Desarrollo de una aplicación Android para el control de una vivienda inteligente mediante Reconocimiento de Voz

Máster Universitario en Ingeniería del Software, Métodos Formales y Sistemas de Información

Autor: ***Javier Rodríguez Llorente***
Directores: **Joan Fons Cors**
Fecha: ***Septiembre 2013***

Tabla de Contenidos:

TABLA DE FIGURAS

1 Introducción

Cada vez más los dispositivos móviles están presentes en nuestra vida para hacer por o con nosotros actividades que nunca pensábamos que un dispositivo de estas características podría realizar. Los avances hacen que estos dispositivos estén equipados con una tecnología cada vez más potente, además de conseguir que los costes sean cada vez más reducidos.

Estos avances están estrechamente ligados a la evolución de internet y a las comunicaciones móviles, en las que actualmente se ha habilitado la red 4G, abriéndonos el mundo desde cualquier sitio en que nos encontremos.

Las investigaciones en estos campos nos dirigen hacia una nueva era, "Internet of Things", llevando más allá la filosofía de dispositivo móvil, y haciendo que cualquier dispositivo electrónico, lavadora, televisión, luces, calefacción, