionar el sistema, al margen de cómo será implementado cada uno de ellos. A continuación se detallan estos casos de uso:

1. ***Ver estado dispositivos***: el *Usuario* debe ser capaz de acceder a un listado de los dispositivos con su estado actual (en el momento de la consulta), por lo que *Dispositivos* participa en dicho caso.
    a. ***Apagar dispositivo***: se trata de un caso de uso extendido. Como añadido al listado, el *Usuario* debe ser capaz de apagar dichos dispositivos, pero de manera estrictamente opcional (siempre y cuando la opción esté habilitada, pues no se puede apagar un dispositivo que ya está apagado). Intrínsecamente, existe una relación de *inclusión* entre este caso de uso y *Cambiar estado dispositivo*[3].
    b. ***Encender dispositivo***: al igual que *Apagar dispositivo*, se trata de un caso de uso extendido que permite al usuario, de manera opcional, encender un dispositivo. De la misma manera, existe una relación de inclusión con el caso de uso *Cambiar estado dispositivo*.
2. ***Dar una orden***: este caso de uso refleja la funcionalidad más importante del sistema en cuanto a los objetivos del proyecto se refiere (desde el punto de vista del usuario). El *Usuario* debe ser capaz de dar una orden indirecta y el sistema, a través de la *Aplicación Android*, debe capturar dicha orden (pudiendo ser, además, una orden tanto vocal como escrita). Intrínsecamente (representado como inclusión), se deben

[3] Esta relación no ha sido añadida al diagrama de casos de uso con el objetivo de mantener la máxima claridad.

poner en marcha los procedimientos de selección de acciones (caso de uso *Selección de acción*).

a. ***Selección de acción***: es el caso de uso clave en el sistema, encargándose de seleccionar entre el conjunto de dispositivos aquel que ofrece una acción acorde a la orden indirecta del *Usuario*. Además, este caso de uso tiene una relación de inclusión y extensión con otro conjunto de casos de uso:

- ***Interpretar orden***: para la selección de la acción el subsistema *Controlador principal* debe hacer uso de un procedimiento de interpretación de la orden indirecta, el cual es representado por este caso de uso.
- ***Confirmar orden***: el sistema debe confirmar con el *Usuario* si la acción que va a ejecutar (aquella seleccionada) es acorde a la petición de este. Sin embargo, deben existir mecanismos que permitan configurar una ejecución directa sin la intervención del usuario, por lo que este caso de uso resulta una extensión que no siempre tiene por qué ocurrir en el transcurso de una ejecución.

3. ***Cambiar estado dispositivo***: como resultado del uso del caso *Selección de acción*, se producirá un cambio en el estado de los dispositivos de la red del *Hogar Digital*. Además, este cambio de estado puede producirse por factores externos a *SCAH* (aplicado directamente por el *Usuario*, tal como se mencionó anteriormente).

a. ***Actualizar información dispositivos***: este caso de uso representa la actualización del estado de los dispositivos en el subsistema *Controlador principal*. Esto debe ocurrir siempre que la red de dispositivos sufra algún cambio en su estado, pues será la información que se utilice en los futuros análisis.

El *diagrama de casos de uso* de la Figura 17 proporciona un primer enfoque de cómo se debe plantear el diseño e implementación del sistema en función de este conjunto de requisitos funcionales. Por un lado tenemos una serie de factores externos entre los que destaca el Contexto, englobando una serie de circunstancias que deben influir en la toma de decisiones. La utilización de la información contextual en la toma de decisiones es quizá el requisito funcional más flexible y a la vez ambiguo, pero también un factor diferencial y clave a la hora de aportar cierta inteligencia al sistema.

Por otro lado tenemos una división bien definida entre los casos de uso contenidos en cada uno de los subsistemas. Esto aporta también una primera visión del reparto de tareas entre subsistemas y la relación entre ellos, aunque la representación es meramente funcional.

Por último, es importante mencionar que, además del reparto de tareas entre subsistemas que ya se ha detallado al comienzo de este punto, se puede inferir un último requisito funcional más concreto: la interacción del usuario debe ser sencilla. Se puede observar como los usos que puede dar el usuario al sistema son muy limitados, pero será tarea del sistema internamente el dar una gran flexibilidad a dichos usos. Esta decisión de simplificar el acceso del usuario al sistema se toma en base a dos criterios:

1. El objetivo principal de este proyecto es el desarrollo de un cierto nivel de inteligencia artificial que, latente en el sistema, sea capaz de interpretar las órdenes del usuario. Para ello, sería suficiente la inclusión de un método de entrada de ordenes por parte del usuario; sin embargo, en un esfuerzo por intentar realizar un sistema más completo (véase apartado 3.1.2.1), se define un requisito más estricto que permita al usuario controlar desde la *Aplicación Android* el estado del conjunto de dispositivos.
2. Hoy en día, es tendencia en el desarrollo de aplicaciones móviles (y otro tipo de interfaces de usuario) la simplificación. En las recomendaciones de diseño (apartado *Simplify My Life*) proporcionada por el sitio web de desarrolladores Android [32] , se listan una serie de recomendaciones entre las que se encuentran las siguientes:
    a. Hazlo breve (del inglés *Keep it brief*)
    b. Las imágenes son más rápidas que las palabras (*Pictures are faster than words*)
    c. Decide por mí, pero déjame tener la última palabra (*Decide for me but let me have the last word).*

3.1.1.2 Requisitos no funcionales.

En este apartado se pretende detallar el conjunto de requisitos no funcionales, que serán divididos en diferentes puntos tal y como se detallará a continuación.

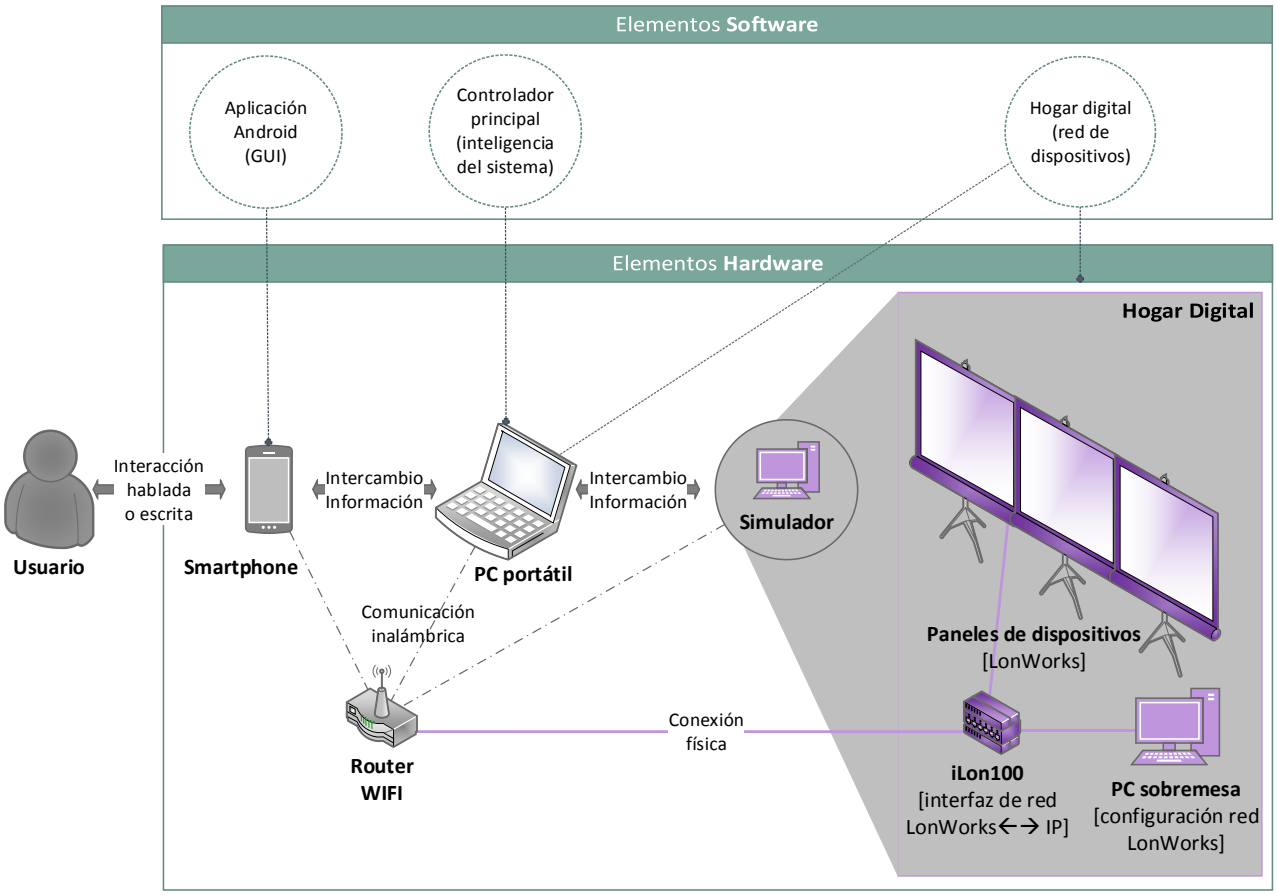


Figura 18. Elementos del sistema

Para la explicación de este apartado se tomará como referencia la Figura 18:

1. **Hardware y Software asociado**: para la implementación del sistema se limita el equipamiento Hardware a un conjunto de elementos que deben proporcionar la totalidad de la funcionalidad descrita en los requisitos funcionales. Estos elementos, tal y como se observa en la Figura 18, se dividen en cuatro bloques:
    a. **Smartphone**: se cuenta con un teléfono inteligente que corre el SO Android 4.4.2 (véase apartado 2.7). Haciendo uso de Android, el Smartphone será el encargado de contener la *GUI* con la que interactuará el usuario (mediante una aplicación). Esto proporciona movilidad y flexibilidad gracias a la plataforma Android y a los muchos servicios y recursos creados para ella.
    b. **PC Portátil**: en el núcleo del sistema estará corriendo el *Controlador principal* sobre un ordenador portátil cuyo SO es Windows 8.1. Se ha decidido implementar el *Controlador principal* mediante el lenguaje de programación **Java**, por su gran extensión y flexibilidad, y la gran cantidad de API y documentación disponibles. Además, su adaptabilidad a la gran mayoría de SO lo convierte en un lenguaje atractivo. Como añadido, el hecho de que la aplicación Android se programe mediante una combinación de Java y XML proporciona un motivo más para encontrar interesante el alineamiento de lenguajes, pues algunas API o mecanismos de control podrían ser reutilizados.
    c. **Hogar digital**: este bloque envuelve a un conjunto de elementos Hardware cuya finalidad, desde el punto de vista del sistema a desarrollar, se puede considerar como una: el control de los dispositivos LonWorks.
        - **Paneles de dispositivos**: este elemento presente en el Hogar digital consta de tres paneles representando un total de 5 habitaciones, cada una de ellas con una reducida cantidad de dispositivos. Toda esta red utiliza LonWorks como medio de interconexión y control de los dispositivos.
        - **iLon100**: se trata de lo que se conoce como servidor de Internet (*Internet Server) o* interfaz de servicios Web. Su función es la de servir de interfaz de red para interconectar una red LonWorks con una red IP o el propio Internet. Sería el nexo de unión entre el resto del sistema y la red LonWorks (véase apartado 2.1.1.1).
        - **PC sobremesa**: para la manipulación del comportamiento de la red LonWorks, este ordenador de potencia y recursos limitados (lo que lo hace poco apropiado para correr el *Controlador principal*) cuenta con las herramientas software necesarias para configurar el comportamiento de la red LonWorks (véase apartado 2.1.1.1).
    d. **Router Wi-Fi**: por último, se cuenta con un router modelo *Belkin G Wireless Router* con conectividad tanto Ethernet (4 puertos) como Wi-Fi, con estándar de conexión IEEE 802.11b/g. Este router serviría de nexo entre los tres bloques anteriores y, aunque físicamente se encuentra ubicado en el *Hogar*

Digital junto con la interfaz de red iLon100, en lo que respecta a este sistema se tomará como elemento común.

Se debe tener en cuenta, sin embargo, que en los objetivos definidos para este proyecto (véase apartado 1.1) se expone cómo el conjunto de elementos del *Hogar Digital* será reemplazado por un *Simulador* que lo represente convenientemente. Este *Simulador* permitirá la independencia completa en el desarrollo del sistema respecto a la topología y tecnología de la red de dispositivos, junto con un aumento significativo de su flexibilidad, tal como se detallará en el apartado 3.2.2.

No obstante, se procurará representar a lo largo de este documento como sería el diseño e implementación de los elementos del sistema si se hiciese uso de la red real disponible (red LonWorks). En cualquier caso, los puntos **c** y **d** no resultan determinantes en el desarrollo final del sistema pues, como se ha dicho, no se hará uso del hardware del *Hogar Digital* y, por extensión, se podrá implantar el sistema en cualquier red privada con conectividad Wi-Fi y acceso a Internet (cualquier router que cumpla estas características será válido en este desarrollo).

El *Simulador*, por su parte, podrá correr en cualquier equipo y no debería requerir grandes recursos hardware, pues su función es meramente representativa y el peso del procesamiento no debería recaer en él.

2. **Rendimiento**: en el ámbito del desarrollo de este proyecto no se considerará prioridad el rendimiento del sistema (véase apartado 3.1.2), sin embargo se define un reducido número de requisitos de rendimiento que afectan al diseño de dicho sistema:

    a. **Duplicidad de información de dispositivos**: analizando de nuevo la Figura 18, concretamente al conjunto de *Elementos Software*, se observa una relación entre el *PC Portátil* y el *Hogar Digital (red de dispositivos)*. Esta relación indica el requisito de que el PC Portátil almacene por algún mecanismo el estado de la red de dispositivos, con intención de evitar consultas innecesarias a la red cuando el estado de esta no haya cambiado respecto a la última consulta. Además, este estado actualizado debe mantenerse asíncronamente respecto al resto de procesos del sistema, es decir, cuando el usuario dé una orden el *Controlador principal* debe ya poseer la información actualizada (de hecho, debería actualizarse tan pronto como el estado de un cierto dispositivo cambie).

    b. **Balance de carga**: desde un punto de vista funcional, el *PC Portátil* no juega ningún rol que no hubiese podido soportar el propio *Smartphone* Android (entiéndase, ambos corren la máquina virtual de Java y disponen de interfaces de conexión Wi-Fi), por lo que a priori la inclusión del PC Portátil podría resultar innecesaria. Sin embargo, aportar un cierto grado de inteligencia artificial al sistema requiere de una gran cantidad de procesamiento. Con el objetivo de no ralentizar el *Smartphone* (que, aunque potente, está lejos de igualar la potencia del *PC Portátil*), lo que implicaría un empeoramiento

considerablemente de la experiencia del usuario con el sistema, se opta por dividir los roles tal como sigue:

- **Interfaz Gráfica de Usuario (GUI)**: el *Smartphone* se limitará únicamente a mostrar la información al usuario, recuperándola por los medios oportunos desde el *PC Portátil,* así como de capturar las órdenes de éste. No deberá haber mecanismos de decisión en la aplicación, sino una mera ilusión de inteligencia en cuanto a la perspectiva del usuario se refiere (véase apartado 3.1.2.1).
- **Inteligencia artificial**: todo el peso del procesamiento en el análisis de órdenes y toma de decisiones recaerá en el *PC Portátil*. Cada nueva orden será analizada en él y devuelta, ya procesada, al Smartphone, de manera que para el usuario será transparente la división entre estos dos subsistemas. De esta manera el *PC portátil* y, por tanto, el *Controlador principal*, se convierte en el cerebro del sistema, pero también en nexo de unión entre el resto de componentes, pues no deberá haber comunicación directa entre *Smartphone* y otro subsistema que no sea el *Controlador principal*. Éste, en su caso, decidirá si la información proveniente de la GUI debe ser encaminada a otro elemento o subsistema o si, por el contrario, puede ser procesada in situ o incluso descartada.

3. **Conectividad**: en el ámbito de la domótica y el hogar digital, el requisito de portabilidad y la ausencia de cableado, dentro de lo posible, pueden resultar esenciales en servicios como el que se pretende desarrollar en este proyecto. Es por ello que, tal como se muestra en la Figura 18, se define una comunicación inalámbrica entre los elementos *Smartphone*, *PC Portátil* y el bloque de elementos *Hogar Digital*, representados en la práctica por el *Simulador*.

   Ciertamente, el *PC Portátil*, de haber sido un servidor dedicado en exclusiva al servicio que se pretende desarrollar, podría perfectamente haberse situado de manera permanente junto al router y ser conectado mediante conexión física (cableado). Sin embargo, al tratarse precisamente de un PC portátil, se considera más adecuado implementar una comunicación inalámbrica y se establece tal condición como requisito.

   Por su parte, de haberse hecho uso de la red LonWorks del *Hogar Digital*, se hubiese mantenido la conectividad ya implantada (previamente al desarrollo de este proyecto), ya que se considera válida y suficiente la conexión física entre los elementos *Panel de dispositivos*, *iLon100*, *PC sobremesa* y *Router Wi-Fi*.

4. **Costo**: obviando el propio coste impugnable al equipamiento hardware y software listado anteriormente, el sistema debe ser desarrollado mediante el uso de recursos *software* gratuitos. Esto implica que cualquier mecanismo adicional (respecto a los ya mencionados) utilizado en el procesamiento de la información, ya sean APIs Web, librerías o entornos de desarrollo, deben ser de coste 0.

5. **Estabilidad**: el sistema debe presentar un nivel de estabilidad aceptable, si bien sólo se considera esencial un control completo en el lado del usuario. De esta manera, aquellos mecanismos de interacción con el usuario deben contemplar todos los eventos posibles para informar correctamente y dar al usuario una buena experiencia de uso.

Tal como se verá en el apartado 3.1.2, las limitaciones que acompañan a un proyecto de este tipo limitará la cantidad de recursos que pueden asignarse, principalmente en tiempo, a la implementación exhaustiva de ciertos requisitos. Por ello, los requisitos arriba mencionados son los únicos que se consideran de carácter obligatorio en cuanto al desarrollo de este proyecto, si bien en el apartado 3.1.2.1 se mencionan un conjunto de especificaciones que, dentro de lo posible, se procurarán respetar para alcanzar el mejor resultado posible.

3.1.2 Limitaciones propias del proyecto.

Dentro del marco de un Proyecto Fin de Grado como el presente surgen una serie de limitaciones tanto en tiempo como en medios, ya sean económicos o de infraestructura. Considerándose este proyecto ambicioso en su objetivo (véase apartado 1.1), se plantean a continuación un conjunto de *licencias* que limitan la funcionalidad de algunos de los componentes del sistema, pero que hacen viable el desarrollo del proyecto, así como la profundización en aspectos clave que de otra manera no podrían ser analizados e implementados con la exigencia que requiere:

1. **Estabilidad limitada**: tal como se ha visto en el apartado 3.1.1.2, se considera esencial que el control del lado de usuario sea completo, de manera que éste no se quede desinformado en caso de error y automatizando, dentro de lo posible, la nueva puesta en marcha del sistema si éste ha caído.

   Sin embargo, ante la gran cantidad de procesos y mecanismos que se deberán llevar a cabo en el sistema propuesto, proporcionar a todos ellos un control exhaustivo de errores y, lo que es más, mecanismos de restablecimiento automáticos, sería una tarea que consumiría una cantidad de recursos temporales de los que no se dispone.

   Por ello, se pondrá como requisito en el lado del *Controlador principal* el control de errores, capturando todos ellos y propagándolos convenientemente hasta un punto donde el administrador (o usuario, en su caso) pueda observar qué ha ocurrido. Sin embargo, todos aquellos procesos cuyos mecanismos automáticos de restauración ante errores sean de complejidad media o alta, no serán necesarios. Por supuesto, tal como veremos en el apartado 3.1.2.1, dichos mecanismos se implementarán siempre y cuando resulte posible en tiempo y forma.

2. **Acciones sobre dispositivos**: se considera suficiente con que el sistema, que ha de manejar tanto sensores como actuadores, sea capaz de realizar únicamente dos acciones sobre estos últimos: encendido y apagado. No se contemplan acciones intermedias ni proporcionales (p.e. una lámpara con porcentaje de luminosidad).

   Proporcionar al sistema un grado razonable de inteligencia artificial y, más concretamente, la capacidad de tomar decisiones sobre órdenes que pueden resultar

ambiguas (tanto más para un sistema informático), es una tarea compleja. Teniendo en cuanta los elementos que utiliza este sistema, incluyendo la propia interfaz de usuario y todos los mecanismos de comunicación entre subsistemas, se considera un objetivo suficientemente ambicioso el intentar aportar el nivel de raciocinio necesario al sistema para que sea capaz de decidir si apagar o encender un dispositivo en función de una orden indirecta y del contexto que envuelve dicha orden.

Además, de esta decisión puede extenderse con cierta facilidad una funcionalidad mayor, pues una vez decidido que, por ejemplo, se desea encender una lámpara (la cual supongamos que puede adquirir ciertos porcentajes de luminosidad), la mayor parte del proceso de decisión (descartando el resto de dispositivos y acciones) ya ha sido llevado a cabo, con lo que bastaría con volver a interrogar al usuario sobre el porcentaje de luz que desea aplicar en la lámpara ya seleccionada.

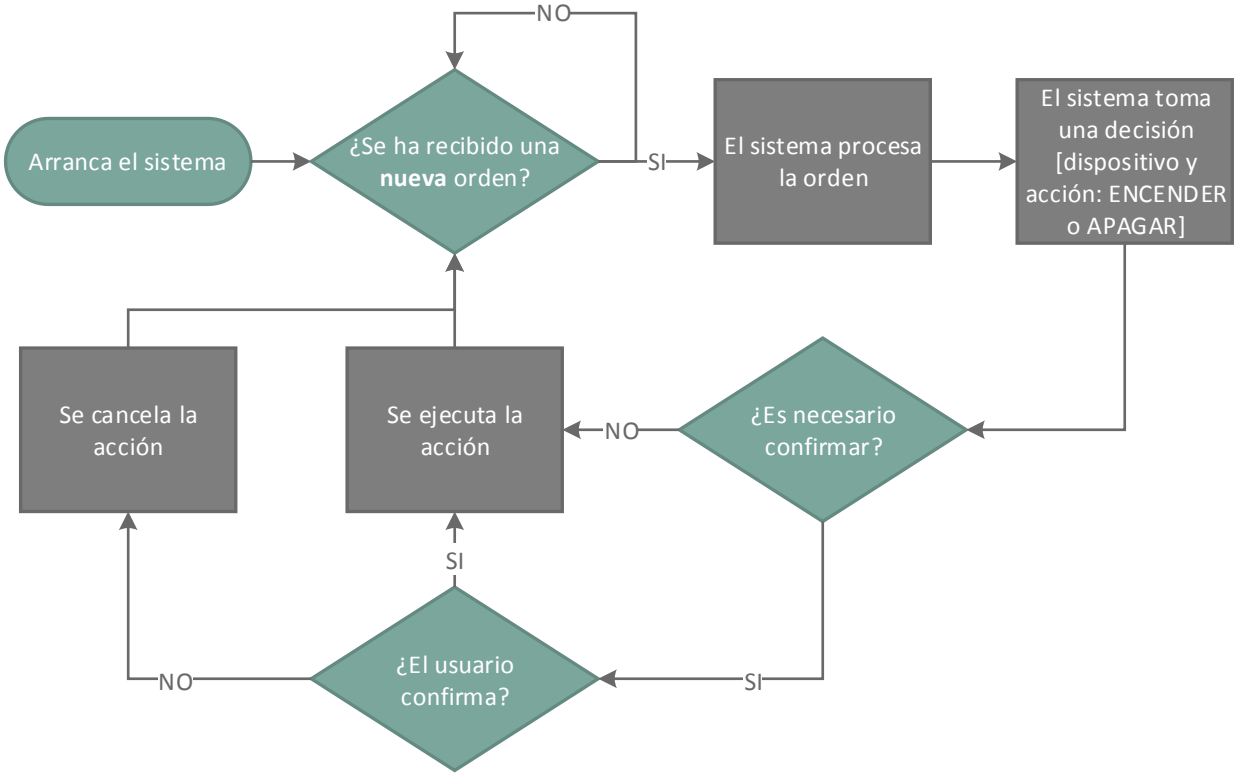


Figura 19. Diagrama de flujo (recepción y procesamiento de órdenes)

3. **Propuesta de acción única**: cuando el sistema tome una decisión a partir de una orden indirecta, el usuario será consultado (siempre y cuando no se haya configurado de otra manera). Si la acción propuesta es cancelada por el usuario, sea por el motivo que sea, el sistema no continuará con el proceso de la orden ni lo mantendrá en memoria. Es decir, si el usuario da una orden y el sistema no propone una acción atractiva para el usuario, el usuario tendrá que volver a realizar la petición y el proceso comenzará de nuevo. Esta situación se refleja en la Figura 19.

   Al igual que se ha mencionado para el apartado anterior, se considera que implementar un mecanismo de rectificación de conclusiones (en caso de cancelación

del usuario) conllevaría de un nivel importante de desarrollo adicional. Sin embargo, esta funcionalidad no aportaría ninguna componente de innovación relevante en cuanto a este sistema se refiere (pues se persigue que la orden sea interpretada correctamente sin necesidad de reiteraciones y eliminación de decisiones fallidas).

4. **Interfaz Android no extensible**: existe gran cantidad de documentación en Internet [37] enfocada a la adaptabilidad de las aplicaciones a diferentes versiones del SO Android y a la variedad de dispositivos que lo soportan: Smartphones, Tablets, SmartTVs, etc. En el mismo contexto de limitación temporal, se ha decidido ajustar la interfaz a las recomendaciones para la versión Android 4.4.2 (alineándolo con el Smartphone disponible) y no se garantizará que el formato y las proporciones de iconos, textos y otros elementos de la aplicación queden correctamente mostrados para la gran variedad de resoluciones existentes, limitando el diseño y *testing* al Smartphone disponible. En cualquier caso, la adaptación a diferentes dispositivos carecería de complejidad y, por otro lado, se encuentra fuera del ámbito de estudio del presente proyecto.

3.1.2.1 Especificaciones complementarias deseables.

Se listan a continuación una serie de especificaciones que se desean alcanzar pero que, por lo mencionado previamente, no pueden fijarse como requisitos:

1. **Producto final**: aunque obviamente el sistema que aquí se desarrolla no tiene como objetivo su comercialización, sino la innovación mediante la investigación y prueba de nuevos mecanismos de razonamiento mediante inteligencia artificial, se pretende realizar un desarrollo enfocado, dentro de lo posible, a un producto final que pudiese ser atractivo en el ámbito comercial. En otras palabras, fijando ciertos estándares de calidad y orientando el desarrollo a la experiencia de usuario tal como se haría en un producto comercializable, se cree que el sistema final desarrollado será de una calidad mayor y más próxima a un producto implantable en un entorno domótico real. Todo ello, por supuesto, teniendo en cuenta la limitación de recursos ya mencionada. Entre las especificaciones deseables se destacan dos:
    a. **Rendimiento**: tal como se ha mencionado en el análisis de los requisitos no funcionales (apartado 3.1.1.2), no se exige al sistema un alto rendimiento, dado que no es la finalidad de este proyecto. De hecho, uno de los objetivos previos al desarrollo es plantear nuevos mecanismos de interpretación de órdenes, cueste lo que cueste desde el punto de vista del procesamiento. Sin embargo, siguiendo el principio de *Producto final*, se procurará proveer al sistema de herramientas, procedimientos y estructuras que le permitan contener el volumen de tiempo necesario para realizar el procesamiento y ejecuciones oportunas. De esta manera, aunque el sistema no resulte óptimo en cuanto a rendimiento temporal se refiere, se procurará que los tiempos de ejecución se encuentren en límites aceptables mediante buenas prácticas en su diseño e implementación.

b. **Enfoque de usuario**: de nuevo, en referencia al diseño de un *Producto final*, se pretende dotar al sistema de mecanismos que permitan a un supuesto usuario (sin conocimientos avanzados del sistema) la configuración del comportamiento tanto de la interfaz de usuario como del propio sistema. Se procurará que, con la ayuda de un pequeño manual de usuario, el usuario final sea capaz de manipular el sistema de una manera intuitiva, sin necesidad de acceder al código y mecanismos subyacentes.

2. **Interoperabilidad**: se ha visto hasta el momento una descripción del sistema y sus requisitos, todo con la finalidad de tomar decisiones sobre un conjunto de dispositivos ubicados en una red accesible para el sistema. Sin embargo, el sistema que se desarrolla en este proyecto deberá interpretar la información de dispositivos de manera independiente al tipo de red. Se considera oportuno este enfoque dado que la interpretación de órdenes se debe basar en la información de los propios dispositivos, no del tipo de red que los aloja. Siendo esto así, parece razonable plantear un sistema en el que, para la mayoría de sus elementos, el tipo de red sea absolutamente transparente.

   Este enfoque permitirá al sistema operar en otros tipos de redes de dispositivos, siempre y cuando la información necesaria sobre dichos dispositivos esté disponible y pueda compartirse entre red y sistema. Este requisito será fácilmente alcanzable con la ayuda del *Simulador* expuesto previamente.

3.2 Análisis de conjunto.

Hasta el momento se han analizado en detalle aquellos requisitos que debe cumplir el sistema para proporcionar la funcionalidad propuesta mediante el uso de los recursos disponibles, así como un conjunto de especificaciones deseables que se procurarán implementar para alcanzar los objetivos definidos.

La finalidad de este apartado no es continuar con un análisis detallado de las funcionalidades y elementos del sistema de manera individual, sino el de analizar todos ellos en conjunto, procurando dar una idea global del sistema y, por otro lado, comenzar a vislumbrar las posibilidades y limitaciones de éste. Además, una visión global obligará a definir nuevas estrategias y particularizar el enfoque para alguno de los requisitos definidos en el apartado anterior.

3.2.1 Comunicación entre componentes

El análisis inicial del modelo de comunicación entre componentes del sistema ha sido previamente caracterizado (véase apartado 3.1.1.2), sin embargo no se ha definido con precisión qué tecnologías o protocolos de comunicación se implementarán.

Repasando brevemente los requisitos establecidos, se limita la conectividad de la siguiente manera:

- Conexión inalámbrica:
  - Entre el *Smartphone* y el *PC Portátil.*
  - Entre el *PC Portátil* y el *Simulador.*
  - Alternativa no implementada: Entre el *PC Portátil* y *Hogar Digital* (concretamente con la interfaz de red *iLon100*). Este enfoque ya ha sido realizado en el Proyecto Fin de Carrera de Roberto Augusto Cuenca [38], por lo que se opta por la simulación de dispositivos como alternativa.
- Conexión cableada (alternativa no implementada):
  - Entre la interfaz de red *iLon100* y el *Panel de dispositivos*.
  - Entre el *Panel de dispositivos* y el *PC de sobremesa*.

La conexión inalámbrica hará uso del router disponible mediante conexión Wi-Fi. No obstante, la interfaz de red *iLon100* estaría conectada mediante Ethernet a dicho router.

Como es bien sabido, las tecnologías tanto Ethernet como Wi-Fi definen estándares que implementan dos capas de la pila de protocolos OSI: *capa de enlace* y *capa física* (cuya representación suele unificarse en la pila de protocolos TCP/IP[4] como un único nivel denominado *capa de acceso a la red*). Por encima, ambos protocolos soportarán IP a nivel de la denominada *capa de Internet* (también conocida como *capa de red*).

Para el desarrollo de este proyecto se ha decidido implementar UPnP como tecnología de comunicación, por su orientación al descubrimiento y presentación de servicios (invocables desde dispositivos remotos), al mismo tiempo que proporciona un servicio de difusión de eventos y subscripción a los mismos (véase apartado 2.4). De esta manera, UPnP, framework que representa un conjunto de protocolos de la *capa de aplicación*, se apoyará sobre dos protocolos de la *capa de transporte*: TCP y UDP.

Respecto a la comunicación entre la interfaz de red *iLon100* y el *Panel de dispositivos*, LonWorks hace uso de su propio protocolo de comunicación denominado LonTalk, que implementa todos los niveles de la capa OSI (véase apartado 2.1.1.1).

En la Figura 20~~Figura 20~~ se pretende representar desde una visión de conjunto los protocolos y dispositivos involucrados en la comunicación del sistema. La funcionalidad de cada una de las capas del protocolo LonTalk ha sido previamente tratada y no se indica en la siguiente figura.

[4] Se tomará la pila de protocolos TCP/IP como referencia a lo largo del proyecto.

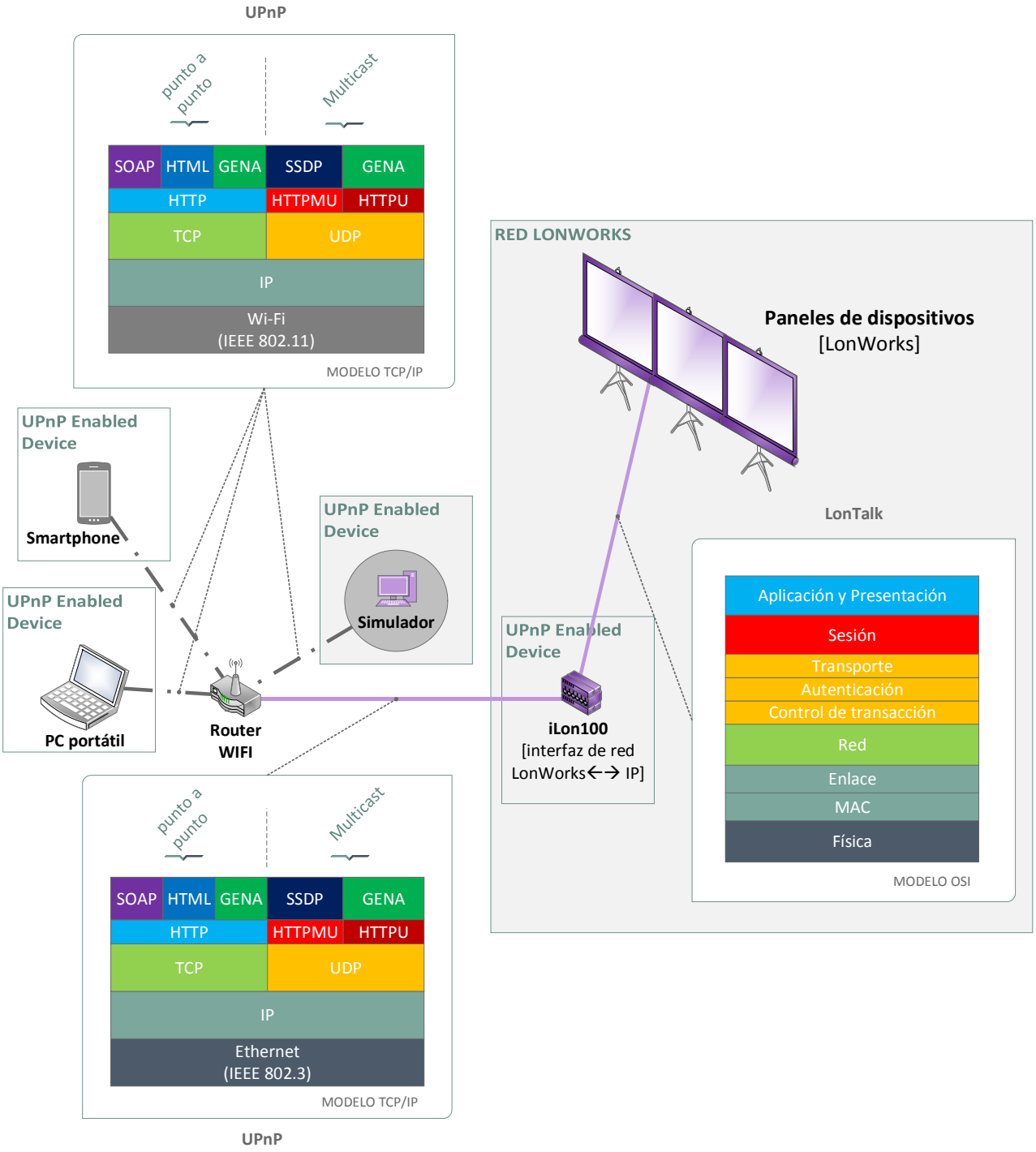


Figura 20. Modelo de comunicación entre elementos del sistema

Esta figura representa el tipo de información que viajará por los diferentes enlaces, indicando los protocolos involucrados en la comunicación. Por supuesto, apoyándose en los protocolos de aplicación viajará la información propia de la arquitectura UPnP y del sistema a desarrollar, pero por simplicidad no se ha reflejado en el modelo. Se puede observar que UPnP sobre IP será lo que prime en el sistema, pero que sin embargo la comunicación en la red LonWorks haría uso de un protocolo de comunicación propio, como ya se ha mencionado.

Ciertamente, la inclusión de UPnP como mecanismo de comunicación entre Smartphone y *PC Portátil*, o lo que es lo mismo, entre la *GUI* y el *Controlador principal*, podría resultar innecesaria (dado que es el *Controlador principal* quien debe manejar la información de los dispositivos). Sin embargo, los mecanismos de difusión de eventos de UPnP resultan útiles para informar a la GUI de que debe mostrar una nueva información, y el hecho de implementar esta tecnología en distintas partes del sistema lo convierte en una buena opción en la implementación de la comunicación en este punto, homogeneizando la red.

El modelo de la Figura 20~~Figura 20~~ identifica tres elementos que deben comportarse como dispositivos reales con UPnP habilitado (o como se le denomina comúnmente en inglés *UPnP Enabled Devices*), que no debe confundirse con el concepto de Dispositivo UPnP, el cual puede no ser un dispositivo real, sino un conjunto de servicios UPnP (véase apartado 2.4). Estos dispositivos deben compartir la información mediante una serie de servicios UPnP que deben exponer. Téngase en cuenta que de ahora en adelante se utilizará los términos *Hogar digital* y *Simulador* indistintamente, pues representan lo mismo para la gran mayoría de componentes: la red de dispositivos. Se contemplan varios modelos preliminares para la asignación de servicios UPnP, que quedan reflejados en la Figura 21:

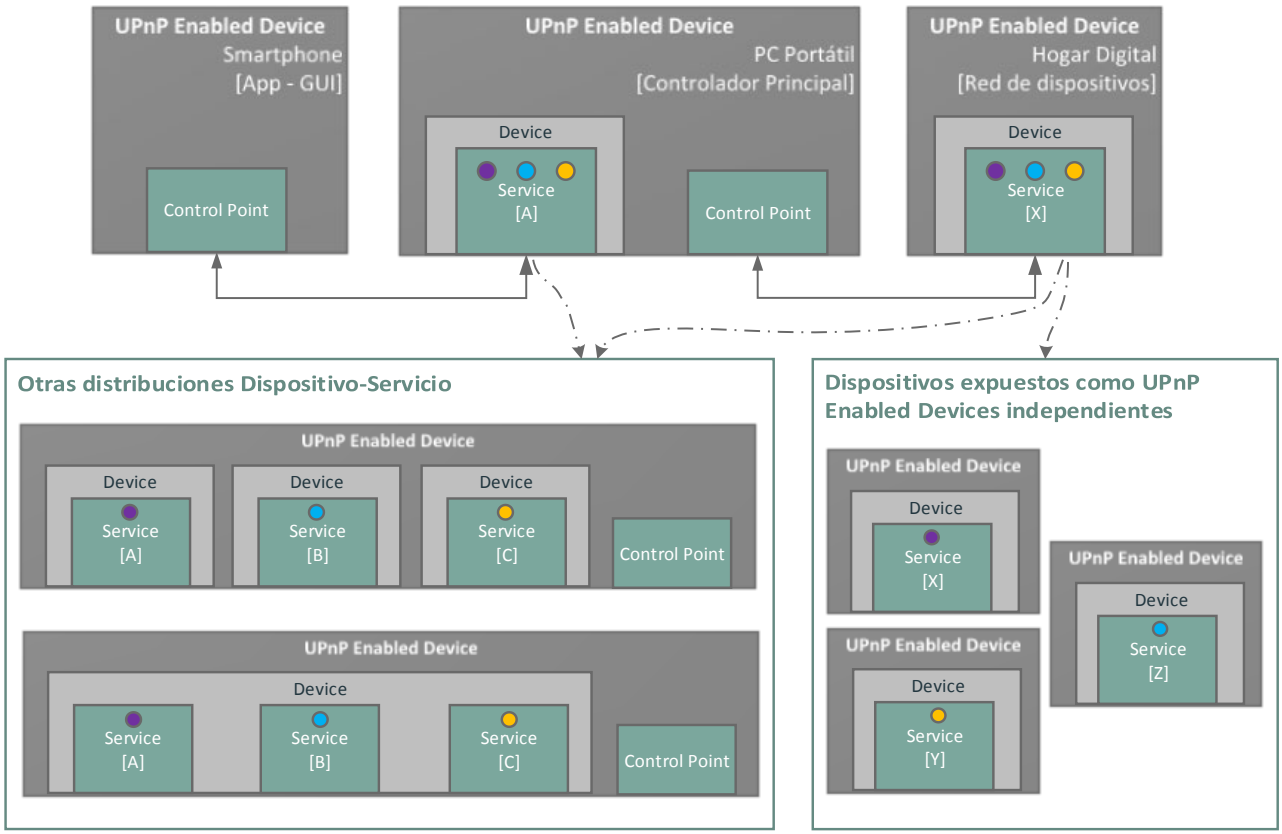


Figura 21. Modelo de Dispositivos-Servicios UPnP

La Figura 21 representa la distribución propuesta para la asignación de dispositivos y servicios UPnP sobre los elementos del sistema. Para su comprensión, debe tenerse en cuenta que los círculos de colores representan tres dispositivos cualesquiera del Hogar Digital (p.e. tres lámparas).

A la hora de implementar los dispositivos y servicios UPnP que van a permitir el control de los dispositivos del Hogar Digital, se pueden tomar diferentes enfoques:

1. Un *dispositivo* conteniendo un único *servicio* que permite actuar sobre el total de los dispositivos del hogar digital.
2. Un *dispositivo* y *servicio* por cada dispositivo del hogar digital.
3. Un *dispositivo* conteniendo varios *servicios*, cada uno de ellos controlando un dispositivo del hogar digital.
4. Asimetría en el conjunto de dispositivos del hogar digital contenidos en cada *servicio* (que también podrían estar a su vez repartidos de manera asimétrica entre diferentes *dispositivos*)[5].

Se decide optar tanto en el *Controlador principal* como en el *Hogar Digital* por el enfoque número 1, pues se ha considerado que con un conjunto muy limitado de acciones (con un alto grado de flexibilidad) se puede dar la funcionalidad fijada en los requisitos del sistema. Siendo esto así, un único servicio puede contener este breve número de acciones.

Se procede a un breve análisis de los elementos de la figura:

- ***Smartphone* (*GUI*)**: la aplicación Android únicamente implementará un Punto de Control (*Control Point*) UPnP, pues no ofrecerá servicios, sino que se limitará a hacer uso de los servicios expuestos por el *Controlador principal*. Estos servicios le permitirán obtener la lista de dispositivos con los correspondientes estados, además de invocar órdenes o acciones[6].
- ***PC Portátil* (*Controlador principal*)**: deberá implementar tanto un dispositivo UPnP (con su correspondiente servicio UPnP), como un Punto de Control para acceder a los servicios del *Hogar Digital*. Nótese que se ha representado dispositivos del hogar digital en el servicio expuesto en el *Controlador Principal*. Esto responde al requisito no funcional 2.a (apartado 3.1.1.2) que fija la necesidad de duplicar la información de los dispositivos en el *Controlador principal*. En lo que respecta a la *Aplicación Android*, el propio *Controlador principal* es el que contiene la red de dispositivos del *Hogar Digital*.
- ***Hogar Digital***: desde el punto de vista del sistema a desarrollar, no existiría diferenciación entre el concepto de *Panel de dispositivos* y la interfaz de red iLon100, sino que forman un único elemento: el *Hogar Digital*. Éste debería comportarse como un *UPnP Enabled Device* que exponga servicios sobre los dispositivos que contiene o bien ocultar al sistema su existencia de manera que cada dispositivo LonWorks hiciese las funciones de un *UPnP Enable Device* independiente (como si implementasen individualmente el conjunto de protocolos UPnP). Este el mecanismo desarrollado en el PFC de Roberto Augusto Cuenca [38].

  Para el *Controlador Principal*, la topología y tecnología utilizada en la red de dispositivos debe ser transparente. Siendo esto así, la utilización del *Simulador* no

[5] Esta opción no se ha reflejado en la figura por simplicidad.
[6] Se entiende *acción* a una ejecución sin interpretación (pulsar un botón) y *orden* como una frase del usuario a interpretar.

penaliza en ninguna medida el desarrollo del resto del sistema, pues la plataforma que se utilice para dar cabida a la red de dispositivos le debe ser indiferente y, por otro lado, el *Simulador* puede implementar perfectamente el dispositivo UPnP en cuestión, proporcionando la funcionalidad necesaria sobre la red de dispositivos de la que hace uso el sistema.

Como última nota respecto a este modelo, es importante destacar que se ha definido una comunicación entre elementos que obligatoriamente debe atravesar el *Controlador principal*, pudiéndose considerar a éste como núcleo y cerebro del sistema.

3.2.2 Simulación de dispositivos

La infraestructura de dispositivos de la que se dispone, concretamente el *Panel de dispositivos*, es sin duda alguna una gran herramienta a la hora de trabajar sobre el control de dispositivos reales mediante el uso de una tecnología muy extendida como es LonWorks. Este panel permite poner a prueba diferentes modelos de automatización y control, trasladando a un entorno real aquellos sistemas que tienen como objetivo el ámbito domótico.

Sin embargo, este proyecto plantea dos objetivos principales:

1. Integración y organización de la información de los dispositivos mediante el uso de ontologías.
2. Implementación de un cierto grado de inteligencia artificial que permita al sistema tomar decisiones en función de indicaciones del usuario y del contexto que envuelve el instante de ejecución.

Para el primero de los puntos mencionados, el panel de dispositivos junto con el resto de elementos definidos en el Hogar Digital proporciona toda la información necesaria para poner a prueba el sistema. No obstante, para el punto número dos una cantidad reducida de dispositivos podría limitar en gran medida tanto el desarrollo como la prueba del sistema. En un sistema que se pretende sepa interpretar órdenes para una cantidad de dispositivos muy amplia (para cualquier dispositivo que el usuario pudiese instalar en el futuro, idealmente), trabajar exclusivamente con los dispositivos instalados en el panel podría acotar excesivamente el marco de estudio.

Este análisis deja entrever la necesidad de implementar algún nuevo mecanismo que amplíe la cantidad y tipo de dispositivos con los que trabaja el sistema. Por todo ello, y considerando que la inversión en tiempo puede rentabilizarse mediante la obtención de un sistema mucho más flexible, se decide desarrollar un simulador que sea capaz de:

1. Representar a tantos dispositivos como se desee.
    a. Y además representar gráficamente dichos dispositivos.
2. Contener tanta información sobre los mismos como aquella que el *Controlador principal* vaya a manejar, así como su estado.
    a. No será necesario que contenga tanta información como un dispositivo real si dicha información no es de utilidad para el sistema a desarrollar.

3. Responder a acciones como lo haría un dispositivo real, cambiando su estado.
    a. Los dispositivos actuadores solo tendrán dos acciones (atendiendo a las *licencias* descritas en el apartado 3.1.2):
        i. Encenderse
        ii. Apagarse
4. Responder a interacciones directas por parte del usuario o desarrollador, es decir, habilitar botones que permitan ejecutar alguna de las acciones descritas en el punto anterior.
5. Organizar en habitaciones, dentro de la representación gráfica, los dispositivos creados.
6. Funcionar de manera transparente para el resto del sistema, como si de dispositivos reales se tratasen.

Las ventajas presentes en este enfoque son evidentes, pues no sólo se amplía la cantidad y tipo de dispositivos sobre los que poder tomar decisiones, sino que se cuenta con una representación de una red de dispositivos totalmente ajena a la tecnología subyacente. Este enfoque facilita el desarrollo de un sistema que no se diseñe para interactuar con una red LonWorks, sino con cualquier red de dispositivos, independientemente de las tecnologías y protocolos utilizados. Además, y no menos importante, permite trabajar con los dispositivos en cualquier momento y en cualquier lugar, lo que propulsa el proceso de desarrollo y pruebas. Se considera que contar con el *Simulador* puede resultar en la aceleración de procesos posteriores de desarrollo y, junto con la mejorada flexibilidad ya mencionada, resultar siendo un elemento clave en cuanto a los estándares de calidad propuestos en el apartado 3.1.2.1 se refiere.

Teniendo todo esto en cuenta, se representa a continuación el comportamiento deseable, desde un punto de vista de conjunto, de este componente:

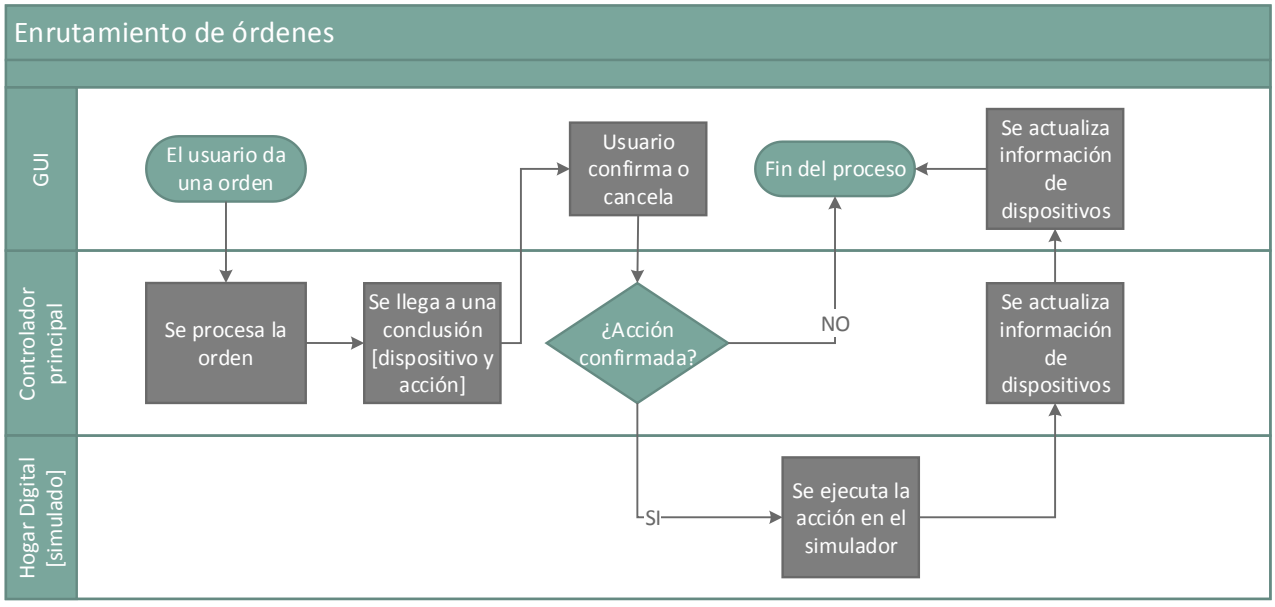


Figura 22. Diagrama de flujo (enrutamiento de órdenes hacia dispositivos)

Como se percibe en la Figura 22, el *Simulador* juega el rol del Hogar Digital. Se detallan a continuación algunos aspectos inherentes a este reparto de funciones:

1. **Acercar la información**. Tal como se definió en los requisitos no funcionales, el *Controlador principal* debe poseer en todo momento la información actualizada de todos los dispositivos de la red a manejar. Esto afecta positivamente al rendimiento tal como se explica a continuación.
2. **Rendimiento**. La comunicación mediante UPnP cuenta con numerosas ventajas, pero la velocidad no es una de ellas. Aunque en general la comunicación es muy rápida, el modelo de propagación de eventos puede ralentizar la comunicación en varios segundos en algunas ocasiones cuando esta tecnología trabajaba sobre Android, principalmente debido al modelo basado en servicios que requiere la aplicación (véase apartado 2.7.1). Por ello, manteniendo tanto el *Controlador Principal* como el *Hogar Digital* alineados de manera asíncrona respecto al instante de ejecución de las órdenes, este enlace de comunicación no es utilizado inmediatamente en el proceso de decisión (y se pospondrá hasta el momento en el que el usuario confirme una orden que deba ejecutarse en la red o el estado de la misma red haya cambiado por factores externos, como la ejecución directa de alguna acción).
3. **Reutilización de hardware**. Tal como se ha modelado el *Simulador*, a pesar de ser un componente completamente externo e independiente del *Controlador principal*, un mismo equipo podría albergar ambos y la independencia entre ellos no se vería afectada, pues el modelo de comunicación se mantiene. De la misma manera y si se considera oportuno, el *Simulador* podrá correr en otro equipo sin necesidad de ningún cambio adicional, siempre y cuando dicho equipo se encuentre en la misma red que el *PC Portátil* o se hayan implementado los mecanismos de enrutamiento apropiados entre diferentes redes (como podrían ser redes privadas virtuales). En cualquier caso, no es objetivo de este proyecto el estudio de alternativas a la red local y se tomará esta como modelo a utilizar en el desarrollo.

3.2.3 Integración de ontologías y contexto en el sistema

En este apartado y como último punto en lo que respecta al análisis preliminar del sistema, se pretende plantear a alto nivel un enfoque para la integración de estos dos conceptos que cimientan gran parte de los objetivos de este proyecto.

Como primer paso en este estudio, se definirán a continuación los roles que ambos mecanismos deben jugar en el sistema:

1. **Ontología**. Tal como se ha visto en el apartado 2.3, el concepto de ontología en el ámbito tecnológico se aplica principalmente a la representación de conocimientos y resulta especialmente útil en la implementación de inteligencia artificial. Su finalidad es la de estructurar la información atendiendo a la semántica, de manera que dichos conocimientos puedan ser fácilmente accesibles por el sistema, en parte o en su todo (el concepto o información que representa).

En resumen, se trata de un mecanismo de representación de conocimiento muy útil en sistemas informáticos. Desde el punto de vista de este sistema, existen dos dominios de conocimiento diferenciados cuya información se considera válida para su estructuración mediante el uso de ontologías:

a. **Dispositivos**. Resulta evidente la necesidad de almacenar y organizar de alguna manera la información referente a los dispositivos. Por necesidades del sistema, la información debería ser rápidamente accesible y estar estructurada de manera que permita localizar aquellos dispositivos que cumplan un cierto tipo de condiciones. Todo esto lo permite el uso de una ontología en la que se definan todas las características propias de los dispositivos que integran este sistema.

b. **Órdenes**. El usuario utilizará el sistema de manera que sus órdenes indirectas puedan ser interpretadas y en última instancia, aplicadas a los dispositivos. Sin embargo, lo que puede resultar obvio para un ser humano resulta extremadamente complejo para un sistema informático. Se ha considerado que la estructuración de la información de la orden mediante una ontología puede proporcionar una organización orientada al significado aprovechable para los objetivos de este proyecto. No obstante, su estructuración no será la de una ontología común, como veremos en el apartado de diseño, pues se trata de una información mucho más ambigua.

Como añadido, la utilización de ontologías basadas en el mismo modelo (en este caso ambas serán ontologías basadas en marcos o *frame-based ontologies*) simplifica los mecanismos de acceso y cruce de la información.

2. **Contexto**. Como uno de los objetivos del sistema y a su vez como un mecanismo esencial en la toma de decisiones, el contexto debe integrarse de manera que influya correctamente en la elección de unas acciones u otras. La representación de la información de contexto podría afrontarse desde muchos y muy variados puntos de vista, pues es información estática en cuanto al procesamiento de la orden se refiere: entiéndase, el contexto varía con el tiempo, pero una vez que se empieza a procesar la orden, el contexto en ese instante es uno y además puede considerarse invariable durante ese breve espacio de tiempo de ejecución. No obstante, surgen algunas necesidades a la hora de aplicar el contexto en un sistema como el que aquí se plantea:

   a. **Aplicación de reglas**. Aun poseyendo la información de contexto necesaria, inferir y aplicar su influencia en la toma de una decisión (ya sea reforzándola o descartándola) no es algo trivial. Se necesita obtener mediante algún mecanismo el peso de cierta situación sobre la posible decisión y, lo que es más, aplicar de algún modo dicho peso sobre el proceso de selección de la decisión final.

   b. **Influencia conjunta**. Cuando se pretende aplicar información contextual sobre la toma de decisiones, se entiende que el contexto no estará compuesto

por un único factor. Es importante plantear un mecanismo capaz de aunar la influencia de varios factores y aplicarla de algún modo sobre el proceso.

c. **Valores absolutos**. Otro problema latente cuando se pretende inferir la influencia de cierta situación en un sistema informático es el de intentar comprender qué significa un valor para el usuario. El hecho de, por ejemplo, obtener que la temperatura exterior es de 40ºC puede resultar un signo evidente de calor para un ser humano, pero no así para una máquina. Por decirlo de otro modo, 40 (grados) no es más que un número para el sistema. Utilizar este valor para influir de una manera razonable en el proceso de decisiones debe hacer uso de mecanismos de interpretación de valores absolutos, tratando de aplicarlos a sensaciones del usuario (que, además, no tendrían por qué ser las mismas entre diferentes usuarios).

A raíz de esta serie de necesidades, se ha decidido utilizar para modelar el contexto lo que se conoce como redes bayesianas, haciendo uso complementario de distribuciones probabilísticas (véase apartados 2.6). Estos modelos matemáticos permiten aproximar el correspondiente conjunto de datos contextuales a una serie de valores porcentuales dependientes entre sí, pudiendo además representar el peso de dicho contexto en el tipo de decisión a tomar.

Tal como se verá en el apartado 4.5, la salida de este tipo de redes es uno o varios valores numéricos, por lo que se deberá implementar mecanismos adicionales para que dichos valores sean aplicables al proceso de decisión. Además, si se quiere poder adaptar el contexto a diferentes usuarios (y sus diferentes percepciones), se deberían ofrecer mecanismos de configuración que permitan aproximar de una manera más personalizada la información contextual, influyendo en cómo se trata la información en las redes de contexto implementadas.

Se muestra la Figura 23 una breve visión de los roles de ambos conceptos en el sistema y, más concretamente, en la toma de decisiones:

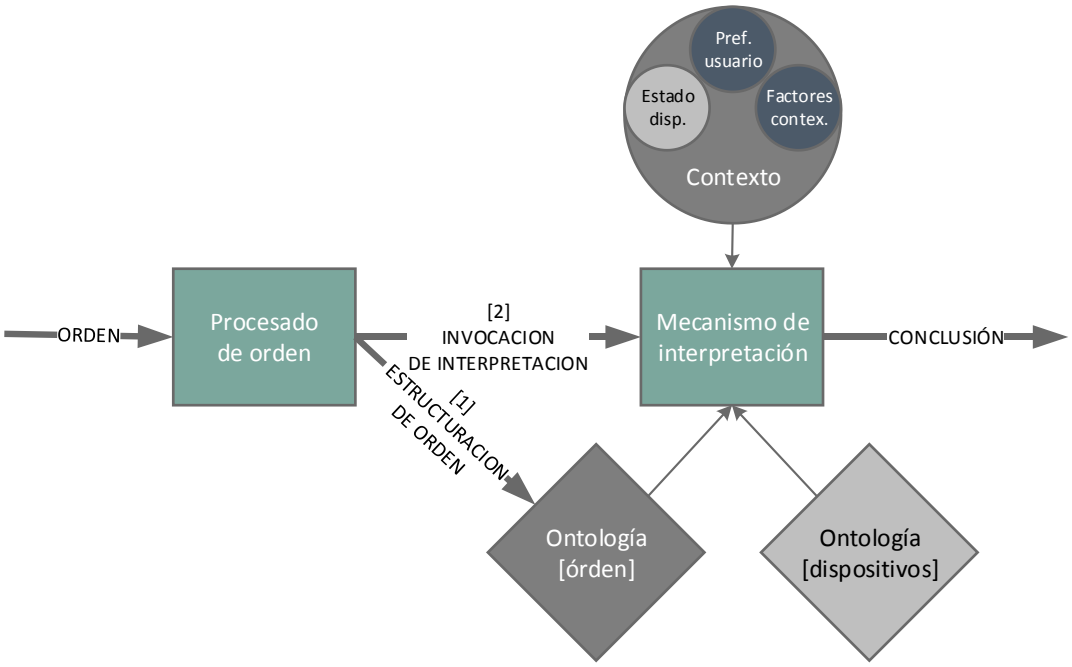


Figura 23. Ontologías y contexto en el proceso de toma de decisiones

En esta figura se muestran algunos aspectos importantes en lo que respecta al papel que desempeñan estos elementos en el conjunto:

1. Cuando una nueva orden se recibe, el sistema debe ejecutar de manera ordenada dos procesos bien distintos:
    a. Se debe procesar la orden para estructurar la información contenida en ella en la ontología de la orden, pues como se observa, dicha información será utilizada en el mecanismo de interpretación.
    b. Una vez el paso anterior se haya completado, se debe invocar el proceso de interpretación, pero la ontología de la orden debe estar previamente formada.
2. La información de dispositivos manejada tanto por el contexto como por la ontología de dispositivos debe estar completamente alineada. De hecho, se elige a la ontología como maestro de los datos de dispositivos (la cual deberá mantenerse actualizada), y será el contexto quien obtenga la información de dicha ontología en el instante justamente anterior al comienzo del procesado de la información contextual correspondiente.
3. Deberá existir un componente en el sistema capaz de:
    a. Obtener la información actualizada tanto de las ontologías como de los resultados extraídos del procesado del contexto.
    b. Cruzar toda esta información de manera que a la salida se tenga una o varias conclusiones basadas en ella.

    A este componente, identificado en la figura como *mecanismo de interpretación*, se le denominará de ahora en adelante el *intérprete*. Este componente será un elemento

base del sistema y, por motivos que se expondrán a continuación, debe tener un alto grado de flexibilidad.

Deteniéndonos en el punto 3 visto anteriormente, surge una última reflexión que será crítica en el desarrollo del sistema, tal como se verá en el capítulo de desarrollo: para lograr un sistema realmente adaptable a los diferentes dispositivos que pudiesen añadirse a la red y sobre los que el sistema debería tomar decisiones, resultaría muy limitado si la implementación del intérprete se definiese de manera invariable. Dicho de otro modo, el intérprete debería ser capaz de enfocar la toma de decisiones de diferentes maneras, en lugar de contar con una programación estática que no permita la modificación de sus mecanismos de decisión.

A continuación se describen ejemplo que permite fundamentar la idea descrita en el párrafo anterior:

Imagínese un hogar digital que cuente con 3 dispositivos que el sistema puede manejar: una lámpara en el salón, un radiador en la cocina y un televisor en el dormitorio (*entorno A*). Ahora supóngase otro hogar digital, esta vez con únicamente 3 persianas y 3 lámparas en el salón (*entorno B*.

En el entorno A, al tratarse de dispositivos que manejan diferentes conceptos (iluminación, temperatura y multimedia) y situados en diferentes habitaciones, resulta evidente que resultaría óptimo primar la información de la habitación o el tipo, dándole más peso en la toma de decisiones. De esta manera, otorgando más peso a las relaciones entre la habitación a la que se refiere el usuario y la habitación de cada dispositivo podría alcanzarse una buena conclusión, y lo mismo ocurre con los diferentes conceptos.

No obstante, en el entorno B, al ser todos los dispositivos de iluminación y ubicados en la misma habitación, el sistema tendría que intentar evaluar más circunstancias, como podrían ser el nombre del dispositivo, el estado actual de cada uno de ellos, etc., lo que exigiría más procesamiento al sistema.

Dicho esto, se considera una buena práctica el desarrollo de mecanismos adicionales que permitan al intérprete evaluar el peso de los diferentes factores de manera adaptable según las indicaciones del usuario o el desarrollador correspondiente. Para ello, se propone dotar al sistema de la capacidad suficiente para leer un cierto algoritmo (de ahora en adelante el *algoritmo del intérprete* o el *algoritmo de interpretación)* que indique cómo y en base a qué deben asignarse los pesos de los diferentes factores en el proceso de toma de decisiones.

El diseño e implementación de todos los elementos y requisitos mencionados a lo largo de este capítulo será detallado en el siguiente, siempre teniendo como base los principios y enfoques ya definidos en este análisis.

Capítulo 4

Diseño e implementación

El análisis previo del sistema ha definido una serie de requisitos y enfoques que deben satisfacerse en el diseño final, junto con otro conjunto de especificaciones deseables que se procurarán cubrir a lo largo del desarrollo (véase apartado 3.1.2.1). Este punto tratará de aclarar todos los detalles de dicho diseño, identificando como se afrontarán los diferentes retos inherentes al sistema.

A lo largo de este capítulo se procurará no ahondar en los conceptos teóricos (cuyos puntos más importantes han sido definidos previamente en el Capítulo 2) ni tampoco en los requisitos ya analizados en el Capítulo 3, sino que se intentará dar un punto de vista más práctico y enfocar los esfuerzos en representar el sistema final tal como ha sido diseñado e implementado.

Por último, antes de comenzar con el contenido de este capítulo, se considera importante aclarar que la descripción de los procesos llevados a cabo por cada elemento, así como el intercambio de información entre ellos, pretende proporcionar una descripción conceptual detallada del funcionamiento del sistema, pero no se explicará cómo esta información es procesada o intercambiada a bajo nivel (es decir, no se detallará cómo ha sido codificada la solución salvo que el contexto lo requiera, y la nomenclatura realmente utilizada podría variar respecto a la indicada en esta memoria).

4.1 Modelo de desarrollo

Tal como se ha definido en el análisis previo, se ha decidido implementar un *Simulador* que permita la interacción con dispositivos simulados y funcione de manera independiente a la red que se encuentra instalada en el *Hogar Digital*, separando completamente el desarrollo del sistema de la topología y tecnología utilizadas en la red de dispositivos. Siguiendo este enfoque, se dividiría todo el desarrollo en tres fases:

- **Fase 1**. Diseño e implementación del *Simulador*. Una vez acabada esta fase, el sistema contará con la infraestructura de dispositivos que requiere para el funcionamiento de las siguientes fases.
- **Fase 2**. Diseño e implementación del mecanismo de toma de decisiones. El Controlador Principal, que será el núcleo del sistema y principal objetivo de este proyecto, será el encargado de llevar a cabo esta tarea. Con el simulador funcionando, se podrán llevar a cabo las pruebas unitarias pertinentes en los que se refiere a la IA y al control de dispositivos.
- **Fase 3**. Diseño e implementación de la Interfaz Gráfica de Usuario (GUI), contenida en la aplicación Android. Con esta fase finalizada el sistema contendrá el conjunto de funcionalidades requeridas.

Desde el punto de vista de los objetivos de este proyecto, la mayor parte del peso del desarrollo recae en la segunda fase, pues engloba los principales retos planteados en este proyecto en cuanto a innovación se refiere. No obstante, el desarrollo de la primera y tercera fase será esencial también para el correcto funcionamiento del conjunto del sistema.

Se describe a continuación, punto por punto, el diseño final con su correspondiente fundamentación para cada uno de los elementos y mecanismos esenciales del sistema. El modelo

de intercambio de información entre los diferentes elementos se tratará en el apartado de implementación (véase apartado 4.10).

En primer lugar se definirá independientemente cada elemento, identificando qué aporta al sistema y cómo se ha decidido implementarlo. Es posible que en algunos casos su papel no quede completamente clarificado en esta primera descripción, pero finalmente se representará el sistema en conjunto, procurando dar un repaso completo a la secuencia de ejecución y el papel que cada uno de estos elementos juega en ella.

4.2 Inteligencia en el sistema

De entre todos aquellos objetivos propuestos para el sistema que se pretende desarrollar en este proyecto hay uno que supone el verdadero reto en lo que a innovación se refiere: proveer al sistema de inteligencia artificial. Concretamente, el sistema procurará interpretar órdenes del usuario, siempre asumiendo que este quiere realizar una acción sobre la red de dispositivos, y obtener una conclusión (selección de dispositivo y acción) que se adapte a dicha orden.

La creación de un mecanismo que sea capaz de llevar a cabo dicho objetivo obliga a responder previamente a un conjunto de preguntas que ayuden a trasladar a un sistema informático una cualidad propia de los seres humanos.

4.2.1 Proceso deductivo en el diseño de la IA

Se exponen a continuación una serie de preguntas en el mismo orden en el que se plantearon en el proceso de diseño y desarrollo, junto con las respuestas que se han dado a partir de las conclusiones obtenidas. Se parte desde una pregunta muy general, descendiendo y profundizando a partir de las respuestas:

¿Cómo respondería un ser humano a una orden[1] o sensación[2] como las que recibirá el sistema?

La imitación de organismos biológicos a la hora de afrontar un nuevo reto en computación es una práctica habitual. Al estudiarse aquí órdenes habladas y, por tanto, el lenguaje, el único candidato es el ser humano. Dicho esto, se definen dos ejemplos para representar a la sensación y a la orden: *hay muy poca luz en esta habitación* y *levanta la persiana* respectivamente. Supóngase además a un hombre tras los muros de la casa, escuchando las órdenes y ejecutándolas él mismo como si del propio sistema se tratara, una especie de impostor.

- **Orden**: parece evidente que el mero hecho de escuchar las palabras *levantar* y *persiana* sería más que suficiente. En una orden tan clara como ésta, tanto la acción como el dispositivo quedan identificados, ¿o no? Lo cierto es que podría haber varias persianas en la misma habitación y, lo que es más, ¿de qué habitación se está hablando?
- **Sensación**: en este punto la interpretación no resulta tan evidente. Desde el punto de vista del hombre escondido tras los muros, se puede suponer que el conjunto de

[1] En este apartado, se entiende orden como una indicación concisa de lo que quiere el usuario (p.e. “apaga la calefacción”).
[2] En este apartado, se entiende sensación como la expresión de una idea o sentimiento enfocado al hogar (p.e. “hace mucho frío”).

> palabras *hay*, *poca luz* y *esta habitación* podrían resultar suficientes para identificar lo que debe hacer (supóngase que deduce que debe encender una lámpara). Sin embargo, en un análisis más profundo, evitando tomar nada como evidente (pues para el sistema no lo será), surgen nuevas preguntas: ¿cómo sabe el hombre que encender una lámpara es lo que requiere la situación?, ¿cómo relaciona el concepto de *luz* al de lámpara?, ¿Por qué se decanta por la lámpara y no por una persiana?

La cantidad de preguntas que surgirían si discutiésemos todos los aspectos de esta decisión en profundidad es casi ilimitada. Se cambia a continuación ligeramente de enfoque para analizar nuevos aspectos respecto al análisis anterior:

¿Habría otras maneras de expresar tanto la orden como la sensación de manera que el hombre llegase a la misma conclusión?

La respuesta es sí, evidentemente. Por ejemplo, la orden podría haber sido *sube la persiana* y el hombre habría alcanzado la misma conclusión. Lo mismo ocurre con la sensación, en la que *en esta habitación tenemos poca luz* contendría el mismo significado pero con una orden e incluso palabras diferentes. Yendo más lejos, incluso órdenes e sensaciones totalmente diferentes podrían requerir la misma acción, como por ejemplo *esta habitación está muy oscura*. De nuevo, varías preguntas surgen, que se resumen en la siguiente: ¿cómo relaciona el hombre tantas y tan variadas estructuras gramaticales con un mismo concepto?

Dando una vuelta más a este análisis, se exponen nuevas preguntas: ¿Si el hombre fuese en realidad un niño de, por ejemplo, 3 años, sería capaz de realizar las mismas interpretaciones? Y lo que es más, ¿obtendría las conclusiones tan rápido como el adulto? Probablemente no. Es muy posible que un niño de 3 años no sepa relacionar ciertos conceptos a los dispositivos que le acompañan, como podría ser encender el aire acondicionado si el usuario manifiesta que tiene calor.

Pero, ¿cómo puede ser que, poseyendo ambas personas un cerebro humano capaz de razonar, no alcancen una misma conclusión de la misma manera? Lo cierto es que si en lugar de un niño de 3 años se hubiese supuesto una persona de la misma edad (30 años) ajena a la tecnología y que jamás hubiese visto una bombilla, la diferencia habría sido incluso mayor.

Tras todas estas preguntas con sus correspondientes respuestas, se alcanza una conclusión que será absolutamente esencial en el planteamiento del proyecto. Esta característica que el hombre debe poseer para satisfacer todas las preguntas es la *experiencia*. La experiencia es el factor diferencial entre el primer hombre planteado y el niño (o el hombre ajeno a la tecnología). El hecho de que el hombre sepa de manera prácticamente inmediata qué debe hacer se basa a la cantidad de veces que a lo largo de su vida ha visto como otras personas realizaban esta acción y la necesidad se resolvía, junto con las veces que él mismo la llevó a cabo.

No se pone en duda que el hombre ajeno a la tecnología, tras un poco de investigación y pruebas, pudiese haber llegado a la misma conclusión, pero su rendimiento habría sido muy inferior y la necesidad del usuario no hubiese sido satisfecha apropiadamente. Además, la próxima vez que el usuario dijese algo parecido, el hombre ajeno a la tecnología (ahora un poco menos ajeno) haría uso de su experiencia para llevar a cabo la acción necesaria inmediatamente.

Por su parte, la interpretación de la propia orden o sensación en sí es un elemento complejo pero necesario, pues sin la compresión de a qué se refiere el usuario es imposible buscar una solución. Es evidente que el cerebro humano posee la capacidad de organizar las palabras en función de su estructura morfosintáctica y darle significado incluso si el orden o las palabras varían. Sin embargo, en muchas ocasiones el mero hecho de haber escuchado una y otra vez una expresión puede darle sentido sin necesidad de buscarle una interpretación racional (es algo habitual en el uso de refranes o incluso aprendiendo idiomas diferente al materno).

Por tanto, como concepto que engloba una respuesta para la gran mayoría de las preguntas, se establece la *experiencia* como algo necesario que aportar al sistema.

Por otro lado, hay un conjunto de preguntas que la experiencia por sí sola no responde. El hecho de que el hombre decida, por ejemplo, encender la lámpara en lugar de levantar la persiana tiene que atender a alguna otra razón. El motivo podría ser el hecho de encontrarse en una franja horaria en la que no haya luz exterior, como por ejemplo la media noche. Además, existen otros motivos probables tales como el hecho de que la persiana ya estuviese levantada o que simplemente esa habitación no cuente con persiana.

Para este último conjunto de preguntas se presenta una nueva conclusión que de nuevo será requerida en el sistema y que por otra parte ya ha sido planteada tanto en los objetivos como en el propio análisis: el *contexto*. Una misma orden debe ser interpretada de diferentes maneras en función de qué situación la envuelva. Por ejemplo: en un contexto en el que existen dos habitaciones, ambas con persianas (una de ellas bajada y otra subida), la orden *levanta la persiana* implicará levantar la persiana que esta bajada, incluso aunque el usuario se encuentre en la habitación contraria. Por su parte, si ambas persianas están bajadas deberá tenerse en cuenta a la hora de tomar una decisión dónde se encuentra el usuario u otros factores como la cantidad de luz ya presente en cada una de las habitaciones.

4.2.2 Implementación de la IA

Tal como se ha visto en el punto anterior, se ha llegado a la conclusión de que el sistema debe manejar el contexto (lo cual fue analizado previamente en el apartado 3.2.3) además de establecerse la necesidad de proporcionar experiencia al sistema de algún modo. En este punto se pretende resolver estas dos cuestiones detallando el diseño del modelo final que se ha implantado en el sistema.

Sin duda alguna, la inclusión de experiencia en el sistema es algo extremadamente complejo. El desarrollo de un sistema de aprendizaje, así como el de una base de datos relacional diseñada para almacenar conocimientos tan heterogéneos que permitan llevar a cabo tan ambiciosa tarea probablemente resultaría una utopía para este proyecto. Incluso en un caso ideal en el que dicho desarrollo se alcanzase, el sistema requeriría de una gran cantidad de tiempo para ampliar esa experiencia y algún mecanismo que le permitiese aplicarla.

Teniéndose presente las limitaciones propias del proyecto (véase apartado 3.1.2), se propone encontrar una alternativa simplificada y accesible con los recursos disponibles de manera que, dentro de lo posible, permita alcanzar los requisitos funcionales propuestos en este proyecto.

A partir de estas premisas surge la pregunta y a la vez la respuesta clave del proyecto y sin duda con más peso en el diseño del sistema:

¿Qué base de datos sobre el lenguaje humano existe en el mundo más completa que Internet?

Internet se nutre cada día con una cantidad de datos casi inimaginable, y en el proceso los seres humanos dejan huella constante de su lenguaje, sus expresiones y sus ideas. Esta información, además, queda reflejada de manera escrita (ideal para un sistema informático) como si de un almacén gigante de experiencias se tratase. Blogs, foros, noticias… ¿A caso no es esto aprovechable en este sistema? La respuesta es sí, por supuesto.

Las preguntas que quedan por resolver son cómo acceder a dicha información y, una vez accedida, como utilizarla en el proceso de decisión.

Para la primera pregunta, la respuesta es *los motores de búsqueda.* No sería viable para el sistema recorrer un conjunto suficientemente amplio de páginas web, ya sean predefinidas o aleatorias, que le diese una visión real de la experiencia humana, dado que el rendimiento sería inaceptable y no habría garantías de tener una buena representación del conjunto de Internet. Recorrer un número limitado de ellas tampoco sería una solución, pues no daría una imagen suficientemente amplia. Sin embargo, los buscadores realizan análisis reiterativos de todo Internet, indexando la información de manera que sea rápidamente accesible por los usuarios (véase apartado 2.5). Además, por lo general, los buscadores tienden a devolver un número mayor de resultados cuanto más común es lo que se busca (entiéndase común como la presencia muy extendida en gran cantidad de páginas y documentos de Internet).

De esta última característica hará uso el sistema, tal como muestra la Figura 24.

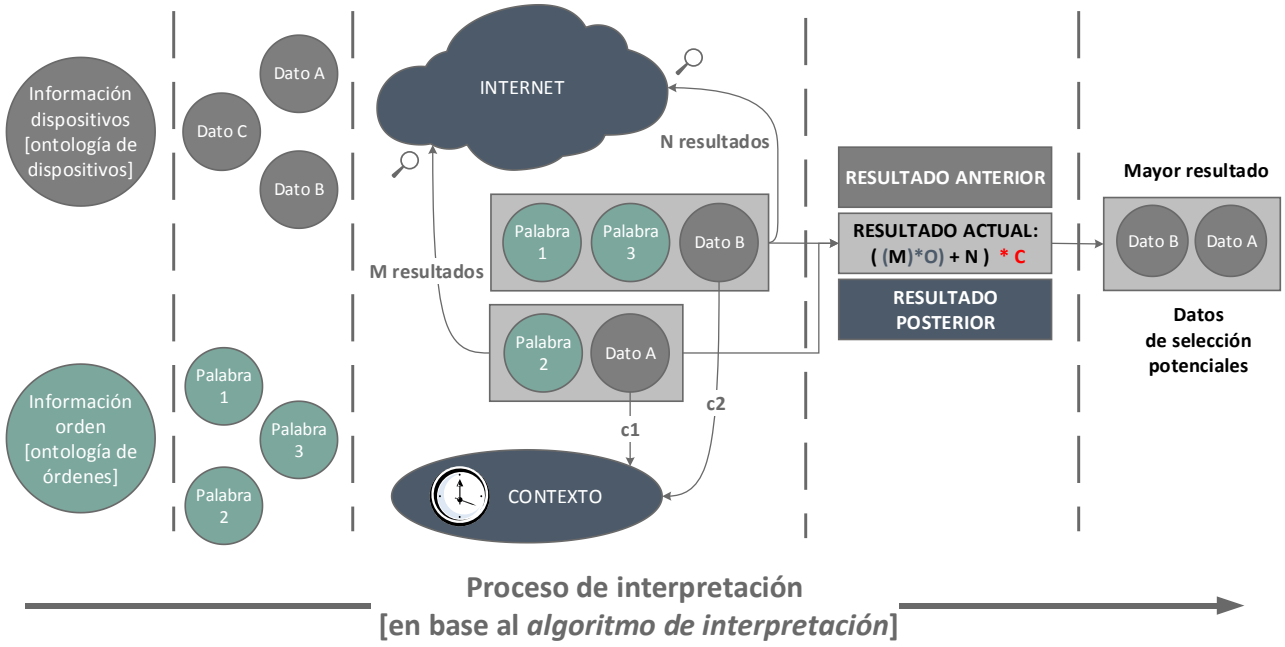


Figura 24. Proceso de interpretación

Tal como se observa en la figura, el proceso de interpretación lleva a cabo diferentes pasos que se detallan a continuación. Para ello, supongamos que la orden en cuestión es *enciende la lámpara del baño*:

1. **Extracción de la información clave**: el primer paso consiste en la extracción de la información relevante para el análisis. Como se representó en la Figura 23, la orden será previamente estructurada en la *ontología de órdenes*, y de la misma manera la información de los dispositivos deberá permanecer ordenada en su correspondiente ontología.

    a. En base al algoritmo de interpretación (que se explica en detalle en el apartado 4.6), extraerá de las ontologías la información relevante para el análisis. Para este ejemplo simplificado[3] supongamos que el algoritmo define dos cruces cualesquiera:

        i. "VERBO SUSTANTIVO concepto[4]" [x100.000]

        ii. SUSTANTIVO habitación

        Las palabras en mayúsculas representan datos de la ontología de órdenes, las minúsculas a datos de la ontología de dispositivos. Cada campo debe rellenarse con todos los valores de ese tipo que existen en la ontología correspondiente, formando diferentes combinaciones. Se trata de un ejemplo meramente representativo ya que el algoritmo final deberá ser considerablemente más elaborado y tener en cuenta más factores.

        Se debe tener en cuenta que las búsquedas de términos exactos devuelven muchos menos resultados, por lo que deben dimensionarse como se considere más apropiado.

2. **Formación de combinaciones**: estas combinaciones darán diferentes resultados en función de los dispositivos presentes en la red y la orden de usuario. Suponiendo que la red cuente sólo con dos dispositivos, una *lámpara en el baño* (con concepto *luz*) y un *radiador en el salón* (con concepto *temperatura*), los cruces quedarían como sigue (se recuerda que la orden era *enciende la lámpara del baño*, con un verbo y dos sustantivos*)*:

        i. "VERBO SUSTANTIVO concepto" [x100.000]

            a. "ENCIENDE LÁMPARA luz"

            [Cruce de la información de la orden –verbo y sustantivo (1º)- con la información de la lámpara –concepto-]

[3] En adelanto al apartado sobre el algoritmo, se aclara que términos encerrados entre comillas serán buscados identificando documentos o páginas que contengan el mismo orden de palabras en alguna frase. El número encerrado entre corchetes representa un factor opcional a aplicar al resultado final.

[4] La palabra *concepto* representa aquel concepto sobre el que actúa un dispositivo. Por ejemplo: una lámpara actúa sobre la *iluminación* o *luz*, que sería lo que se entiende como concepto, ya sea reduciéndola o aumentándola. El concepto del radiador, por su parte, se podría definir como la *temperatura*.

b. "ENCIENDE LÁMPARA temperatura"

[Cruce de la información de la orden –verbo y sustantivo (1º)- con la información del radiador –concepto-]

c. "ENCIENDE BAÑO luz"

[Cruce de la información de la orden –verbo y sustantivo (2º)- con la información de la lámpara –concepto-]

d. "ENCIENDE BAÑO temperatura"

[Cruce de la información de la orden –verbo y sustantivo (2º)- con la información del radiador –concepto-]

ii. SUSTANTIVO habitación

a. LÁMPARA baño

[Cruce de la información de la orden –sustantivo (1º)- con la información de la lámpara –habitación-]

b. LÁMPARA salón

[Cruce de la información de la orden –sustantivo (1º)- con la información del radiador –habitación-]

c. BAÑO baño

[Cruce de la información de la orden –sustantivo (2º)- con la información de la lámpara –habitación-]

d. BAÑO salón

[Cruce de la información de la orden –sustantivo (2º)- con la información del radiador –habitación-]

Para este ejemplo se obvian por simplicidad otras acepciones que formarían más combinaciones (p.e. *baño* por sí mismo bien podría entenderse como primera persona del presente indicativo del verbo bañar, es decir, como una forma verbal).

3. **Obtención de puntuación (presencia en Internet)**: el algoritmo dictará como se debe lanzar la consulta, dado que los motores de búsqueda permiten diferentes tipos (mediante el uso de la correspondiente sintaxis).

   Por ejemplo, se puede hacer una búsqueda por coincidencia exacta, dando por válido únicamente los documentos o páginas web que contienen esa correlación de palabras. También se puede indicar, entre otras alternativas, espacios que puedan ser completados por cualquier valor dentro de una frase. Todo este conjunto de opciones permite hacer búsquedas que nos den una idea de cuan común es la relación entre los conceptos que forman la consulta.

   De esta manera, suponiendo el ejemplo planteado, el buscador devolvería una serie de resultados que se muestran a continuación (se ha utilizado Bing para su obtención):

a. "VERBO SUSTANTIVO concepto" [x100.000]
   i. "ENCIENDE LÁMPARA luz" → 25 x100.000 → 2.500.000
   ii. "ENCIENDE LÁMPARA temperatura" → 4 x100.000 → 400.000
   iii. "ENCIENDE BAÑO luz" → 0 x100.000 → 0
   iv. "ENCIENDE BAÑO temperatura" → 0 x100.000 → 0

b. SUSTANTIVO habitación
   i. LÁMPARA baño → 14.300
   ii. LÁMPARA salón → 35.000
   iii. BAÑO baño → 46.700.000
   iv. BAÑO salón → 11.600.000

4. **Aplicación del contexto**: aunque no se tratará en este ejemplo por simplicidad (se verá en detalle en el apartado 4.5), el peso obtenido del contexto se aplicará al valor de salida de los pasos anteriores, eliminando si es necesario algunas opciones tales como encender una dispositivo que ya está encendido[5] o incluso aplicar preferencias sobre ciertos tipos de dispositivos (por ejemplo, reduciendo el valor de levantar una persiana si es de noche, pues el concepto sobre el que actúa, la iluminación, no aumentaría mediante esta acción).
5. **Identificación del dispositivo con mayor puntuación**: como último paso se deben unificar las puntuaciones del proceso atendiendo al dispositivo que las representa. De esta manera, cada dispositivo obtiene del cruce de sus puntuaciones los siguientes valores:
   a. *Lámpara en el baño*:
      2.500.000 + 0 + 14.300 + 46.700.000 = 49.214.300
   b. *Radiador en el salón*:
      400.000 + 0 + 35.000 + 11.600.000 = 12.035.000
6. **Consulta y ejecución**: el dispositivo seleccionado es la *lámpara del baño*, por adquirir la mayor puntuación. Dada la simplicidad de este algoritmo, no se ha cruzado la información del estado (encendido o apagado) y este no ha ejercido peso en la selección, de manera que ejecutará la acción que pueda, encenderse o apagarse, según su estado actual. El usuario será quien tenga la última palabra a la hora de ejecutar o no la acción, siempre que lo haya configurado así.

Tal como se ha indicado, este es un ejemplo muy simplificado respecto al algoritmo que requiere el sistema para obtener un buen índice de aciertos en sus elecciones, pero representa fielmente cómo mediante un algoritmo que indique qué y con qué peso deben cruzarse conceptos estáticos, tanto de la orden como de los dispositivos, puede alcanzarse una conclusión correcta.

[5] En realidad, las eliminaciones se aplican antes del paso de consulta (evitando las innecesarias y mejorando el rendimiento), pero se ha considerado en este punto más entendible el orden de exposición mostrado.

Además, se puede percibir como un algoritmo que es válido para un conjunto de dispositivos sería insuficiente para otro, pero también se vislumbra como un algoritmo más complejo requiere de muchas más combinaciones y consultas, reduciéndose su rendimiento drásticamente (en cuanto a tiempo de ejecución se refiere).

He aquí el motivo por el que se requiere un sistema capaz de procesar diferentes algoritmos. Por un lado, se necesita que el sistema se adapte rápidamente a nuevas formas de procesar las órdenes (en busca de la óptima) y, dada la imprevisibilidad del contenido en Internet, permita un proceso de depuración de prueba y error sin grandes esfuerzos de adaptación por parte del desarrollador.

En un entorno de implantación real a gran escala (siguiendo las especificaciones complementarias deseables del apartado 3.1.2.1), se podrían poner al servicio de los usuarios algoritmos prediseñados y testados en función de los dispositivos y distribución de habitaciones de viviendas tipo, permitiendo al usuario además modificarlo a sus necesidades y particularidades.

En definitiva, lo que se ha pretendido mostrar en este punto es que el análisis del número de ocurrencias que tiene la relación de dos o más conceptos (p.e. *luz* y *lámpara*) en Internet puede proporcionar al sistema esa experiencia necesaria para comprender a qué se puede referir el usuario dentro de la red de dispositivos y, junto con el análisis del contexto, tomar una decisión que procure resolver las necesidades dicho usuario.

4.3 Estructuración del conocimiento

En este apartado se define la estructura que deberán utilizar las ontologías a la hora de organizar y jerarquizar la información, además de los mecanismos o recursos que utilizarán a la hora de seleccionar la información a cargar en dichas ontologías, lo que se conoce como crear instancias. Como se ha descrito previamente, se contempla el uso de dos ontologías, las cuales serán denominadas de ahora en adelante *ontología de órdenes* y *ontología de dispositivos*.

Aunque existen diferentes tipos de ontologías ya definidas enfocadas a estructurar conocimientos tales como los referentes a una red de dispositivos, se ha considerado que un sistema como el que aquí se plantea requiere de flexibilidad absoluta a la hora de estructurar la información (principalmente aquella que pretende almacenar información referente a la orden de usuario). Además, existen algunos factores adicionales a tener en cuenta:

1. Se considera muy recomendable que ambas ontologías utilicen el mismo modelo de conocimiento para que el acceso a ellas y el cruce de información procedente de ambas sea más eficiente. De esta manera, ambas ontologías utilizarán el modelo que pueda representar a la más restrictiva.
2. No se contempla en este proyecto la externalización de la información contenida en las ontologías.
3. Se pretende simplificar las ontologías al máximo, representando solo la información que vaya a ser utilizada por el sistema. Se ha comprobado durante el desarrollo que este factor influye significativamente en el rendimiento del sistema.

En virtud del punto 2 y 3, utilizar en este proyecto lenguajes de ontologías extendidos tales como OWL se considera ineficiente desde el punto de vista de la funcionalidad propuesta para este sistema. Para el desarrollo del proyecto que aquí se presenta se ha decidido utilizar ontologías basadas en marcos, utilizando como herramienta de desarrollo el software Protégé-Frames.

Entre las ventajas de este enfoque se destaca la flexibilidad en el desarrollo, el ajuste de la ontología a los requisitos concretos de este sistema y la simplificación del proceso de definición gracias a la herramienta Protégé y a la baja complejidad de este tipo de ontologías. Además, este software está escrito en Java, lo que permite importar las librerías oportunas al sistema en desarrollo (también escrito en Java) para manejar la información que contiene y utilizar todas sus herramientas. Esta característica resulta esencial en la implementación del sistema y comprende uno de los motivos principales de su elección como editor.

Por último, y antes de comenzar con la descripción detallada de las ontologías desarrolladas en este proyecto, se aclara que se afrontará su exposición desde un punto de vista funcional, estableciendo en detalle las estructuras y mecanismos lógicos utilizados, si bien no se planteará en este apartado cómo se definen en el software Protège las ontologías expuestas. Para más información sobre el modo de funcionamiento de este software, véase el apartado (véase apartado 2.3.2.1).

4.3.1 Ontología de dispositivos

La base de conocimiento a definir en la ontología de dispositivos puede modelarse asumiendo que un dispositivo contendrá una serie de características similares a otros dispositivos de su mismo tipo. A partir de los requisitos definidos y a la base de intentar limitar al máximo su complejidad, se diseña la ontología de dispositivos tal como se muestra en la Figura 25.

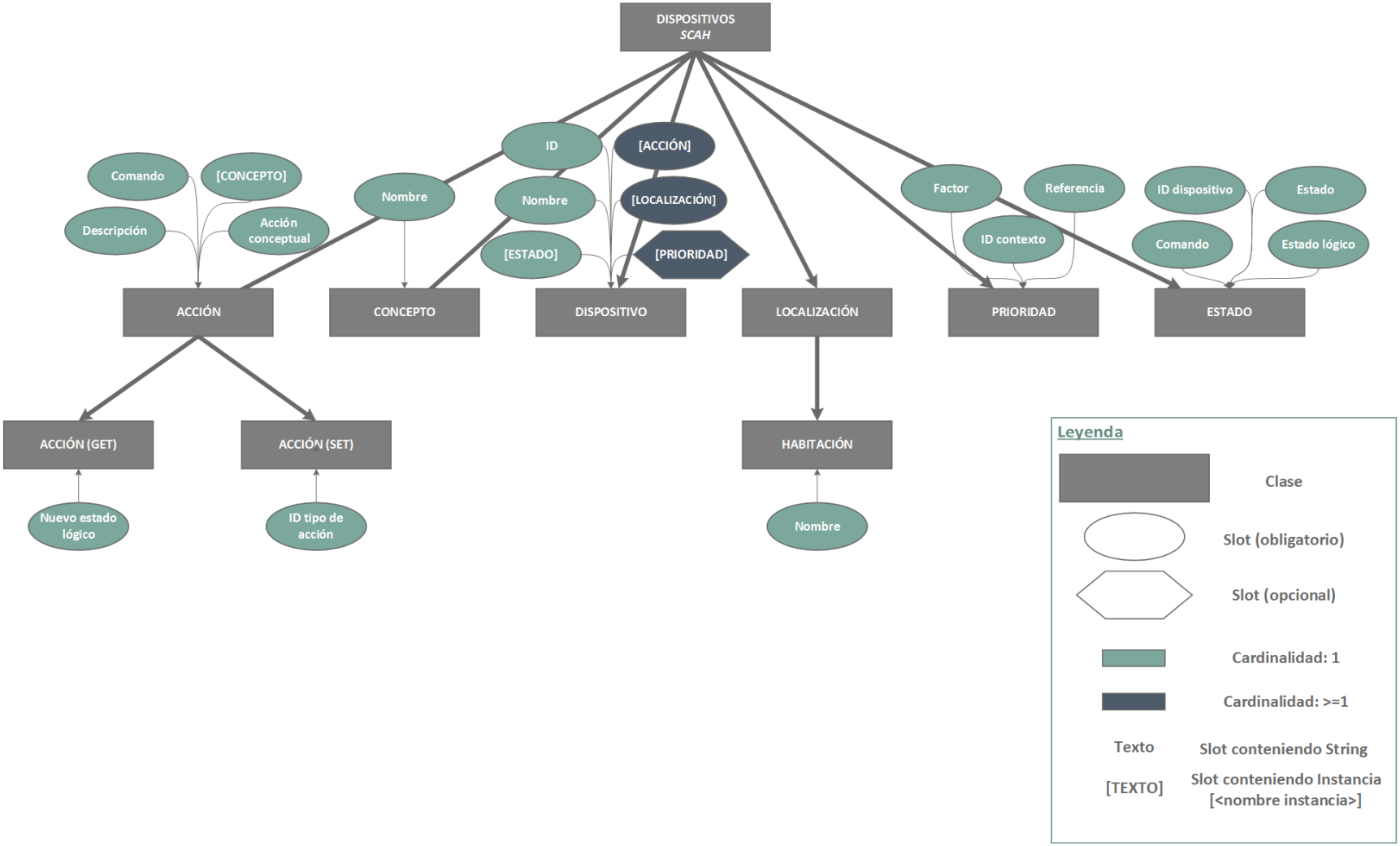


Figura 25. Ontología de dispositivos

Como se observa, se definen tres niveles de clases en la ontología, siempre persiguiendo la máxima simplicidad. El conjunto de slots (atributos) y facetas (restricciones) se ha elegido considerando la información esencial que puede ser requerida en el *algoritmo del intérprete*. En este modelo ontológico (*frame-based*) las facetas se definen fácilmente limitando la cardinalidad y el tipo de un slot. Por razones de diseño, se decide que todo slot que no sea de tipo *Instancia de otra clase* sea de tipo String (cadena de texto) y será el sistema el encargado de interpretarlo y transformarlo cuando corresponda. Dada la heterogeneidad de la información de los dispositivos, no se definen slot al primer nivel (del que heredarían todas las clases).

El diseño de una ontología nunca es único y cerrado, sino que multitud de posibilidades podrían ser válidas y los retoques en la estructura podrían seguir sucediéndose indeterminadamente. La elección de este modelo se sustenta en la intención de definir el mínimo número de atributos, y aunque la mayor parte de las clases se definen a segundo nivel, teniendo en cuenta los tipos definidos en los slots la estructura lógica quedaría como se muestra en la figura Figura 26:

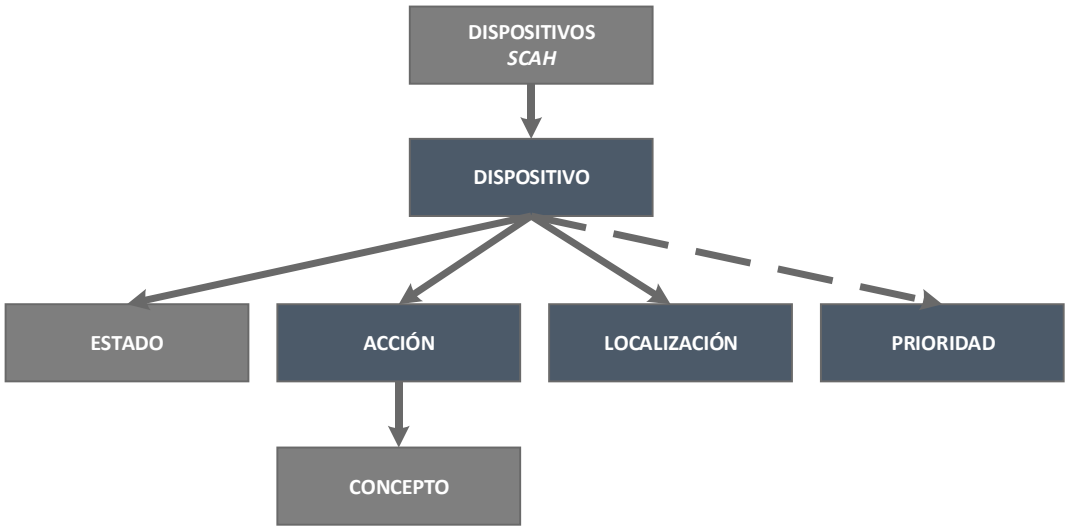


Figura 26. Modelo lógico de la ontología de dispositivos

En la jerarquía, toda clase por debajo de otra (a un nivel inferior) hereda sus slots y facetas. Mediante la combinación de ambas figuras se procede a explicar brevemente la estructura:

La ontología guardará tantas instancias de dispositivos como dispositivos haya en la red. Cada dispositivo queda definido por un nombre y un identificador, junto con el siguiente conjunto de atributos:

- **Estado**: representa en qué estado se encuentra un dispositivo (encendido o apagado) y que debe ser único. El estado tiene atributos que indican el dispositivo que puede alcanzar dicho estado, el comando que se requiere ejecutar para alcanzarlo y el estado lógico que representa (ON / OFF).

  Se podría pensar que los atributos que identifican el dispositivo no serían necesarios dado que se puede trazar la dependencia (el dispositivo del que cuelgan) y así es, pero por cuestión de rendimiento en la implementación del sistema algunos procesos se

simplifican si el estado contiene esta información, reduciéndose el tiempo de ejecución.

- **Acción**: un dispositivo puede ejecutar un conjunto de acciones, la cuales se aplican sobre un **Concepto** determinado (p.e. sobre la *iluminación*). La acción queda definida por una descripción breve (p.e. encender), un comando (secuencia que identifica a la acción inequívocamente), y la definición conceptual de la acción (sobre el *concepto*, es decir, *aumentar* o *reducir* la presencia de dicho concepto)
    - Existen además dos tipos de acciones definidas en el tercer nivel que representan acciones del segundo nivel con pequeñas particularidades:
        - Las acciones de tipo SET representan aquellas que lleva a cabo un actuador
            - Su acción implica un cambio de estado, representado mediante su estado lógico ON/OFF
        - Las acciones de tipo GET representan aquellas que lleva a cabo un sensor
            - Estas acciones contienen un identificador del tipo de acción (p.e. *detectar usuario* en un detector de presencia que indica si está detectándolo o no en un momento preciso)
- **Localización**: un dispositivo está ubicado en uno o varios puntos (p.e. un dispositivo *calefacción* puede aplicar su acción sobre varias estancias). Aunque para el sistema sólo se ha contemplado la habitación al trabajarse con un entorno doméstico, se ha mantenido la estructura conceptual de que un dispositivo podría estar ubicado en distintos tipos de localización como un piso completo o una zona exterior.

  Tal como se ha desarrollado el sistema, si se desea introducir un dispositivo presente en varias habitaciones (o incluso todas), se pueden añadir dichas habitaciones individualmente. Estas habitaciones están definidas por su nombre.
- **Prioridad**: un dispositivo puede tener una serie de prioridades o bien no tener ninguna. Una *prioridad* indica que este dispositivo tiene prioridad respecto a otros que manejen el mismo concepto siempre y cuando se cumplan unas ciertas condiciones. Si esas condiciones no se cumplen la elección de este dispositivo debe ser penalizada. Las prioridades se definen por la situación contextual en la que se aplican, la referencia a tener en cuenta y el factor a aplicar.

  Por ejemplo, una persiana debería tener prioridad sobre una lámpara en un día soleado al medio día, pero no debería tenerse muy en cuenta por la noche o en un día nublado. Por tanto, la situación contextual a tener en cuenta sería la *iluminación exterior*, la referencia sería el hecho de que ésta fuese *alta* y el factor el que se considere oportuno, el cual, aunque será independiente al algoritmo del intérprete, será configurable y jugará su papel en el proceso de interpretación.

4.3.2 Ontología de órdenes

Tal como se adelantó en el diseño de la inteligencia artificial, las particularidades y ambigüedades de una orden hablada desde el punto de vista de un sistema informático requiere de una estructura algo más peculiar.

Existen trabajos previos orientados al reconocimiento del lenguaje hablado con la intención de inferir cual es el significado subyacente, es decir, a qué conceptos se está refiriendo el orador.

En [39], Marwa Graja, Maher Jaoua y Lamia Hadrich Belguith desarrollan un sistema de interpretación basado en ontologías y que realiza el procesamiento en dos pasos: anotación semántica e interpretación semántica. En este *paper* se menciona cómo realizan una primera evaluación de las palabras o conjuntos de palabras, identificando los posibles conceptos asociados, lo que corresponde al primero de los pasos. En el segundo paso, se analiza la relación entre conceptos consecutivos, todo ello apoyado por el dominio aplicado a cada concepto.

En [40], Silvia Quarteroni, Marco Dinarelliy Guiseppe Riccardi trabajan en un sistema capaz de analizar los predicados de las oraciones pronunciadas y relacionar las acciones que hacen referencia a los diferentes conceptos. Además, hacen hincapié en definir un mecanismo capaz de relacionar conceptos de diferentes dominios. En este *paper* se desarrolla una ontología que define las diferentes características gramaticales de una oración, inclusive una clase abstracta para los conceptos.

La ontología de órdenes desarrollada en este proyecto, no obstante, no se ha diseñado con la intención de discernir a través de ella los conceptos clave de la oración para extraer una conclusión única. Esta ontología pretende únicamente organizar las diferentes palabras que componen la oración de manera que exista una relación de dependencia semántica entre ellas. Esta dependencia no se analiza a través de la propia ontología, sino que tal como se explicará a lo largo de este capítulo, será la información que proporciona los motores de búsqueda la base de esta clasificación, lo que simplifica en gran medida la complejidad requerida en el diseño de esta ontología.

El fin último de esta ontología no es otro que almacenar las palabras de una orden de manera que aquellas con más relación semántica aparezcan en más ocasiones y, por tanto, puedan ser tratadas de manera que a lo largo del proceso de decisión resulten tener un peso mayor.

A lo largo del desarrollo de este proyecto se plantearon diferentes estructuras, llegándose a la conclusión de que para alcanzar los requisitos definidos de la manera más simplificada posible y siempre teniendo en cuenta el diseño del proceso de interpretación de la IA, se debía crear una ontología ideada en exclusiva para facilitar dicho proceso de interpretación. De esta decisión surge una ontología que desde un punto de vista ajeno al sistema podría no tener un significado claro ni representar conceptos fácilmente entendibles. Sin embargo, sí se ha enfocado este diseño de manera que cumpla una de las principales funciones de una ontología, la estructuración semántica de la información.

Para este diseño de la ontología de órdenes se ha tenido muy presente cómo será accedida y utilizada su información. Su finalidad es la de almacenar las palabras que componen una orden de manera que existe una relación de dependencia entre ellas. La pregunta que surge en este punto es cómo estructurar estas palabras.

Tal como se ha analizado en el diseño de la IA, una orden podría contener diferentes conceptos e ideas que incluso podrían haber sido organizados de manera diferente en otra orden equivalente. De la misma manera, podría contener palabras irrelevantes, pues una orden puede ser muy elaborada y contener en realidad una información muy simple. Dicho esto, se ha procurado diseñar una estructura que almacene sólo las palabras que frecuentemente cuentan con más peso en una orden y que, por otro lado, contenga diferentes significados a partir de las distintas dependencias entre las ellas.

El hecho de reducir las palabras candidatas para el análisis mejora considerablemente el rendimiento, y la búsqueda de posibles estructuras alternativas permite reforzar ciertos conceptos tal como se explicará a lo largo de este punto. Se comienza mostrando el modelo de la ontología en la Figura 27:

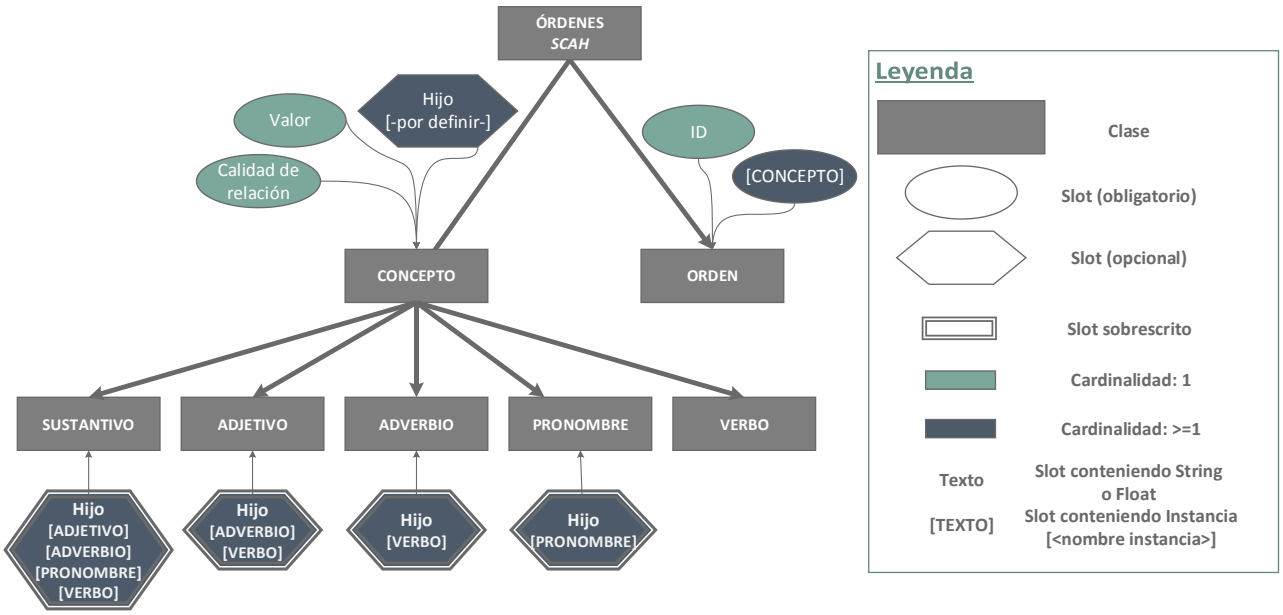


Figura 27. Ontología de órdenes

Al igual que ocurría en la ontología de dispositivos, se definen tres niveles de Clases, siendo el primer nivel aquel del que heredan el resto (se ha decidido no asignar slots a este nivel). En el segundo nivel se definen dos clases bien diferenciadas:

- **Orden**: esta Clase representa la orden a analizar, contiendo todas aquellas palabras necesarias. Una orden está representada por un identificador y por un **Concepto.**
- **Concepto**: en el ámbito de esta ontología, se considera que toda palabra representa un concepto (objetos, acciones, cualidades, cantidades…). Por debajo, a tercer nivel, se definen los 5 tipos morfológicos de palabras que se utilizarán: **Sustantivo**, **Adjetivo**, **Adverbio**, **Pronombre** y **Verbo**. Cada Concepto está definido por su valor (la palabra) y su peculiaridad es que puede contener como atributos otros conceptos (hijos), lo que representa una dependencia entre estas palabras. Por último, otro slot define la calidad de la relación entre padre e hijo.

- En el tercer nivel se sobrescribe por cada clase el tipo de clases permitidas como hijos, creando una relación jerárquica.

Esta ontología sólo tiene sentido si se tiene en cuenta el mecanismo que creará las instancias y las condiciones que tendrá en cuenta para ello. La Figura 28 muestra los pasos que sigue el sistema a la hora de cargar la orden en esta ontología.

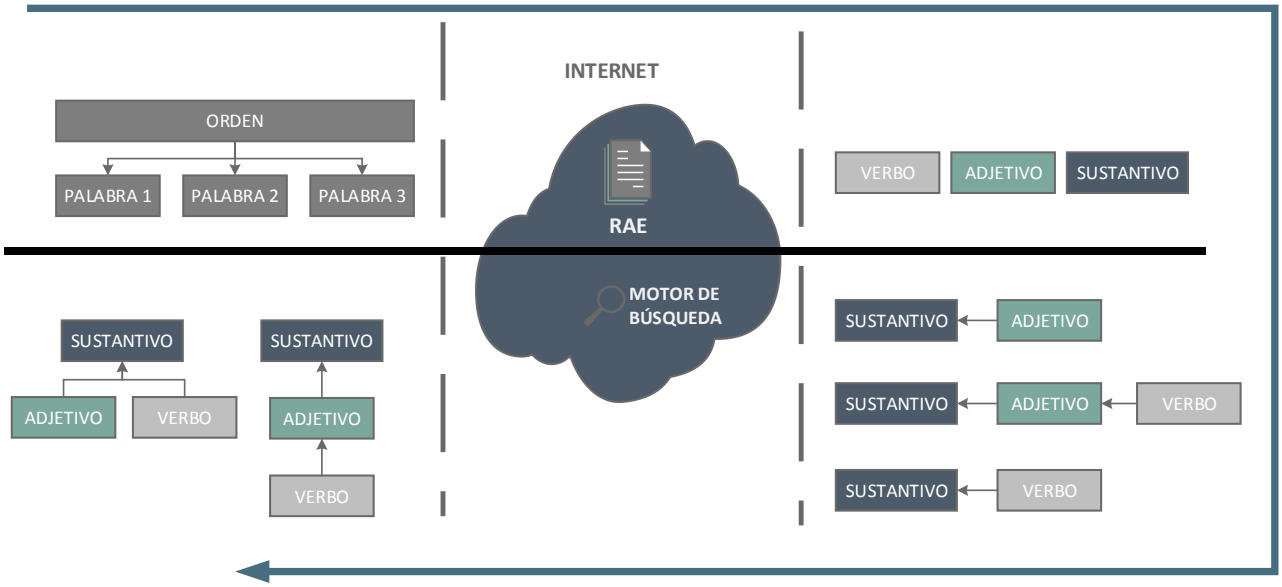


Figura 28. Análisis de la orden y carga en la ontología

A continuación se detallan los pasos mostrados en la figura (supóngase para el ejemplo que la orden era *hay poca luz*)

1. Se divide la orden por palabras: *hay*, *poca*, *luz*
2. Se clasifican dichas palabras consultando la RAE:
   a. **Hay**: verbo
   b. **Poca**: adjetivo
   c. **Luz**: sustantivo

   En ocasiones, una misma palabra puede pertenecer a diferentes tipos morfológicos, siendo, por ejemplo, sustantivo y verbo a la vez. Se ha decidido que el sistema únicamente tome el primer tipo indicado por la página web de la RAE (que los ordena de más a menos común). Este enfoque proporciona un buen índice de acierto y minimiza la cantidad de combinaciones que pueden surgir (mejorando el rendimiento considerablemente).
3. Se establece un orden jerárquico en la morfología de las palabras tal como sigue:
   a. **Sustantivo**: si existe un sustantivo, este estará a la cabeza de la jerarquía. De él puede colgar cualquier otro tipo.
   b. **Adjetivo**: es el segundo en la jerarquía y por debajo admite adverbios y verbos.

c. **Adverbio**: los adverbios sólo permiten verbos colgando de ellos, y son los terceros en la jerarquía.

d. **Verbo:** es el último en la jerarquía y puede colgar de cualquier tipo, salvo de los pronombres. Del verbo no puede colgar nada.

e. **Pronombres**: es un tipo especial que se mantiene en una rama diferente de la jerarquía. Sólo pueden colgar de los sustantivos y de ellos cuelgan, únicamente, otros pronombres.

Además, una vez definido el tronco, las siguientes palabras deben intentar adherirse a él (es decir, si el adjetivo ya cuelga del sustantivo, no se puede crear otro tronco que sólo contenga al adjetivo y al verbo o adverbio colgando de él). Esta jerarquía surge, entre otros diseños alternativos, como resultado de procurar que toda estructura mantenga significado desde el tallo hasta su tronco, siguiendo la rama (véase Figura 29). Siguiendo este enfoque, en el ejemplo propuesto se tienen cuatro estructuras:

a. Hay luz

b. Hay poca luz

c. Poca luz

4. Para establecer qué relaciones tienen sentido y cuáles no se recurre de nuevo a los motores de búsqueda. Esto aporta la experiencia necesaria para discernir si la relación es buena y, además, el mecanismo de consulta ha sido ya implementado, luego resulta un enfoque óptimo. Este proceso consiste en realizar tantas consultas como estructuras se hayan definido (en este ejemplo, tres estructuras) y obtener el número de ocurrencias en la red de la misma manera que se hace en el proceso de decisión.

   a. Si el resultado supera una cierta marca, se considera bueno y la palabra se adhiere al árbol. Cuando una palabra se adhiere como tallo, la consulta debe hacerse teniendo en cuenta todas las palabras hasta el tronco (p.e. en el caso *b* la palabra *hay* se ha adherido a *hay poca luz* porque el conjunto de la frase ha devuelto suficientes resultados, no porque únicamente el conjunto *hay poca* lo haya hecho).

   b. Si una palabra puede adherirse de diferentes maneras al árbol, el árbol original se duplica de manera que todas las combinaciones queden reflejadas. De ninguna manera un árbol contendrá más de una única vez una misma palabra.

   c. La marca a superar para considerarse una relación válida se ha decidido definir mediante una prueba previa que sirva de baremo.

      i. La prueba consiste en realizar una consulta con una conjunto de palabras configurable por el usuario (p.e. “hace mucho calor”), junto con la aplicación de un factor también configurable. Se ha detectado a lo largo del desarrollo como la cantidad de resultados que devuelve un motor de búsquedas puede variar muy significativamente entre diferentes días en incluso 4 o 5 órdenes de magnitud (lo que un día puede devolver 250.000 resultados al día siguiente puede devolver 32,

por ejemplo)[1]. El hecho de realizar una consulta previa permite detectar el estado del buscador y mantener la proporcionalidad a la hora de decidir qué combinaciones son válidas.

ii. Por ejemplo, si el motor de búsqueda devuelve a la consulta *"hace mucho calor"* 54.000 resultados y el factor a aplicar de 0.5, cualquier combinación que supere 27.000 resultados se dará por válida.

5. A partir de los resultados obtenidos del paso 4, se añaden las palabras al árbol correspondiente siguiendo las reglas establecidas en el paso 3. En la Figura 28 se ha supuesto que todas las combinaciones eran válidas y, como el verbo puede colgar de los otros dos tipos y sólo puede aparecer una vez en cada estructura, se duplica el árbol. En cada instancia de la clase *Concepto* (cada palabra adherida al árbol) el valor corresponderá a la propia palabra y la calidad del enlace será la puntuación obtenida al conectar dicha palabra con sus ascendientes en la jerarquía (para el tronco este valor será irrelevante).

Con el objetivo de dar una imagen más completa de este proceso de estructuración, se muestra en la Figura 29 un ejemplo real de un conjunto de estructuras surgidas del análisis de la orden "Hay muy poca luz en esta habitación".

Estas estructuras son resultado de un requisito de calidad de enlace bajo (el cual es configurable), pero en el diseño del proceso de interpretación puede resultar interesante incrementar este valor de corte para que la ontología solo contenga los valores esenciales, incluso descartando algunos que podrían parecer relevantes a priori, pues como ya vimos en el ejemplo propuesto en el diseño del AI, puede resultar suficiente con un número muy limitado de conceptos.

Reduciendo la cantidad de conceptos almacenados, el número de combinaciones en el proceso de decisión resulta mucho menor, lo que mejora en muy alta medida el rendimiento. Una vez más, la búsqueda del equilibrio resulta clave a la hora de proporcionar una buena experiencia al usuario. En cualquier caso, para este ejemplo resulta interesante observar como mediante diferentes estructuras, todos los caminos definidos desde el tallo al tronco tienen un significado válido.

[1] Por fortuna para el enfoque de este proyecto, los buscadores suelen mantener cierta proporcionalidad, de manera que cuando se reducen los órdenes de magnitud, se reduce para todos los resultados. Esto provoca, por supuesto, que la diferencia entre resultados disminuya drásticamente y el sistema sea menos preciso, pero se ha comprobado que se consigue mantener un índice de aciertos aceptable.

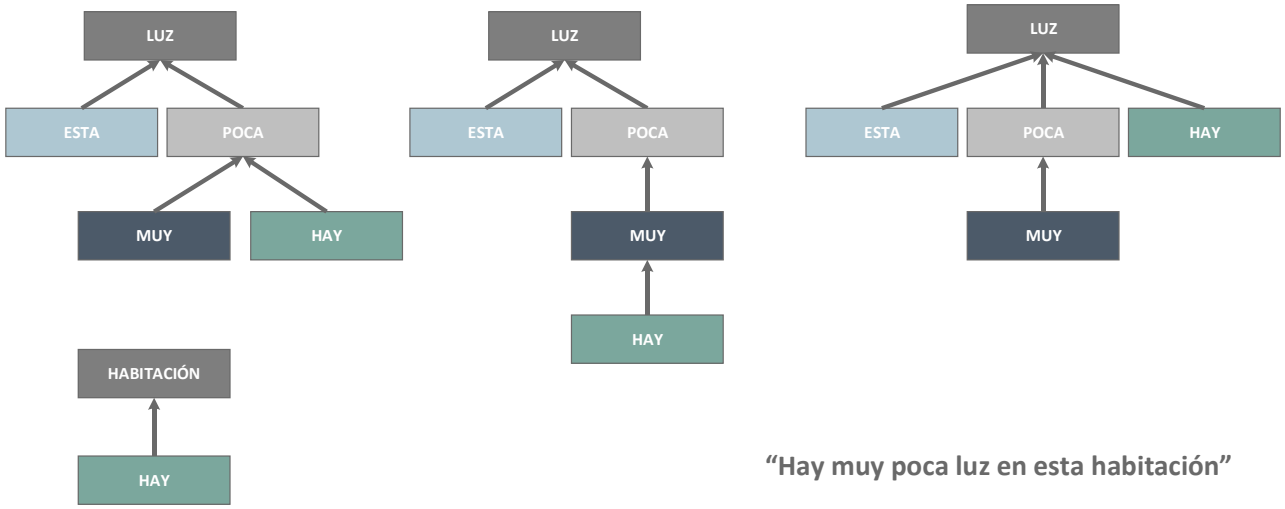


Figura 29. Ejemplo de instancias en la ontología de órdenes

Tal como se observa en la Figura 29, las dependencias mantienen relación semántica (del tallo al tronco) y se eliminan las combinaciones inválidas (p.e. *muy luz, muy habitación, muy esta luz, poca habitación*…). Sin embargo, Internet no es únicamente una gran base de datos de conversaciones, y pueden aparecer repetidamente relaciones que a priori no tienen sentido. Por ejemplo, *hay habitación* no resulta a una combinación común, pero al existir en Internet infinidad de páginas web de alquiler o venta de pisos, de hoteles, museos, etc., un término como este enmarcado en, por ejemplo, un anuncio que reza "Hay habitación disponible" resulta tener más sentido. Siendo esto así, es tarea del conjunto del sistema intentar depurar la información que obtiene cada parte individualmente.

4.4 Base de datos

Tras el análisis de dos componentes esenciales en el desarrollo del proyecto como son la inteligencia artificial y la implementación de ontologías, se ha hecho patente el papel importantísimo que realiza el motor de búsqueda en el proceso de estructuración de la información así como su uso incluso más crucial en el proceso de interpretación.

Por otro lado, ya se ha descrito cómo el sistema accede a la red, concretamente a la página web de la RAE, para clasificar morfológicamente las distintas palabras que componen una orden y almacenarlas convenientemente en la ontología. Este proceso conlleva, por supuesto, accesos a Internet mediante determinadas consultas, procesado de la respuesta (parseando la página web) y en ocasiones requiere repetir el proceso de manera reiterativa hasta obtener la información relevante.

Complementariamente y tal como se verá en el apartado 4.5, la información del contexto será obtenida también mediante el acceso a una API web que proporcionará los datos de interés para el sistema.

Todos estos accesos a Internet, con sus correspondientes procesados, requieren de gran cantidad de tiempo y ralentizan considerablemente la ejecución. Siendo esto así, se hace evidente la necesidad de implementar algún mecanismo que permita no realizar más consultas de las

estrictamente necesarias, mejorando el rendimiento del sistema (en cuanto a tiempo de ejecución se refiere) muy significativamente. Tanto es esto así, que sin este mecanismo los tiempos de ejecución podrían considerarse inaceptables en función de la complejidad de la red, de la orden y de distintos factores de la configuración.

Por todo ello, se decide implantar en el sistema una base de datos capaz de almacenar mediante distintas tablas los resultados obtenidos para las consultas previamente ejecutadas. Concretamente, se ha decidido utilizar SQLite por su ligereza y simplicidad (véase apartado 2.8.1), pues el sistema no requiere de funciones más avanzadas.

Para esta base de datos se contemplan tanto tablas estáticas, con una estructura fija y predefinida (dado que almacenarán siempre el mismo tipo de información) como tablas dinámicas[2] que responderán a las necesidades del *algoritmo del intérprete* (que se verá en detalle en el apartado 4.6).

Existirán tres tablas estáticas con el objetivo de almacenar:

- Información climatológica
- Resultados previos del motor de búsqueda
- Resultados del análisis morfológico

Cada una de estas tablas estáticas debe cumplir dos requisitos:

1. Debe contener las columnas que se adapten al tipo de información que almacenarán, con su correspondiente tipo.
2. Debe contener una marca de tiempo que indique el instante en el que cada registro fue añadido. Esto resulta de especial utilidad en dos tipos de tabla:
    a. Tabla de información climatológica: en la configuración de usuario se definirá cada cuanto tiempo esta información debe ser actualizada para mantener una imagen fidedigna de los diferentes factores climatológicos.
    b. Tabla de búsquedas: tal como ya se ha comentado en apartados anteriores, los motores de búsqueda resultan inestables si comparamos la cantidad de resultados que devuelve un día y los que puede devolver una semana después. Aunque la proporción entre resultados devueltos en una misma ronda de consultas aproximadamente se mantenga, no se podrá tener certeza de que el resultado recuperado de la BD está en el mismo nivel que los devueltos en el último instante de ejecución.

       Esta situación implica que si en un proceso de interpretación se recupera un dato de la BD y otro, por inexistente, se tiene que buscar por primera vez, el resultado a la salida del proceso podría no ser correcto si el estado del buscador ha variado excesivamente. Es por esto que se define por configuración el periodo de caducidad de un registro, de manera que cuando

[2] En el ámbito de este proyecto, se denomina tabla dinámica a una tabla cuya definición es creada a raíz de otros factores ajenos a la BD, concretamente a partir de la definición del *algoritmo del intérprete* (sin embargo, una vez la tabla es creada su estructura no se modifica hasta que se carga un nuevo algoritmo, resultando durante todo este periodo una tabla estática).

se recurra a la BD, si el registro existe pero está caducado se actualizará con la cantidad de resultados que devuelve en ese instante el buscador. Este diseño se ha comprobado que consigue estabilizar el sistema, aunque tiene repercusiones negativas en el rendimiento (pues se requiere de más consultas al buscador).

Aunque se ha decidido conservar la marca de tiempo en la tabla de resultados del análisis morfológico para proporcionar información adicional al desarrollador, esta información no caducará nunca pues se entiende que los resultados no deberían variar en el tiempo de vida de este sistema (p.e. *luz* es un sustantivo y continuará siéndolo de manera indefinida).

Se presentan a continuación las tres tablas estáticas que contienen la Base de Datos:

Tabla 2. Tabla de búsquedas previas (BD)

BÚSQUEDA (QUERY)	RESULTADOS	MARCA DE TIEMPO
TEXT (UTF-8)	INT	DATETIME
PK (NOT NULL)	(NOT NULL)	DEFAULT (CURRENT_TIMESTAMP)
LUZ+%2B+PERSIANA	**1.356.237**	**2014-05-07 19:54:18**

Tal como se ha definido esta tabla, ni la búsqueda ni los resultados pueden quedarse vacíos al añadir un registro y, por su parte, la marca de tiempo se añadirá con el instante en el que se añade dicho registro sin necesidad de indicar un valor concreto. Además, no podrá existir más de una vez una misma búsqueda, dado que la *Primary Key* fuerza la condición *Unique*.

Aunque desde el punto de vista de la BD es indiferente si el texto se introduce con formato UTF-8 o no, se ha decidido forzar la entrada en esta columna con este formato, ya que será la codificación requerida a la hora de lanzar la consulta al motor de búsqueda.

Tabla 3. Tabla de análisis morfológico (BD)

PALABRA	TIPO	MARCA DE TIEMPO
TEXT	TEXT	DATETIME
PK (NOT NULL)	(NOT NULL)	DEFAULT (CURRENT_TIMESTAMP)
LÁMPARA	**nombre femenino**	**2014-05-07 19:54:12**

Esta tabla, con una estructura parecida a la anterior, contiene dos columnas de texto junto con la marca de tiempo. De la misma manera, ninguna de las dos primeras columnas puede quedar vacía y, además, la *palabra* debe ser única.

En este caso no existen requerimientos de formateado en la *palabra*, pues es información manejada localmente que no se refrescará. Es importante destacar que el tipo morfológico de una palabra puede ser cualquiera indicado por la RAE, no necesariamente uno de los que se han decidido manejar en el sistema. El objetivo de esta tabla es evitar futuras consultas, no preseleccionar información.

Por ejemplo, si se clasifica la palabra "la" y la RAE indica que se trata de un artículo, descartarla sería un error de diseño incluso cuando se sabe con certeza que no será utilizada, pues ello implicaría que la próxima vez que el usuario utilizase la palabra "la" en una frase el sistema volvería a acudir a la RAE para clasificarla al no tener su clasificación disponible previamente. Manteniendo en la BD todas las palabras, útiles o no para el procesamiento posterior, el sistema se asegura el no clasificar más de una vez cada palabra.

Tabla 4. Tabla de información climatológica (BD)

AMANECER	ATARDECER	TEMPERATURA	PORCENTAJE NUBES	DESCRIPCIÓN	MARCA DE TIEMPO
DATETIME	**DATETIME**	**INT**	**INT**	TEXT	DATETIME
-	-	-	-	-	DEFAULT (CURRENT_TIMESTAMP)
2014-04-12T05:40:28	**2014-04-12T18:51:01**	**17.71**	**32**	**lluvia ligera**	**2014-04-12 20:56:33**

La tabla de información climatológica, por su parte, es una tabla algo más peculiar. Esta tabla solo permite la inserción de un registro y, cuando se añade otro, el primero desaparece. No se ponen restricciones de no nulidad ni de unicidad dado que:

- No unicidad: al haber un único registro la unicidad está garantizada para todas las columnas.
- No nulidad: esta restricción no se impone con el fin de adaptarse a posibles variaciones en la API climatológica (que constituye un módulo independiente tal como se verá en el apartado 4.10). Dado que el sistema es capaz de adaptarse a la falta de algunas evidencias (repartiendo equitativamente la probabilidad de que un estado sea el actual), el cambio a otra API que no pudiese proporcionar toda la información no sería crítico.
    - Por ejemplo: si el sistema pretende determinar si el cielo está despejado o no y no hay evidencias en la tabla, repartirá la probabilidad de manera que:
        - Soleado → 33.3%
        - Intermedio → 33.3%
        - Nublado → 33.3%

Se muestra a continuación el diagrama de flujo que representa el acceso a cierta información en Internet, consultando previamente la Base de Datos.

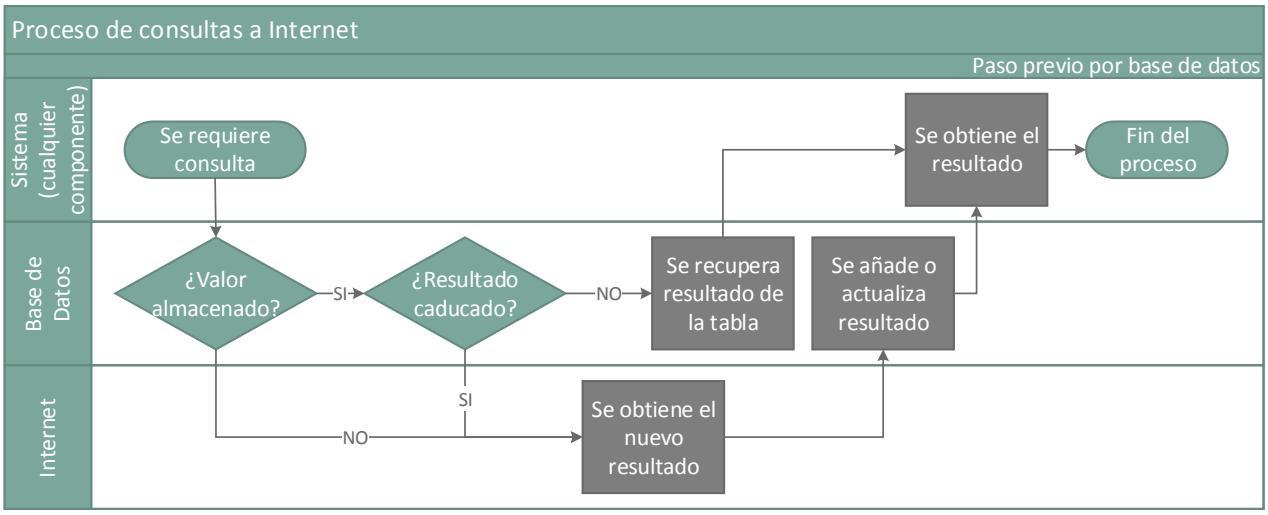


Figura 30. Consultas a internet a través de la BD

Por su parte, las tablas denominadas dinámicas responden a las necesidades de almacenamiento y procesado del algoritmo y, por tanto, deben ser tan flexibles como este. Qué factores tienen en cuenta y cómo se aplican en la BD se explicará en detalle en el apartado 4.6.

Se ha visto a lo largo de este punto como un conjunto muy sencillo de tablas independientes (sin relaciones de *Foreign Key* entre ellas) puede resultar extremadamente útil en la mejora del rendimiento del sistema, aunque no añadan una funcionalidad adicional y la información que proporcionen sea la misma que la que se hubiese obtenido sin su existencia. Sin embargo, en este proyecto se hará uso de otras funcionalidades que proporciona la base de datos para servir como soporte en el proceso de interpretación.

Aunque se detallará en el correspondiente apartado del *algoritmo*, se puede adelantar que los mecanismos de selección condicional de información, agrupación e inserción en nuevas tablas serán las herramientas principales a la hora de proporcionar soporte al proceso de interpretación. Mediante estas funciones el intérprete podrá definir cómo unificar valores de diferentes tablas para obtener una conclusión global a partir de las condiciones que establezca. Mediante un conjunto de *queries* relativamente sencillas (que, eso sí, deberán ser construidas dinámicamente para satisfacer las necesidades del algoritmo) se puede realizar el procesado, suma y selección de gran cantidad de datos de una manera muy simplificada respecto al equivalente mediante codificación en Java.

A modo de breve repaso, se definen a continuación las principales funciones utilizadas en la base de datos para dar soporte al intérprete:

- Selección: se puede recuperar selectivamente conjuntos concretos de información basándose en dónde debe encontrarse la información y qué condiciones debe cumplir tanto esta información como el resto de información del registro que la contiene (otras columnas).
- Unión: se puede recuperar información de diferentes tablas que satisfagan ciertos criterios y unificarla en nuevos conjuntos de registros.

- Inserción: se puede mover la información seleccionada, unificada o procesada a otras tablas.
- Ordenado: se pueden aplicar distintos tipos de ordenación en base a varios criterios.
- Borrado: se puede descartar información que cumpla ciertos criterios.
- Procesado de información: se puede aplicar ciertas operaciones, matemáticas y lógicas, de manera que la información sufra ciertas transformaciones mediante herramientas como:
    - Truncado: selección de un conjunto concreto de caracteres en una secuencia.
    - Concatenación: añadido de información a un valor (tanto predefinida como procedente de otros valores).
    - Operaciones aritméticas: sobre uno o varios valores.
    - Limitaciones: permitiendo la selección sobre un número de registros concreto.

4.5 Procesado del contexto

El contexto, tal como se ha mencionado en varios puntos a lo largo de este documento, resulta uno de los factores esenciales en la aportación de inteligencia al sistema. Como breve repaso, se enumeran a continuación los requisitos que debe cumplir el mecanismo seleccionado para la aplicación del contexto:

1. Debe poder interpretar valores absolutos e inferir su porcentaje de influencia en el contexto.
2. Debe poder representar diferentes factores contextuales y su respectiva influencia en la toma de decisiones.
3. Debe poder inferir la influencia conjunta de todos los elementos que afectan en la toma de decisiones en el instante de ejecución.
4. Debe poder responder a ciertos parámetros de configuración que acerquen la toma de decisiones al contexto tal como lo entiende un usuario determinado.

Para afrontar los puntos 1 y 2 se hará uso de un modelo probabilístico conocido como *red bayesiana*, *red de Bayes*, *red de creencias* o *modelo Bayesiano* (de ahora en adelante, *red bayesiana*). Tal como se ha visto en el apartado 2.6, este modelo probabilístico permite representar una serie de relaciones de dependencia en función de un conjunto de probabilidades mediante un grafo acíclico.

Para la creación de esta red o grafo, se ha hecho uso de la herramienta UnBBayes (véase apartado 2.6.2), que permite mediante una interfaz gráfica la definición de nodos, tablas de probabilidad y relaciones. Además, este software está codificado en Java. Esta última característica permite importar las propias librerías que utiliza UnBBayes y leer la información de las redes generadas mediante la interfaz gráfica, modificando la red en tiempo de ejecución y llevando a cabo los pasos oportunos para obtener la salida correspondiente.

Por todo ello, este software resulta especialmente útil en este proyecto, dado que se cuenta de un software muy intuitivo a la hora de crear redes bayesianas y la posibilidad posterior de manejar toda esta información desde el sistema.

Se muestra a continuación, en la Figura 31, el modelo de red bayesiana más completo que se ha implementado en el sistema. Tal como se detallará a lo largo de este apartado, se implementan tres tipos de redes, siendo las otras dos redes una simplificación de esta. El modelo implementado en esta figura aglutina la influencia tanto de factores externos como internos y define la relación entre los diferentes factores desde los extremos (evidencias) hasta la decisión (nodo representado con el nombre *Action*).

Debe aclararse que la red de contexto se ha diseñado para ser analizada una vez por cada dispositivo y de hecho, tal como se detallará, se analiza para cada posible acción de este. Sin embargo, en la práctica, sólo una acción será posible por cada uno (pues no se analizará si se debe encender un dispositivo ya encendido). Siendo esto así, el diseño de la red se enfoca al estudio del concepto que maneja un dispositivo, así como a la acción que puede ejecutar.

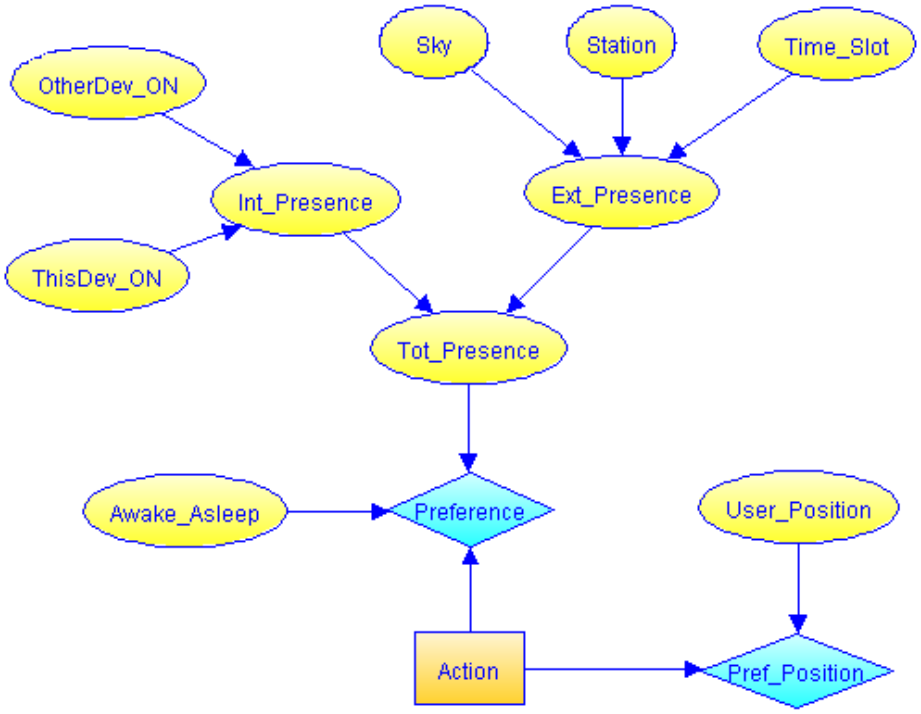


Figura 31. Red bayesiana completa (modelo de nodos)

Para el análisis de la red bayesiana representada en la Figura 31, se describe en primer lugar el tipo de nodos aquí representados, aunque una descripción previa ha sido realizada en el apartado 2.6.2.1 (téngase en cuenta que se expone la función de cada nodo tal como se ha diseñado en este sistema, aunque existen modelos de red alternativos donde los nodos pueden jugar diferentes roles):

- Nodos de variables probabilísticas: representado por elipses, este tipo de nodos representa la probabilidad de que se produzca cada una de las circunstancias representadas en dicho nodo, de manera que la suma debe ser 1.

- Si un nodo de probabilidad no depende de nodos anteriores, es decir, son extremos (evidencias), entonces el porcentaje de cada uno de los valores representados en el nodo debe asignarse mediante algún mecanismo externo. En caso de no hacerlo, los porcentajes de probabilidad quedarán repartidos equitativamente.
- Si un nodo de variables probabilísticas depende de nodos anteriores, debe rellenarse una tabla de potencias que indica la relación entre la presencia de los valores de los nodos anteriores y la presencia de los valores de este nodo. En la Tabla 5 se ilustra esta circunstancia para el nodo *Tot_Presence* (que representa la presencia total de un cierto concepto[3] en función de la presencia interna y externa de este, tal y como se verá en la descripción de los nodos que se expondrá tras este punto).

Tabla 5. Tabla de potencias (red bayesiana)

Int_Presence	YES			NO		
Ext_Presence	high	medium	low	high	medium	low
NO	0	0,1	0,15	0,3	0,7	0,9
YES	1	0,9	0,85	0,7	0,3	0,1

- Para entender esta tabla, debe aclararse que:
  - El nodo *Int_Presence* representa dos posibles valores: Sí (*YES*) y No (*NO*), representando qué probabilidades se cree que existen de que el concepto en estudio en esta red de contexto ya se encuentre presente (YES) en el interior de la casa en el momento de la ejecución o no (NO).
  - Por su parte, el nodo *Ext_Presence* representa, esta vez mediante tres niveles (alto, medio o bajo, con su correspondencia en inglés *high*, *medium*, *low*), la probabilidad de que la presencia del concepto en cuestión en el exterior de la casa corresponda a uno de estos tres niveles.
  - El nodo Tot_Presence, que es el representado por esta tabla de potencias, contiene dos valores, al igual que el nodo Int_Presence: *YES* y *NO*, en función de la probabilidad de que el concepto, como resultado de la presencia interna y externa, esté presente en el momento de la ejecución.

  De esta manera, en la tabla de potencias se define arbitrariamente qué porcentaje deben adquirir los posibles valores del nodo Tot_Presence en función de la probabilidad de cada uno de los nodos anteriores. Para su definición, se utiliza como referencia el cruce de los valores absolutos (por ejemplo, en la primera columna se indica qué probabilidad debería adquirir este nodo si los anteriores tienen toda la probabilidad concentrada en los

[3] Se entiende concepto, al igual que en puntos anteriores, como aquel sobre el que actúa un dispositivo (p.e. *iluminación*, *temperatura*, *música*, etc.).

valores *YES* y *high*, considerándose que por extensión la presencia total es *YES* al 100%).

- Nodo de variables de utilidad: representado por un rombo, este tipo de nodos aglutina una serie de puntuaciones que son asignadas en función de la probabilidad de los distintos valores de los nodos precedentes. Estas puntuaciones pueden utilizarse para representar cualquier concepto, por ejemplo la cantidad de dinero en una inversión, si bien no tienen para la red ningún significado y deben utilizarse con el propósito que requiera cada sistema. Este tipo de nodos son ideales para añadir preferencias a este tipo de redes e incluso mecanismos de aprendizaje, pues la variación en la puntuación asignada a cada valor o combinación de valores previos puede modificar significativamente la salida de la red.

  En la Tabla 6 se ilustra la tabla correspondiente al nodo *Pref_Positio*n (preferencia en función de la posición del usuario).

Tabla 6. Tabla de asignación de valores (red bayesiana)

User_Posit...	Same room			Other room		
Action	INCREASE	REDUCE	NOTHING	INCREASE	REDUCE	NOTHING
Utility	75	75	75	25	25	25

  - De nuevo, para entender esta asignación debe conocerse que:
    - El nodo *Action* contiene tres posibles valores (Aumentar, reducir y no hacer nada, en inglés *INCREASE*, *REDUCE*, *do NOTHING*). Este nodo representa qué puntuación debe adquirir cada una de estas posibles acciones (sobre el concepto) en función del contexto.
    - El nodo User_Position contiene los valores *Same room* (en la misma habitación) y *Other room* (en otra habitación), representando si el usuario se encuentra en la misma habitación que el dispositivo en cuestión o no.
  - Este nodo, pese a su simplicidad, muestra con claridad cómo se asigna más o menos cantidad a cada una de las acciones en función de las evidencias (probabilidad de los nodos extremos), en este caso en función del nodo User_Position. En este sistema, se entienden los valores indicados en el nodo *Utility* como puntuación, siendo más favorable una cierta acción cuanto mayor es esta. De esta manera, en el ejemplo mostrado este nodo asigna el triple de puntuación a aplicar cualquier acción sobre un dispositivo que se encuentre en la misma habitación que el usuario frente a uno que este en otra habitación.

- Nodos de variables de decisión: representados mediante un cuadrado, estos nodos representan la puntuación obtenida desde los nodos de utilidad. Tal como se han utilizado en este sistema, estos nodos sirven como punto de salida de la red, indicando a partir de los diferentes factores introducidos (el contexto) qué acción es más probable que deba ser ejecutada para un cierto dispositivo. Este tipo de nodos no tienen dependencia probabilística con los nodos anteriores y su implementación debe

responder únicamente a qué nodos de variables de utilidad deben utilizarse para extraer la conclusión final.

De esta primera descripción de la red se puede ya intuir que, por un lado, el tamaño de la tabla de potencias crece considerablemente cuanto más nodos precedentes y valores dentro de estos nodos existen (por ejemplo, la tabla de potencias del nodo *Ext_Presence* contiene 108 celdas). Teniéndose en cuenta que esta red será analizada para cada dispositivo, añadir una gran cantidad de nodos a la red para hacerla más precisa podría repercutir muy negativamente en el rendimiento del sistema.

Por otro lado, se observa que de requerirse modificación de estas tablas de potencias (por ejemplo para adaptar la red a las preferencias del usuario), ésta no puede ser afrontada celda a celda, sino que debe existir algún mecanismo que simplifique la tarea, dado que pedirle al usuario que rellene a placer las 108 celdas del nodo *Ext_Presence* no parece razonable. El mecanismo desarrollado para tratar esta casuística se verá en el apartado 4.8.2. Además, la influencia de factores externos podría variar mucho en función del contexto que se esté manejando (p.e. en la iluminación la estación no es tan determinante como puede serlo en la temperatura).

Este grafo hace uso de uno de los algoritmos más extendidos a la hora de resolver redes bayesianas, el algoritmo conocido como *árbol de unión* (del inglés, *Junction Tree*). Este algoritmo (el único habilitado en UnBBayes por otra parte), se encarga de reducir el número de ramas mediante su unión probabilística, hasta que puede inferir la probabilidad de ciertas variables en función del resto de nodos. Para más información respecto a este algoritmo, véase apartado 2.6.2.2.

4.5.1 Nodos implementados en la red bayesiana

Una vez que la red es compilada (asegurándose de que todos los nodos suman 1 a partir del conjunto de probabilidades de los diferentes valores), se puede presentar de una manera más intuitiva cómo quedaría la relación probabilística entre los diferentes nodos, lo cual se representa mediante la Figura 32.

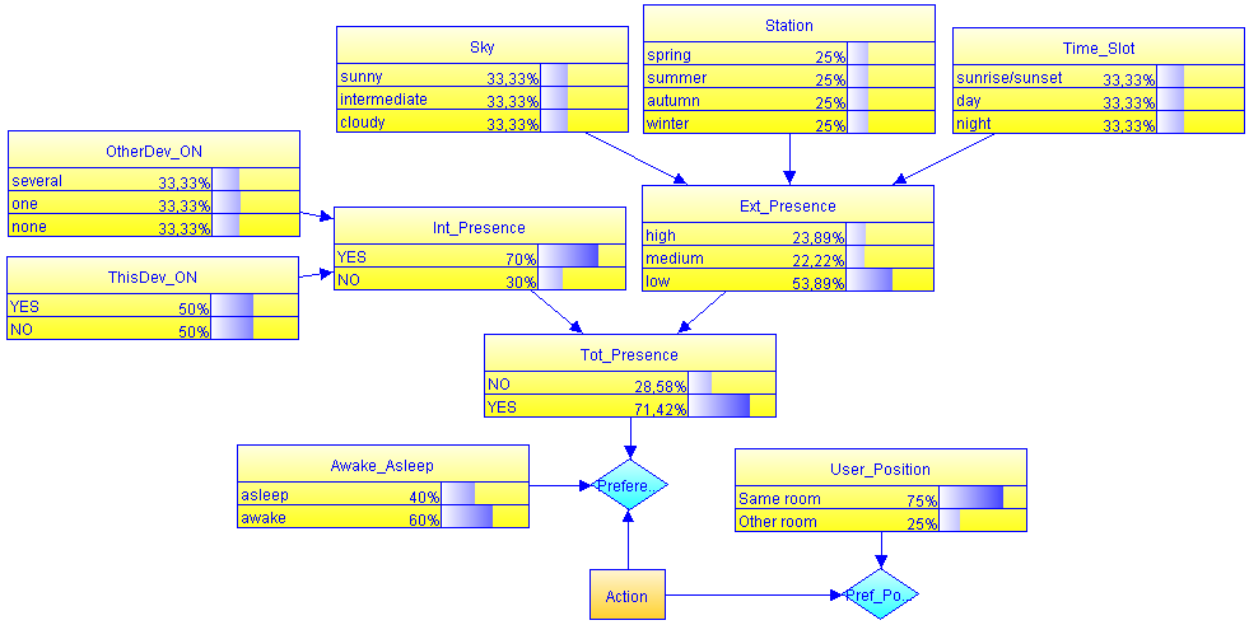


Figura 32. Red bayesiana completa (representación porcentual)

La figura de esta red ha sido capturada sin la asignación de las evidencias (que se realiza en tiempo de ejecución) y, por tanto, los nodos extremos aparecen con probabilidades sin inicializar, si bien no tiene ninguna relevancia pues todos los nodos extremos serán cargados antes de la ejecución del algoritmo *Junction Tree*. Sin embargo, este grafo resulta más claro a la hora de entender qué representa cada nodo y cómo influye en el conjunto.

A continuación, una vez definido el papel de cada tipo de nodo en la red, se define cada nodo implementado y los valores que representa, así como sus posibles relaciones probabilísticas (no se dará en esta descripción información numérica de las tablas de potencias con la intención de simplificar la explicación).

- **Sky (Estado del Cielo)**. Representa el estado del cielo y su información se obtiene del porcentaje de nubosidad que proporciona la API climatológica.
- **Station (Estación)**. Representa la estación actual (en el momento de la ejecución).
- **Time_Slot (Franja horaria)**. Representa en qué franja horaria se ubica el momento de la ejecución. Como se observa en la Figura 32, dada la similitud entre el amanecer y el atardecer en lo que a la presencia de diferentes conceptos se refiere, se toma ambos como una única variable.
- **Ext_Presence (Presencia externa)**. En dependencia con el valor que tomen los nodos anteriormente mencionados, la presencia externa del concepto en cuestión es asignada, con los tres niveles indicados.
- **OtherDev_ON (Otros dispositivos encendidos)**. Este nodo representa cuantos dispositivos que manejan el mismo concepto (p.e. iluminación) están encendidos en la habitación en la que se encuentra el dispositivo en estudio (recuérdese que la red se analiza para cada dispositivo). El estado de dispositivos que manejen otros conceptos

no influye (por ejemplo, que la calefacción esté encendida no se tendrá en cuenta si se analiza un dispositivo de iluminación).

- **ThisDev_ON (Este dispositivo encendido)**. Representa si el dispositivo en estudio está encendido, pues influirá en la presencia interna del concepto.
- **Int_Presence (Presencia interna)**. Al igual que ocurría con la presencia externa, en función de la cantidad de dispositivos encendidos que proporcionen el concepto en estudio, este nodo tomará diferentes valores, esta vez representado únicamente con dos posibilidades.
- **Tot_Presence (Presencia total)**. Este nodo engloba la presencia del concepto en función de la presencia interna y externa.
- **Awake_Asleep (Despierto o dormido)**. Representa la probabilidad de que el usuario se encuentre actualmente en una franja horaria en la que o bien está dormido o pretende estarlo (y viceversa). Se ha decidido no relacionar este nodo con el nodo Time_Slot dado que no necesariamente todos los usuarios dormirán en las mismas franjas horarias.
- **Preference (Preferencia)**. En este caso el nombre no representa de manera obvia su utilidad, pues tal como se ha explicado, su función es la de asignar puntuaciones en función de los valores que han tomado los nodos precedentes. Este nodo pretende puntuar el tipo de acción a realizar a partir de la presencia del concepto y de la probabilidad de que el usuario esté activo o dormido.
- **User_Position (Posición de usuario)**. Este nodo representa si el usuario se encuentra en la misma habitación que el dispositivo en análisis o no.
- **Pref_Position (Preferencia de posición)**. Tal como se ha explicado previamente, se asignan distintas puntuaciones a realizar una acción sobre un dispositivo en función de la posición del usuario.
- **Action (Acción)**. Nodo de decisión que sirve de salida del sistema. A partir de los dos nodos de utilidad, se asigna una puntuación global a la posible ejecución de una acción sobre un dispositivo.

En este punto pueden surgir dos preguntas más en lo que respecta a la información que maneja esta red de contexto: cómo se añaden estos valores y cómo se aplican las preferencias de usuario. La respuesta depende del tipo de nodo, tal como se expone en el siguiente punto.

4.5.2 Asignación de valores absolutos en los nodos

Para los nodos que dependen de datos climatológicos, es evidente que la información debe obtenerse de algún repositorio y que, además, esta información debe estar relativamente actualizada. Para ello, se ha hecho uso de la API *Open Weather Map* [41], que ofrece un acceso sencillo y la posibilidad de recuperar la información mediante diferentes formatos como XML. Aunque esta API ofrece datos adicionales, aquellos que se han considerado de relevancia para el

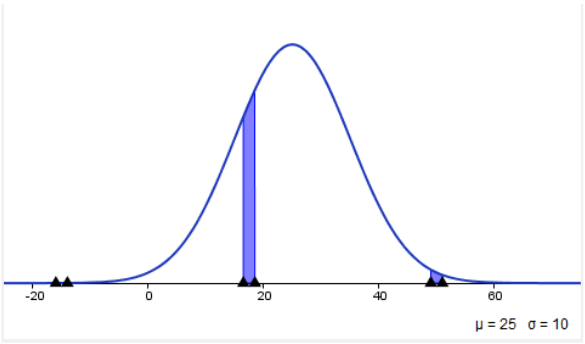
sistema propuesto y, por tanto, los únicos datos que se han utilizado, son aquellos que se indicaron en la Tabla 4.

Sin embargo, puede no resultar suficiente obtener una cierta información desde la API si esta no puede trasladarse a una evidencia en la red bayesiana (que como se ha visto debe estar representada porcentualmente). La interpretación de valores absolutos en sensaciones o proporciones suele ser algo complejo para un sistema informático. Existen estudios enfocados a la creación de los denominados *reasoners* (razonadores) y de las *fuzzy machines* (máquinas difusas) que pretenden dar sentido a estos datos, pero en el ámbito de este proyecto y persiguiendo la máxima simplicidad en cada uno de sus componentes (eso sí, garantizando un buen nivel de funcionamiento), se ha decidido implementar una transformación simple mediante el uso de distribuciones probabilísticas.

Concretamente, se ha utilizado la distribución normal, también conocida como distribución de Gauss. La gráfica de su función de densidad tiene una forma acampanada (a esta curva se la conoce como campana de Gauss). Este modelo permite distribuir la probabilidad en un eje que, bien adaptado, puede representar los distintos valores que puede tomar un nodo.

Por ejemplo, supóngase que se pretende asignar a un nodo que representa temperatura la probabilidad de que dicha temperatura sea *alta*, *media* o *baja*. Un buen reparto de la probabilidad podría ser una aproximación aceptable a la sensación que tiene el usuario frente a una temperatura concreta. Sin embargo, bien es sabido que no todos las personas experimentan la misma sensación térmica ante una misma temperatura y esto debe ser, por tanto, adaptable.

Teniendo todo esto en cuenta, si el usuario tiene mecanismos para indicar qué temperaturas considera máxima y mínima (lo cual es a su vez útil para adaptar el sistema a viviendas en diferentes partes del planeta), se puede establecer los límites del eje. Para este ejemplo supóngase que el usuario delimita -15 ºC como el punto más frío y 50ºC como el más cálido. En este ejemplo, el valor medio estaría ubicado en 17.5 ºC.




```
-15 ºC   (-16 <= X <= -14)   :        0 -->      0 -->  0%
17.5 ºC  (16.5 <= X <= 18.5) :   0.0602 -->   0.94 --> 94%
50ºC     (49 <= X <= 51)     :   0.0035 -->   0.06 -->  6%
```

Figura 33. Distribución normal aplicada al contexto

Mediante una distribución normal como la que se representa en la Figura 33, se puede obtener una aproximación válida para el reparto de probabilidades. Para ello, se siguen los siguientes pasos:

1. Se definen los puntos extremos del eje (correspondientes a temperatura *baja* y *alta* con probabilidad máxima[4]), así como los puntos que reflejen la probabilidad máxima de otros valores (en este ejemplo 17,5ºC representa la temperatura *media*, si bien podría haberse establecido el valor en otro punto cualquiera teniendo en cuenta que la sensación térmica no es lineal).
2. Se elige una varianza apropiada para que el reparto de probabilidad sea razonable a lo largo del eje. En este ejemplo se ha elegido $\sigma = 10$
3. Se asigna la media en correspondencia al valor absoluto recuperado que se quiere procesar (en este ejemplo se ha supuesto que la temperatura devuelta por la API es 25ºC): $\mu = 25$
4. Por definición, el resultado de la distribución será 0 para un *x* determinado, y deberá seleccionarse un rango para obtener valores no nulos. Para ello, se elige un margen de oscilación con respecto a los valores que identifican la probabilidad máxima de las distintas circunstancias (*alta*, *media* o *baja*). En este caso se ha seleccionado un margen de 1 unidad, lo que implica rangos de 2 unidades entorno al valor en cuestión, tal como se observa en los datos numéricos de la Figura 33.
5. Se obtienen los valores de salida para las condiciones expuestas en los puntos anteriores y se normalizan, obteniendo valores válidos para introducir en los nodos de evidencias.

El planteamiento explicado, sin embargo, puede resultar insuficiente para algunas representaciones. Supóngase, por ejemplo, que se quiere estimar la probabilidad de que el usuario esté en una franja horaria en la que debería estar dormido respecto a una en la que debería estar despierto tal como lo entiende el sistema. En este caso el eje es cíclico, pues en dicho eje el valor 25 correspondería igualmente al valor 1 (las horas del día). En estos casos, el rango de la distribución que sobrepase los extremos debe reflejarse en los extremos contrarios. Para ello, se ha tomado el enfoque de la Figura 34, donde se calcula dos distribuciones normales diferentes con distintas medias pero idéntica varianza. Siempre y cuando estas no se solapen se podrán entender como una única distribución probabilística normal reflejada a lo largo del eje.

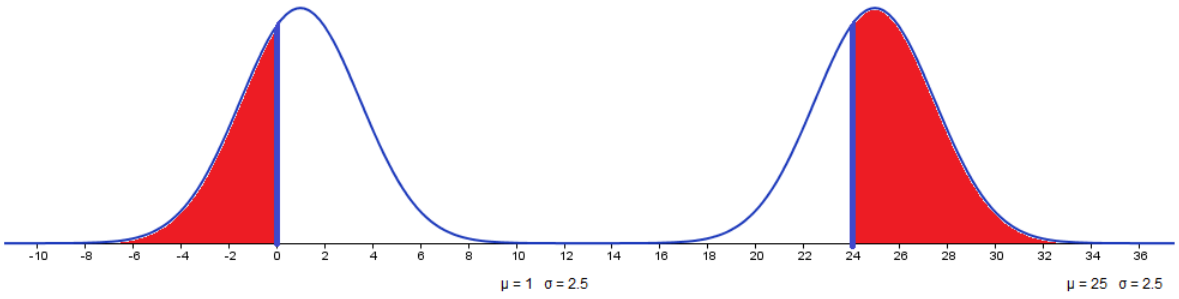


Figura 34. Distribución normal aplicada al contexto en ejes cíclicos

[4] El concepto de probabilidad máxima hace referencia a aquel valor en la medición que haría corresponder el 100% de la probabilidad a un estado o circunstancia concreta. En este ejemplo, si la temperatura medida fuese 17.5ºC (que corresponde con el valor medio), la distribución probabilística para temperatura *alta*, *media* y *baja* sería *0*, *1* y *0* respectivamente (lo que correspondería a la citada probabilidad máxima).

Tal como se observa en la figura, ignorando aquellos rangos marcados en rojo, se obtiene una distribución normal válida para el eje que representa las horas del día. Si por ejemplo se estimase que el usuario se acuesta sobre las 23:00 y se despierta sobre las 06:00, siendo las 01:00, se puede deducir a simple vista cómo la mayor parte de la probabilidad recaerá en el hecho de que el usuario está dormido, si bien una pequeña parte corresponderá a la posibilidad de que esté despierto.

Este tipo de planteamiento resulta flexible, pero evidencia la necesidad de manejar preferencias de usuario. No todos los usuarios determinarán que 50ºC es el máximo nivel de calor y algunos preferirán que el sistema interprete una situación de 40ºC como una situación calor extremo. Más determinante es esta adaptabilidad al usuario cuando hablamos de las horas a las que suele acostarse y despertarse, pues varía mucho entre culturas e incluso entre personas de una misma sociedad. Aunque se verá con más detalle en el apartado 4.8, se puede adelantar que el sistema manejará las preferencias del usuario y las aplicará a la hora de determinar los puntos del eje en la distribución normal explicada.

4.5.3 Tipos de redes implementados

En el diseño e implementación de este sistema se ha decidido optar la utilización de tres redes bayesianas distintas adaptadas a las necesidades de cada dispositivo. La más completa de ellas ha sido analizada en los puntos anteriores de este apartado, pero esta red plantea algunos inconvenientes a la hora de adaptarse a ciertos dispositivos.

Tal como se verá en el apartado 4.10, en el proceso de carga de los dispositivos en el sistema (en el arranque) se definirán tantas redes bayesianas como sea necesario para manejar todos los posibles conceptos de los distintos dispositivos. Esto no quiere decir que existan tantas redes como dispositivos, sino que cada red será reutilizada por los dispositivos que puedan adaptarse a ella y, si esta adaptabilidad no es posible, se debe crear una nueva red de manera que todos tipos de dispositivos estén contemplados. Siguiendo este planteamiento, se ha llegado al diseño de tres redes diferentes, tal como se expone a continuación.

Supóngase que la red va a ser atravesada por un dispositivo que maneja el concepto *iluminación*. Este concepto depende de factores internos, como el hecho de que haya lámparas encendidas en la habitación en cuestión, sin embargo la iluminación depende mucho de factores externos, pues una persiana subida en un día soleado, al mediodía, puede ser la solución óptima cuando se requiere luz en el interior de la habitación. Por tanto, para conceptos que dependen de factores externos, todos los nodos de evidencias orientados al contexto exterior son útiles, y el nodo de presencia externa (que aúna los resultados de estos nodos) resulta necesario.

Sin embargo, si el dispositivo en cuestión es un radiador, manejando *temperatura* como concepto, determinar la temperatura externa en función de la estación o el estado del cielo resultaría muy poco preciso. Recuérdese, no obstante, que la API climatológica informaba sobre la temperatura exterior. Siendo esto así, parece óptimo cargar en el nodo de influencia externa los valores obtenidos directamente desde la API, sin la necesidad de inferirlo. Para ello, a su vez, resulta necesario eliminar la dependencia con los nodos precedentes, pues de no hacerlo la propagación de probabilidades mediante el algoritmo *Junction Tree* sobrescribiría cualquier valor que se cargase previamente en el nodo. Teniendo esto en cuenta, surge la red de la Figura 35 (a), en

la que el nodo de presencia externa del concepto se convierte en un nodo de evidencias y puede escribirse arbitrariamente antes de ejecutar el algoritmo.

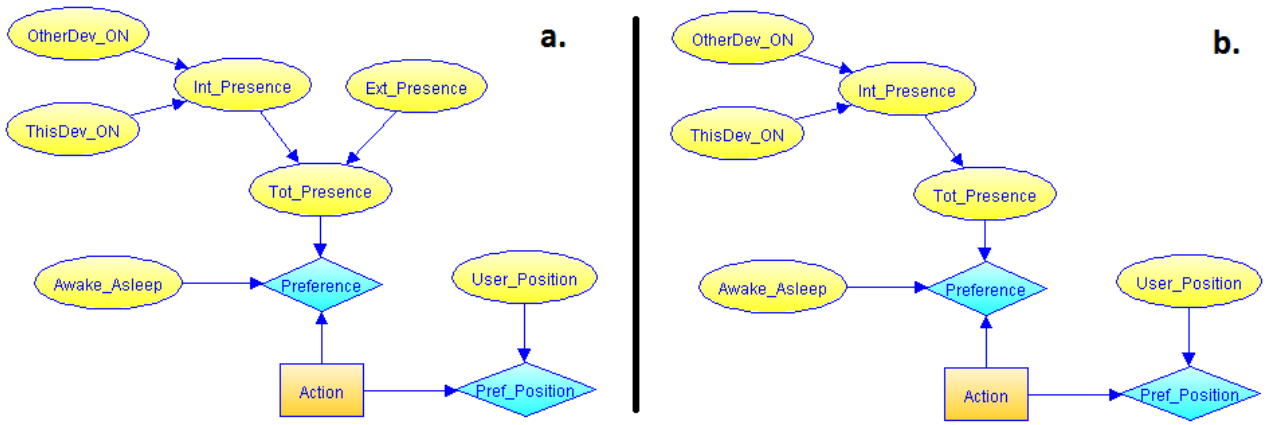


Figura 35. Redes bayesianas alternativas

Por su parte, puede darse el caso de manejar dispositivos que manejan conceptos ajenos a factores externos (p.e. una televisión o cualquier otro dispositivo multimedia). Para este tipo de dispositivos, la red de la Figura 35(b) se adapta a este hecho, con la única peculiaridad de que la presencia total depende únicamente de la presencia interna.

En la práctica, se agrupan las redes creadas a partir de los dispositivos de la siguiente manera:

- **Red completa para un único tipo de concepto**. Esto es así porque la red completa necesita rellenar para cada concepto la tabla de potencias del nodo *Ext_Presence*, dado que la dependencia respecto a los nodos precedentes varía en función del concepto. Esto se hace, como se verá en el apartado de configuración, aplicando un mecanismo que convierte un breve número de datos introducidos por el usuario en valores suficientes para completar la tabla de potencias. Como estos valores variarán en función del concepto, se debe almacenar una red diferente por cada uno de ellos.

- **Red parcial para un único tipo de medida externa (nodo de presencia externa como evidencia)**. De la misma manera, aunque la estructura sea válida para todos aquellos dispositivos que manejen conceptos cuya presencia externa se pueda obtener de manera precisa, depende de cuál sea esta medida, este nodo contendrá distintos valores. Se ha considerado que cada tipo de medida (entiéndase medida como, por ejemplo, la medida de *temperatura exterior*) es única para todos los dispositivos que hagan uso de dicha medida, y el rendimiento mejora si se reutiliza la red sin necesidad de cargar una y otra vez los mismos datos en el nodo, independientemente del concepto manejado.

  El siguiente ejemplo ilustra esta situación: un radiador, que maneja *temperatura* como concepto, hará uso de la medida de *temperatura exterior* en su análisis contextual, aunque también podría hacerlo un supuesto dispositivo de riego automático, cuyo concepto fuese otro (por ejemplo, la *humedad*), pero en el que pudiese ser interesante incluir la temperatura en el análisis de su contexto para determinar si es necesario o

no volver a regar las plantas. De esta manera, ambos dispositivos harían uso de la misma red sin recargar los valores del nodo y mejorando el rendimiento, a pesar de manejar diferentes conceptos.

- **Red común para cualquier dispositivo no incluido en los dos puntos anteriores (sin tener en cuenta la presencia externa del concepto)**. Esta red es única y la utilizarán todos los dispositivos que no se adapten a los casos anteriores, independientemente del concepto que manejen.

Este planteamiento lleva a almacenar un número reducido de redes bayesianas y, por otro lado, reduce la necesidad de modificar en exceso estas redes en tiempo de ejecución, reutilizando aquellos nodos que no varían entre dispositivos. Todos los nodos que no varían durante el tiempo de ejecución, como los que reflejan factores de franja horaria (incluido si el usuario está despierto o no) o los climatológicos, son cargados una sola vez en estas redes almacenadas antes de procesar la orden, limitando las modificaciones durante el proceso de decisión a aquellos nodos que varían en función del dispositivo, como el hecho de que el usuario se encuentre en la misma habitación (que dependerá de la habitación del dispositivo) o que otros dispositivos de su mismo concepto en dicha habitación estén encendidos (que de nuevo variará por cada uno de ellos).

4.5.4 Utilización de las redes bayesiana en el proceso de decisión

Las redes bayesianas, como se ha expuesto en los puntos anteriores, permiten inferir a partir de ciertas circunstancias contextuales la necesidad probable de que un dispositivo deba ser encendido, apagado o dejado tal como está. Esta salida de la red la proporciona el nodo *Action* tal y como se ha explicado previamente, pero cómo aplicar esa salida al proceso de decisión aún es un punto abierto.

Este mecanismo se podrá presentar con más detalle cuando se expongan los enfoques del algoritmo de interpretación en el siguiente apartado o incluso en el momento en el que se estudie la relación entre todos estos componentes en el apartado 4.10. No obstante, se pretende adelantar brevemente la utilización de estos valores en el proceso de decisión para que el análisis de la implementación de las redes bayesianas no quede incompleto. Los pasos para la aplicación de la salida de estas redes, a modo de resumen, son los siguientes:

1. Se inicializan todas las redes bayesianas definidas para los diferentes conceptos y medidas externas tal como se ha explicado, asignando valores a los nodos que no variarán durante el breve proceso de interpretación. Este paso incluye cargar todos los valores de presencia externa y el nodo que representa si el usuario está activo o la hora actual corresponde a una hora donde el usuario debería estar dormido[5].
2. Se lee de la base de datos, a partir de las normas establecidas en el algoritmo de interpretación, los datos de uno de los registros de las tablas de análisis de dispositivos candidatos, obteniendo sus datos contextuales (concepto que maneja, estado y habitación).

[5] Tal como se ha planteado el sistema, dado que el usuario debe dar órdenes para arrancar los distintos procesos, es evidente que el usuario estará siempre despierto a la hora de ejecutar una cierta acción. Lo que se pretende representar es el hecho de que, por el horario actual, es posible que el usuario pretenda reducir la presencia de ciertos conceptos que pueden molestarle durante el sueño, como podría ser la iluminación.

3. Se carga la red bayesiana que se adapta a este dispositivo y se leen los datos desde la ontología de dispositivos que permiten obtener el número de otros dispositivos encendidos que manejan el mismo concepto. Con esta información se inicializan los valores de los nodos de presencia interna.
4. Se obtiene si el usuario se encuentra o no en la misma habitación que el dispositivo. Si este dispositivo se encuentra en una habitación que no dispone de detector de presencia, se carga el nodo de presencia asignando el 50% a cada posibilidad. Si existe sensor de presencia, se da el 100% al hecho de que el usuario esté o no en dicha habitación.
5. Con todos los nodos de evidencias cargados, se ejecuta el algoritmo *Junction Tree* y se lee la salida para el posible estado que puede ejecutar este dispositivo. Por ejemplo, si un dispositivo está encendido, la opción de encender el dispositivo no se estudiará, por imposible. Sólo se tendrá en cuenta la puntuación de salida para apagar el dispositivo o no hacer nada. El razonamiento inverso se aplica también para dispositivos apagados.
6. Con el resultado de salida oportuno puede ocurrir varias cosas. En la configuración de usuario existen mecanismos que permiten seleccionar que si una acción ha tenido menos puntuación que "no hacer nada" ésta se anule (o bien que se reduzca mediante un factor), aunque también puede dejarse tal como se lee. De esta manera, aunque no es parte del algoritmo de interpretación propiamente dicho, la configuración de usuario puede tener repercusiones importantes en la conclusión final.
7. Se lee de la base de datos la puntuación que había obtenido para cada consulta el registro en cuestión (se verá en detalle en el próximo apartado) y ésta se multiplica por la puntuación de salida del contexto. De esta manera, las puntuaciones del contexto afectan directamente a las puntuaciones propias de la interpretación de la orden, fusionándose y dando lugar a una puntuación total que el sistema tomará por definitivas (y que se almacenará también en la base de datos).
    a. En realidad existe algún paso posterior que aplica las preferencias a los dispositivos, modificando de nuevo la puntuación total obtenida en este punto, pero se reserva la explicación para el próximo apartado.
8. Se vuelve al punto 2, pasando al siguiente registro de la tabla de análisis, paso que se repetirá hasta que se hayan procesado todos.

4.6 Algoritmo de interpretación

El algoritmo de interpretación o algoritmo del intérprete es, sin duda alguna, el mecanismo que más flexibilidad otorga al sistema. Una vez establecidas las bases de los apartados anteriores en lo que respecta a los diferentes módulos (cómo simular la experiencia mediante los buscadores, cómo manejar el contexto, cómo estructurar la información, cómo almacenar y recuperar datos, etc.), se ha llegado al punto clave de todo el sistema, el mecanismo que regula y unifica todos estos planteamientos.

En apartados anteriores, con el objetivo de ir introduciendo brevemente la lógica subyacente tras el intérprete, se han mencionado algunas de sus características y se ha visto (ejemplo del apartado 4.2.2) cómo una versión muy simplificada de este algoritmo podía utilizarse para inferir una conclusión a partir de una orden.

En este apartado, además de exponer las características de dicho algoritmo en detalle, se terminará de definir el uso que se le da a otras partes del sistema que han quedado pendientes de explicación en espera de que el algoritmo estuviese completamente presentado (como por ejemplo la utilización de las herramientas de la Base de Datos y la aplicación de preferencias a partir del contexto).

4.6.1 Sintaxis y rellenado de tablas de análisis

El algoritmo, tal como se verá, posee una cierta complejidad inherente al requisito de flexibilidad que se le ha impuesto. Para que esta complejidad quede acotada y la representación de los datos pueda ser manejada de una manera fluida por el sistema, se ha diseñado e implementado el algoritmo de manera que, en su fondo, representa cómo rellenar una o varias tablas de la Base de Datos. Este enfoque tiene varias ventajas, entre las que se destacan:

- La información queda bien estructurada, de manera que puede ser consultada en el proceso de desarrollo y depuración, así como a la hora de trazar cómo se ha obtenido un cierto resultado.
- Se pueden utilizar los mecanismos propios de la BD para procesar la información contenida, lo que reduce en gran medida la cantidad de código requerido. Además, estos mecanismos ya optimizados permiten acelerar el procesado y aumentar el rendimiento.

A modo de primera aproximación, la Tabla 7 representa cómo el algoritmo relaciona la estructura de la tabla con el contenido que debe albergar. En esta ilustración queda reflejada una única tabla, pero el algoritmo puede manejar tantas tablas como el desarrollador considere necesario para garantizar un buen funcionamiento del sistema.

Tabla 7. Relación algoritmo - tabla

NOMBRE DE TABLA									
KEY 1	KEY ...	KEY m	QUERY 1	QUERY ...	QUERY n	RES 1	RES ...	RES n	TOTAL

Observando esta tabla vemos que hay, además del nombre, tres bloques:

- Columnas conteniendo valores clave (*key values*): estos valores serán referencia para el siguiente bloque (*queries*) y entre otras características, servirán de base en la unión de registros. Su utilidad se verá en detalle más adelante.
- Columnas conteniendo consultas (*queries*) y sus resultados (*results*): estas columnas contendrán los valores que posteriormente serán utilizados en el motor de búsqueda, cuyo resultado será a su vez almacenado en la columna *Res x* correspondiente.

- Columna de resultado total (*total*): esta columna contendrá la suma de todas las columnas de resultados, con el correspondiente peso del contexto aplicado a ella.

Para la definición de la sintaxis y los mecanismos que se ejecutarán tras ella, se comienza exponiendo en la Figura 36 un algoritmo tal como los que puede utilizar en el sistema. De hecho, este algoritmo es uno de los más eficientes en cuanto a la relación *aciertos – tiempo de ejecución* que se han diseñado a lo largo del desarrollo del proyecto. Por supuesto, la sintaxis ha sido definida had-hoc durante el diseño del sistema con el fin de asemejar su estructura a aquella que poseerá en la tabla correspondiente.

```
<ANALYSIS_DEVICE_TABLE>
  {$O_o_id , $D_d_id}
  {
  (x0,wa,ct,cu2)#[$D_sa_conceptActionDescription]
  & (x0,wc,ct,cu3)#[$D_c_conceptName]
  & (x0,wr,ct,cu4)#[$D_r_roomID]
  & (x0,ws)#[$D_s_logState]
  & (x0,wn)#[$D_sa_newLogState]
  & (x0,cu1)#[$D_sa_actionStatement]
  & (x15,e350)#[$O_n_value + "$D_d_deviceName"]
  & (x50)#["$O_v_value $D_d_deviceName"]
  & (x30,e200)#[ "$O_n_value $D_r_roomID" ]
  & (x30,e200)#[ "$D_r_roomID $O_v_value"]
  }
&&
<ANALYSIS_CONCEPT_TABLE>
  {$O_o_id , $D_d_id}
  {
  (x0,wa,ct)#[$D_sa_conceptActionDescription]
  & (x0,wc,ct)#[$D_c_conceptName]
  & (x0,wr,ct)#[$D_r_roomID]
  & (x0,ws)#[$D_s_logState]
  & (x0,wn)#[$D_sa_newLogState]
  & (x250)#["$O_n_value $D_c_conceptName"]
  & (p,x200)#["$D_sa_actionDescription * $O_n_value"]
  & (p,x200)#["$D_sa_conceptActionDescription * $O_n_value"]
  & (p,x200)#["$Ox_adj_value" + "$D_sa_actionDescription * $O_n_value"]
  & (p,x200)#["$Ox_v_value" + "$D_sa_actionDescription * $O_n_value"]
  }
```

Figura 36: algoritmo de interpretación

Se define a continuación la sintaxis utilizada para representar el algoritmo que define el comportamiento del sistema, si bien los mecanismos subyacentes se expondrán más adelante con el fin de no sobrecargar esta descripción, así como las restricciones propias del sistema. En esta descripción se tratará de representar únicamente cada parte del algoritmo, indicando qué y cómo es procesada:

- **<** *NOMBRE_DE_TABLA* **>** ***{*** *INFO_KEYS* ***}{*** *INFO_COLUMNAS* ***}***
  &&
  < *NOMBRE_OTRA_TABLA* **>** ***{*** *INFO_KEYS* ***}{*** *INFO_COLUMNAS* ***}***
  …

El algoritmo permite definir tantas tablas como se desee, pero al menos debe definirse una, que además será la *tabla principal* (la primera definida). Por cada tabla debe definirse un conjunto de valores clave (*keys*) y otro conjunto de columnas, que se transformarán en las columnas de *queries* y *resultados* presentadas previamente.

- ***{** KEY_1**,** KEY_2**, … ,** KEY_N **}***

  Lo que anteriormente se ha denominado como *INFO_KEYS* es en realidad un conjunto de valores clave, que pueden ser de uno de estos tipos:

  - Una variable: en este caso el valor debe comenzar con el carácter *$* y su sintaxis debe respetar el siguiente patrón:
    - *$<ID de ontología>_<ID de clase>_<ID de slot>*

    Como se puede deducir, una variable en el algoritmo representa un valor recuperable de alguna de las ontologías, definido por su *clase* y *slot*. Esta variable, por otra parte, podría ser cargada con muchos valores, tantos como *instancias* de la clase en cuestión haya en la ontología. En algunos casos, incluso para una misma *instancia* podría haber varios valores válidos si el *slot* tiene cardinalidad múltiple, es decir, si puede contener más de un valor. En otros, si es *slot* no es obligatorio, podría no haber ningún valor para esta variable.

    Por su parte, se puede observar en el ejemplo de la figura que tanto la ontología como la *clase* están representadas mediante una única letra. Esto es así por una cuestión meramente de diseño (reduciendo la cantidad de texto necesario en el algoritmo y mejorando su legibilidad). No se entrará en detalle en este apartado sobre la nomenclatura de estos identificadores, sino que se reserva dicha descripción para el anexo correspondiente al manual de usuario.

  - Una constante: cualquier valor que no sea una variable (que no comience con el carácter *$*) se tomará como constante y no se procesará, sino que el dato será cargado con el valor tal cual es descrito. En el ejemplo de la Figura 36 no se ha utilizado constantes[6] en los valores clave.

- ***{***
  (** LISTA_DE_OPCIONES **)#[** QUERY_1 **]
  &
  (** LISTA_DE_OPCIONES **)#[** QUERY_2 **]
  &
  …
  }

  Este punto corresponde al contenido que previamente se denominó *INFO_COLUMNAS*. En lo que respecta a la Tabla 7, cada bloque de información (separado por el carácter *&*)

[6] Aunque las constantes podrían resultar útiles en el diseño de algunos algoritmos y depende del desarrollador tener en cuenta su influencia, en el desarrollo de este proyecto se ha observado que no aportan funcionalidad destacable. No obstante, en pro de una mayor flexibilidad y dejando la puerta abierta a futuros diseños, se ha mantenido la posibilidad de manejar estos valores.

corresponde a una columna de tipo *query* y, a su vez, a una columna de tipo *result* (Tabla 7).

Por cada tabla definida en el algoritmo, se pueden definir tantos bloques de información de columna (y, por extensión, tantas columnas de tipo *query* y *result*) como se quiera, si bien existirán algunas columnas obligatorias con ciertos requisitos, lo cual se explicará más adelante. La explicación de la *LISTA_DE_OPCIONES* se pospone y se pasa a la descripción de la información contenida bajo el identificador *QUERY_n*:

- Todo lo que se encuentre contenido en este bloque *QUERY_n* corresponde a una futura consulta en el motor de búsqueda y será almacenada en una columna de tipo *query*, cuyo resultado será recopilado a su vez en la columna *result* correspondiente (véase Tabla 7). Esto explica por qué hay siempre el mismo número de columnas de cada uno de estos dos tipos y su estrecha relación.
- Los tipos de valores contenidos en este bloque pueden ser los mismos que en las *keys* (variables o constantes), pero además la combinación de ambos es posible y, de hecho, necesaria para poder relacionar conceptos. Las variables, que de la misma manera se obtienen desde la ontología correspondiente, pueden combinarse con otras variables dentro de la misma *query* y con una serie de constantes.

  El intérprete se encargará de procesar palabra a palabra y determinar cuáles deben ser transformadas por su valor y cuales deben permanecer tal como son. Este proceso permite construir secuencias complejas que combinen datos de las diferentes ontologías y aprovechen la sintaxis propia de los motores de búsqueda para obtener diferentes tipos de resultado.

  No obstante, resulta evidente que de una misma *query* definida en este bloque pueden resultar diferentes combinaciones de valores. Esta circunstancia es manejada por el intérprete replicando las filas (registros en la tabla) con los datos almacenados hasta el momento y añadiendo cada nueva posible combinación en uno de estos registros. Este proceso implica una rápida expansión del tamaño de la tabla, de manera exponencial según aumenta la cantidad de combinaciones posibles y, a su vez, el número de *queries* que se quieren procesar (el número de columnas definidas). Tal como se ha mencionado en varias ocasiones a lo largo de este documento, la búsqueda del equilibrio entre la cantidad de datos a procesar y el índice de aciertos es absolutamente clave.

  Aunque no se mencionó con anterioridad, los datos introducidos en las columnas *key* siguen este mismo mecanismo y podrían rellenarse con combinaciones de variables y constantes. Por su parte, el número de registros se multiplicará para albergar todas las combinaciones, tanto de valores simples (una sola variable en una misma columna) como de valores compuestos (varias variables combinadas en una misma columna).

  Como añadido, se puede observar que la referencia a variables de la ontología de órdenes tiene una pequeña variante en algunos casos. Si el campo <*ID de ontología*> está identificado por el valor *Ox*, esa *x* añadida identifica que esa variable debe

contener los valores de esa *clase* y *slot* para todas las instancias de la estructura de la orden desde esa clase hasta el tronco. Por ejemplo, si en una estructura de una orden para *"hay poca luz"* existe un verbo (*hay*) colgando de un adjetivo (*poca*) que a su vez cuelga de un sustantivo (*luz*), la variable *$O_v_value* (que referencia a un verbo) correspondería al valor "*hay"*, mientras que la variable *$Ox_v_value* correspondería a "*hay poca luz"*. Por su parte, como sólo tiene en cuenta el sentido ascendente (hacia el tronco), la variable *$Ox_adj_value* (que referencia a un adjetivo) correspondería al valor *"poca luz"*.

- Existe una relación de dependencia entre los valores cargados en las variables de las columnas *query* y los valores cargados en las columnas *key*, así como una relación de dependencia de todas las columnas *query* entre sí. Esto quiere decir que un mismo registro sólo puede albergar información de un mismo dispositivo en todas sus columnas.

  Supóngase el siguiente ejemplo simplificado para representar esta situación:

  - *{ $D_d_id } { [" $D_d_name * $D_r_roomID "] }*

  En este ejemplo se tiene única *key*, que hace referencia al ID de los dispositivos, y una única *query,* que utiliza el nombre y ubicación de los dispositivos (se obvia la sintaxis de definición de la tabla y la correspondiente a las opciones).

  A la hora de procesar la *key* y rellenar la tabla de análisis, el intérprete obtendrá tantos Identificadores de dispositivos como dispositivos haya, lo que implica que la tabla pasará a tener tantos registros como dispositivos para contener todos los posibles valores.

  En el momento de procesar la *query* resulta evidente que, de nuevo, si se analizase de manera independiente la variable *$D_d_name* se obtendrían tantos nombres de dispositivos como dispositivos contiene la ontología. No obstante, la tabla carecería de utilidad en el proceso de decisión si se entremezclasen datos de dispositivos sin atender a su relación, mezclando el ID de un dispositivo con el nombre de otro dispositivo diferente.

  Para mantener la relación entre los datos de las diferentes columnas, el intérprete requiere que todos los valores cargados para las variable de un mismo registro pertenezcan al mismo dispositivo, de manera que si el ID del dispositivo en un registro es *lam_1* (correspondiente a una lámpara del salón) la *query* del ejemplo sólo podría contener el valor *"lámpara * salón"* en dicho registro, resultando este en su conjunto de la siguiente manera:

  - *{ lam_1 } { [" lámpara * salón "] }*

  Como se ha mencionado, resultaría inútil para el proceso de decisión si los registros quedasen ordenados con valores como el siguiente:

  - *{ lam_1 } { [" radiador * cocina "] }*

Esta circunstancia se repite para todas las columnas de un mismo registro y obliga al intérprete a procesar gran cantidad de información, pues requiere contrastar cada nuevo posible valor con los valores añadidos anteriormente en el registro.

En contrapartida, este requisito permite que la puntuación de un registro represente la puntuación de una acción para un único dispositivo y, por tanto, convierte a la tabla en una herramienta eficaz a la hora de procesar y seleccionar o descartar dichas acciones. Además, conteniendo el identificador en la columna *key*, el intérprete es capaz de identificar fácilmente dicho dispositivo y actuar en consecuencia.

Aunque en este ejemplo sólo se han planteado variables de la ontología de dispositivos, se pueden entremezclar variables de la ontología de órdenes (se debe, de hecho, para inferir conclusiones de la relación de la orden y la red de dispositivos).

En resumen, el valor que se haya almacenado en la columna *key* determina qué valores serán cargados a partir de las variables de las columnas *query* en el registro en cuestión, tanto para la ontología de órdenes como para la de dispositivos. A su vez, cada nuevo valor para una columna *query* dependerá de los valores previamente cargados en el registro. Por ejemplo, un mismo registro correspondiente a una lámpara no podrá contener una *query* que haga referencia a *encender* la luz y otra *query* que haga referencia a *reducir* la iluminación, sino que en este registro sólo sería válida la combinación *encender* luz con *aumentar* iluminación o la combinación *apagar* luz y *reducir* iluminación. De esta manera, cada registro, además de identificar a un dispositivo, también identifica una acción concreta de ese dispositivo.

Para terminar en cuanto al procesamiento de variables y combinaciones se refiere, se considera oportuno aclarar que las celdas pueden permanecer vacías si las variables no devuelven valores o los valores que devuelven no son válidos en algún registro debido a la dependencia previamente mencionada. En la práctica, el intérprete carga un valor de escape (configurable) para identificar que una cierta variable no tiene valores válidos. Este enfoque permite identificar estos casos incluso en *queries* combinadas. La ubicación de celdas con variables no cargadas resulta muy práctica para el intérprete, pues puede anular el resultado de esta celda por no estar representando correctamente la combinación de conceptos.

Previamente a la exposición del resto de componentes del algoritmo, se muestra la Figura 37 representando de manera simplificada lo ya analizado en los puntos previos. La figura sólo muestra los primeros pasos en el proceso de relleno de una tabla cualquiera y las celdas se representan mediante nombres representativos (aquellos que contienen la *O* pertenecen a la ontología de órdenes, mientras que la *D* hace referencia a la ontología de dispositivos). En el ejemplo se contemplan únicamente dos instancias de órdenes (dos estructuras de árbol de la orden del usuario, tal como se vio en el apartado 4.3.2), dos dispositivos y dos acciones por cada uno de estos dispositivos.

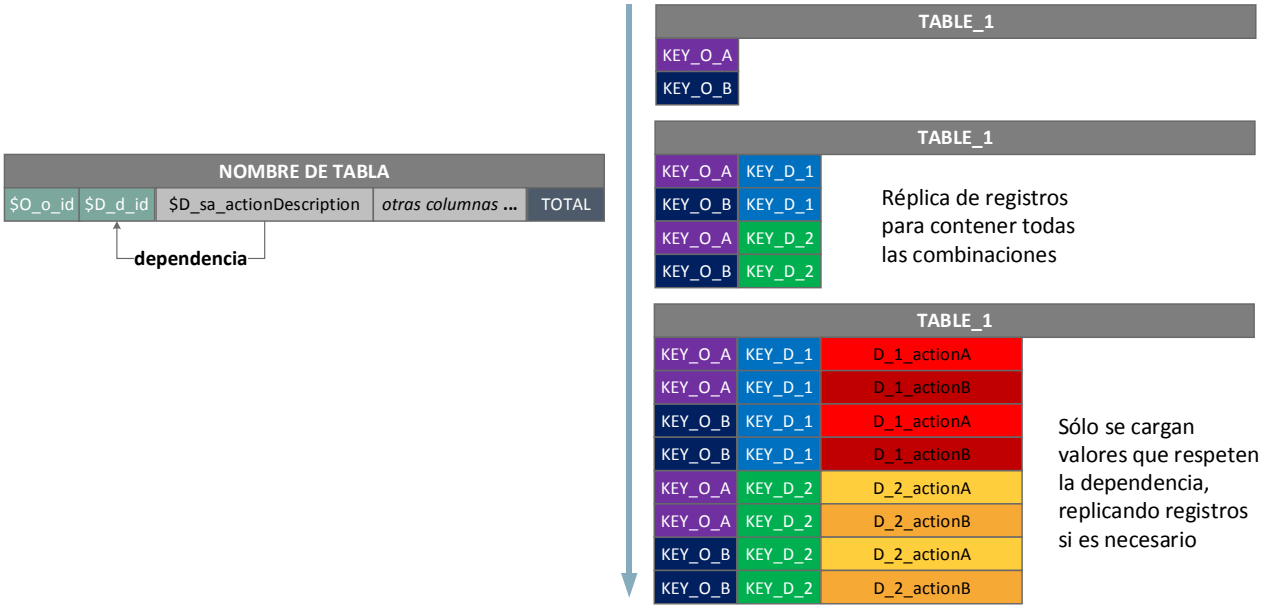


Figura 37. Proceso de rellenado de la tabla de análisis

Una vez alcanzado este punto, la correspondencia entre cómo se desea que sea formada la tabla en la base de datos y cómo debe esto declararse en el algoritmo ha quedado reflejada. Como se observa, es en realidad un proceso bastante intuitivo, declarando tablas por su nombre y definiendo las *keys* y las *queries* que se quieren cargar en sus columnas.

Tal como se refleja en la Figura 37, lo primero que hace el intérprete es leer el algoritmo y rellenar las tablas declaradas con todas las combinaciones válidas (extrayendo la información desde las ontologías), hasta tener completas todas las celdas correspondientes a *keys* y *queries*, pero dejando vacías aquellas pertenecientes a resultados parciales y resultado total. Para ello, el intérprete requiere de la definición de nuevas tablas en la BD (aquellas que se denominaron tablas dinámicas en el apartado 4.4 por su dependencia del algoritmo).

El siguiente paso que realiza el intérprete, una vez las tablas y los registros están completamente definidos a falta de rellenar las celdas de resultados, es recorrer dichos registros, uno a uno. Se expone a continuación la secuencia de ejecución:

1. Se leen del registro las columnas que contienen la información del estado actual y la posible acción del dispositivo. Si son incompatibles (p.e. encender una lámpara que ya está encendida), este registro se ignora dejando vacías todas las celdas de resultados, pues no es una acción que se quiera como conclusión. El mecanismo que utiliza el intérprete para obtener esta información de estado y acción se explicará más adelante.
2. Si en el paso 1 el registro no es descartado, se recupera el valor cargado en cada columna *query* y se consulta al motor de búsqueda, recuperando un resultado. La base de datos entra en juego, pues hace de intermediaria en este proceso y si el valor estaba ya contenido lo devuelve directamente sin necesidad de una nueva búsqueda. Si el valor no existía en la tabla de resultados, se realiza, ahora sí, la consulta al buscador.

Este resultado, no obstante, no es el valor que se cargará en la columna *result*. En este paso entra en juego la previamente denominada *LISTA_DE_OPCIONES*:

- ***(** opción_1 **,** opción_2**, … ,** opción_n **)#[** QUERY **]***

  Se trata de una lista de opciones que permite añadir tantas opciones como se deseen o bien no añadir ninguna. Estas opciones tienen muy diferentes funciones y se irán exponiendo en el momento que su papel resulte relevante. En este punto se introducen las opciones que permiten aplicar un cierto factor al resultado obtenido desde el motor de búsqueda.

  - *x<número>*: permite aplicar una cierta multiplicación al resultado obtenido (p.e. *x10* aumentará el resultado en *10 veces*). Se permiten valores decimales para decrementar el valor.
  - *p*: indica al intérprete que debe aplicar un aumento proporcional al número de palabras que contiene la búsqueda.

    Téngase en cuenta que, por ejemplo, los resultados que devuelve una consulta como *"persiana"* incluye, a su vez, todos los resultados de una búsqueda como *"levantar persiana"* (dado que una incluye a la otra). Para intentar poner al mismo nivel estos dos tipos de resultados se ha implementado esta opción. Si multiplicar únicamente por el número de palabras resulta insuficiente, se puede combinar con la opción anterior para aumentar este factor (p.e. la combinación *(p, x10)* en una *query* de 2 palabras resultaría en la multiplicación por *20* del resultado).
  - *e<número>*: aplica el factor en función de la cantidad de letras coincidentes entre palabras de una misma query.

    Para este análisis, el intérprete toma en cuenta sólo las palabras que provienen de variables e ignora las constantes. Si existen más de dos variables, se toma la coincidencia mínima entre todas ellas. Supóngase el siguiente ejemplo para ilustrar esta situación:

    - *(e10)#["$O_o_name + $D_r_roomID"]* → *(e10)#["cocina + cocina"]*

    En este ejemplo la orden contiene el sustantivo *cocina* (p.e. *apaga el radiador de la cocina*) y el registro en cuestión pertenece a un dispositivo ubicado en la cocina (supóngase el *radiador* de la *cocina*).

    Si el usuario ha utilizado la palabra *cocina*, es muy posible que los dispositivos de interés sean exclusivamente aquellos ubicados en ella. Esta opción permite primar esta serie de circunstancias más allá del resultado del buscador. De esta manera, como la coincidencia es del 100% entre ambas variables, se aplica el factor completo, que es *10*, luego se multiplica por *10* el resultado del buscador.

Si la relación de palabras hubiese sido, por ejemplo, *["cocinar + cocina"]*, la coincidencia sería de 6 letras respecto a 7 (palabra de mayor longitud), lo que supone un *86%* de coincidencia. En este caso, el factor de multiplicación quedaría como $10 \cdot 0{,}86 = 8{,}6$

Si el porcentaje de coincidencia es demasiado bajo, podría llegar a penalizar excesivamente el resultado, reduciéndolo o incluso anulándolo. Existe un parámetro en la configuración que regula si se permite que los resultados sean penalizados en estos casos o si se establece el mínimo valor del factor a *1*, para incrementar o mantener (pero de ninguna manera decrementar).

3. Una vez se ha recuperado el valor del motor de búsqueda para la query en cuestión (recuérdese que el valor del motor de búsqueda refleja la cantidad de resultados obtenidos en la consulta) y aplicado sobre ellos las opciones de manera acumulativa de manera que si hay varias opciones se aplicarán todas, una a una, el valor es almacenado en la columna *result* correspondiente.
4. Si hay más registros sin procesar, se procesa el siguiente (vuelta al punto 1).

4.6.2 Aplicación del contexto en el proceso de interpretación

Tal como se ha ido exponiendo, en este punto de la ejecución el intérprete ya ha cargado todas las columnas de resultados, pero aún no ha rellenado la columna *total*. Esto se debe a que, para ello, el intérprete debe aplicar el contexto a esta serie de resultados para tenerlo en cuenta en la decisión.

Lo cierto es que una primera aplicación del contexto ya ha sido llevada a cabo mediante el mecanismo que, a partir del estado de los dispositivos, ha descartado algunos registros. En este apartado se pretende exponer cómo se aplica el resto de factores contextuales analizados en el apartado 4.5 y cuya influencia se obtiene mediante una puntuación de salida de la red bayesiana correspondiente.

Tal como ya se adelantó, el contexto se había diseñado para ser procesado mediante el uso de redes bayesianas una vez por cada registro de estas tablas de análisis. En la práctica, dado que una de las acciones del dispositivo se ha anulado, solo se analizará un tipo de acción por cada dispositivo. Para obtener los valores requeridos en la carga de los nodos evidencias de la red bayesiana, entra en juego, de nuevo, la lista de opciones. Se exponen a continuación las opciones que resultan útiles en esta tarea:

- *wa*: esta opción le indica al intérprete que la información contenida en esta columna *query* es la información que identifica la acción conceptual[7] que se debe utilizar en el contexto.

[7] Se entiende acción conceptual como la acción que se aplica sobre el concepto, es decir, *AUMENTAR* o *REDUCIR* la presencia de este concepto (aumentar temperatura, reducir iluminación, etc.).

- *wc*: indica que la información contenida en esta columna hace referencia al concepto del dispositivo, el cual es necesario para seleccionar la red bayesiana correspondiente.
- *wr*: indica que esta columna informa de la habitación en la que se encuentra el dispositivo.
- *ws*: indica que esta columna contiene el estado del dispositivo.
- *wn*: indica que la información de esta columna refleja el nuevo estado que obtendría el dispositivo si se ejecuta esta acción (es decir, que si se selecciona la acción que representa este registro, por ejemplo *encender radiador*, el dispositivo pasará a estado lógico *ON*).

Todos los algoritmos que se diseñen y, más concretamente, las tablas que se definan, deben poseer estas columnas, pues de no ser así, aunque el intérprete pueda alcanzar los puntos anteriores y rellenar gran parte de la tabla, no podrá aplicar la influencia del contexto.

Por otro lado es interesante recalcar tres puntos respecto a estas opciones:

- Estas opciones son las que permitieron al intérprete obtener el estado y la acción y determinar qué registros eran inválidos para ignorarlos.
- Si el desarrollador considera que esta información no es apropiada para consultarse en el buscador (por ser palabras aisladas y no relacionar ningún concepto), se puede combinar con la opción *x0*. El intérprete ignora todas las *queries* con esta opción por no aportar peso a la decisión, así que el sistema se ahorra la necesidad de la consulta en el motor de búsqueda. De esta manera, la tabla puede contener información útil para varios componentes del sistema, no solo información a consultar en el buscador.
- Tal como se verá en el apartado de configuración, la entrada de datos de dispositivos en el simulador fuerza al usuario a definir esta información de manera que encaje a la hora de contrastarla con el contexto.

Recapitulando, el intérprete carga la red bayesiana oportuna, inicializa los nodos que dependen del dispositivo e invoca la ejecución del algoritmo *Junction Tree*. Después, en función de si la acción reflejada en este registro es de tipo *aumentar* o de tipo *reducir*, lee la puntuación de salida del contexto del tipo correspondiente.

Se recuerda que, tal como se mencionó previamente, en la configuración se permitirá definir un factor de reducción (o anulación) en caso de que la salida del contexto sea mayor para *no hacer nada* que para ejecutar la acción correspondiente. Es tarea del desarrollador decidir cómo tratar esta situación. En las configuraciones más eficientes que se han diseñado en el desarrollo del algoritmo, resulta suficiente con reducir ligeramente el valor de salida, pero la flexibilidad del algoritmo y la configuración permiten definir gran cantidad de enfoques que podrían ser igualmente válidos.

En este punto, el intérprete posee la puntuación resultante de la suma de las columnas de resultados (en función de los resultados devueltos por el motor de búsquedas) y el resultado del contexto. El siguiente paso consiste en multiplicar ambos valores y almacenar el valor resultante en

la columna *total*. Se puede afirmar que, ahora sí, las tablas de análisis están completamente rellenas y las relaciones de conceptos entre la orden y los dispositivos más exitosas contarán en este instante con una puntuación considerablemente mayor.

La Figura 38 pretende representar este proceso de obtención de resultados a partir de los valores ya presentes en el registro, así como su actualización con el resultado total obtenido. Esta figura proporciona una primera visión de conjunto en el proceso de decisión y evidencia la influencia conjunta de la *experiencia* y *sentido semántico* (mediante el uso del motor de búsqueda) y el *contexto* (mediante las redes bayesianas).

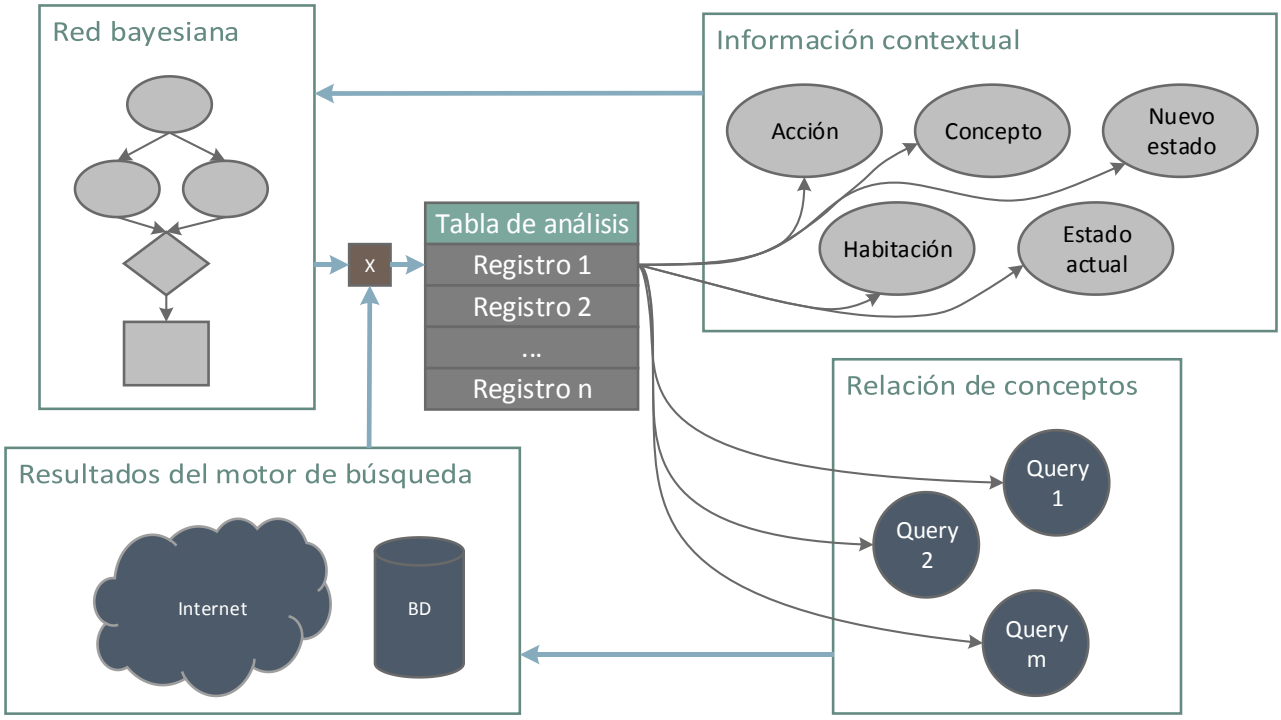


Figura 38. Asignación de puntuación por registro

4.6.3 Unificación de resultados para la obtención de conclusiones

El intérprete, en los pasos correspondientes del proceso de toma de decisiones que se han mencionado hasta el momento, ha obtenido una serie de registros y puntuaciones organizados, tal vez, en varias tablas. No obstante, la conclusión debe ser una.

Si se realiza un análisis más profundo de la serie de valores cargados en los diferentes registros se llegará a la conclusión de que, en realidad, existen varios registros que hacen referencia a un mismo dispositivo y acción. Esto es así debido a que la columna *key* que hace referencia al ID de un dispositivo se ha duplicado tantas veces como haya sido necesario para que la tabla contuviese todas las combinaciones con los ID de las distintas órdenes (esto ha sido ejemplificado en la Figura 37). De esta manera, los datos del dispositivo y acción se han combinado con diferentes estructuras de la ontología de órdenes y, en definitiva, existirán varios registros que hacen referencia a un mismo dispositivo y acción.

Surge la necesidad, por tanto, de unificar los valores de los distintos registros referentes a una misma acción para aunar todas las puntuaciones y obtener, finalmente, el dispositivo y acción con más probabilidades de éxito (tal como lo entiende el sistema). Este planteamiento permite que la decisión no se limite a seleccionar el registro que mejor se ha combinado con una estructura de la orden, sino que permite elegir el dispositivo que mejor se ha combinado con el conjunto de estructuras de la ontología de órdenes.

De nuevo, para que el desarrollador del algoritmo pueda indicar al intérprete cuáles son las columnas por las que debe unificar los registros, se hace uso de la lista de opciones. Téngase en cuenta, además, que los datos unificados se volcarán en una nueva tabla que se denominará *tabla de conclusiones* (que se entiende como otras de las tablas dinámicas, pues su definición depende del algoritmo), y que además en esta tabla se unificarán los registros de todas las tablas de análisis definidas en el algoritmo. Efectivamente, este es el mecanismo que permite inferir una conclusión independientemente del número de tablas de análisis definidas.

Se explica a continuación la opción de utilidad para este paso del proceso de decisión:

- *ct*: esta opción le indica al intérprete que esta columna debe utilizarse en el proceso de unificación de tablas (*join*). Siendo un poco más precisos, en la nueva tabla de conclusiones se unificarán todos los registros que contengan los mismos valores en las columnas con opción *ct*.

  La aplicación de esta opción tiene unos ciertos requisitos que se listan a continuación:

  - Se pueden definir tantas columnas con esta opción como se quiera, si bien deben ser las mismas columnas en todas las tablas de análisis. Debe existir al menos una columna de este tipo o el intérprete no sabrá qué tomar como base en la unificación.
  - Cuando se define que, por ejemplo, la columna *query_1* es de tipo *ct*, el intérprete entiende que debe unificar todas las tablas por esta columna, luego la condición para los registros que se añadan a la nueva tabla de conclusiones será que *tabla1.query_1 = tabla2.query_1* (y así sucesivamente). Es por esto que las columnas *ct* deben ser las mismas en todas las tablas.

  Para garantizar que no se unifican valores de dispositivos diferentes, la *key* del dispositivo, que debe corresponder a su ID (al menos una de las *keys* para que las *queries* tengan una referencia a la hora de cargar variables), se toma automáticamente como columnas *ct*.

  El ID de la orden, sin embargo, puede configurarse para que se unifique por la palabra clave de la orden, en lugar de por la estructura creada en la ontología. Por ejemplo, si a partir de una orden como *"enciende la luz de la cocina"*, se generaron dos estructuras cuyo concepto principal (el tronco de la estructura) está formado por la palabra *luz*, existirán registros de cada dispositivo combinados con cada una de estas estructuras (*luz_1* y *luz_2*), a pesar de que puedan contener

ambas una información similar y, en definitiva, los registros sean casi idénticos[8]. Recuérdese, no obstante, que cada estructura en la ontología de órdenes contenía palabras y/o dependencias distintas, a pesar de que la palabra principal sea la misma, luego podría resultar interesante no unificarlas (será decisión del desarrollador).

En resumen, esta opción permitiría que si, por ejemplo, la información referente a encender una lámpara se ha cruzado con dos estructuras de la orden cuyo concepto principal es el mismo (*luz* en el ejemplo anterior), se unifiquen como si de una sola estructura se tratase.

Desde el punto de vista semántico, esta opción permite favorecer aquellas combinaciones que pertenecen a estructuras de órdenes que se hayan formado en varias ocasiones (y, por tanto, que deban tener más sentido y por extensión más peso en la decisión). En definitiva, si se unifican los registros de estas dos combinaciones planteadas, el valor será mayor que aquellos registros que resultan de la combinación de estructuras en la ontología de órdenes que sólo se han formado una vez (lo que se puede entender como palabras con menos peso semántico).

La tabla de conclusiones, por su parte, contendrá las columnas *key*, cada una de las columnas *ct* definidas y, por último, una columna que contiene el resultado total como suma de los resultados parciales de los diferentes registros unificados. Se ilustra en la Tabla 8 el formato de esta tabla.

Tabla 8. Tabla de conclusiones

TABLA DE CONCLUSIONES						
KEY_Disp	KEY_Ord	QUERY ct 1	QUERY ct 2	...	QUERY ct n	TOTAL

4.6.4 Obtención de conclusiones

En este apartado se pretende exponer los últimos pasos en la obtención de la conclusión final que se le presentará al usuario, si bien la mayor parte del proceso ya ha sido planteado.

El intérprete, una vez ha rellenado la tabla de conclusiones, es capaz de ordenar los registros por la columna *total* y seleccionar el registro con mayor puntuación. No obstante, no se seleccionará directamente el dispositivo identificado en este registro, sino que lo que realmente se extraerá son los valores de las columnas *ct* de este registro.

Este enfoque pretende determinar qué factores son los que debe cumplir un dispositivo para satisfacer la necesidad. En el ejemplo de la Figura 36, estas columnas identificaban las siguientes características: *acción conceptual*, *concepto* y *habitación*. Es decir, si el dispositivo y acción con más puntuación tras todo el proceso de decisión hasta este punto ha sido, por ejemplo, *encender* una *lámpara* del *salón*, lo que el intérprete realmente extrae es que las características que un

[8] Este es otro de los motivos por los que almacenar búsquedas previas en la base de datos resulta útil. Aunque se generen registros con gran parte de la información duplicada, sólo se realizará búsquedas la primera vez; la segunda y sucesivas serán recuperadas desde la base de datos de manera rápida y eficiente y no sobrecargará en exceso el sistema esta duplicidad de información.

dispositivo debe cumplir para satisfacer la necesidad es que sea un dispositivo que pueda *aumentar* la *iluminación* del *salón*.

En este último paso entra en juego el concepto de *tabla principal* que se introdujo en puntos anteriores. Lo que realiza el intérprete en este momento es extraer de la primera tabla de análisis definida o *tabla principal* las puntuaciones de los dispositivos que cumplan los requisitos (en este ejemplo, *aumentar* la *iluminación* del *salón*). Este proceso podría recuperar dispositivos como persianas u otros dispositivos que aporten luz al salón. Nótese que la puntuación recuperada en este punto por cada dispositivo es aquella que contenía la primera *tabla de análisis*, no la compuesta entre todas las tablas de la *tabla de conclusiones* (el motivo de este planteamiento se verá en el apartado de diseño del algoritmo).

El intérprete maneja en este momento un número reducido de dispositivos (incluso puede que sólo uno) y sus puntuaciones. El último proceso por ejecutar es la aplicación de lo que se ha denominado las *prioridades*. El usuario o desarrollador puede configurar una serie de preferencias para ciertos tipos de dispositivos, de manera que su puntuación se aumente o se reduzca en función de ciertos factores contextuales.

Un ejemplo muy sencillo de *prioridad* es aquella que se le puede asignar a una persiana. El usuario puede preferir que, si la iluminación exterior es alta, se levante la persiana en lugar de encenderse una bombilla. En este supuesto, si el usuario indica que *"hay poca luz"*, es posible que el algoritmo determine que la solución sea encender una lámpara y que, sin embargo, lo propio sea levantar la persiana. No obstante, si el usuario indica "enciende la lámpara", a pesar de que sea mediodía y el sol brille intensamente, se puede deducir que el usuario ha sido muy explícito y, aunque la persiana tenga preferencia, debe encenderse la lámpara.

En el diseño de este paso debe tenerse muy en cuenta que el intérprete maneja puntuaciones, y la clave del éxito radica en poder variar dichas puntuaciones de una manera suficientemente importante como para que la persiana aumente su valor y rebase la puntuación de la lámpara pero que, por otro lado, si el usuario ha sido muy explícito, la puntuación de la lámpara sea suficientemente alta como para, aun aumentando la puntuación de la persiana, la lámpara siga teniendo el valor más alto. Esta situación se refleja en la Figura 39, donde se muestra la correspondencia de resultados deseada.

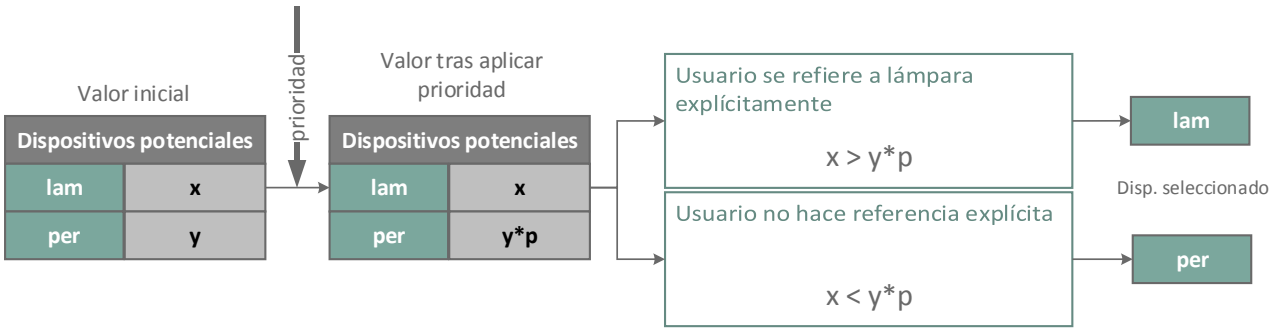


Figura 39. Resultados ideales en la selección de dispositivo

Conseguir que la situación de la figura sea la que realmente suceda en todas las ocasiones es, cuanto menos, complicado. Para ello se puede hacer uso de alguna de las opciones de

dimensionamiento mencionadas, como aquella que prima la coincidencia entre palabras, procurando que un dispositivo cuyo nombre (u otra característica) ha sido mencionado explícitamente por el usuario obtenga una puntuación tan elevada que la aplicación de la prioridad no consiga sobrepasar dicha puntuación.

Por otra parte, una prioridad no tiene por qué aumentar siempre la puntuación inicial de un dispositivo, sino que debe ser capaz de reducirla si las condiciones contextuales no son propicias. Utilizando de nuevo el ejemplo de la persiana, si el usuario requiere luz y la orden se ejecuta de noche, poco podrá hacer una persiana para proveer la requerida luz, luego su puntuación debe ser penalizada.

En la búsqueda de la simplicidad pero, a la vez, flexibilidad, se ha diseñado un mecanismo matemático sencillo que permite dimensionar la puntuación en función de algún factor contextual tal como sigue:

- El usuario, mediante la configuración (al añadir un dispositivo), puede o no añadir una o varias *prioridades* a este dispositivo.
- Al añadir la prioridad, el usuario determina en base a qué nodo de la red bayesiana contextual se debe aplicar esta prioridad, el valor de referencia de dicho nodo y el factor a aplicar. Por ejemplo, se puede determinar que una persiana depende del nodo de *presencia externa*, tomando como referencia el valor *alta* y aplicando un *factor = 1,5*.
  - El intérprete recupera la red bayesiana correspondiente al dispositivo y lee el valor del nodo en cuestión, concretamente el correspondiente al valor de referencia indicado. Imagínese que se recupera que la *iluminación exterior* es *alta* en un 80%, por lo que el valor recuperado es *0.8*.
  - Se aplica la siguiente ecuación, siendo *r* la puntuación base, p el valor recuperado del nodo y *f* el factor a aplicar:

$$r \cdot [(p \cdot 2)^f] \qquad (7)$$

  - El resultado de esta ecuación será la puntuación final del proceso de decisión. Si *f* es positiva, téngase en cuenta que siempre y cuando $p > 0.5$ (es decir, que acumule más la mitad de la probabilidad) el resultado final será mayor que el inicial. Por el contrario, el resultado disminuirá si $p < 0.5$. Por su parte, si $p = 0.5$ el resultado final será igual al inicial.

    Esta sencilla ecuación permite incrementar o reducir la puntuación de salida de un dispositivo concreto en función de si es más o menos probable que el factor contextual de referencia se esté produciendo. Siguiendo con el ejemplo planteado, si la iluminación exterior es alta con probabilidad 0.8, y se aplica por ejemplo $f = 1.5$, la salida sería:

$$r \cdot [(0.8 \cdot 2)^{1.5}] = 2.024 \cdot r \approx 2r \qquad (8)$$

    Por el contrario, si en el atardecer la luminosidad se reduce y $p = 0.2$, entonces:

$$r \cdot [(0.2 \cdot 2)^{1.5}] = 0.253 \cdot r \approx \frac{r}{4} \qquad (9)$$

Como añadido, es importante comentar que el intérprete puede manejar un factor directo (que se aplica sin atravesar la ecuación). Aunque en los algoritmos más eficientes no se ha hecho uso de esta opción, puede resultar interesante en caso de que se observe que un dispositivo sale seleccionado de manera demasiado habitual (reduciendo su valor) o, por el contrario, incrementar el valor porque su puntuación sea siempre demasiado baja (quizás por una pobre presencia en Internet dado que el dispositivo es una novedad).

Como último paso, una vez seleccionado un único dispositivo a través del procedimiento de prioridades, se debe extraer la información de interés. Esta información está indicada por el último conjunto de opciones que aún no se había explicado:

- *cu1*: indica que esta columna contiene el comando que identifica inequívocamente la acción del dispositivo.
- *cu2*: establece que la información sobre la *acción conceptual* (*aumentar* o *reducir* la presencia del concepto) está contenida en esta columna.
- *cu3*: identifica la columna que almacena el *concepto* que maneja este dispositivo.
- *cu4*: indicando que la columna en cuestión contiene la *habitación* donde se encuentra el dispositivo.

Estos valores, que pueden ubicarse en cualquier columna pero que sólo se procesarán en la primera tabla, son los que se utilizarán para informar al usuario a través de la GUI (aplicación Android), preguntándole si efectivamente desea *aumentar* la *luz* del *salón*, por ejemplo. El comando de ejecución se reserva para que, en caso de confirmación, el *Controlador principal* pueda invocar la acción en el *Simulador*. De nuevo, acompañar estas columnas con la opción x0 hace su procesamiento más liviano, al no tener que interrogar a la BD ni al motor de búsqueda y no afectando al proceso de decisión por ser información que no se ha cruzado con la orden.

4.6.5 Diseño del algoritmo

Como último punto en lo que se refiere al algoritmo y los procesos subyacentes que aplica el intérprete, se presentan brevemente algunos factores a tener en cuenta en el diseño del algoritmo.

La división en varias tablas de análisis puede resultar útil a la hora de obtener una buena conclusión. En el algoritmo de la Figura 36, se observan dos tablas con nombres que dan una primera idea de lo que representan:

- Tabla principal (de dispositivos): contiene, además de las columnas requeridas, un conjunto de combinaciones entre las órdenes y los dispositivos orientadas a puntuar los dispositivos por su posible correspondencia con las palabras del usuario. De esta manera, esta tabla asignará más puntuación a dispositivos con más relación con las palabras pronunciadas (o escritas) y no tanto por el concepto que representen. Así, valores como el nombre o la habitación pueden ser determinantes.

- Tabla secundaria (de conceptos): esta tabla está más orientada a inclinar el peso hacia los dispositivos cuyos conceptos (iluminación, temperatura, humedad, música, etc.) y acciones (encender, levantar, etc.) guarden más relación con las palabras del usuario.

Dado que el intérprete obtendrá el tipo de dispositivo necesario como unión de estas dos tablas, tanto los datos propios de los dispositivos como aquellos referentes al concepto o acciones que manejan dichos dispositivos serán tenidos en cuenta (como si de una sola tabla se tratase). Sin embargo, a la hora de realizar la selección final (con los dispositivos ya filtrados por el concepto, acción y habitación oportunos), la puntuación a tener en cuenta será únicamente la de la primera tabla (*tabla principal*). Esto facilita que el hecho de que el usuario se refiera explícitamente a un dispositivo y/o habitación quede mejor reflejado a la hora de aplicar la *preferencia*.

Se ha comprobado, entre los diferentes diseños, que éste resulta el más óptimo en cuanto a índice de aciertos, aunque en contrapartida ralentiza el tiempo necesario para la ejecución, dado que el incremento de registros requiere de más procesamiento.

En cuanto al control de errores en la formulación del algoritmo, y siguiendo el principio de limitación temporal planteado en el apartado 3.1.2, se ha procurado que el sistema capture todos los posibles errores cuyo origen sea el algoritmo y su errónea definición, pero el sistema no se repondrá en caso de que esto suceda, sino que informará del tipo de error y cerrará convenientemente. Por tanto, es tarea del desarrollador implementar cuidadosamente el algoritmo y testarlo hasta que el sistema responda tal como se espera, pues incluso sin errores aparentes podría estar devolviendo resultados erróneos como resultado de una mala configuración.

El proceso de desarrollo del algoritmo, aunque puedan establecerse a priori los conceptos que se quieren cruzar entre la orden y la información de los dispositivos, requiere siempre de una depuración mediante prueba y error. Esto se debe a la imprevisibilidad de los resultados devueltos por el motor de búsqueda. Internet no resulta un contenedor de información estable y predecible, y ni mucho menos proporciona una garantía de que los resultados sean los esperables. En ocasiones los resultados se salen de lo que podría considerarse razonable y el algoritmo debe intentar contener estas variaciones.

Como ya se ha mencionado en varias ocasiones, el algoritmo que requiere un tipo de red puede ser muy distinto al que requiere otra, dependiendo del número de dispositivos, su ubicación y su tipo. Atendiendo a estas características, el desarrollador debería dar más peso a los factores más diferenciales.

Por último, se reitera que la clave en un funcionamiento apropiado del sistema radica en la búsqueda de un punto medio entre índice de aciertos y tiempo requerido para la ejecución. Se podría pensar que añadir un gran número de cruces (*queries*) en el algoritmo incrementará siempre este índice de aciertos, pero en la práctica añadir un cruce no suficientemente diferencial con gran peso puede penalizar el resultado, así como el tiempo necesario de ejecución.

4.7 Simulador

El simulador es una pieza fundamental en el sistema desarrollado, no sólo por representar a la red de dispositivos e independizar el proceso de decisión de la tecnología subyacente, sino por flexibilizar en gran medida la cantidad y tipo de dispositivos que el sistema maneja.

Tal como se ha mencionado, el algoritmo depende en gran medida del número y tipo de dispositivos presentes en la red, y su nivel de desarrollo podría no haber alcanzado un nivel de detalle suficiente si la red hubiese contado con una cantidad reducida de dispositivos de una manera invariable (como sucedería en la red LonWorks presente en el Hogar Digital). El hecho de poner a prueba la IA con diferentes dispositivos ha conducido a la implementación de más opciones y procesos en el algoritmo, que en última instancia han flexibilizado en mayor medida la adaptabilidad de éste, mejorando el resultado.

No se pretende en este apartado entrar en gran detalle sobre cómo se hace uso del Simulador (ello se tratará en el Anexo I). En este apartado se procurará exponer brevemente las características del Simulador, así como los procesos de comunicación subyacentes entre éste y el Controlador Principal.

4.7.1 Características del Simulador

La interfaz gráfica de usuario del Simulador, creada mediante las herramientas proporcionadas por la *API Java 2D* [42], se puede definir como una única ventana que contiene información de todos los dispositivos de la red. El diseño de este componente se ha basado en los *Paneles de dispositivos* del *Hogar Digital*, que divide un conjunto de dispositivos LonWorks por habitaciones.

Dado que la finalidad de este *Simulador* es jugar el papel de dichos paneles, resulta apropiado este diseño orientado a la representación de dispositivos por habitaciones. Se muestra en la Figura 40 una captura del Simulador en funcionamiento, con la GUI desplegada.

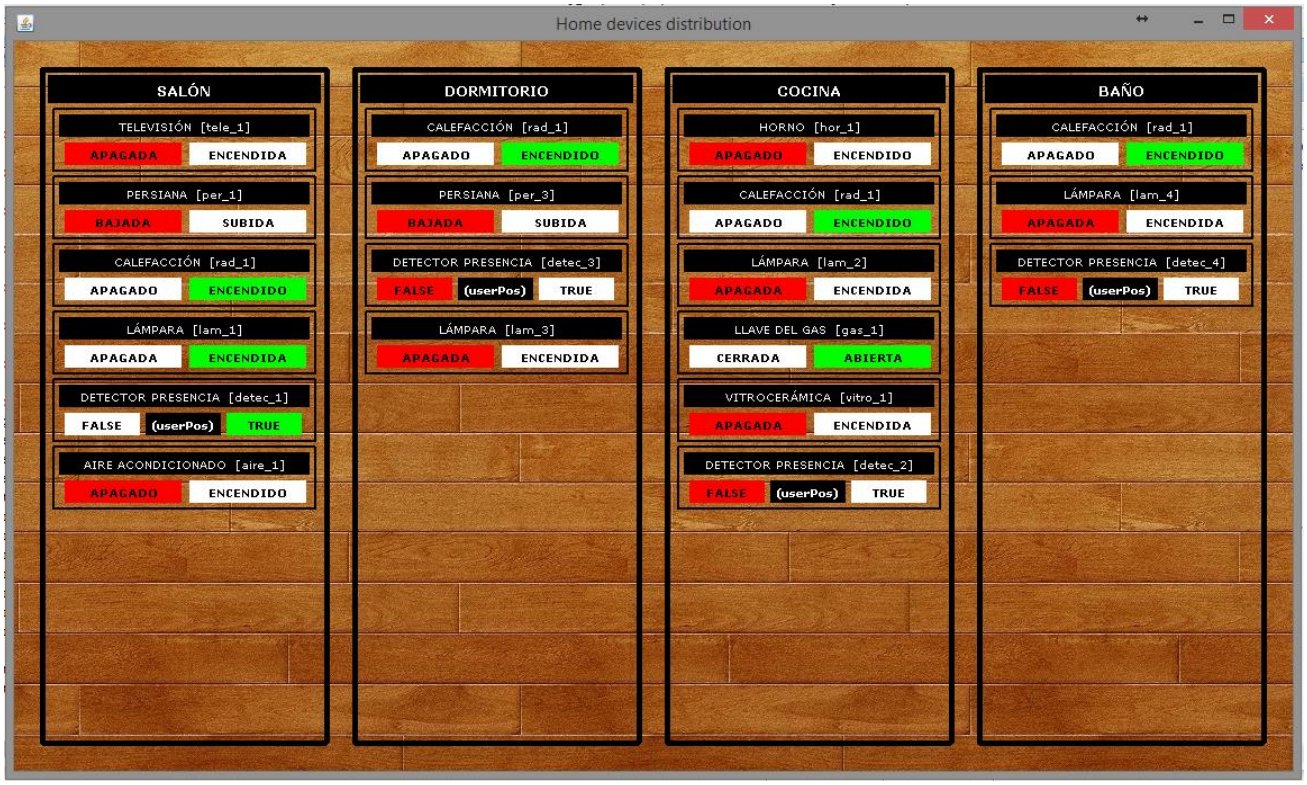


Figura 40. GUI del Simulador

La GUI del *Simulador* se ha diseñado de modo que aporte la máxima flexibilidad posible, siempre enfocado a una representación sencilla y funcional. El usuario, en el proceso de añadido de dispositivos, no necesitará tener en cuenta el *Simulador* de ninguna manera, y creará los dispositivos atendiendo únicamente a la red que quiere representar. Es tarea del sistema interpretar esta información y adaptarla convenientemente a un entorno gráfico interactivo.

Se debe tener en cuenta que el *Simulador* es un módulo completamente independiente y, de hecho, puedo correr en otro equipo diferente al *Controlador Principal* si se considera oportuno (adaptándose bien a equipos con bajos recursos). Es este módulo el encargado de leer y almacenar los dispositivos creados por el usuario.

Por su parte, el *Controlador Principal* no tendrá acceso a la configuración de dispositivos del usuario, luego tendrá que obtener la información directamente del Simulador como si de una red real se tratase.

Se exponen a continuación algunas de las tareas que realiza el Simulador a la hora de cargar y representar los dispositivos:

1. Carga el fichero de configuración de dispositivos y almacena la información referente a cada uno de ellos, organizándola apropiadamente.
2. Realiza un proceso de validación de los datos introducidos por el usuario. Si un dispositivo no cumple todos los requisitos de validez es descartado y se toma como si no hubiese sido añadido (informando al usuario/desarrollador del dispositivo y problema). El resto de dispositivos válidos se añaden y utilizan correctamente.
3. Procesa el número de habitaciones definido y adapta el espacio de la ventana a ello. Siempre que sea posible, mostrará una única fila y varias columnas, pero si hay varias habitaciones, divide el espacio en más filas y columnas.
4. En cada habitación se muestran los dispositivos correspondientes, adaptando a su vez el espacio para representar etiquetas y botones si se sobrepasa el disponible.
5. Proporciona botones por cada dispositivo en función de sus posibles acciones, habilitando solo aquellos que pueden ejecutarse (por representar el estado contrario al actual).

El *Simulador* entiende que si se cierra la GUI se está cerrando todo el sistema y finaliza los procesos convenientemente. A su vez, cada vez que el *Simulador* arranca, carga de nuevo el fichero de configuración, luego el estado vuelve a ser el inicial. Se puede decir que el *Simulador* no tiene memoria (entre distintos arranques) y sólo controla cambios y estados durante el tiempo de ejecución.

4.7.2 Control de estado e intercambio de información Simulador-Controlador Principal

A pesar de que la representación de los dispositivos resulta muy importante tanto para permitir su modificación directa como para controlar que el sistema está funcionando correctamente (aplicando bien las acciones), la función principal del Simulador no es ésta.

El Simulador es una representación de una posible red real de dispositivos, y su función principal es la de controlar sus estados y proporcionar métodos de acceso tanto a la información de estos como a las posibles acciones que pueden realizar. Además, debe ser capaz de propagar los cambios que ocurran internamente para que aquellos sistemas que lo requieran mantengan la información actualizada (como es el caso del *Controlador Principal*).

Tal como se ha adelantado en el punto anterior, el Controlador Principal, al arrancar, interrogará al Simulador sobre los dispositivos que contiene la red y su estado, de manera que pueda cargar las instancias correspondientes en la ontología de dispositivos. Para el intercambio de esta información se utiliza, como ya se definió en el análisis y parte del diseño, UPnP como tecnología de comunicación. En primer lugar se desea representar el servicio expuesto por el *Simulador*, que será explicado a continuación.

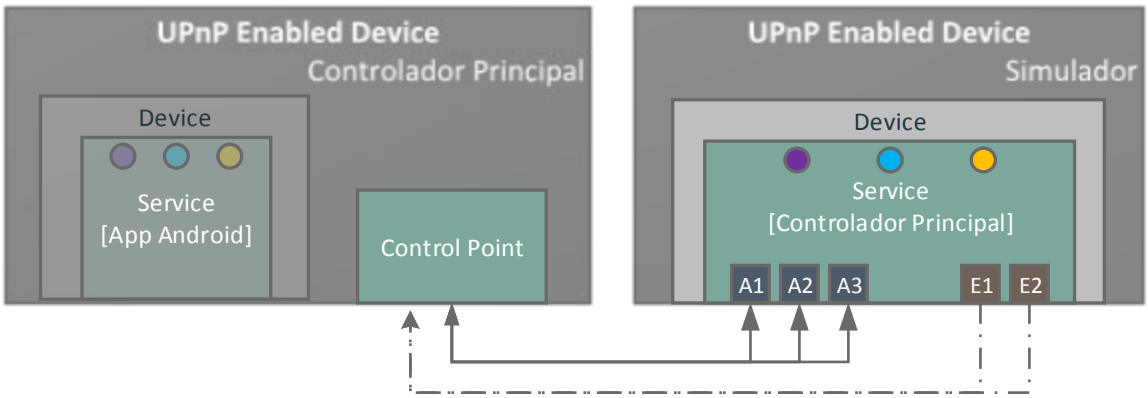


Figura 41. Servicios y eventos UPnP (Simulador-Controlador Principal)

En la Figura 41 se representa el diseño por el que se ha optado a la hora de implementar dispositivos, servicios, acciones y eventos UPnP en el Simulador. Tal como se observa, la distribución es la siguiente:

- Se implementa un único dispositivo UPnP capaz de proporcionar información sobre todos los dispositivos de la red (sobre los supuestos dispositivos simulados[9]). Existen muchos enfoques válidos, pero se ha optado por utilizar un único dispositivo UPnP de manera que resulte indiferente la cantidad de dispositivos reales a manejar. De esta manera, el Controlador Principal siempre sabe de dónde extraer la información de los dispositivos simulados.
- De la misma manera, se implementa un único servicio UPnP conteniendo todas las posibles acciones (así como los eventos que puedan lanzarse).
- Se exponen tres acciones en este servicio:
  - **A1**: devuelve la información completa de toda la red, de manera que el *Controlador Principal* pueda recuperar toda esta información y estructurarla en la ontología de dispositivos.

[9] Para facilitar su distinción, se denominara en este apartado *dispositivo* (o *dispositivo UPnP*) a la implementación de un dispositivo UPnP, mientras que se denominará *dispositivo simulado* a un dispositivo de los que supuestamente se encontrarían en el Hogar Digital.

        - No requiere parámetros de entrada
        - Salida → la información de la red (ordenada por dispositivos)
    - **A2**: ejecuta una acción en un dispositivo (actuador).
        - Entrada → ID de dispositivo + comando de ejecución
        - Salida → un booleano indicando si la acción ha sido ejecutada convenientemente o no.
    - **A3**: recupera el valor leído por un dispositivo (sensor).
        - Entrada → ID de dispositivo + comando de ejecución
        - Salida → ID de dispositivo + resultado de la lectura
- Se lanzan 2 tipos de eventos, ambos relacionados. Cuando se produce un cambio en la red por ejecución interna (si el usuario *clica* un botón de la interfaz para cambiar el estado de un dispositivo), el *Simulador* informa de esta situación mediante la propagación de dos eventos:
    - **E1**: se informa de que un dispositivo ha cambiado de estado (con los datos del cambio).
        - Formato → ID Dispositivo + comando de ejecución
    - **E2**: Se informa de que la red ha cambiado (con la nueva información de la red en conjunto, equivalente a la acción A1).
        - Formato → la información de la red (ordenada por dispositivos).

    El hecho de que se propaguen dos eventos distintos para una misma circunstancia (conteniendo distinta información) responde a una decisión de diseño, mejorando el rendimiento de ciertos procesos en el *Controlador Principal* al no tener que analizar toda la red para deducir el cambio, sino conocerlo tanto en conjunto como por separado.

El formato de estos mensajes se representa en la Figura 42, indicando el contenido de cada uno de ellos. Como se observa, la estructura es reutilizada en varias ocasiones, si bien cómo se tratará la información dependerá de lo descrito en el punto anterior.

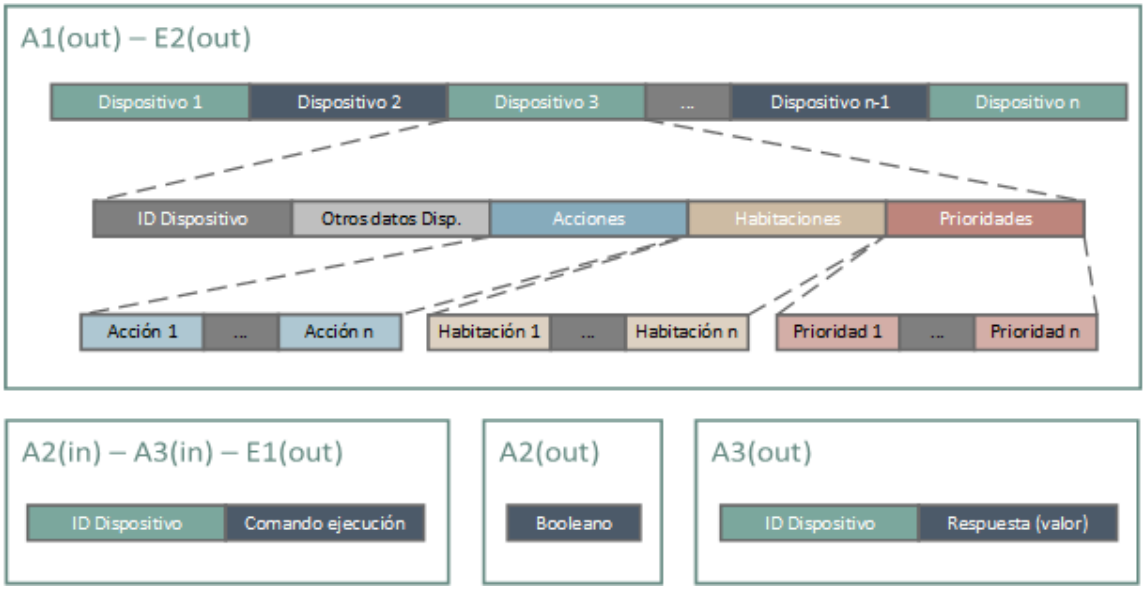


Figura 42. Formato de mensajes UPnP (Simulador-Controlador Principal)

Como ya se ha mencionado en diferentes ocasiones, esta figura refleja que una cantidad reducida de operaciones sobre el *Simulador* permite manejar la red completamente e independizar la complejidad de esta del intercambio de información. De haberse diseñado cada dispositivo real como un dispositivo UPnP independiente, se hubiesen requerido mecanismos adicionales que pudiesen adelantar cierta información de acceso a estos, y la lógica resultante sería más compleja, si bien la funcionalidad final sería similar.

Por último, se considera importante recalcar que la red LonWorks del Hogar Digital podría haberse utilizado de la misma manera, siempre y cuando se hubiese implementado un componente que adaptase la información de la red a este modelo de dispositivo, servicios, acciones y eventos UPnP. En definitiva, con un desarrollo relativamente sencillo, una red de dispositivos reales como la presente en el Hogar Digital hubiese sido válida para su uso mediante el sistema desarrollado.

4.8 Configuración de usuario

A lo largo de todo el documento se ha hecho reiteradas menciones a la configuración de usuario y, como se ha visto, en algunos puntos ha quedado patente que ésta resulta determinante a la hora de definir cómo debe funcionar el sistema.

La configuración de usuario se ha diseñado para albergar muchos y muy variados parámetros. Sin embargo, se ha pretendido mantener una cierta similitud en el modo de introducción de todos ellos con el fin de simplificar el proceso de configuración al usuario.

El objetivo de este punto, al igual que ha sucedido en puntos anteriores, no es el de definir cómo hacer uso de la configuración, lo que se verá en el Anexo I, sino la exposición de los mecanismos que proporcionan la verdadera funcionalidad a este enfoque.

4.8.1 Utilización de hojas de cálculo Excel

Ha quedado patente a lo largo del documento la gran cantidad de tecnologías y mecanismos que se han utilizado en el diseño e implementación de este sistema. La posible creación de una interfaz de configuración exclusiva del sistema se ha considerado innecesaria. Además, los recursos temporales consumidos en el proceso podrían haber afectado al desarrollo de otros componentes del sistema más importantes.

Por ello, se ha decidido utilizar hojas de datos Excel para la entrada de parámetros de configuración, las cuales toman formas tales como las representadas en la Figura 43.

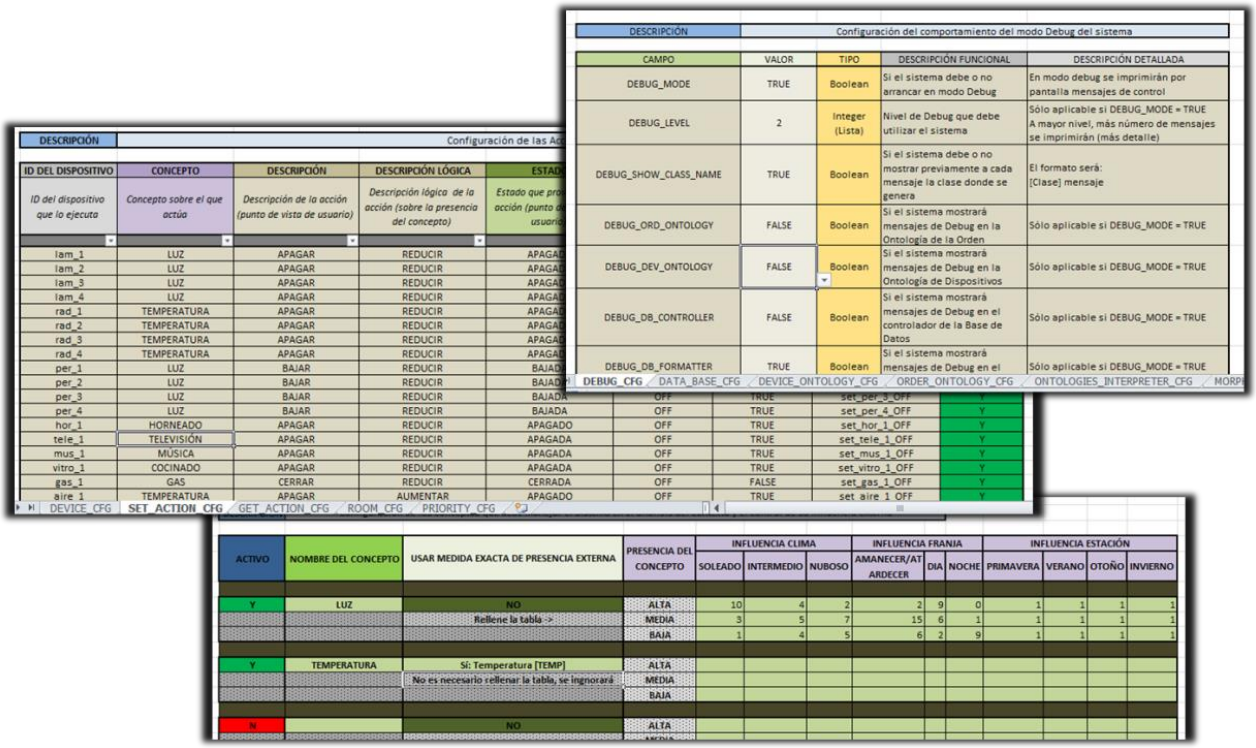


Figura 43. Hojas de configuración

Las hojas de cálculo, también conocidas como hojas de datos proporcionan un modo sencillo de introducir datos de diferentes tipos (como texto y números), aplicar filtros, realizar cálculos sobre ellos e incluso aplicar operaciones lógicas. Además, las hojas de cálculo Excel [43] soportan las denominadas Macros (operaciones programables mediante Visual Basic) que permiten manipular con muchísima flexibilidad todos los datos de este tipo de ficheros. Por estas características, además de por su gran número de herramientas de edición y representación de datos, su uso se ha extendido al de contenedor y organizador de considerables volúmenes de información.

Tal como se observa en la Figura 43, los ficheros de configuración cuentan con varias pestañas, cada una conteniendo el conjunto de parámetros que afectarán a un módulo concreto. Existe un componente, tanto en el *Controlador Principal* como en el Simulador (replicado en ambos sistemas), que se encarga de interpretar estas hojas de datos y cargarlas en el sistema

correspondiente. Para ello, hace uso de la API denominada *Java Excel API* [44], que permite manejar este tipo de ficheros y extraer los datos apropiadamente.

Se listan a continuación los ficheros y su finalidad, en función del sistema que los maneja:

- Simulador
  - Configuración general: parámetros de depuración (activar o desactivar el modo de depuración, así como regular su nivel y la información que mostrará).
  - Configuración de dispositivos: pestañas para añadir cada grupo de información de los dispositivos a simular.
    - Información general del dispositivo
    - Acciones que puede realizar el dispositivo (actuadores)
    - Acciones que puede realizar el dispositivo (sensores)
    - Habitaciones donde ubicar el dispositivo
    - Prioridades a aplicar sobre el dispositivo

    Esta configuración permite activar y desactivar datos o dispositivos completamente para probar rápidamente diferentes configuraciones sin necesidad de reintroducir los datos.
- Controlador Principal
  - Configuración general: parámetros referentes a cada módulo de este sistema:
    - Depuración
    - Base de datos
    - Ontologías
    - Análisis morfológico
    - Motor de búsqueda
  - Configuración de contexto: parámetros referentes a la información que manejará el contexto:
    - Parámetros generales (ubicación de datos, nomenclatura, etc.)
    - Parámetros particulares (información de hábitos del usuario, márgenes de tiempo, ubicación del hogar, etc.)
    - Conceptos a manejar: descripción de los diferentes conceptos que dependen de factores externos (y requerirán redes de contexto independientes)

Además de las ventajas ya mencionadas, existen otras tantas que merece la pena destacar del uso de este tipo de ficheros para la configuración:

- Las hojas pueden bloquearse, de manera que el usuario solo pueda modificar los campos de datos configurables (evitando errores de formato). De esta manera, funciona de manera similar a una interfaz con cajas de texto y desplegables.
- Se pueden definir listas de valores predefinidas (desplegables) en las casillas a rellenar, garantizando que los valores introducidos son compatibles con el resto del sistema.
- Se puede añadir control de datos *in situ*, como el tipo, valor o longitud introducido (p.e. asegurar que el usuario introduce un número no nulo) y configurar mensajes de aviso en caso de que el usuario introduzca mal dicho dato, forzando su reintroducción.
- Permite al usuario, en las pestañas habilitadas a tal efecto, introducir tantos registros como desee sin afectar al modo de lectura ni requerir controles adicionales.

Estas características convierten a las hojas de datos en interfaces eficientes, controlando que los datos introducidos son válidos para el sistema y previniendo posibles defectos por su incorrecta manipulación.

Aunque se entiende que en el ámbito de este proyecto no habrá usuarios inexpertos manipulando la configuración, este diseño responde al requisito deseable de *producto final* (enfoque al usuario) planteado en el apartado 3.1.2.1.

4.8.2 Ampliación de datos de conceptos

Antes de finalizar este apartado de configuración, se considera interesante hacer mención al mecanismo que convierte una reducida cantidad de datos introducidos por el usuario al añadir un concepto en la configuración en la gran cantidad de datos que deberán poblar la tabla de potencias del nodo de influencia externa en la red bayesiana correspondiente (véase apartado 4.5.1).

El usuario, en este proceso de configuración, puede definir conceptos de dos tipos: influencia externa a partir de datos medidos (p.e. la temperatura, que se recupera directamente desde la API climatológica) o influencia externa a partir de los tres nodos precedentes (estado del cielo, estación y franja horaria).

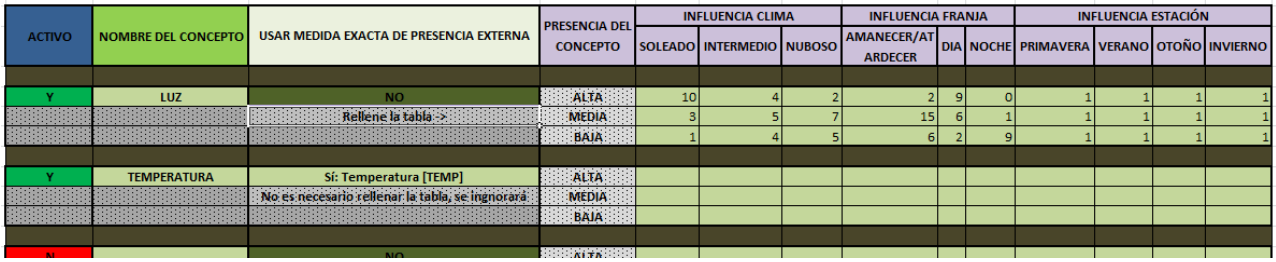
El primero de los casos no requiere de información adicional, pues la API, junto con los mecanismos de conversión de valores absolutos a un conjunto de probabilidades, resulta suficiente.

El segundo caso, no obstante, plantea un reto mayor. Tal como vimos en la descripción de los nodos de la red bayesiana, el nodo de influencia externa, tal como ha sido implementado, cuenta con una tabla de potencias (dependencias probabilísticas) con 108 celdas. Resultaría un diseño cuanto menos discutible solicitar al usuario que rellene toda esta información manualmente. Sin embargo, si se pretende que el sistema pueda añadir cualquier concepto que el usuario requiera para sus dispositivos, deberá existir algún mecanismo que permita obtener esta información.

Para enfocar la explicación de este mecanismo se tomará como referencia la Figura 44, que muestra un fragmento de la pestaña de configuración de conceptos. En ella se muestran dos conceptos configurados: la *temperatura* (que depende de un factor externo que el sistema es capaz de manejar, por lo que es seleccionable desde un desplegable) y la *luz*, que al ser un concepto que

depende de los tres nodos mencionados, requiere de algo más de información mediante el rellenado de una tabla.

Téngase en cuenta que si, por ejemplo, en un futuro desarrollo se implementase la lectura del porcentaje de *iluminación* desde la API, este concepto podría añadirse a las posibilidades que muestra el desplegable y manejarse de la misma manera que lo hace la *temperatura*, sin necesidad de ningún dato adicional.



ACTIVO	NOMBRE DEL CONCEPTO	USAR MEDIDA EXACTA DE PRESENCIA EXTERNA	PRESENCIA DEL CONCEPTO	INFLUENCIA CLIMA			INFLUENCIA FRANJA			INFLUENCIA ESTACIÓN			
				SOLEADO	INTERMEDIO	NUBOSO	AMANECER/ATARDECER	DIA	NOCHE	PRIMAVERA	VERANO	OTOÑO	INVIERNO
Y	LUZ	NO	ALTA	10	4	2	2	9	0	1	1	1	1
		Rellene la tabla ->	MEDIA	3	5	7	15	6	1	1	1	1	1
			BAJA	1	4	5	6	2	9	1	1	1	1
Y	TEMPERATURA	Sí: Temperatura [TEMP]	ALTA										
		No es necesario rellenar la tabla, se ignorará	MEDIA										
			BAJA										
N		NO	ALTA										

Figura 44. Configuración de conceptos

Como se observa en la figura, la cantidad de celdas de esta tabla es considerablemente menor a la necesaria en el nodo. En esta tabla se le pide al usuario que únicamente puntúe de manera individual la influencia que considera que tiene una situación concreta sobre este concepto (considerándose la influencia como la capacidad de esta situación para aumentar la presencia de dicho concepto).

Estas celdas, que deben contener valores enteros no negativos, pueden ser rellenadas de manera intuitiva, de modo que su puntuación sea mayor cuanto más probable crea el usuario que es la influencia. Además, no existe número máximo, de manera que si el usuario, según rellena la tabla, se encuentra de repente un valor que considera más alto que los anteriores, no tiene que volver atrás para decrementar los valores que ya fuesen máximos.

Para ilustrar este enfoque, considérense los datos de la Figura 44. En ella, se ha indicado que, de manera intuitiva, la relación entre *día soleado* y *alta presencia de luz* tiene una puntuación de 10, reduciendo los valores asignados de presencia *media* y *baja* (y así sucesivamente). Sin embargo, se ha considerado que la presencia o ausencia de luz depende poco de la estación (pues un día soleado en invierno puede ser más luminoso que un día nuboso en verano).

Lo que hace el sistema al cargar esta configuración es multiplicar las puntuaciones que el usuario ha asignado a las diferentes celdas (en referencia a la combinación que requiere cada celda de la tabla de potencias, tal como se observa en el la Tabla 9).

Una vez que posee las tres combinaciones de una columna en cuestión, el sistema las normaliza y las carga en el nodo, resultando un reparto apropiado (suma de probabilidad igual a 1) y con una proporción entre posibilidades que reflejan con bastante precisión los valores que requería la tabla.

Tabla 9. Tabla de potencias (nodo de presencia externa)

Station	spring									
Time_Slot	sunrise/sunset			day			night			
Sky	sunny	intermediate	cloudy	sunny	intermediate	cloudy	sunny	intermediate	cloudy	su
high	0,2	0,15	0,05	0,85	0,55	0,3	0,05	0	0	0,2
medium	0,5	0,3	0,2	0,1	0,3	0,4	0,1	0,1	0	0,5
low	0,3	0,55	0,75	0,05	0,15	0,3	0,85	0,9	1	0,3

De esta manera tan sencilla (en la línea de la simplificación que se ha seguido a lo largo de todo el desarrollo) y a pesar de que el usuario introduzca los valores en la configuración por mera intuición, se ha comprobado que el la salida resulta válida para un buen funcionamiento de la red de contexto.

Este mecanismo de configuración añade gran flexibilidad al tipo de dispositivos con influencia externa que puede manejar el sistema (cualquiera que desee añadir el usuario). Téngase en cuenta, no obstante, que aquellos dispositivos cuyo concepto no corresponda a ninguno de los conceptos definidos en esta hoja de configuración (p.e. *música* en una minicadena), se tomarán como dispositivos cuyo contexto no depende de factores externos a la casa y harán uso de la red bayesiana diseñada para ello (aquella sin nodo de presencia externa que se presentó en el apartado 4.5.3).

4.9 Aplicación Android (GUI)

En el apartado 4.7 sobre el *Simulador*, ya se mencionó el diseño de una Interfaz Gráfica de Usuario o GUI, pero esta surgía únicamente de la necesidad de representar los datos de los dispositivos simulados. Si el sistema se hubiese trasladado a un entorno real, esa GUI no estaría presente en el sistema. Desde el punto de vista de este proyecto, la GUI realmente importante, y la cual sería en todo caso utilizada por el usuario corresponde a la aplicación Android.

De nuevo, no se pretende describir cómo el usuario debe hacer uso de la aplicación (de ahora en adelante *app* o *GUI*), sino hacer mención a las diferentes decisiones de diseño, los mecanismos implementados y el modelo de comunicación utilizado.

4.9.1 Características de la aplicación Android

Continuando con la línea de simplicidad que se ha pretendido utilizar en cada uno de los módulos del sistema, la app se ha diseñado concentrando toda la funcionalidad en una cantidad reducida de acciones sobre ella.

No obstante, la app es la cara visible del sistema para el usuario. Recuérdese la especificación deseable del apartado 3.1.2.1 denominada *producto final*, en la que se establecía que se procuraría diseñar el sistema enfocándolo a un producto que, en última instancia, sería utilizado por usuarios ajenos al sistema. Siguiendo esta corriente, la *app*, como interfaz del sistema y herramienta de interacción con el usuario, debe poseer un diseño cuidado.

La Figura 45 muestra una serie de capturas de la aplicación con su apariencia final. Aunque la imagen no contiene todas las ventanas de la aplicación, representa en gran medida el modelo tomado como referencia en su implementación.

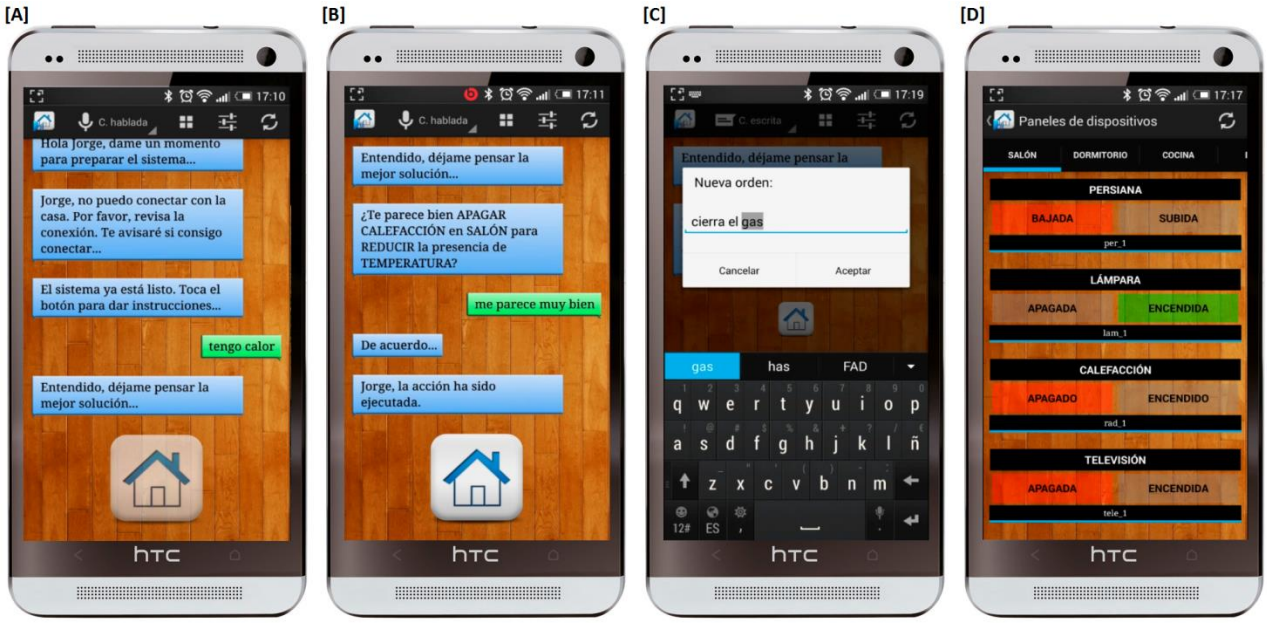


Figura 45. Aplicación Android (GUI)

Probablemente, lo primero que salte a la vista de este diseño es el modelo de comunicación con la casa implementado (tipo chat). Efectivamente, la aplicación, como puerta de acceso al sistema por parte del usuario, tiene como finalidad crear la ilusión de inteligencia. El usuario no debe preocuparse de cómo interactuar con una máquina, sino que puede expresar directamente sus sensaciones u órdenes, como si hubiese una persona al otro lado de este chat.

En las capturas *[A]* y *[B]* se observa este modelo de comunicación *usuario-sistema*. Al abrir la aplicación, se alerta al usuario de que aún tiene que establecerse la comunicación con el hogar y que debe esperar un instante. En el ejemplo *[A]* la app se abrió antes de levantar el resto del sistema, luego esta se quedó esperando hasta que finalmente pudo conectar.

Una vez establecida la comunicación, el usuario solo tiene que *clicar* el botón de la casa para intervenir en esta conversación. Este proceso puede realizarse tanto de manera vocal como escrita (lo cual es regulado desde la barra de acción). Esta opción escrita se puede observar en la captura *[C]*. Aunque el proceso se realice de manera vocal (y el sistema también hable en sus respuestas mediante un motor *TextToSpeech*), toda la conversación queda reflejada también de manera escrita. Esto permite al usuario revisar la conversación que se ha tenido hasta el momento y, además, comprobar que la orden pronunciada ha sido correctamente procesada y convertida a texto por la aplicación, esta vez mediante un motor *SpeechToText*.

La *app* hará llegar la orden en cuestión al *Controlador Principal*, que la procesará tal como se ha visto a lo largo de este documento, obteniendo la conclusión que será devuelta. Durante este proceso, el botón parpadeará y permanecerá inactivo indicando al usuario que el sistema aún está procesando la orden y debe esperar un poco antes de intervenir de nuevo.

Una vez se recibe una conclusión, el usuario podrá confirmar o negar dicha acción (con sus propias palabras). Finalmente, haciendo llegar esta confirmación al *Controlador Principal*, la acción es propagada hasta el *Simulador* y ejecutada. A su vez, una vez ejecutada, el usuario es

avisado de nuevo, teniendo la certeza en ese punto de que todo el proceso se ha completado correctamente.

Sin embargo, tal como se habrá podido deducir a estas alturas del documento, con los mecanismos y enfoques utilizados ya expuestos, el sistema no tomará la decisión correcta en todas las situaciones. El diseño del algoritmo es clave en este aspecto (variando completamente este índice de aciertos) y, junto con el resto de limitaciones, se dará el caso de que el sistema no sepa interpretar correctamente algunas órdenes. En estos casos cabe la posibilidad de dar la orden de nuevo, cambiando la manera de expresarla, quizás de una manera más explícita (lo que en muchas ocasiones resulta suficiente).

No obstante, en el empeño por proporcionar al usuario una herramienta completa y que permita en todos los casos la manipulación apropiada de los dispositivos del hogar, se ha desarrollado también dentro de la aplicación lo que se ha venido a denominar el *panel de dispositivos* (siguiendo la nomenclatura que se ha utilizado a lo largo del proyecto), representado en la captura *[D]*.

El *panel de dispositivos* es una versión reducida de los paneles que contiene el *Simulador* o de los que podrían haberse utilizado en la red LonWorks del *Hogar Digital*. Este panel muestra los actuadores organizados por habitaciones, junto con el estado actual de éstos. El usuario puede *clicar* un estado actualmente inactivo para ejecutar una orden directa sobre el sistema, sin necesidad de interpretación.

La orden es propagada hasta el simulador y ejecutada, y una vez este paso es completado, se confirma en dirección contraria para que finalmente se refleje en la *app*. Una vez el estado cambia en la *app*, el usuario puede tener la certeza de que el éste ha cambiado previamente en el *Simulador* de manera correcta.

La aplicación permite configurar ciertos comportamientos como la necesidad o no de confirmar la ejecución de una acción, el nombre del usuario al que referirse y el modo de inicio de la *app* (vocal o escrito). Además, se proporciona al usuario control adicional sobre el estado de la conexión entre la app y el Controlador Principal, pero todo ello será reflejado en más detalle en el Anexo I.

4.9.2 Control de estado e intercambio de información App-Controlador Principal

La aplicación Android con la que interactúa el usuario contiene únicamente la GUI y el mecanismo de comunicación oportuno. La *app* se limita a adaptar la información para el usuario y a controlar la concurrencia de los diferentes eventos que debe manejar (cambios en la interfaz, control de conexión, procesado de eventos UPnP, control de eventos del sistema Android, etc.). Además, la app debe controlar los errores del sistema con mayor exigencia a la impuesta en otros componentes, procurando recuperarse automáticamente y reconectar sin la intervención del usuario, así como informar a éste de dichos errores.

No obstante, todo el peso del procesado de información necesario para la toma de decisiones que se ha expuesto en los distintos apartados de este documento recae en el *Controlador Principal*.

Dado que este reparto de carga de trabajo debe ser transparente para el usuario, debe existir un mecanismo de comunicación entre estos elementos que permita el intercambio de información. Tal como se expuso en el análisis del sistema, la tecnología utilizada es, de nuevo, UPnP.

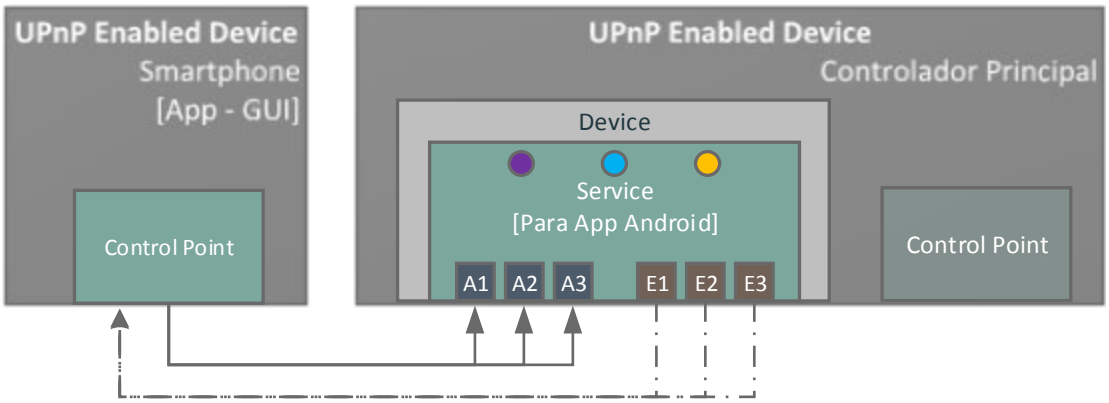


Figura 46. Servicios y eventos UPnP (App-Controlador Principal)

La Figura 46 muestra el otro lado de la comunicación que se había quedado sin exponer en la Figura 41. Como se observa, la similitud entre ambas es muy alta, si bien los servicios y eventos no son los mismos:

- Se implementa un único dispositivo UPnP, siguiendo la misma lógica planteada en el simulador (véase apartado 4.7.2).
- Se expone un único servicio, conteniendo todas las acciones y eventos. El *Controlador Principal* contiene la información completa de la red actualizada, incluida estados, organizada en su ontología de dispositivos.
- Existen tres acciones disponibles:
  - **A1**: procesar una orden, la acción principal del sistema. Cuando el usuario interviene en la conversación, en la *app* esta es entendida como una orden y se envía al *Controlador Principal*. Para identificarla correctamente en los distintos pasos de la comunicación, la aplicación genera un identificador aleatorio único (de ahora en adelante *randomID*).
    - Entrada → randomID + orden
    - No hay salida
  - **A2**: confirmar/cancelar la acción que el sistema ha propuesto ejecutar a raíz de la orden.
    - Entrada → randomID + flag confirmación/negación (Y/N)
    - No hay salida
  - **A3**: ejecutar acción directa. Este tipo de acciones hacen referencia a cuando el usuario *clica* un botón del panel de dispositivos y no se requiere interpretación, sino simplemente la ejecución.

- Entrada → ID dispositivo + Comando de ejecución
- No hay salida

- Existen tres eventos:
  - **E1**: informa de que se ha llegado a una conclusión a partir de una orden previa.
    - Formato → randomID + conclusión
  - **E2**: informa del estado de una acción que ha sido previamente confirmada/cancelada.
    - Formato → randomID + flag confirmación/cancelación (Y/N)
  - **E3**: informa de un nuevo estado de los dispositivos de la red, conteniendo toda la información necesaria para su representación en el *panel de dispositivos* de la *app*. Esta información es una adaptación (organizando por habitaciones los dispositivos y eliminando cualquier información que no sea requerida en la aplicación) de la información recibida por el *Controlador Principal* desde el *Simulador*, con el fin de facilitar y agilizar el trabajo de la app a la hora de representarla.
    - Formato → información de dispositivos simplificada (por habitaciones)

En el diseño de este modelo de comunicación se ha tenido muy presente que los procesos a llevar a cabo en el *Controlador Principal* pueden requerir de una cantidad considerable de tiempo. Es por ello que se ha decidido no tratar valores de salida en ninguna acción, procurando un funcionamiento totalmente asíncrono entre la *app* y el *Controlador Principal*.

De esta manera, las órdenes y confirmaciones se informan en la propia acción junto con un identificador único, y se continúa posteriormente con otras tareas que puedan ser requeridas por el usuario o el sistema, manteniendo a su vez el control sobre los eventos que puedan llegar. Cuando se reciba dicho evento, la *app* podrá identificar si se trata de una orden o confirmación pendiente a partir del identificador y continuar con el proceso que corresponda.

Es importante resaltar que la *app* no hará control alguno sobre el estado de los dispositivos y se limitará a mostrar al usuario los estados que vaya recibiendo desde el *Controlador Principal*. La *app* no procesa la información más allá de lo requerido para mostrarla en el *panel de dispositivos* e invocar, cuando se requiera, una acción a partir de la pulsación de algún botón.

El modo de comunicación es el siguiente:

- Acciones directas:
  1. Se invoca la acción cuando el usuario *clica* el botón, identificando el dispositivo y comando único asociado a dicho botón (acción). No se hace más control de esta acción.
  2. Cada vez que la red de dispositivos del *Simulador* sufre un cambio, se recibirá un evento con la nueva información y se actualizará en el panel de

dispositivos. Será suficiente para el usuario observar como el estado de la red cambia para saber que la acción ha sido ejecutada, sin necesidad de mecanismos adicionales.

- Órdenes a procesar:
    1. Una vez capturada la orden, ésta se le hace llegar al *Controlador Principal* junto con un identificador único y se continúa con otras tareas, si bien se seguirá pendiente de la recepción de eventos.
    2. Se recibe un evento con la conclusión (identificada con el ID asignado a la orden previamente), y la aplicación la muestra al usuario.
    3. Tanto si se confirma como si se cancela, se le hace llegar al *Controlador Principal* el flag correspondiente, de nuevo con el identificador único de la acción adjunto.
    4. El *Controlador Principal* propaga la ejecución al *Simulador* o la borra de la lista de órdenes pendientes, según corresponda. Una vez este paso esté completo, lanza un evento confirmando el estado de la acción (identificador y flag).
    5. La aplicación informa al usuario de si la acción se ha ejecutado o cancelado correctamente y se da por terminado el procesamiento de la orden. Se espera a que el usuario de una nueva orden (en tal caso se vuelve al punto 1).

La Figura 47 representa la información intercambiada en función de la acción o evento correspondiente. En ella se observa cómo la información de la red se ha reorganizado y la información irrelevante para la aplicación (como las *prioridades*) no se tiene en cuenta.

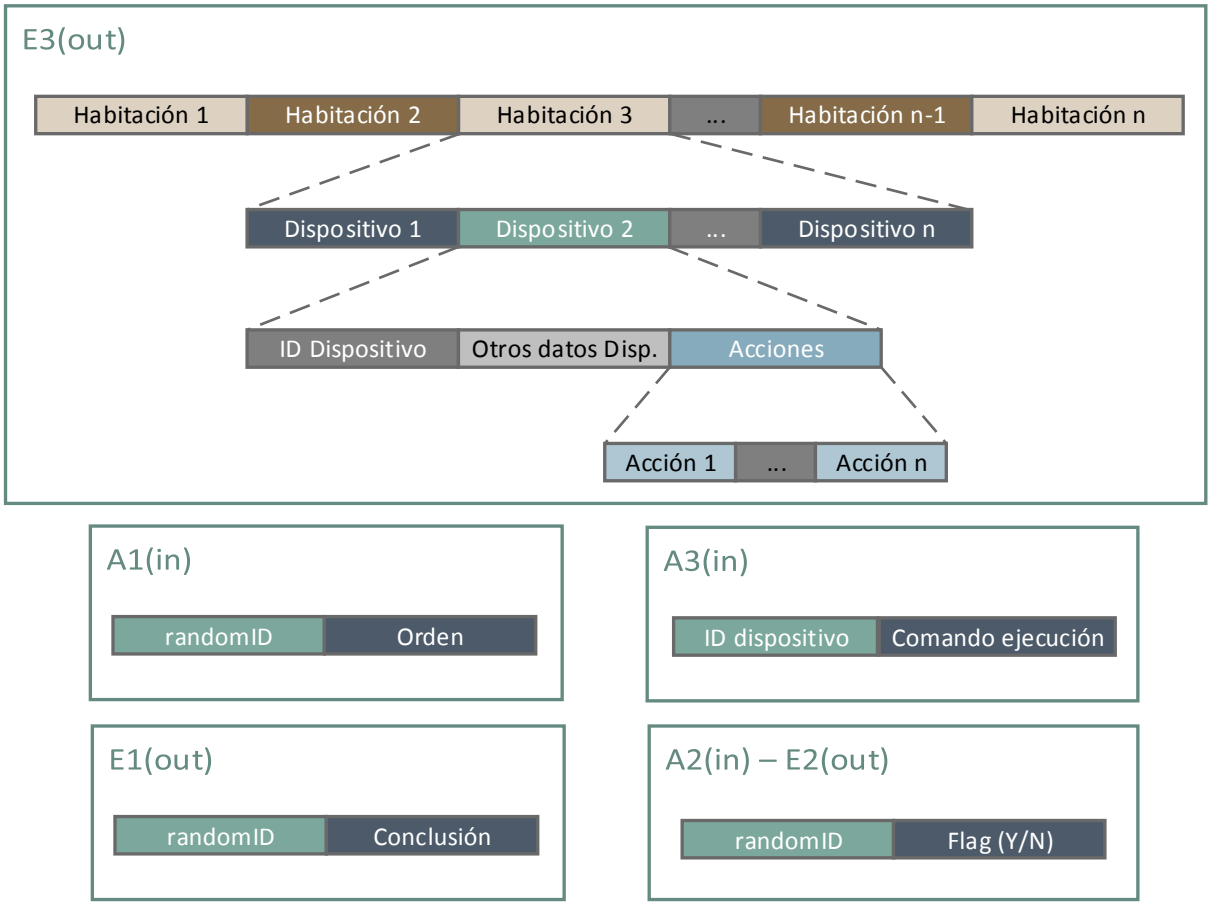


Figura 47. Formato de mensajes UPnP (App-Controlador Principal)

4.10 Sistema en conjunto

En este punto del documento, todos los mecanismos y tecnologías que se han considerado esenciales para el proyecto y, por tanto, dignas de analizar en profundidad, ya han sido expuestas. Esta exposición se ha enfocado desde un punto vista funcional y de alto nivel, procurando no entrar en demasiado detalle salvo en aquellos casos en los que se ha considerado importante para la comprensión del conjunto.

Resulta evidente que por debajo de cada uno de estos componentes hay gran cantidad de lógicas y código que los soportan y hacen posible la ejecución de los diferentes procesos, lo que requiere de una buena organización de componentes a bajo nivel. Aunque se pretende mantener el enfoque del documento y no dar más detalles de los necesarios, se considera interesante proporcionar una breve visión del modelo de subsistemas que sustenta el conjunto del sistema.

El diagrama de componentes que se muestra en la Figura 48 no solo servirá para dar esta visión de conjunto, sino que permitirá reflejar una característica que se la ha pretendido proporcionar al sistema durante todo el desarrollo: un diseño modular.

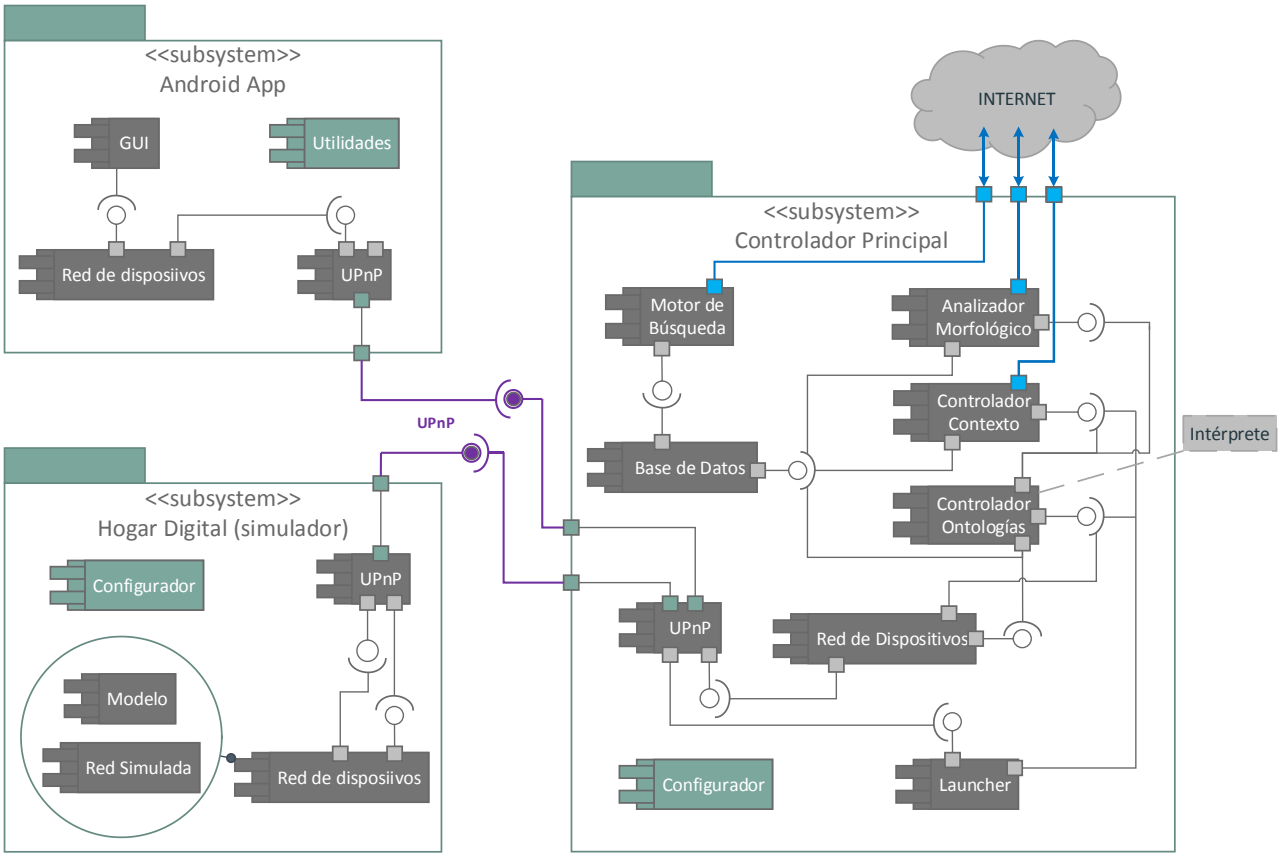


Figura 48. Diagrama de componentes

Pretendiendo mantener un nivel complejidad bajo en la figura para mejorar su comprensión, ésta no refleja las relaciones entre los componentes *Configurador* (*Simulador* y *Controlador Principal*) y *Utilidades* (*App*) con el resto de los componentes del subsistema correspondiente. Estos componentes son accedidos desde el resto de componentes de su subsistema y su función es la siguiente:

- *Configurador*: cargar y contener los parámetros de configuración que afectan a todo el subsistema. Son accesibles a nivel global dentro de dicho subsistema.
- *Utilidades*: contenedor global de parámetros de configuración y datos compartidos entre diferentes hilos de la aplicación. Proporciona mecanismos de acceso concurrente.

Sin intención de entrar en profundidad en cada uno de los componentes, continuación se listan los puntos más importantes en este diseño representado en la Figura 48:

- Todos y cada uno de los componentes del sistema son accedidos mediante una interfaz determinada. A su vez, estos hacen uso de otros componentes a través de la interfaz correspondiente.
  - Los únicos componentes del sistema con información accesible sin el uso de interfaces son los componentes *Configurador* y *Utilidades* mencionados previamente, por su carácter global y por servir principalmente de contenedores de información compartida que afectan al conjunto del sistema.

- En el *Controlador Principal*, el componente denominado *Launcher* es el orquestador de todo el proceso de decisión, controlando la invocación paulatina de los diferentes pasos requeridos.
- Sólo un conjunto determinado de componentes acceden a Internet, cada uno con diferentes objetivos:
  - Análisis morfológico (RAE)
    - Sólo accederá si los datos no están almacenados en la Base de Datos
  - Información climatológica (API climatológica)
    - Sólo accederá si los datos no están almacenados o han caducado.
  - Consultas al motor de búsqueda
- Todos los accesos al *Motor de búsqueda* se hacen a través del componente de la *Base de Datos*, de manera que resulta transparente para el resto del sistema si los datos ya estaban o no presentes y se evitan accesos innecesarios que pudiesen ralentizar el sistema.
- Todos los subsistemas realizan sus correspondientes gestiones contra el componente *Red de dispositivos*. Para los subsistemas resulta transparente dónde se encuentra la información que quieren compartir u obtener del resto y cómo acceder a ella.

  Son estos componentes de nivel inferior los encargados de recuperar la información oportuna: *Red de dispositivos* y *Upnp*.
- Cada componente concentra un conjunto de tareas enfocadas a un único tipo de gestión, individualizando cada paso requerido para el procesamiento; con la única excepción del intérprete, que por una cuestión de diseño se ha decidido ubicar junto con el resto de controles sobre las ontologías, si bien hace uso también de la Base de Datos y del contexto.

En definitiva, se ha procurado implementar cada componente de una manera totalmente independiente y dividir bien las diferentes operaciones. Este enfoque permitirá, en un supuesto trabajo futuro (tal como se expondrá en el apartado 5.2), mejorar los diferentes componentes por separado.

Se considera, en definitiva, que este diseño permitirá al sistema crecer por fases (enfocando el esfuerzo en la sustitución de componentes individuales) y, por tanto, aumenta en gran medida su capacidad de mejora y crecimiento.

4.11 Resumen de ejecución

La gran cantidad de tecnologías, mecanismos y lógicas aplicados al sistema, junto con la estrecha relación entre ellos, pueden dificultar la comprensión de cada uno de los elementos expuestos en los apartados anteriores y el papel que juegan en el sistema.

La comprensión del modo de funcionamiento y utilidad de cada uno de los componentes que se han ido exponiendo depende, en la mayoría de los casos, de la comprensión de otros

componentes que estaban por exponer. Por ello, y no existiendo componentes sin influencia de otros, se ha pretendido ordenar esta explicación de la manera más clara posible, intentando recalcar aquellos aspectos más importantes.

Aunque se espera que la base del sistema en su conjunto haya quedado bien definida, se considera interesante presentar una breve descripción, punto por punto, de los pasos que ejecuta el sistema y qué papel juega los elementos en cada uno de ellos, dando un último repaso al diseño e implementación llevado a cabo.

4.11.1 Secuencia de ejecución

Para la exposición de la secuencia se supondrá el arranque del sistema y la ejecución de una orden. Aunque existen otras secuencias de ejecución, al ser ésta la más completa se podrán extrapolar el resto por similitud.

Se intentará representar todos los pasos significativos, aunque por supuesto habrá otros pasos menores entre medias de algunos de ellos. Por su parte, los indicados serán brevemente descritos con el fin de minimizar el volumen de esta secuencia. Por otro lado, se muestran los procesos de manera secuencial por simplicidad, si bien algunos de ellos se ejecutan en paralelo.

1. Se arranca el *Simulador*.
    a. Creación e inicialización de todos sus componentes.
    b. El *Configurador* carga la configuración de usuario, incluyendo los dispositivos a simular (lectura de ficheros Excel)
    c. Se crea y expone el servicio UPnP.
    d. Se procesa la información de dispositivos y se muestra la interfaz gráfica (panel de dispositivos mediante Java 2D).
2. Se arranca el *Controlador Principal*.
    a. Creación e inicialización de todos sus componentes.
    b. El *Configurador* carga la configuración de usuario (lectura de ficheros Excel), incluyendo la información de contexto y todos los parámetros del subsistema (entre los que se encuentra el algoritmo de interpretación).
    c. Se crea el punto de control UPnP y el servicio UPnP, que se expone inmediatamente.
        i. El punto de control busca dispositivos UPnP y obtiene la información de acceso al *Simulador* y su servicio UPnP.
            1. Se obtiene la información de la red de dispositivos desde el *Simulador* mediante la acción UPnP correspondiente (se obvia, por simplicidad, el procesamiento en el lado del Simulador, que no va más allá de adaptar la información y devolverla).
    d. Se estructura la información de la red en la ontología de dispositivos, cuya estructura fue definida mediante el software Protégé-Frames.

e. Se obtiene la información climatológica desde la API.

i. Se almacena en la *Base de datos*

f. Se definen, a partir de las redes de dispositivos, las redes bayesianas necesarias, replicando y modificando las redes definidas desde UnBBayes.

i. Se inicializan los nodos de las redes contextuales (bayesianas) correspondientes a datos climatológicos, temporales y de configuración; es decir, aquellos nodos que no dependen de los dispositivos.

3. Se arranca la aplicación Android.

a. Creación e inicialización de todos sus componentes.

b. Se carga la configuración de usuario (con los mecanismos propios de Android).

c. Se muestra la interfaz en el modo apropiado (según configuración), pidiendo al usuario de que espere hasta que el sistema esté listo.

d. Se crea el punto de control UPnP.

i. Se busca el dispositivo UPnP del *Controlador Principal* y se obtiene la información de acceso a su servicio UPnP.

1. Se recibe la información adaptada de la red, que puede ser mostrada en el panel de dispositivos si el usuario lo requiere.

e. Se avisa al usuario de que el sistema está listo y espera a que este de una orden.

4. El usuario *clica* el botón principal de la aplicación y da una orden, como por ejemplo "tengo calor" (siguiendo el ejemplo de la Figura 45).

a. La *app* captura la orden e invoca la acción UPnP correspondiente en el *Controlador Principal* para procesarla. Esta orden va acompañada de un identificador único generado en este instante.

b. El botón de nueva orden se bloquea y parpadea, indicando al usuario que debe esperar (aunque puede seguir navegando por el resto de la aplicación, incluido el panel de dispositivos).

5. El *Controlador Principal* recibe la orden, la cual se conduce hasta el *Launcher*, que irá invocando los siguientes pasos:

a. Fragmentación y estructuración de la orden en la ontología de órdenes.

i. Se consulta a la RAE para la clasificación morfológica (si la BD no contenía previamente la información) y se almacena toda nueva información para futuras consultas

ii. Implica consultas reiteradas al motor de búsqueda para el análisis semántico, las cuales se irán almacenando también en la base de datos para futuros usos.

b. Se lanza el intérprete, que llevará a cabo, a su vez, los siguientes pasos:

i. Lee el algoritmo e interpreta la estructura de tablas que necesitará.

1. Se crean en la base de datos estas tablas.

ii. Se recorre cada bloque del algoritmo y se procesa:

1. Se obtiene la información de los dispositivos y órdenes correspondientes, creando todas las combinaciones necesarias para cada columna.

2. Se rellenan progresivamente las tablas de análisis, duplicando los registros que sean necesarios.

a. Una vez completado el proceso, se tienen todos los cruces de información que se deben consultar en el buscador.

iii. Se recorre registro a registro las tablas de análisis definidas, lanzando una consulta en el buscador por cada *query*.

1. El resultado devuelto es procesado, aplicando aquellas opciones de dimensionamiento definidas en el algoritmo.

a. El valor resultante se guarda en la columna *result* correspondiente.

2. Una vez todas las columnas *result* están cargadas, se lee la información contextual del registro y se carga en la red bayesiana que corresponda.

a. Se ejecuta el algoritmo *Junction Tree* en la red, leyendo el valor de salida que se adapta a la acción que representa este registro (aumentar o reducir la presencia del concepto del dispositivo).

3. Se multiplica la suma de resultados de las columnas *result* con el valor de salida de la red contextual, cargando el resultado final en la columna *total*.

iv. Se unifican los registros de las distintas *tablas de análisis* en la *tabla de conclusiones,* en función del algoritmo.

v. Se extrae la conclusión provisional, que consta de la acción conceptual, el concepto y la habitación. Supóngase que esta conclusión provisional es *reducir* la *temperatura* del *salón*.

vi. Se lee de la primera *tabla de análisis* (*tabla principal*) los dispositivos capaces de llevar a cabo esta acción. Supóngase que se cuenta con un

radiador (que podría apagarse) y un aire acondicionado (que podría encenderse) para satisfacer la necesidad.

1. Se recupera la puntuación que tenían estos dispositivos en esta primera tabla.

vii. Se aplican las prioridades, si estos dispositivos tuviesen alguna definida.

1. Se recupera, desde la red bayesiana apropiada, el valor de referencia del nodo en cuestión y se aplica la formula correspondiente. Esta salida se multiplica por la puntuación inicial, dando lugar a la puntuación final.

a. Si hay varias prioridades para el mismo dispositivo, se aplican una tras otra.

b. Una vez acabado este punto, se tienen las puntuaciones finales del proceso de interpretación y toma de decisiones.

viii. Se selecciona el dispositivo y acción con mayor puntuación final.

1. Se recupera la acción entendible por el usuario, es decir, acciones del tipo apagar (un dispositivo), en lugar de reducir (la presencia de un concepto). Supóngase que se ha decidido *reducir* la *temperatura* del *salón* mediante la acción *apagar* la *calefacción* del *salón*.

2. Se almacena el comando que representa la acción, por si es requerido más adelante (tras la confirmación del usuario).

c. Se propaga la conclusión, acompañada del ID de la orden, hasta la *app* mediante un evento UPnP.

6. La aplicación recibe la conclusión, la cual identifica mediante el identificador adjunto.

a. Se le muestra al usuario y se le permite intervenir de nuevo.

i. Aquello que diga el usuario se interpretará como una confirmación o una negación (en función de si contiene o no ciertas palabras clave).

b. Tanto si el usuario confirma como si cancela la acción propuesta, ésta se envía de nuevo hacia el *Controlador Principal* con el identificador de la orden original adjunto.

i. Se ejecuta la acción correspondiente en el *Controlador Principal*.

7. El *Controlador Principal*, mediante la acción invocada, procesa la confirmación/cancelación de la orden, la cual identifica, de nuevo, por el ID aleatorio generado para la orden original.

a. Si se trata de una cancelación, éste se limita a eliminar la información que manejase al respecto.

    i. Envía el estado de esta acción mediante un nuevo evento, confirmando que se ha tratado tal como el usuario espera (lo que sería indicado al usuario por la aplicación). No se desarrolla esta opción por simplicidad.

b. Si se trata de una confirmación, se recupera el comando de ejecución previamente almacenado.

    i. Se envía el identificador del dispositivo junto con el comando al *Simulador*.

        1. Se ejecuta la acción UPnP correspondiente.

8. El *Simulador* ejecuta la acción identificada mediante el dispositivo y comando.

    a. Devuelve el estado de la ejecución en la misma operación.

    b. Al cambiar el estado de la red, el *Simulador* propaga un evento UPnP con la nueva información de la red.

9. El *Controlador Principal* sabe casi inmediatamente el estado de la ejecución de la orden (pues es parámetro de salida de la acción UPnP).

    a. Se envía un evento UPnP con el estado de la ejecución (más el identificador de la orden), que será recibido por la *app*.

    b. Como consecuencia del cambio de estado de la red del Simulador, el Controlador Principal recibe también el evento UPnP correspondiente.

        i. El Controlador principal adapta la información y la simplifica, generando un nuevo evento UPnP de cambio de red para la Aplicación.

10. La *app* recibe dos eventos UPnP (no necesariamente en el mismo instante).

    a. Uno de ellos indica si la acción se ha ejecutado o no.

        i. Se informa al usuario del estado de la orden (si se ha ejecutado o si ha habido algún problema).

        ii. Se da por finalizado el procesamiento de la orden.

        iii. Se habilita el botón principal (que deja de parpadear) y se esperan nuevas órdenes.

    b. El otro evento refleja el cambio en la red. Este evento se trata igual en los casos en los que se aplica una acción directamente sobre el simulador (desde la interfaz gráfica del simulador), razón por la que no se unifica con el evento anteriormente mencionado.

        i. La *app* procesa la información para ser mostrada convenientemente la próxima vez que el usuario entre en el *panel de dispositivos* (o refresca la ventana actual si se encuentra actualmente abierto).

11. Se da por concluido la secuencia de ejecución correspondiente al procesamiento, confirmación y ejecución de una orden.

Con esta descripción de la secuencia se espera que el papel de los distintos elementos del sistema haya quedado completamente reflejado. Como se ha expuesto, el intercambio de información de otras posibles secuencias se puede extrapolar fácilmente, pues el planteamiento es muy similar entre todas ellas. Además, el proceso descrito es el más complejo y todas las demás secuencias no son más que una simplificación de la expuesta.

Con la finalización de este apartado se da por terminado el Capítulo 4.

Capítulo 5

Conclusiones y trabajos futuros

5.1 Resultados obtenidos.

En este apartado se expondrán las diferentes conclusiones adquiridas a raíz de los resultados obtenidos una vez finalizado este Proyecto Fin de Grado.

Esta exposición de conclusiones se dividirá en dos apartados. El primero de ellos hará referencia a los éxitos que se consideran haber obtenido tras el desarrollo de este proyecto. El segundo apartado hablará de los puntos débiles del sistema, así como de las causas subyacentes.

5.1.1 Éxitos alcanzados.

Los éxitos obtenidos una vez finalizado el proyecto y con el sistema plenamente funcional, siempre descritos desde un punto de vista personal, pueden resumirse en los puntos que se listarán a continuación:

- Desarrollo de un sistema inteligente funcional e innovador.

  De entre todos los objetivos propuestos para el proyecto, el principal y a la vez más complejo era el diseño e implementación de un sistema capaz de "razonar". Lo que para los seres humanos puede resultar tan sencillo es, de hecho, extremadamente complicado para un sistema informático.

  El sistema, a su vez, no requería únicamente de la interpretación de una orden, lo que por sí solo ya resulta objeto de estudio de grandes equipos de investigación, sino que además debía ser capaz de analizar el contexto que envolvía el momento en cuestión y las particularidades del propio usuario, aplicando toda esa información al proceso de decisión.

  Una vez finalizado el proyecto, y sin perjuicio del trabajo que han requerido otros elementos del sistema, se puede confirmar que el diseño e implementación de este componente ha sido sin duda el que más esfuerzo ha requerido, exigiendo un constante rediseño (con las pertinentes pruebas unitarias y de conjunto) durante el desarrollo del proyecto, para adaptarse a las necesidades que surgían a raíz de la incorporación de nuevos elementos y resolviendo todos aquellos problemas que surgen en un proyecto de innovación como éste.

  Además, su estrecha relación de dependencia con muchos otros componentes que sustentan el concepto de inteligencia artificial ha planteado un gran reto en su resolución. Aunque sobre el papel y mediante diferentes pruebas unitarias el sistema debía funcionar, no ha sido hasta los últimos pasos del desarrollo de este proyecto cuando se ha podido observar el comportamiento del sistema en su conjunto y verificar que el diseño, finalmente, resultaba válido para los objetivos propuestos.

  En definitiva, se considera que el resultado obtenido cumple las expectativas y objetivos definidos para este proyecto.

  El éxito de la capa de inteligencia desarrollada no se limita a su capacidad de llevar a cabo las tareas que se plantearon como objetivo, sino que recae, sobre todo, en la

manera de conseguirlo: se ha diseñado desde el primer instante con el requisito de coste cero, es decir, usando únicamente software gratuito e información de Internet.

- Desarrollo de un sistema completo y heterogéneo.

  En el desarrollo de este proyecto se han diseñado e implementado gran cantidad de elementos de muy diferente índole. Desde que se plantearon los objetivos del sistema se definió que dicho sistema debería resultar completo y funcional.

  No se ha pretendido en este proyecto desarrollar únicamente una idea teórica, sino que además se han dedicado grandes esfuerzos en llevar esa idea a cabo y construir un sistema funcional, desde la red de dispositivos hasta la interacción con el usuario.

- Simplicidad final desde el punto de vista de usuario.

  A pesar de la complejidad subyacente, uno de los requisitos que se impusieron para la creación del sistema fue el de simplificar la interacción con el usuario. Este planteamiento es tendencia en los últimos años en el desarrollo de aplicaciones móviles. El modelo conversacional desarrollado en este proyecto simplifica al usuario en gran medida la utilización del sistema, pues sólo ha de expresarse tal como lo hace en el día a día.

- Flexibilidad y estándar de calidad.

  Otro de los objetivos definidos para el desarrollo del sistema era el de enfocar su diseño, dentro de lo posible, a un producto final. Además, este enfoque ha exigido un diseño no completamente cerrado, sino que los distintos elementos se han implementado con capacidad de crecimiento y adaptabilidad.

- Simulación versátil de dispositivos.

  La creación del *Simulador* ha flexibilizado los procesos de desarrollo y prueba de diferentes partes del sistema, permitiendo un diseño más avanzado y completo del proceso de interpretación.

  Este componente ha proporcionado al sistema la capacidad de añadir cualquier dispositivo a la red y, por extensión, la posibilidad de poner a prueba las diferentes configuraciones.

- Reducido tamaño.

  Cuando se indaga sobre la tendencia que han llevado a cabo diferentes grupos o compañías a la hora de implementar sistemas capaces de interpretar órdenes, un mecanismo habitual suele ser el de contar con extensas base de datos, junto con los mecanismos de acceso y consulta pertinentes, con el objetivo de relacionar rápidamente los términos utilizados por el usuario con ciertas acciones mediante el conocimiento previo.

  Por ello, se considera un éxito el hecho de que el sistema desarrollado, en conjunto, tenga un peso total de 40MB (incluyendo al Simulador, el Controlador Principal y la aplicación Android).

- Adquisición de conocimientos muy diversos

  Por último, pero no por ello menos importante, el trabajo intensivo con diferentes tecnologías y la necesidad de integrar todas ellas en un sistema con un objetivo muy concreto ha implicado la adquisición de variados conocimientos en áreas muy diversas.

  De esta manera, conocimientos sobre tecnologías que a lo largo de los estudios correspondientes a esta titulación no han podido estudiarse en profundidad quedan ampliados o incluso adquiridos por primera vez, enriqueciendo el conjunto de recursos técnicos incorporados a mi *curriculum*.

5.1.2 Puntos débiles

A pesar de que, en conjunto, se considera que el proyecto ha alcanzado los objetivos planteados, bien es cierto que el sistema también tiene sus puntos débiles, siendo los principales los siguientes:

- Mala escalabilidad de la red de dispositivos.

  El sistema ha demostrado ser muy rápido cuando el número de dispositivos presentes en la red es reducido (en torno a 5-10 dispositivos), devolviendo respuestas al usuario de manera casi inmediata.

  No obstante, el diseño de cruces de conceptos implica un crecimiento exponencial de las operaciones y consultas que debe realizar el sistema con respecto al número de dispositivos que este maneja, por lo que el rendimiento decrece rápidamente (por ejemplo, una red de unos 15-20 dispositivos requerirá de un tiempo de procesamiento en torno a los 20-40 segundos).

  Este rendimiento, además, varía considerablemente en función de los tipos de dispositivos presentes en la red, pues dispositivos similares reutilizan gran parte del procesamiento en el proceso de decisión, pero dispositivos muy diversos requerirán de nuevos cruces y consultas. También influye considerablemente el hecho de que la orden haya sido procesada previamente o sea la primera vez, pues habrá mayor o menor número de cruces previamente almacenados en la Base de Datos.

- Limitación de acciones disponibles en los dispositivos

  Tal como se planteó en el apartado 3.1.2, se definieron una serie de *licencias* para hacer el desarrollo factible teniendo en cuenta las limitaciones propias de un Proyecto Fin de Grado. Entre estas licencias se encontraba el hecho de limitar las acciones de los actuadores a dos, *encenderse* o *apagarse*.

  Dispositivos con acciones intermedias (como un regulador de luz) han quedado fuera de los dominios de este proyecto, si bien, como ya se explicó, una vez decidido qué dispositivo debe encenderse, sólo requeriría un pequeño cuestionario al usuario definir qué porcentajes o valores deben aplicarse en la acción.

- Decisiones erróneas.

  El índice de aciertos en la toma de decisiones, tal como se ha visto en el apartado de diseño e implementación, depende casi en exclusiva del correcto diseño del algoritmo. No obstante, ni siquiera un algoritmo bien diseñado podrá garantizar el éxito, y siempre existirá la posibilidad de que el sistema proponga erróneamente una acción, si bien se ha procurado que esto suceda el menor número de veces posible.

- Componentes con bajo desarrollo.

  Como consecuencia de las limitaciones temporales propias del proyecto, así como la necesidad de trabajar en un gran número de elementos muy distintos, el grado de desarrollo de algunos componentes del sistema ha sido limitado.

  Se ha pretendido simplificar cada uno de dichos desarrollos para proporcionar la funcionalidad requerida, pero en ocasiones este enfoque ha dejado elementos con ciertas circunstancias sin controlar completamente. Por ejemplo, el analizador morfológico obtendrá siempre el primer tipo morfológico que proporcione la RAE para una palabra, sin atender a si realmente la palabra, en su contexto, tiene otra acepción.

  Por último, aunque se ha procurado mantener un buen control de errores y proporcionar la información oportuna cuando estos suceden, la gran mayoría de elementos del sistema no tienen capacidad de recuperación automática ante errores de cierta magnitud (con la única excepción de la aplicación, que como interfaz de usuario y punto visible del sistema procura simplificar al máximo la participación del usuario).

En referencia a estos puntos débiles del sistema, así como a posibles desarrollos alternativos, se expondrá en el próximo apartado los posibles trabajos futuros que surgen de este proyecto.

5.2 Trabajos futuros.

Como último punto de este apartado, se presentan a continuación una lista de posibles trabajos futuros que mejorarían el sistema en diferentes aspectos, muchos de ellos en consonancia con los puntos débiles expuestos previamente:

- Mejora de la escalabilidad de la red de dispositivos.

  Aunque la base lógica de cruces de contextos podría ser mantenida, existen medidas de paralelización de procesos y filtrados previos que podrían intentar proporcionar al sistema un rendimiento más constante e independiente del número de dispositivos.

- Mejora del análisis morfológico.

  Tal como se ha expuesto en el punto anterior, el análisis morfológico podría ser mejorado para garantizar la correcta asignación de tipos en función del contexto de la frase en cuestión.

- Incorporación de nuevos factores contextuales.

  En este proyecto se han hecho uso de los factores contextuales más habituales y generales, pero podría existir la posibilidad en el futuro de incorporar algunos nuevos que particularicen a las circunstancias del usuario o la vivienda algunas decisiones.

- Mejora del control de dispositivos para realizar cualquier tipo de acción.

  Por supuesto, una de las mejoras más importantes que podría sufrir el sistema sería la extensión de su funcionalidad para soportar acciones y estados intermedios. Recuérdese que sólo se manejan dos estados: completamente encendido o completamente apagado. Aunque ya se ha mencionado algún posible enfoque para lograr este objetivo, posiblemente esta mejora requeriría de un considerable nivel de esfuerzo, pudiendo ser necesario incluso modificar parte del diseño del sistema.

- Introducción de más medidas climatológicas.

  La API climatológica utilizada en este sistema proporciona un conjunto reducido de datos, si bien mediante la incorporación de nuevas fuentes de información estas podrían ampliarse para adaptarse a nuevos tipos de dispositivos.

- Aprendizaje extendido a la interpretación

  El sistema se ha diseñado para almacenar en la base de datos todas las consultas que se llevan a cabo en Internet, ya sean a los motores de búsqueda a otros recursos. Esto proporciona cierto nivel de aprendizaje, pues las futuras consultas que se requieran podrán ser obtenidas directamente desde la BD sin acceder a Internet y reduciendo considerablemente el tiempo de ejecución.

  No obstante, existen otros mecanismos en la interpretación de órdenes que no se han diseñado para trabajar con resultados anteriores.

  Sería interesante para el sistema almacenar las órdenes junto con los factores contextuales y la conclusión obtenida en una cierta ejecución con el objetivo de recuperarla en futuros procesamientos. Si el usuario realiza una orden previamente procesada con un contexto suficientemente similar, el sistema podría recuperar la conclusión sin necesidad de procesar la orden de nuevo, lo que aceleraría el tiempo de respuesta.

- Mejora de la sintaxis de definición del algoritmo de interpretación

  Con el fin de flexibilizar al máximo el diseño del algoritmo de interpretación para adaptarlo rápidamente a nuevos requisitos que pudiesen surgir durante el desarrollo de este proyecto, se ha utilizado una sintaxis ad-hoc que requiere un procesamiento único para el sistema y resulta poco extensible.

  Una vez completado su diseño y con la estructura final del algoritmo definida, se considera recomendable que dicho algoritmo pudiese ser procesado de una manera más estandarizada, por lo que su migración a una sintaxis basada en XML podría resultar interesante para el sistema.

Anexo I

Manual de usuario

Contenido del anexo

Este anexo tiene como objetivo describir brevemente la estructuración de ficheros que componen el sistema desarrollado en este proyecto, así como proporcionar las indicaciones necesarias para poder utilizarlo.

En lo que respecta a la configuración, la gran mayoría de parámetros cuentan con una descripción junto a ellos en los propios ficheros, luego no se entrará en detalle con cada uno de ellos. Sin embargo, sí se describirán en conjunto, indicando a qué factores se orienta cada grupo.

El anexo tratará los siguientes puntos:

1. Requisitos
2. Estructuración de ficheros
3. Configuración del sistema
4. Arranque del sistema
5. Uso de la aplicación Android

Requisitos

El sistema se compone de tres subsistemas independientes: dos programas Java y una aplicación Android. Para correr el sistema se requieren los siguientes equipos y software:

- Dispositivo móvil con SO Android 2.2 o superior y conectividad Wi-Fi.

  La aplicación se ha optimizado para su uso con Android 4.4.x, si bien todas las librerías de las que hace uso son soportadas por versiones anteriores, desde la versión 2.2 (*Froyo*).

  El diseño de la interfaz gráfica ha tomado como referencia el dispositivo *HTC One M7*, con resolución de pantalla *1.920 x 1.080* (*4,7 pulgadas*). No se garantiza la correcta representación de gráficos en otros dispositivos con resoluciones distintas, en especial aquellos con otras distribuciones de pantalla[1] (véase apartado 3.1.2).

- PC con conectividad a Internet y máquina virtual de Java instalada (JRE 1.7).

  Las pruebas del sistema se han llevado a cabo sobre un PC Portátil con SO Windows 8.1 (64 bits) y procesador Intel Core i3 (2,4GHz y 4GB RAM).

  El sistema puede funcionar sobre equipos con menos recursos, pero el tiempo de ejecución necesario para los diferentes procesos podría aumentar (en diferentes medidas en función de la complejidad de la red de dispositivos simulada).

  Se pueden utilizar dos PC distintos y correr el *Simulador* y el *Controlador Principal* en diferentes equipos. Se recomienda alojar el *Controlador Principal* en el más potente.

Todos los equipos deben estar conectados a la misma red local (por ejemplo, conectados al mismo router Wi-Fi de una vivienda). El Equipo que aloje el *Controlador Principal* debe tener salida a Internet desde dicha red[2].

[1] La aplicación se ha puesto a prueba en varias ocasiones en una tablet con pantalla de 10,1 pulgadas, resolución 1,280 x 800 y Android 4.2. El funcionamiento de todas las características de la aplicación han sido correctos, pero los gráficos (bocadillos, tamaños de letra, etc.) se han desproporcionado, como cabía esperar.

[2] La aplicación Android, en principio, no hace uso de datos de Internet. Sin embargo, si el usuario no tiene descargados de manera local los motores de habla y reconocimiento de voz (como ocurría en versiones anteriores del SO Android), el móvil podría requerir su uso.

Estructuración de ficheros

En lo referente a la aplicación Android, no se hará mención a la estructuración de ficheros, pues todos ellos se encontrarán empaquetados en un único fichero con extensión *.apk*.

CONTROLADOR PRINCIPAL

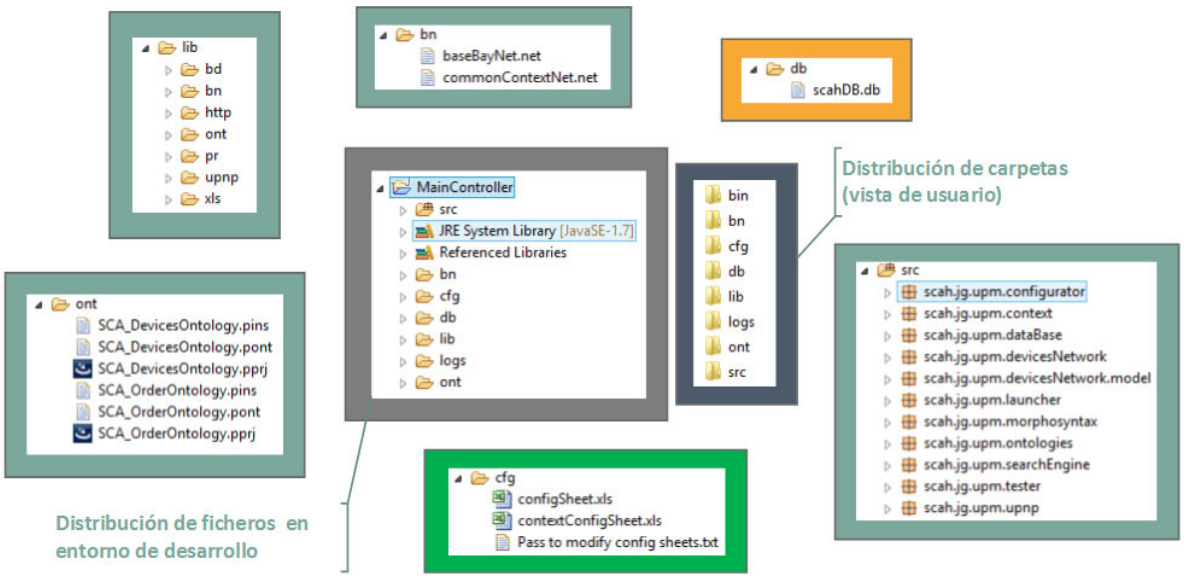


Ilustración 1. Estructuración de ficheros (Controlador Principal)

El entorno de desarrollo mostrado es Eclipse 4.2.1 (*Juno*).

La estructura desde la carpeta raíz (con nombre *MainController*) debe mantenerse estable y su contenido no debe variar, salvo en aquella con nombre *cfg*.

Desde el punto de vista del usuario, sólo resulta interesante la carpeta *cfg* por contener las hojas de datos de configuración. La Base de Datos de la carpeta *bd* puede eliminarse si se desea para limpiar el conocimiento del sistema y el programa generará otra, aunque no se recomienda (pues cuanto más información contenga más rápido responderá el sistema). El resto de elementos no deben ser modificados por el usuario.

APLICACIÓN ANDROID

Todos los ficheros de esta aplicación estarán empaquetados en un fichero con nombre *scah.app.apk.*

El usuario únicamente debe ejecutar este fichero en su dispositivo (*clicar* sobre él) y éste se encargará de instalarlo en el sistema Android correctamente.

SIMULADOR

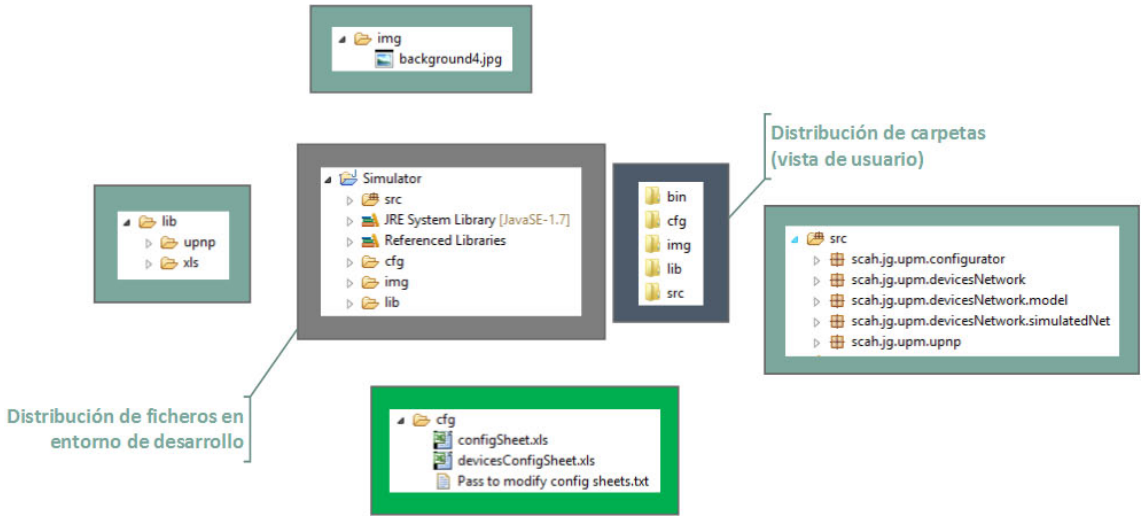


Ilustración 2. Estructuración de ficheros (Simulador)

De nuevo, toda la estructura de carpetas desde la raíz (*Simulator* en la imagen) debe permanecer estable.

El usuario solo debe acceder a la carpeta *cfg*, concretamente a las hojas de datos que contienen la configuración de dispositivos simulados y la configuración general.

Configuración del sistema

La configuración referente a la aplicación, la cual se llevará a cabo mediante los mecanismos propios de Android (en tiempo de ejecución) no se describirán en este apartado, sino que se hará mención a ellos en el correspondiente apartado de uso de la aplicación.

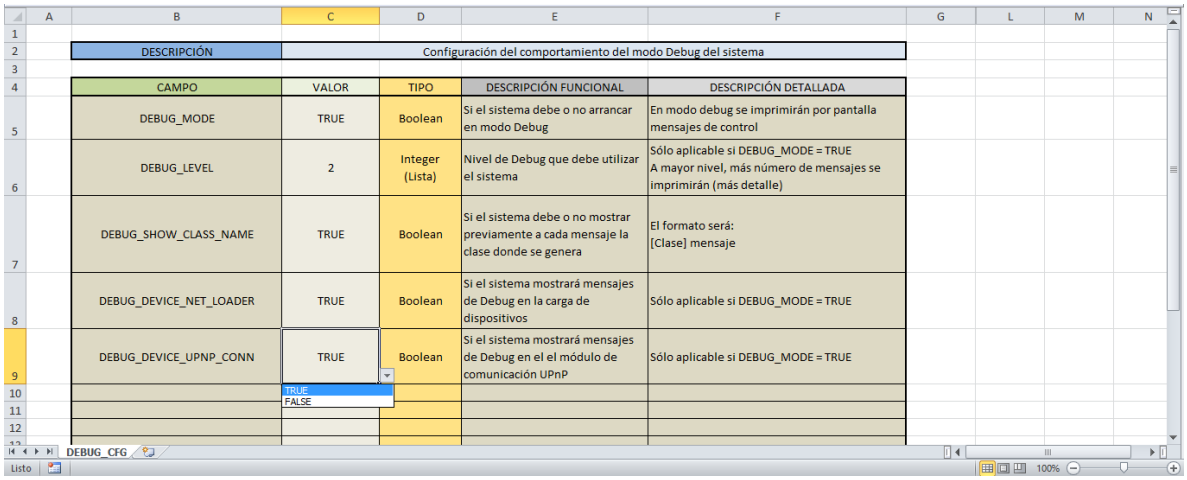
Existen tres conjuntos de parámetros de configuración:

1. Configuración general, con presencia en:
    a. *Controlador Principal*: gran cantidad de parámetros.
    b. *Simulador*: únicamente parámetros de depuración.
2. Configuración de conceptos con dependencia externa, con presencia en:
    a. Controlador Principal
3. Configuración de dispositivos a simular, con presencia en:
    a. Simulador

CONFIGURACIÓN DEL SIMULADOR

Configuración general

[cfg/configSheet.xls]



DESCRIPCIÓN	Configuración del comportamiento del modo Debug del sistema			
CAMPO	VALOR	TIPO	DESCRIPCIÓN FUNCIONAL	DESCRIPCIÓN DETALLADA
DEBUG_MODE	TRUE	Boolean	Si el sistema debe o no arrancar en modo Debug	En modo debug se imprimirán por pantalla mensajes de control
DEBUG_LEVEL	2	Integer (Lista)	Nivel de Debug que debe utilizar el sistema	Sólo aplicable si DEBUG_MODE = TRUE A mayor nivel, más número de mensajes se imprimirán (más detalle)
DEBUG_SHOW_CLASS_NAME	TRUE	Boolean	Si el sistema debe o no mostrar previamente a cada mensaje la clase donde se genera	El formato será: [Clase] mensaje
DEBUG_DEVICE_NET_LOADER	TRUE	Boolean	Si el sistema mostrará mensajes de Debug en la carga de dispositivos	Sólo aplicable si DEBUG_MODE = TRUE
DEBUG_DEVICE_UPNP_CONN	TRUE	Boolean	Si el sistema mostrará mensajes de Debug en el el módulo de comunicación UPnP	Sólo aplicable si DEBUG_MODE = TRUE

Ilustración 3. Configuración general (Simulador)

La configuración general del Simulador solo contiene parámetros referentes al modo de depuración (Pestaña: *DEBUG_CFG*).

El usuario debe seleccionar en la columna C (no podrá modificar el resto) el valor que quiere otorgar a la variable en cuestión (que puede estar limitado por conjuntos de valores predefinidos o ciertos requisitos de formato y valor).

Una vez se haya completado la configuración, el usuario debe guardar los cambios. El sistema podrá leer los cambios al arrancar incluso con la hoja abierta, pero se recomienda que se cierre antes del arranque para evitar posibles fallos de lectura imprevistos.

Configuración de dispositivos

[cfg/devicesConfigSheet.xls]

Esta configuración, única del Simulador, es la encargada de añadir o quitar dispositivos simulados al sistema.

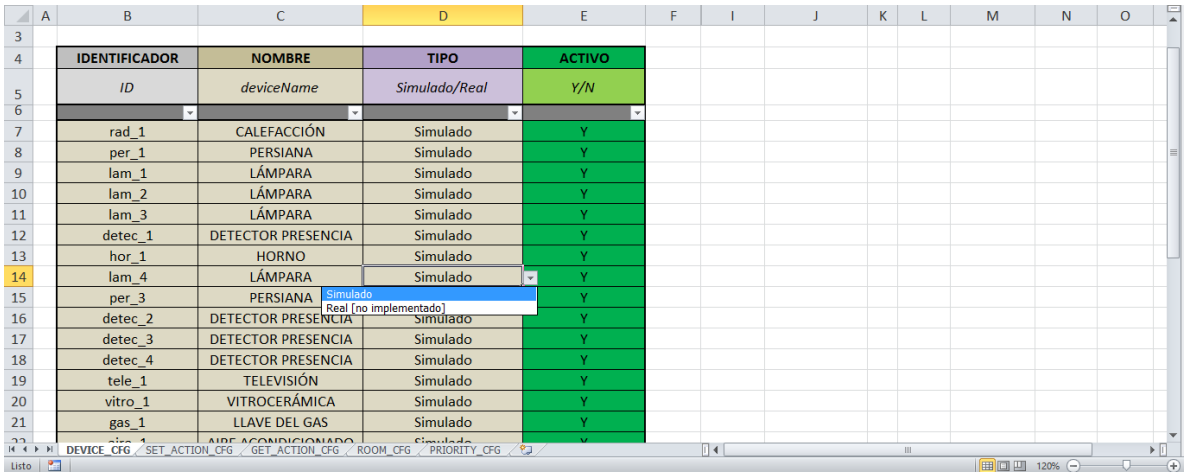
La configuración cuenta con las siguientes pestañas que el usuario debe rellenar:

- *DEVICE_CFG* (obligatorio): configuración de la información base de los dispositivos (ID y nombre)
- *SET_ACTION_CFG* (obligatorio si el dispositivo es un actuador): configuración de las acciones de dispositivos actuadores.
- *GET_ACTION_CFG* (obligatorio si el dispositivo es un sensor): configuración de las acciones de dispositivos sensores.
- *ROOM_CFG* (obligatorio): configuración de la ubicación se los dispositivos.

- *PRIORITY_CFG* (opcional): configuración de las prioridades de los dispositivos.

Se detalla a continuación el formato de cada una de estas pestañas.

DEVICE_CFG



IDENTIFICADOR	NOMBRE	TIPO	ACTIVO
ID	*deviceName*	*Simulado/Real*	*Y/N*
rad_1	CALEFACCIÓN	Simulado	Y
per_1	PERSIANA	Simulado	Y
lam_1	LÁMPARA	Simulado	Y
lam_2	LÁMPARA	Simulado	Y
lam_3	LÁMPARA	Simulado	Y
detec_1	DETECTOR PRESENCIA	Simulado	Y
hor_1	HORNO	Simulado	Y
lam_4	LÁMPARA	Simulado	Y
per_3	PERSIANA	Simulado	Y
detec_2	DETECTOR PRESENCIA	Simulado	Y
detec_3	DETECTOR PRESENCIA	Simulado	Y
detec_4	DETECTOR PRESENCIA	Simulado	Y
tele_1	TELEVISIÓN	Simulado	Y
vitro_1	VITROCERÁMICA	Simulado	Y
gas_1	LLAVE DEL GAS	Simulado	Y

Ilustración 4. Configuración de dispositivos

El usuario debe otorgar un identificador único a cada dispositivo e indicar su nombre. En la configuración, pensada para adaptarse a posibles futuros desarrollos, se puede indicar si el dispositivo es real o simulado, si bien se ignorarán todos aquellos que se indiquen como reales al no haberse implementado su control.

Se debe procurar que el nombre del dispositivo refleje apropiadamente a este, pues será un concepto clave a la hora de identificar a qué dispositivo puede estar refiriéndose el usuario (en función del algoritmo de interpretación).

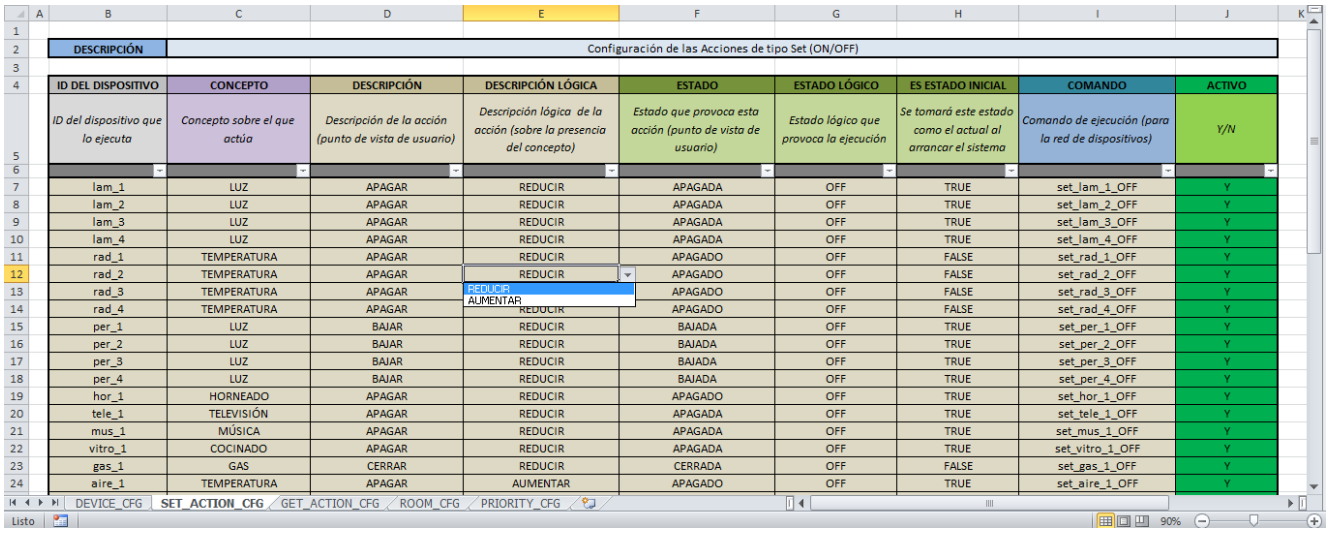
La hoja permite desactivar algunos dispositivos, de manera que el sistema los ignore. Este enfoque, seguido en todas las hojas, permite ignorar información sin necesidad de eliminarla, lo que favorece su reactivación (especialmente útil en fase de desarrollo y pruebas).

Si el usuario introduce de manera errónea algún dato, tanto de esta como de otras pestañas, el dispositivo y todos sus datos son ignorados, como si no se hubiese añadido (por lo que desactivar un campo necesario en cualquiera de las pestañas implica la desactivación del dispositivo entero, salvo que se añadan los valores necesarios en otra fila de dicha pestaña).

En conjunto, un dispositivo debe contener datos de todas las pestañas, salvo aquella referente a las prioridades (que son opcionales). No obstante, un dispositivo solo puede ser sensor o actuador, por lo que sólo deben añadirse acciones a una de las pestañas, según corresponda.

Todas las hojas permiten aplicar filtros (fila 6) para ubicar rápidamente los valores de la columna de interés (como dispositivos de un cierto tipo o aquellos ubicados en una misma habitación).

SET_ACTION_CFG



DESCRIPCIÓN	Configuración de las Acciones de tipo Set (ON/OFF)

ID DEL DISPOSITIVO	CONCEPTO	DESCRIPCIÓN	DESCRIPCIÓN LÓGICA	ESTADO	ESTADO LÓGICO	ES ESTADO INICIAL	COMANDO	ACTIVO
ID del dispositivo que lo ejecuta	*Concepto sobre el que actúa*	*Descripción de la acción (punto de vista de usuario)*	*Descripción lógica de la acción (sobre la presencia del concepto)*	*Estado que provoca esta acción (punto de vista de usuario)*	*Estado lógico que provoca la ejecución*	*Se tomará este estado como el actual al arrancar el sistema*	*Comando de ejecución (para la red de dispositivos)*	*Y/N*
lam_1	LUZ	APAGAR	REDUCIR	APAGADA	OFF	TRUE	set_lam_1_OFF	Y
lam_2	LUZ	APAGAR	REDUCIR	APAGADA	OFF	TRUE	set_lam_2_OFF	Y
lam_3	LUZ	APAGAR	REDUCIR	APAGADA	OFF	TRUE	set_lam_3_OFF	Y
lam_4	LUZ	APAGAR	REDUCIR	APAGADA	OFF	TRUE	set_lam_4_OFF	Y
rad_1	TEMPERATURA	APAGAR	REDUCIR	APAGADO	OFF	FALSE	set_rad_1_OFF	Y
rad_2	TEMPERATURA	APAGAR	REDUCIR	APAGADO	OFF	FALSE	set_rad_2_OFF	Y
rad_3	TEMPERATURA	APAGAR	REDUCIR / AUMENTAR	APAGADO	OFF	FALSE	set_rad_3_OFF	Y
rad_4	TEMPERATURA	APAGAR	REDUCIR	APAGADO	OFF	FALSE	set_rad_4_OFF	Y
per_1	LUZ	BAJAR	REDUCIR	BAJADA	OFF	TRUE	set_per_1_OFF	Y
per_2	LUZ	BAJAR	REDUCIR	BAJADA	OFF	TRUE	set_per_2_OFF	Y
per_3	LUZ	BAJAR	REDUCIR	BAJADA	OFF	TRUE	set_per_3_OFF	Y
per_4	LUZ	BAJAR	REDUCIR	BAJADA	OFF	TRUE	set_per_4_OFF	Y
hor_1	HORNEADO	APAGAR	REDUCIR	APAGADO	OFF	TRUE	set_hor_1_OFF	Y
tele_1	TELEVISIÓN	APAGAR	REDUCIR	APAGADA	OFF	TRUE	set_tele_1_OFF	Y
mus_1	MÚSICA	APAGAR	REDUCIR	APAGADA	OFF	TRUE	set_mus_1_OFF	Y
vitro_1	COCINADO	APAGAR	REDUCIR	APAGADA	OFF	TRUE	set_vitro_1_OFF	Y
gas_1	GAS	CERRAR	REDUCIR	CERRADA	OFF	FALSE	set_gas_1_OFF	Y
aire_1	TEMPERATURA	APAGAR	AUMENTAR	APAGADO	OFF	TRUE	set_aire_1_OFF	Y

Ilustración 5. Configuración de acciones (actuadores)

El usuario debe añadir la información de los dos posibles estados que puede adquirir un actuador en este sistema (*encendido* o *apagado*) y los conceptos relacionados.

Las columnas *E*, *G* y *H* sólo pueden adquirir uno de los valores permitidos, haciendo referencia a la acción conceptual (cómo afecta la acción a la presencia del concepto[3], si se *aumenta* o se *reduce*), al estado lógico (cómo queda el dispositivo tras ejecutar esta acción, *ON/OFF*) y a si ese estado es el que debe tomar por inicial el Simulador al arrancar (*TRUE/FALSE*). Por su parte, la columna *J* (Activo) sólo puede tomar los valores *Y/N*.

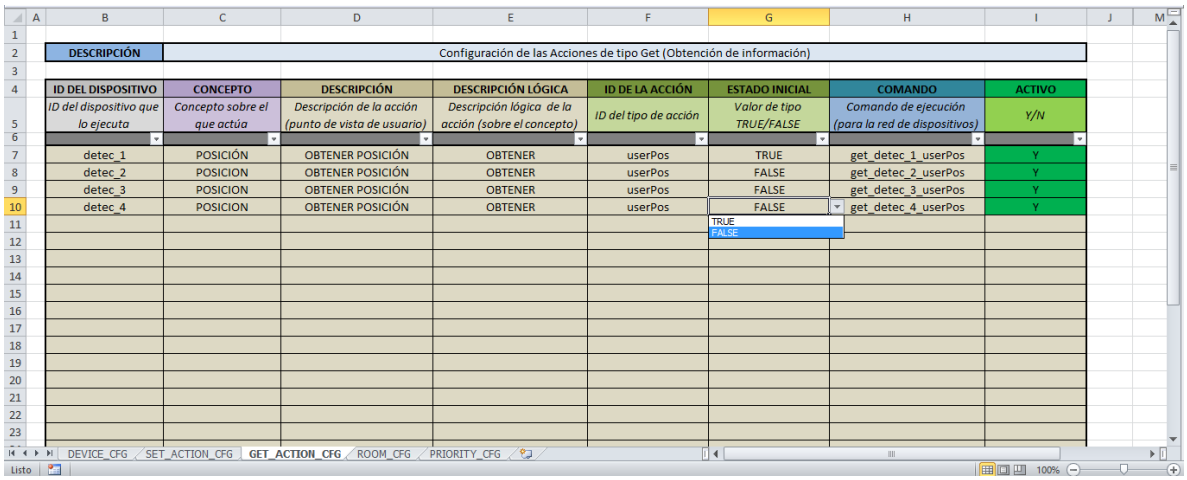
El resto de las columnas son de entrada libre, pero deben respetarse las siguientes reglas para darse el dispositivo por válido:

1. El ID del dispositivo debe corresponder a uno de los dispositivos habilitados en la pestaña anterior.
2. Debe haber exactamente dos registros activos por cada ID (por cada dispositivo), uno que haga referencia a *encender* el dispositivo y otro a *apagarlo*. Es decir, debe haber un registro con estado lógico *ON* y otro con estado lógico *OFF*.
3. El comando de ejecución debe ser único (columna *I*) para un ID concreto. A la hora de ejecutar acciones, siempre se propagará la dupla dispID-comando, que debe ser única.
4. Sólo uno de los dos registros activos (*ON/OFF*) puede ser el estado inicial (columna *H* a *TRUE*).

Si cualquiera de las condiciones anteriores no se cumple, el sistema dará por errónea la entrada de datos para el dispositivo referenciado en el ID y éste no se cargará en el *Simulador*.

[3] Se recuerda que en el ámbito de este proyecto, el término concepto hace referencia a aquella cualidad o estado que se ve afectado por la acción del dispositivo. Por ejemplo, el concepto de una lámpara es la iluminación (que es aumentada o reducida en función del estado de la lámpara).

GET_ACTION_CFG



DESCRIPCIÓN	Configuración de las Acciones de tipo Get (Obtención de información)						
ID DEL DISPOSITIVO	**CONCEPTO**	**DESCRIPCIÓN**	**DESCRIPCIÓN LÓGICA**	**ID DE LA ACCIÓN**	**ESTADO INICIAL**	**COMANDO**	**ACTIVO**
ID del dispositivo que lo ejecuta	*Concepto sobre el que actúa*	*Descripción de la acción (punto de vista de usuario)*	*Descripción lógica de la acción (sobre el concepto)*	*ID del tipo de acción*	*Valor de tipo TRUE/FALSE*	*Comando de ejecución (para la red de dispositivos)*	*Y/N*
detec_1	POSICIÓN	OBTENER POSICIÓN	OBTENER	userPos	TRUE	get_detec_1_userPos	Y
detec_2	POSICION	OBTENER POSICIÓN	OBTENER	userPos	FALSE	get_detec_2_userPos	Y
detec_3	POSICION	OBTENER POSICIÓN	OBTENER	userPos	FALSE	get_detec_3_userPos	Y
detec_4	POSICION	OBTENER POSICIÓN	OBTENER	userPos	FALSE	get_detec_4_userPos	Y

Ilustración 6. Configuración de acciones (sensores)

Cuando el dispositivo sea un sensor, el usuario debe añadir al menos una acción en esta pestaña. Salvo la columna *G* (que limita a *TRUE* o *FALSE* sus posibles valores) y la columna *I* para activar o desactivar filas completas (*Y/N*), el resto son columnas de entrada libre.

Para el desarrollo de este proyecto, los únicos sensores cuya funcionalidad se maneja son los sensores de presencia. Por ello, es suficiente con indicar el estado inicial (si detectan o no a un usuario en la habitación en la que se encuentran en el momento del arranque, indicado mediante un booleano).

Dada las limitaciones temporales en el desarrollo del sistema (véase apartado 3.1.2), solo se ha implementado el uso de detectores de presencia cuyo ID de acción sea *userPos*. Cuando el sistema requiera obtener la posición del usuario, interrogará a los sensores capaces de llevar a cabo esta acción (cualquier otro sensor no tendrá funcionalidad en el sistema actualmente desarrollado).

No obstante, para procurar mantener una estructura capaz de albergar y utilizar nuevos sensores con bajo nivel de desarrollo, se permite la introducción de cualquier valor en estas columnas (aunque los dispositivos no cumplan ninguna función en el sistema actual).

Las columnas *C*, *D* y *E* no tienen peso en el funcionamiento de estos dispositivos, aunque son necesarias para respetar la estructura de dispositivos en la ontología, por lo que deben ser rellenados (independientemente del valor).

Por su parte, la columna *A* debe identificar el ID de un dispositivo añadido en la primera pestaña y la columna *H* debe contener un comando único para ese ID. Si Esto no se respeta, se ignorará el dispositivo.

ROOM_CFG

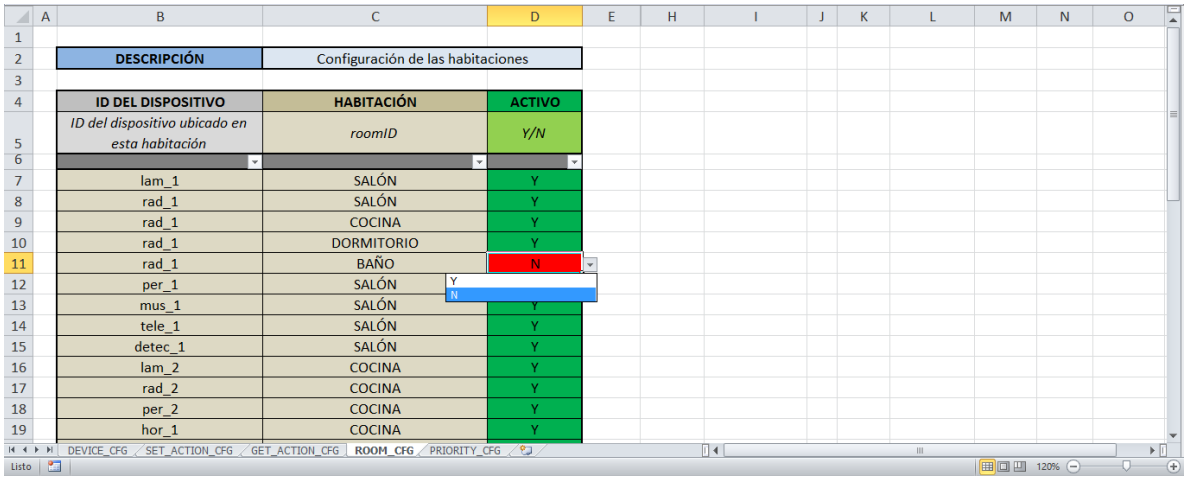


Ilustración 7. Configuración de ubicación de dispositivos

La última información obligatoria que debe introducir el usuario a la hora de añadir dispositivos es la ubicación de estos.

Salvo la columna *D* (Activo), el resto de columnas son de entrada libre. La columna *B* debe hacer referencia a un dispositivo añadido en la primera pestaña y la columna *C* debe indicar el nombre de la habitación.

El Simulador, cuando arranque, creará tantas habitaciones como valores diferentes existan en la columna C (por lo que se debe indicar el nombre de una misma habitación siempre de la misma manera).

Se permite que un dispositivo esté en varias habitaciones, pero todos deben estar al menos en una. Cualquier dispositivo cuya ubicación no haya sido indicada al menos una vez se ignorará y no se cargará en el Simulador.

Cuando un dispositivo se asocia a varias habitaciones, encenderlo (o apagarlo) en una implica hacerlo en todas. Esto puede resultar útil en dispositivos que afecten a varias habitaciones (por ejemplo, la calefacción).

De nuevo, la columna A debe contener el ID de un dispositivo añadido en la primera pestaña. De no ser así, esta fila se ignorará al cargar la configuración.

PRIORITY_CFG

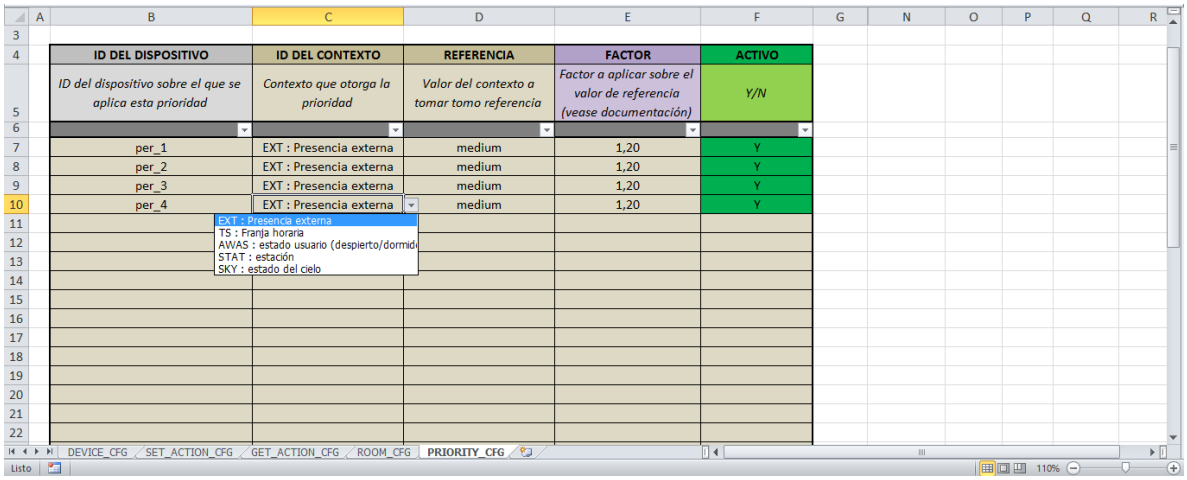


Ilustración 8. Configuración de prioridades

La configuración en esta hoja sólo debe aplicarse para aquellos dispositivos sobre los que quiera aplicarse prioridades.

Tal como se explicó en el apartado 4.6.4, para aplicar una prioridad debe indicarse el nodo de la red contextual (bayesiana) que se debe leer (columna *C*), el valor de referencia (columna *D*) y el factor a aplicar (columna *E*).

La columna *C* permite únicamente un conjunto de valores predefinidos que identifican nodos de la red contextual entendibles para el usuario. No obstante, el valor de referencia de la columna *D* es texto libre y debe identificar correctamente el valor del nodo correspondiente que se quiere leer (véase Figura 32).

La columna *E*, por su parte, debe ser un número (forzado a la hora de introducir valores, alertando en caso de introducción incorrecta).

Una vez más, debe identificarse correctamente el ID del dispositivo sobre el que debe aplicarse o la fila entera será ignorada.

Dependencia entre pestañas

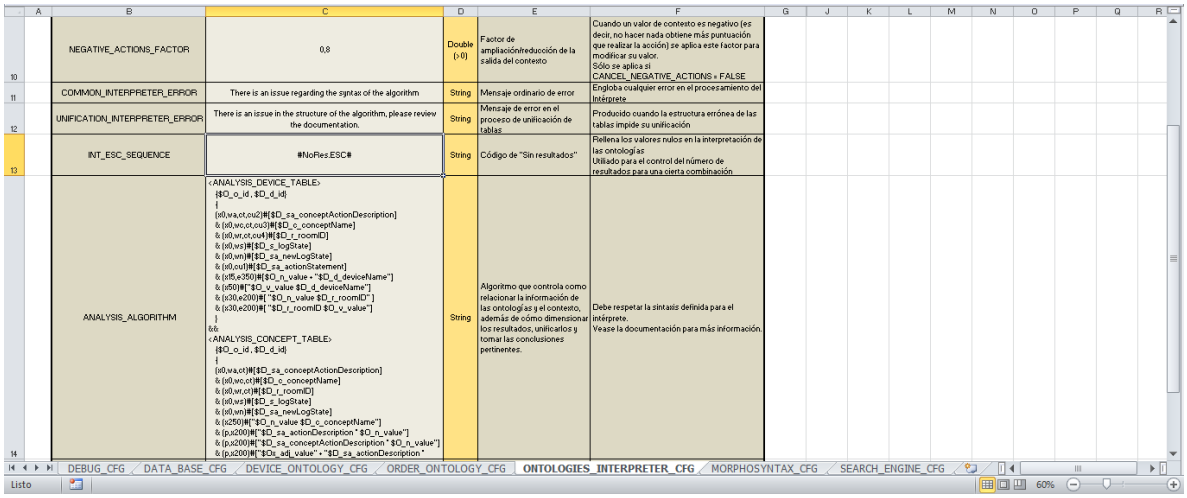
Cuando el simulador arranque y lea la información de los dispositivos que tiene que cargar, validará la información introducida por el usuario y determinará si el dispositivo es válido para su manejo por el sistema. Esto implica que las condiciones impuestas para cada pestaña deben cumplirse y garantiza que los dispositivos añadidos a la red virtual son operativos.

Aquellos dispositivos que han fallado en la validación serán indicados si el modo de depuración está activo, de manera que el usuario pueda corregir los datos erróneos.

CONFIGURACIÓN DEL CONTROLADOR PRINCIPAL

Configuración general

[cfg/configSheet.xls]



	B	C	D	E	F
10	NEGATIVE_ACTIONS_FACTOR	0,8	Double (>0)	Factor de ampliación/reducción de la salida del contexto	Cuando un valor de contexto es negativo (es decir, no hacer nada obtiene más puntuación que realizar la acción) se aplica este factor para modificar su valor. Sólo se aplica si CANCEL_NEGATIVE_ACTIONS = FALSE
11	COMMON_INTERPRETER_ERROR	There is an issue regarding the syntax of the algorithm	String	Mensaje ordinario de error	Engloba cualquier error en el procesamiento del Intérprete
12	UNIFICATION_INTERPRETER_ERROR	There is an issue in the structure of the algorithm, please review the documentation.	String	Mensaje de error en el proceso de unificación de tablas	Producido cuando la estructura errónea de las tablas impide su unificación
13	INT_ESC_SEQUENCE	#NoRes:ESC#	String	Código de "Sin resultados"	Rellena los valores nulos en la interpretación de las ontologías Utilado para el control del número de resultados para una cierta combinación

14	ANALYSIS_ALGORITHM	<ANALYSIS_DEVICE_TABLE>
		[$O_o_id , $D_d_id]
		{
		(s0,w,a,ct,cu2)#[$D_sa_conceptActionDescription]
		& (s0,w,c,ct,cu3)#[$D_c_conceptName]
		& (s0,w,r,ct,cu4)#[$D_r_roomID]
		& (s0,w,s)#[$D_s_logState]
		& (s0,w,n)#[$D_sa_newLogState]
		& (s0,cu1)#[$D_sa_actionStatement]
		& (x15,e350)#[$O_n_value + "$D_d_deviceName"]
		& (x50)#["$O_v_value $D_d_deviceName"]
		& (s30,e200)#["$O_n_value $D_r_roomID"]
		& (s30,e200)#["$D_r_roomID $O_v_value"]
		}
		&&
		<ANALYSIS_CONCEPT_TABLE>
		[$O_o_id , $D_d_id]
		{
		(s0,w,a,ct)#[$D_sa_conceptActionDescription]
		& (s0,w,c,ct)#[$D_c_conceptName]
		& (s0,w,r,ct)#[$D_r_roomID]
		& (s0,w,s)#[$D_s_logState]
		& (s0,w,n)#[$D_sa_newLogState]
		& (s250)#[" $O_n_value $D_c_conceptName"]
		& (px200)#["$D_sa_actionDescription " $O_n_value"]
		& (px200)#["$D_sa_conceptActionDescription " $O_n_value"]
		& (px200)#["$Ds_adj_value"+ "$D_sa_actionDescription "
			String	Algoritmo que controla como relacionar la información de las ontologías y el contexto, además de cómo dimensionar los resultados, unificarlos y tomar las conclusiones pertinentes.	Debe respetar la sintaxis definida para el intérprete.
					Véase la documentación para más información.

Ilustración 9. Configuración general (Controlador Principal)

El fichero de configuración general del *Controlador Principal* contiene gran cantidad de parámetros de configuración, cada pestaña enfocada a configurar el comportamiento de uno de los componentes del sistema.

Dado que todos los parámetros tienen su correspondiente descripción en la tabla de configuración, no se entrará más en detalle. Únicamente debe tenerse en cuenta por parte del usuario que la única columna modificable es la columna *C*, y pueden existir listas o condiciones en la introducción de datos para adaptarlos a los requisitos del sistema (en general, se indica en la columna *D* tanto el tipo como las condiciones a cumplir en los valores que se quieran introducir, si bien una ventana de alerta avisará cuando se cometa algún error).

La única descripción que se hará respecto a este fichero pretende describir la sintaxis de variables del algoritmo de interpretación, pues los patrones que deben seguirse en su definición (y su significado) han sido descritos en detalle en el apartado 4.6.

Variables en el algoritmo de interpretación

El formato de una variable es el siguiente:

- **$**<*ID de ontología*>_<*ID de clase*>_<*ID de slot*>

El campo <*ID de slot*> debe corresponder exactamente al nombre del Slot en cuestión, tal como se ha definido en la ontología. La nomenclatura utilizada en la definición de *Clases* y *Slots* puede ser consultada en los propios ficheros de la ontología, con la utilización del software Protège-Frames:

- Ruta de ontología de dispositivos: *ont/SCA_DevicesOntology.pprj*
- Ruta de ontología de órdenes: *ont/SCA_OrderOntology.pprj*

El nombre utilizado en la definición de *Clases* y *Slots* está también presente en las pestañas *DEVICE_ONTOLOGY_CFG* y *ORDER_ONTOLOGY_CFG*, donde puede consultarse rápidamente (estos valores deben estar alineados con los definidos en la ontología, por lo que no deben ser modificados si no se ha modificado previamente la ontología a la que hacen referencia).

Por su parte, tanto el campo <*ID de ontología*> como el campo <*ID de clase*> deben ser introducidos mediante un alias diseñado con el objetivo de mejorar la legibilidad del algoritmo, pues se trata de un conjunto limitado de valores que aparecerían de manera repetitiva a lo largo de dicho algoritmo, incrementando considerablemente su volumen.

La correspondencia de cada alias con el valor de referencia queda como sigue:

- <*ID de ontología*>

  Este campo identifica la ontología a la que pertenece el valor a cargar.

  - *D*: variable perteneciente a la ontología de dispositivos
  - *O*: variable perteneciente a la ontología de órdenes

- <*ID de clase*>

  Este valor hará referencia a una clase contenida en la ontología identificada en el campo anterior.

  Es importante recalcar que el algoritmo accederá a las diferentes clases por el valor configurado en los parámetros de las pestañas *DEVICE_ONTOLOGY_CFG* y *ORDER_ONTOLOGY_CFG* mencionadas previamente.

  Los alias aquí utilizados no hacen más que cargar el valor de dichos parámetros, luego aunque la nomenclatura de las clases varíe, el algoritmo permanecerá siendo válido (siempre y cuando los parámetros hayan sido actualizados en su pestaña correspondiente).

  No ocurre así con los Slot, que un cambio de nomenclatura en la ontología implicaría un cambio en el algoritmo (dado que se carga el valor directamente). No se ha implementado el mismo mecanismo que para las *Clases* por ser su volumen mucho mayor.

  - Clases de la ontología de dispositivos:
    - *d*: dispositivo
    - *a*: acción del dispositivo
    - *sa*: acción de un actuador
    - *ga*: acción de un sensor
    - *c*: concepto que maneja el dispositivo

- *l*: localización del dispositivo
- *r*: habitación donde se encuentra el dispositivo
- *s*: estado del dispositivo
- *p*: prioridad del dispositivo

- Clases de la ontología de órdenes:
  - *o*: orden
  - *c*: concepto
  - *n*: nombre
  - *adj*: adjetivo
  - *adv*: adverbio
  - *p*: pronombre
  - *v*: verbo

A continuación se muestra a modo de ejemplo un algoritmo en el que puede observarse con facilidad cómo han sido definidas diferentes variables.

```
<ANALYSIS_DEVICE_TABLE>
  {$O_o_id , $D_d_id}
  {
  (x0,wa,ct,cu2)#[$D_sa_conceptActionDescription]
  & (x0,wc,ct,cu3)#[$D_c_conceptName]
  & (x0,wr,ct,cu4)#[$D_r_roomID]
  & (x0,ws)#[$D_s_logState]
  & (x0,wn)#[$D_sa_newLogState]
  & (x0,cu1)#[$D_sa_actionStatement]
  & (x15,e350)#[$O_n_value + "$D_d_deviceName"]
  & (x50)#["$O_v_value $D_d_deviceName"]
  & (x30,e200)#[ "$O_n_value $D_r_roomID" ]
  & (x30,e200)#[ "$D_r_roomID $O_v_value"]
  }
&&
<ANALYSIS_CONCEPT_TABLE>
  {$O_o_id , $D_d_id}
  {
  (x0,wa,ct)#[$D_sa_conceptActionDescription]
  & (x0,wc,ct)#[$D_c_conceptName]
  & (x0,wr,ct)#[$D_r_roomID]
  & (x0,ws)#[$D_s_logState]
  & (x0,wn)#[$D_sa_newLogState]
  & (x250)#["$O_n_value $D_c_conceptName"]
  & (p,x200)#["$D_sa_actionDescription * $O_n_value"]
  & (p,x200)#["$D_sa_conceptActionDescription * $O_n_value"]
  & (p,x200)#["$Ox_adj_value" + "$D_sa_actionDescription * $O_n_value"]
  & (p,x200)#["$Ox_v_value" + "$D_sa_actionDescription * $O_n_value"]
  }
```

Ilustración 10. Definición de variables en el algoritmo

Configuración de contexto

[cfg/contextConfigSheet.xls]

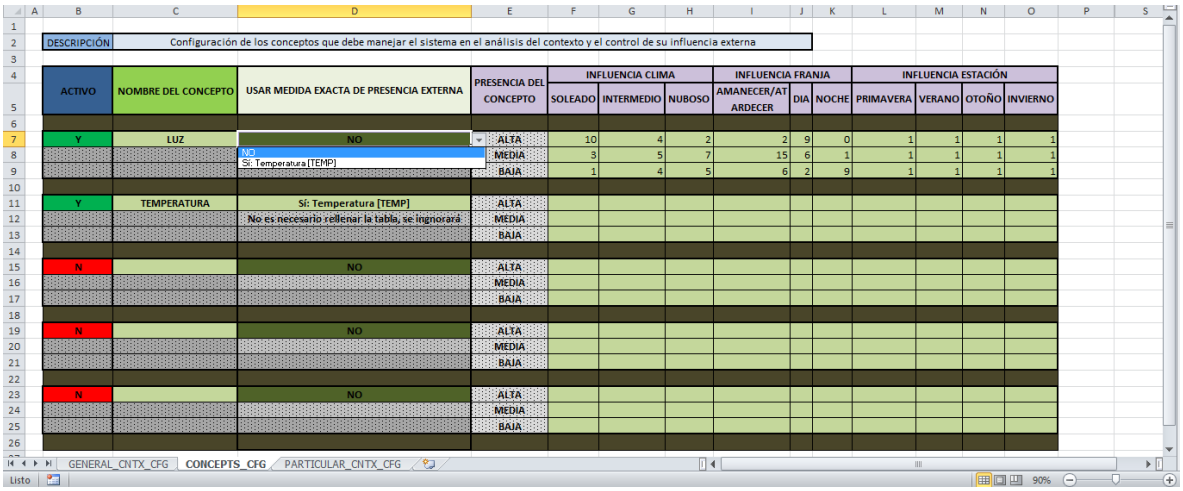


Ilustración 11. Configuración de contexto

La configuración del contexto consta de tres pestañas:

1. *GENERAL_CNTX_CFG*: configuración de parámetros referentes a ubicación de ficheros, nomenclatura, etc.
2. *PARTICULAR_CNTX_CFG*: configuración de parámetros orientados a las preferencias de usuario y comportamiento del sistema en la interpretación (horarios adaptados al usuario, ajuste de márgenes y valores para el estudio probabilístico, etc.)
3. *CONCEPTS_CFG*: configuración de conceptos (en lo que se refiere a conceptos manejados por dispositivos, como *luz* o *temperatura*) que dependen de factores externos. Para más detalles en lo que al modo de empleo de esta hoja se refiere, véase apartado 4.8.2.

Las dos primeras pestañas mencionadas tienen el mismo formato que las utilizadas en la configuración general, y la descripción de cada parámetro viene indicada en la propia tabla de configuración.

Por su parte, la descripción de cómo utilizar la pestaña *CONCEPTS_CFG* y las implicaciones de cada una de las configuraciones fueron vistas en detalle durante en el apartado indicado en el punto 3. Se recuerda que el único valor de tipo *medida* (sin deducción probabilística) soportado por el sistema es la temperatura, si bien el diseño permite con un muy bajo nivel de desarrollo implementar nuevas medidas en el futuro si se amplían los datos que se recuperan desde la API climatológica.

Arranque del sistema

El orden a seguir en el arranque de los diferentes subsistemas (aplicación Android, Controlador Principal y Simulador) no está definido, pues todos inicializarán sus correspondientes elementos y prepararán la comunicación UPnP en cuanto se inicie cada uno de ellos.

Si al arrancar, un subsistema con el que deben conectarse aún no está levantado, esperarán hasta que lo esté y conectarán lo antes posible. No obstante, para agilizar el establecimiento de la comunicación entre subsistemas y seguir un orden lógico (teniendo en cuenta las dependencias entre elementos), se recomienda que los subsistemas se arranquen en el siguiente orden:

1. Arrancar el *Simulador*

   Los dispositivos simulados se cargan y se muestra la red simulada al usuario. El paso más natural, antes de arrancar el resto de elementos, es que la red esté presente en el sistema, luego este subsistema debería estar corriendo antes de arrancar el resto, si se pretende representar una red real.

2. Arrancar el *Controlador Principal*.

   Este subsistema interrogará al *Simulador*, en cuanto pueda establecer la comunicación, sobre los dispositivos de la red. Además, será el nexo de unión entre la *app* y la red de dispositivos, por lo que el sentido lógico establece este subsistema como segundo en el orden de arranque.

3. Arrancar la *aplicación* Android.

   La aplicación hará esperar al usuario hasta que consiga establecer la comunicación con el Controlador Principal, por lo que en vistas a una buena experiencia de usuario (minimizando la espera) se recomienda arrancar este subsistema en último lugar.

INSTRUCCIONES DE ARRANQUE

Las diferentes clases que componen los programas Java que sustentan los subsistemas *Controlador Principal* y *Simulador* deben estar correctamente compiladas antes de su ejecución.

En el proceso de desarrollo se ha utilizado el JDK 1.7.0, y las versiones compiladas de las diferentes clases se encuentran bajo el directorio *bin* de las estructuras de ficheros indicadas en la Ilustración 1 y la Ilustración 2. Para la ejecución del sistema precompilado debe instalarse en el equipo o equipos correspondientes Java 7 (JRE 1.7).

No obstante, existe la posibilidad de recompilar las clases de ambos programas bajo versiones anteriores del JDK. Los diferentes elementos que componen estos sistemas indican no tener dependencias más allá de la versión 1.4, si bien esta circunstancia no ha sido testada y no se garantiza su correcto funcionamiento.

La aplicación Android, por su parte, requiere de Android 2.2 o superior, como ya se adelantó. De la misma manera, la *app* ha sido testada únicamente en los SO Android 4.2 y 4.4, y no se puede garantizar su compatibilidad completa con versiones anteriores.

Finalmente, se indican a continuación las clases (con sus correspondientes rutas) que deben ejecutarse para lanzar estos dos subsistemas:

- *Simulador*
  - *Simulator\bin\scah\jg\upm\upnp\SCAHDevice.class*
- *Controlador Principal*
  - *MainController\bin\scah\jg\upm\upnp\SCAHDevice.class*

Como se observa, el arranque del dispositivo (*UPnP Enabled Device*) que representa a cada uno de estos subsistemas de cara al resto del sistema conlleva la inicialización del resto de elementos representados tras ellos.

Tanto el *Simulador* como el *Controlador Principal* son, desde el punto de vista del conjunto del sistema, dos *UPnP enabled devices* que exponen ciertas funcionalidades útiles para el usuario.

Con el objetivo de facilitar la ejecución de estos dos subsistemas y evitar problemas de dependencias con librerías, paquetes y otras clases del sistema, se proporcionan también las versiones comprimidas de éstos mediante sus correspondientes ficheros .jar (Java ARchive). La estructura de directorios queda como se muestra en la siguiente figura:

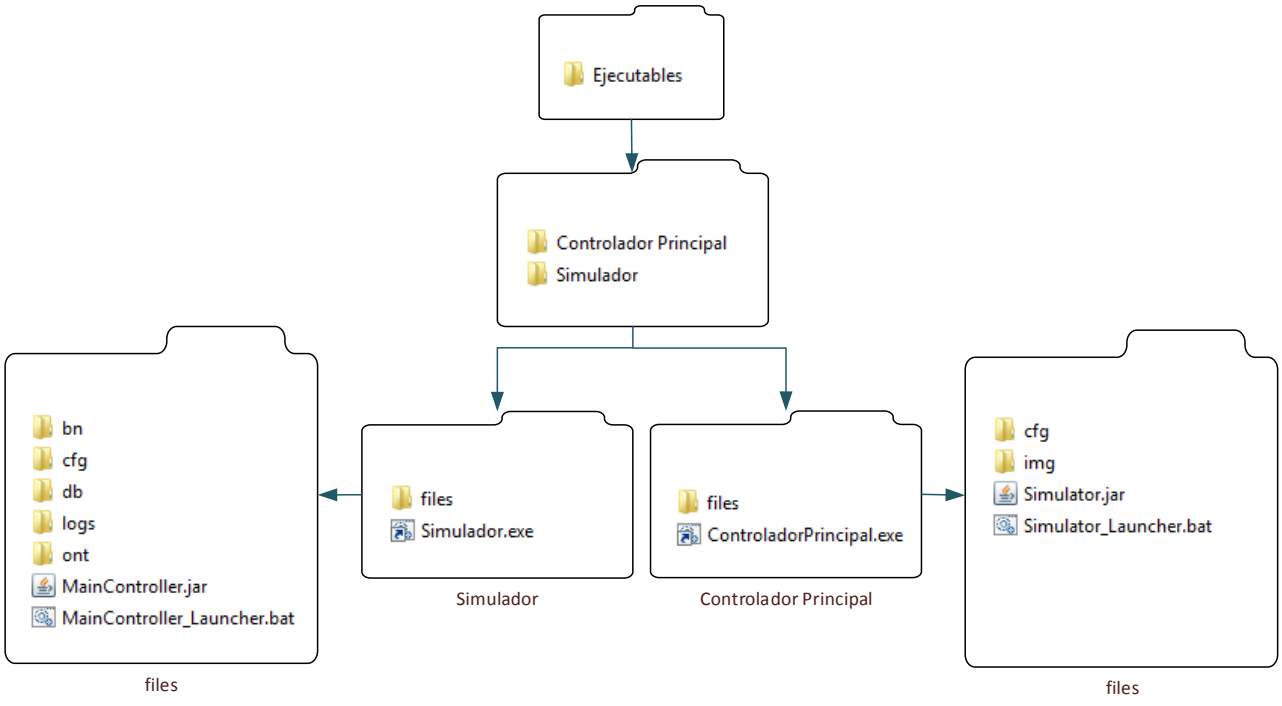


Ilustración 12. Estructura de ficheros ejecutables

Manteniendo esta estructura y ubicado en el directorio correspondiente, será suficiente con ejecutar los ficheros *.jar* desde consola mediante el siguiente comando:

- *java –jar <nombre_fichero>.jar*

Es importante añadir la opción *–jar* para forzar que la ventana de la consola se muestra en la ejecución, en lugar de correr en segundo plano.

Si el sistema va a lanzarse sobre Windows, el usuario puede hacer doble *click* en los ficheros *.bat* para ejecutar directamente el programa, sin necesidad de utilizar el comando mencionado. De la misma manera, los ficheros *.exe* ubicados en el directorio superior lanzarán estos sistemas sin necesidad de entrar en la estructura de ficheros por debajo del directorio *files*.

Al igual que sucedía antes, el usuario puede acceder al directorio *cfg* de esta estructura de ficheros (al mismo nivel que el fichero *.jar*) para modificar la configuración de cada sistema.

Finalmente, sólo quedaría pendiente arrancar la aplicación Android. Gracias a las facilidades que este SO proporciona en su manejo a los usuarios, será suficiente con *clicar* sobre el icono de la app (que debe haber sido instalada previamente). La nomenclatura de la aplicación queda como sigue:

- Nombre del fichero: *scah.app.apk*

  Este será el fichero que empaqueta la estructura del resto de ficheros de la *app*. El usuario debe cargarlo en su dispositivo móvil y posteriormente *clicar* sobre él. El propio SO Android se encargará de llevar a cabo el proceso de instalación sin más requisitos que la confirmación por parte del usuario.

- Nombre de la aplicación: *SCAH*

  El usuario verá en su dispositivo que se ha añadido un nuevo icono perteneciente a esta aplicación con nombre SCAH (siglas de *Semantic and Context Aware Home*, cuya significado en español corresponde a *Hogar Consciente de la Semántica y el Contexto*, en referencia a las principales características del sistema desarrollado).

Uso de la aplicación Android

Como últimas indicaciones que se proporcionarán en el manual, se expone a continuación una breve descripción de cómo puede el usuario hacer uso de las diferentes herramientas que proporciona la aplicación Android desarrollada.

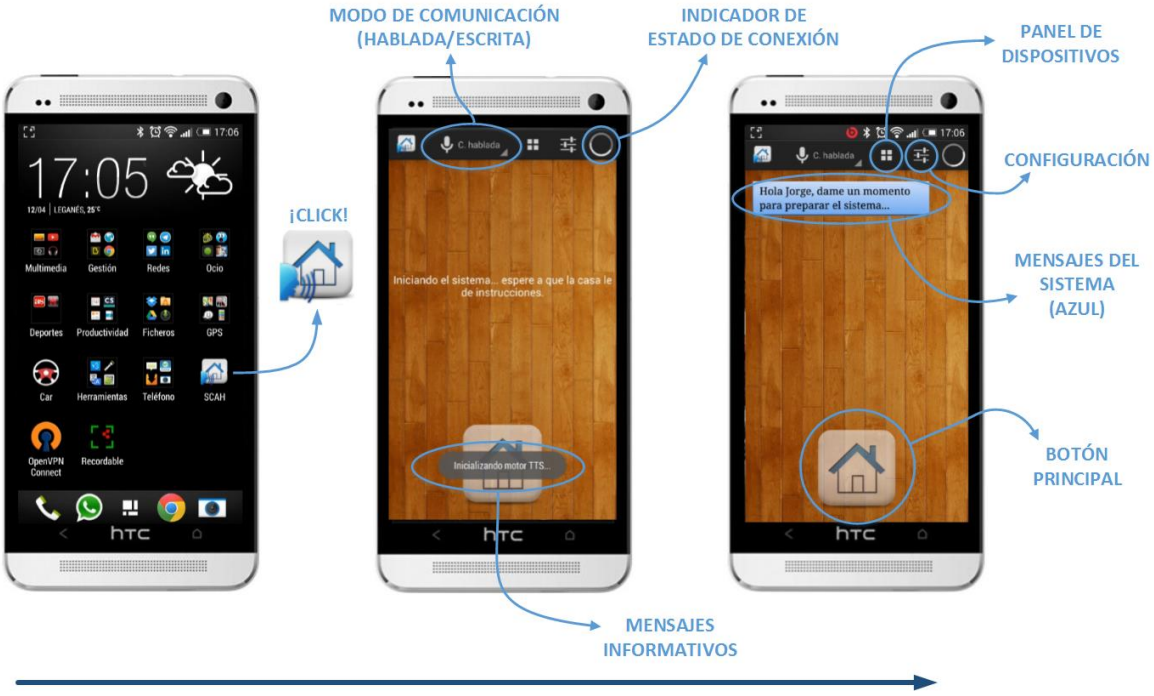


Ilustración 13. Uso de la app (actividad principal 1/3)

Al iniciar la app, el usuario verá las ventanas mostradas en la Ilustración 13. Se definen a continuación los distintos elementos:

- **Modo de comunicación**: el usuario puede elegir, mediante un desplegable, cómo desea que continúe la conversación.
    - Comunicación hablada: el sistema responderá a través de los altavoces de manera hablada (aunque también aparecerá el mensaje escrito en el bocadillo correspondiente de color azul). El usuario, por su parte, deberá dar las indicaciones también en voz alta, de manera que el sistema reconocerá lo que diga y lo mostrará también de manera escrita mediante un bocadillo (color verde).
    - Comunicación escrita: el sistema no dirá en voz alta sus mensajes, sólo aparecerán escritos en el bocadillo. El usuario, al intervenir en la conversación, introducirá su mensaje mediante una caja de texto emergente.
- **Mensajes informativos**: cuando la aplicación quiere informar al usuario de tareas no relacionadas con el conjunto del sistema (no orientadas a la red de dispositivos ni a la conexión con el resto de subsistemas), lo hará mediante mensajes *toast* como el de la ilustración.

Este tipo de mensajes muestran exclusivamente el estado de algunos elementos de la aplicación (si están listos o no para ser usados o si se encuentran en proceso de inicialización).

El resto de mensajes provendrán de la IA (como si fuese la propia casa quien estuviese conversando con el usuario e informándole de lo que sucede).

- **Indicador de estado de conexión**: este indicador tiene dos funciones.

  Por un lado, cuando muestra una barra de progreso circular como la de la ilustración está indicando al usuario que aún no se ha establecido correctamente la comunicación con el *Controlador Principal*.

  Por otro lado, cuando la comunicación está establecida, aparece un icono de refresco que, si es *clicado*, elimina la conexión actual y vuelve a reconectar. Esto puede ser útil si el usuario percibe que la conexión está siendo excesivamente lenta o inestable, de manera que puede ser necesario establecerla de nuevo.

- **Panel de dispositivos**: permite acceder a la actividad que muestra el panel de dispositivos y habilita la interacción directa con ellos. Se verá en detalle este panel en las siguientes ilustraciones.

- **Configuración**: permite acceder a la actividad que permite configurar el comportamiento de la aplicación. Se verá en detalle en las siguientes ilustraciones.

- **Mensajes del sistema**: como si de una conversación de chat se tratara, todos los mensajes del sistema (el hogar inteligente desde el punto de vista del usuario) estarán situados en bocadillos a la izquierda de la pantalla y el fondo de estos será azul.

  Los mensajes del usuario, como se verá en la siguiente ilustración, estarán situados a la derecha (siguiendo la convención establecida por la mayoría de las aplicaciones tipo chat, resultándole más familiar al usuario) y el fondo de los bocadillos será verde.

- **Botón principal**: el botón principal será el modo que tendrá el usuario de indicar que quiere participar en la conversación. Cuando se *clica*, en función del modo activo, el usuario deberá hablar o escribir su mensaje.

  Este botón, además, parpadea (permaneciendo inactivo) cuando el sistema está procesando órdenes pendientes, de manera que por un lado le indica al usuario que el sistema aún está *pensando* y por otro lado controla cuando este interviene en la conversación para limitar los posibles errores en la ejecución de órdenes.

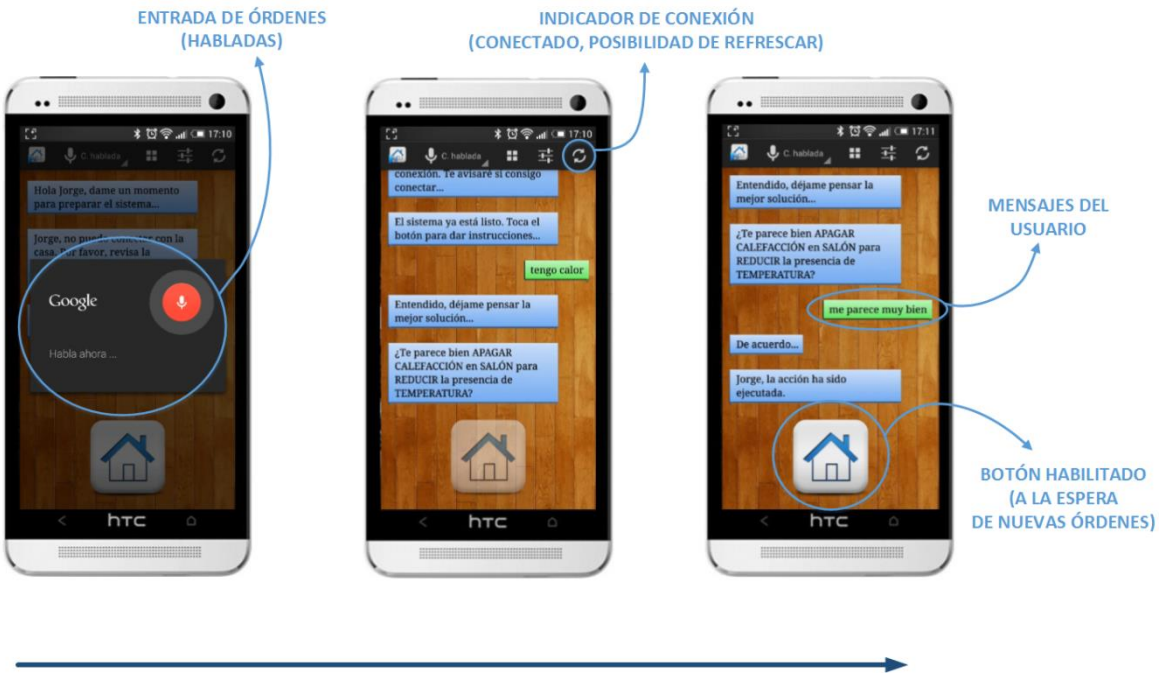


Ilustración 14. Uso de la app (actividad principal 2/3)

La Ilustración 14 muestra algunas de las características mencionadas anteriormente.

El usuario, al *clicar* el botón principal y encontrarse en modo *C. hablada* tendrá que dar la orden correspondiente cuando la ventana emergente se lo indique.

La conexión, que en este punto ya ha sido establecida, puede ser refrescada *clicando* el indicador que ahora, tal como se adelantó, está representado mediante el icono de refresco.

Entre la segunda y la tercera captura habría una nueva entrada de orden de manera hablada, pero se obvia por simplicidad.

Una vez la ejecución de la acción elegida ha sido llevada a cabo, la aplicación vuelve a habilitar el botón principal, que hasta este punto parpadeaba y no podía ser *clicado*, y espera nuevas órdenes.

Se debe tener en cuenta que en estado de espera (botón principal habilitado), la orden que indique el usuario se tomará como una orden a procesar, es decir, una nueva frase que hace referencia a una nueva acción que debe deducirse. Sin embargo, cuando el sistema proponga una acción a raíz de esta orden, aquello que indique el usuario se interpretará como una confirmación o una negación y no se procesará como una nueva orden de ninguna manera. Para introducir una nueva orden, el usuario deberá esperar a que el proceso actual concluya con la confirmación/negación correspondiente.

Téngase en cuenta que a la hora de confirmar o negar una acción propuesta, el sistema analiza la frase del usuario y busca una serie de palabras clave (o, al menos, una consecución de caracteres dentro de una palabra) que considera que en la gran mayoría de casos hace alusión a una confirmación, por lo que el usuario puede expresarla como mejor le parezca sin atender a palabras concretas. Toda frase que no contenga alguna de estas palabras clave será tomada como una negación. Si el usuario se encuentra ante una situación en la que el sistema toma como negación lo

que, en realidad, es una confirmación, siempre puede recurrir a expresar dicha confirmación de otra manera más sencilla: sí, vale, está bien, es correcto, perfecto, etc.

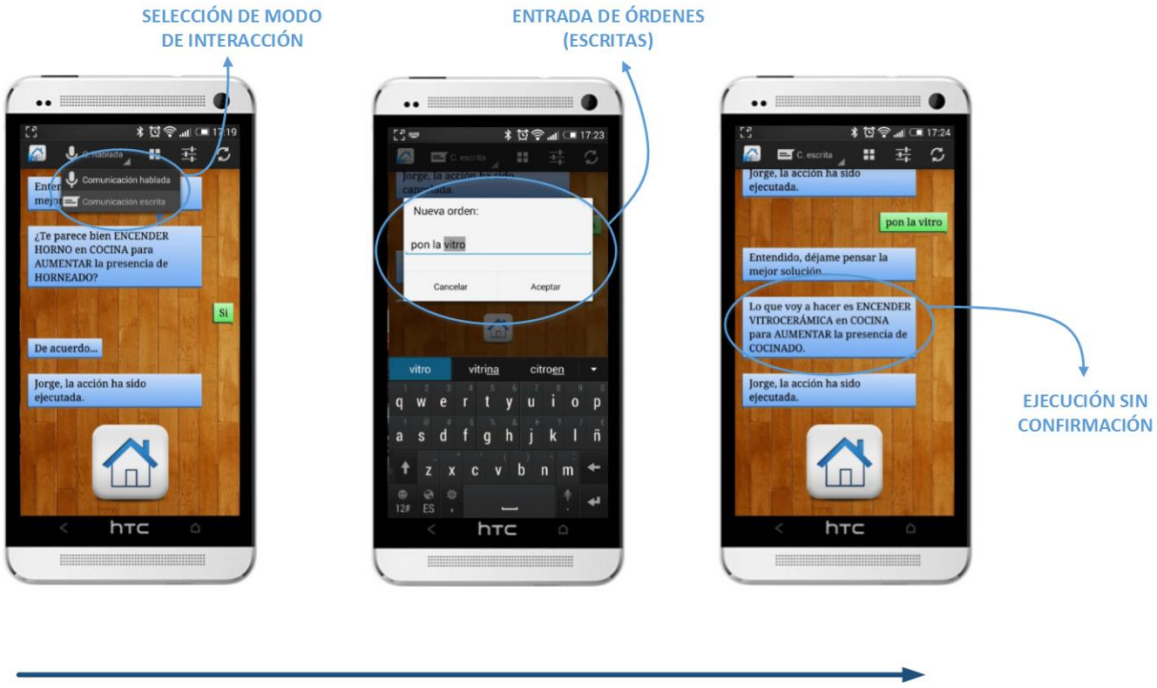


Ilustración 15. Uso de la app (actividad principal 3/3)

La Ilustración 15 muestra dos variantes de la configuración de la app. En la segunda captura vemos como, al cambiar de modo, el usuario introduce las órdenes de manera escrita mediante una caja de texto emergente.

La confirmación, aunque no se muestra por simplicidad, es otra ventana emergente con una caja de texto que indica cual es la acción que se propone. El usuario, en esa misma caja, debe responder tal como lo hacía de manera hablada con sus propias palabras.

La segunda variante de configuración que se muestra en esta ilustración es la ejecución sin confirmación. Si el usuario decide habilitar esta opción, toda acción que proponga el sistema será inmediatamente confirmada por la aplicación sin intervención del usuario. Esto podría resultar útil al usuario en hogares con pocos sensores en los que tenga un alto grado de certeza de que el sistema responderá bien a su petición.

Si el sistema ha errado en la conclusión y ha ejecutado la opción sin confirmación, el usuario siempre puede recurrir rápidamente al panel de dispositivos para cambiar el estado del dispositivo afectado. Por supuesto, también podría realizar este paso mediante una nueva orden, pero resultaría más lento y con menos garantías de éxito.

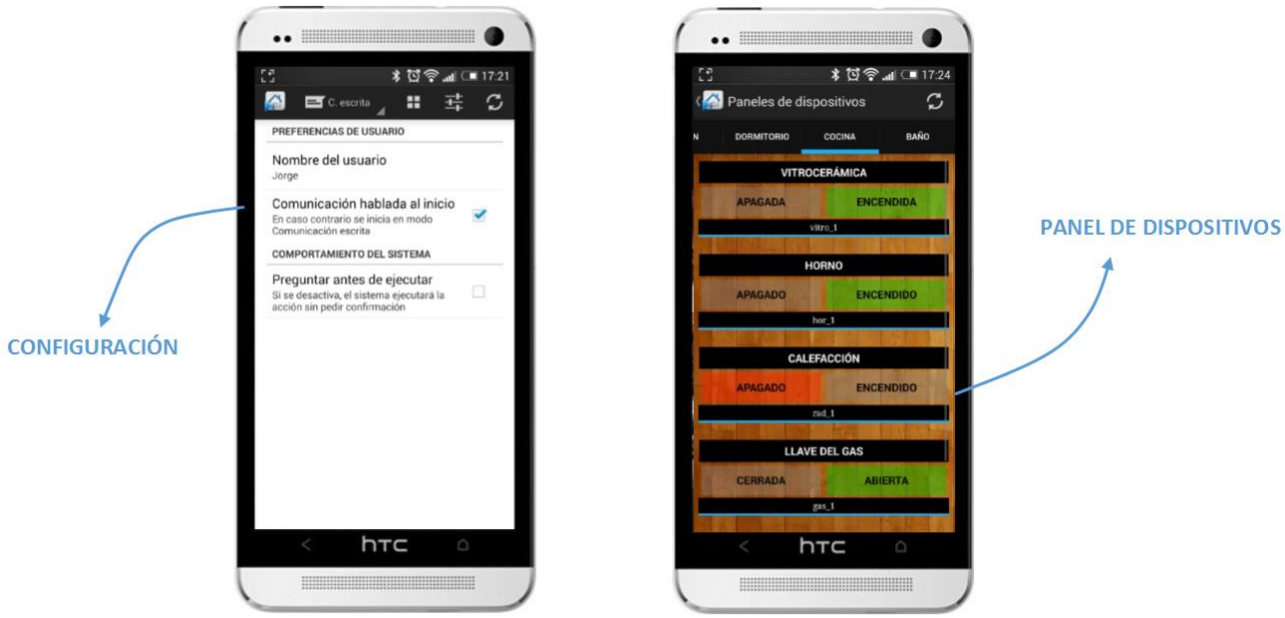


Ilustración 16. Uso de app (configuración y panel de dispositivos)

Por último, se exponen en la Ilustración 16 las dos vistas restantes que tendrá el usuario disponible en la app.

La primera captura, correspondiente a la configuración, permite al usuario configurar los siguientes parámetros:

- **Nombre del usuario**: nombre que utilizará el sistema para referirse al usuario (al instalar la app por primera vez el sistema se referirá al usuario como *humano*).
- **Comunicación hablada al inicio**: si se activa, al abrir la aplicación el modo activo será el de *C. hablada*, por el contrario, se arrancará con el modo *C. escrita*. Esta opción puede resultar útil para evitar que el sistema, nada más abrirse la app, empiece a hablar en voz alta (con las indicaciones de preparación que hemos visto previamente).
- **Preguntar antes de ejecutar**: si esta opción esta activada, el sistema siempre requerirá de confirmación antes de ejecutar la acción propuesta. Si se desactiva, tal como se ha visto previamente, la ejecución se lleva a cabo sin confirmación.

El usuario puede volver a la actividad principal mediante el botón de retroceso o pulsando de nuevo el botón de configuración.

La segunda captura muestra el panel de dispositivos. Este panel, muy similar al visto en el Simulador, permite navegar por las diferentes habitaciones y ejecutar acciones directamente sobre los dispositivos *clicando* el botón gris de algún dispositivo para convertir ese estado en el actual.

El usuario puede experimentar cierto retraso en la ejecución de una acción en el panel de dispositivos (unos pocos segundos en el peor de los casos) debido a que el panel solo muestra el cambio de estado una vez este se ha propagado por todo el sistema y ha sido ejecutado en el *Simulador*, el cual responde de vuelta. El usuario, una vez observe que el dispositivo cambia de estado, puede estar seguro de que la acción se ha ejecutado satisfactoriamente en la red.

Referencias bibliográficas

[1] C. R. Morales, R. V. Serrano y C. d. C. Lozano, Domótica e Inmótica: Viviendas y Edificios inteligentes, Madrid: RA-MA, 2006.

[2] Asociación Española de Domótica e Inmótica - CEDOM, «CEDOM,» [En línea]. Available: http://www.cedom.es/es. [Último acceso: 02 06 2014].

[3] Federación de la Energía de la Comunidad de Madrid - FENERCOM, «FENERCOM,» [En línea]. Available: http://www.fenercom.com/pdf/publicaciones/la-domotica-como-solucion-de-futuro-fenercom.pdf. [Último acceso: 03 06 2014].

[4] EL PAÍS, «Tecnología | EL PAÍS,» [En línea]. Available: http://tecnologia.elpais.com/tecnologia/2014/01/14/actualidad/1389688619_310380.html. [Último acceso: 25 06 2014].

[5] Red Eléctrica de España - REE, «REE,» [En línea]. Available: http://www.ree.es/es/red21/redes-inteligentes/que-son-las-smartgrid. [Último acceso: 06 06 2014].

[6] Red Española de Ciudades Inteligentes - RECI, «SmartCity: red respañola de ciudades inteligentes,» RECI; FUNDETEC; FEMP, [En línea]. Available: http://www.redciudadesinteligentes.es/. [Último acceso: 06 06 2014].

[7] ZigBee Alliance, «ZigBee Alliance,» [En línea]. Available: http://www.zigbee.org/. [Último acceso: 07 06 2014].

[8] KNX Association, «KNX Association,» [En línea]. Available: http://www.knx.org/es/. [Último acceso: 07 06 2014].

[9] Internet Engineering Task Force - IETF, «6lowPan,» [En línea]. Available: http://www.6lowpan.org/#. [Último acceso: 06 06 2014].

[10] Echelon Corp., «LonWorks,» [En línea]. Available: http://www.echelon.com/technology/lonworks/. [Último acceso: 06 06 2014].

[11] V. Torra, «Fundación General CSIC,» [En línea]. Available: http://www.fgcsic.es/lychnos/es_ES/articulos/inteligencia_artificial. [Último acceso: 07 06 2014].

[12] Apple Inc., «Apple - iOS7- Siri,» [En línea]. Available: http://www.apple.com/es/ios/siri/. [Último acceso: 08 06 2014].

[13] Microsoft, «Windows Phone 8.1 and cortana,» [En línea]. Available: http://www.windowsphone.com/en-us/features-8-1. [Último acceso: 08 06 2014].

[14] Universidad de Princeton, «Eugene Goostman,» [En línea]. Available: www.princetonai.com/bot. [Último acceso: 08 06 2014].

[15] N. F. Noy y D. L. McGuinness, *Desarrollo de Ontologías-101: Guía para crear tu primera ontología,* Standford: Standford University, 2005.

[16] W3C, «OWL Web Ontology Language Reference,» [En línea]. Available: http://www.w3.org/TR/owl-ref/. [Último acceso: 09 06 2014].

[17] W3C, «Guía Breve de Web Semántica,» [En línea]. Available: http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica. [Último acceso: 09 06 2014].

[18] Standford University, «Getting Started with Protégé-Frames,» [En línea]. Available: http://protege.stanford.edu/doc/tutorial/get_started/get-started.pdf. [Último acceso: 09 06 2014].

[19] Standford University, «Getting started with protégé-owl,» [En línea]. Available: http://protege.stanford.edu/doc/owl/getting-started.html. [Último acceso: 09 06 2014].

[20] UPnP Forum, «UPnP Forum - What is UPnP,» [En línea]. Available: http://upnp.org/about/what-is-upnp/. [Último acceso: 12 06 2014].

[21] UPnP Forum, «UPnP Forum - Device Architecture Documents,» [En línea]. Available: http://upnp.org/sdcps-and-certification/standards/device-architecture-documents/. [Último acceso: 12 06 2014].

[22] Microsoft corp., «TN Universal Plug and Play in Windows XP,» [En línea]. Available: http://technet.microsoft.com/en-us/library/bb457049.aspx. [Último acceso: 25 06 2014].

[23] Teleal, «4th Line - Cling,» [En línea]. Available: http://4thline.org/projects/cling/. [Último acceso: 12 06 2012].

[24] Teleal, «4th Line - Cling Core manual,» [En línea]. Available: http://4thline.org/projects/cling/core/manual/cling-core-manual.html. [Último acceso: 12 06 2014].

[25] Wikipedia, «Wkipedia - Motores de búsqueda,» [En línea]. Available: http://es.wikipedia.org/wiki/Motor_de_b%C3%BAsqueda. [Último acceso: 12 06 2014].

[26] -, «Search Commands,» [En línea]. Available: http://www.searchcommands.com/. [Último acceso: 12 06 2014].

[27] C. bielza y P. Larrañaga, «Departamento de Inteligencia Argificial - UPM,» [En línea]. Available: http://www.dia.fi.upm.es/~concha/Intro%20a%20RB_muia2.pdf. [Último acceso: 12 06 2014].

[28] University of Brasilia (UnB) ; George Mason University (GMU), «UnBBayes - Framework for probabilistic graph models,» [En línea]. Available: http://unbbayes.sourceforge.net/index.html. [Último acceso: 13 06 2014].

[29] Wikipedia, «Wikipedia - Treewidth,» [En línea]. Available: http://en.wikipedia.org/wiki/Treewidth. [Último acceso: 12 06 2014].

[30] S. M. (. o. Massachusetts), «School of Computer Science - University of Massachusetts - The Junction Tree Algorithm,» [En línea]. Available: http://www-anw.cs.umass.edu/~cs691t/SS02/lectures/week7.PDF. [Último acceso: 13 06 2014].

[31] Google Inc., «Android - Descubre android,» [En línea]. Available: http://www.android.com/meet-android/. [Último acceso: 13 06 2014].

[32] Google Inc., «Android Developer,» [En línea]. Available: https://developer.android.com/design/get-started/principles.html. [Último acceso: 2014 05 12].

[33] Google Inc., «Application Fundamentals | Android Developer,» [En línea]. Available: http://developer.android.com/guide/components/fundamentals.html. [Último acceso: 13 06 2014].

[34] C. J. Date, Introducción a los sistemas de bases de datos, Pearson Educación, 2001.

[35] solid IT, «DB-Engines Ranking,» [En línea]. Available: http://db-engines.com/en/ranking. [Último acceso: 13 06 2014].

[36] SQLite Consortium, «SQLite,» [En línea]. Available: http://www.sqlite.org/about.html. [Último acceso: 13 06 2014].

[37] Google Inc., «developer.android.com,» [En línea]. Available: http://developer.android.com/training/basics/supporting-devices/index.html. [Último acceso: 13 05 2014].

[38] R. A. C. Cuenca, *Android en el Hogar Digital usando UPnP y Redes LonWorks,* Madrid, 2013.

[39] M. GRAJA, M. Jaoua y L. H. Belguith, *Building ontologies to understand spoken tunisian dialect,* Túnez, 2011.

[40] S. Quarteroni, M. Dinarelli y G. Riccardi, *Ontology-based Grounding of Spoken Language Understanding,* Trento (Italia), 2009.

[41] Extreme Electronics Ltd., «openweathermap,» [En línea]. Available: http://openweathermap.org/API. [Último acceso: 02 06 2014].

[42] Oracle Corp., «Java 2DTM Graphics and Imaging,» [En línea]. Available: http://docs.oracle.com/javase/1.5.0/docs/guide/2d/index.html. [Último acceso: 05 06 2014].

[43] Microsoft, «Microsoft Excel: software de hoja de cálculo,» [En línea]. Available: http://office.microsoft.com/es-es/excel/. [Último acceso: 13 06 2014].

[44] E. H. Jung y A. Caldwell, «jexcelapi,» [En línea]. Available: http://jexcelapi.sourceforge.net/. [Último acceso: 05 06 2014].

[45] N. Dopico, «CasaFutura - Diatel - UPM,» [En línea]. Available: http://casafutura.diatel.upm.es/html/tecs/upnp.pdf. [Último acceso: 12 06 2014].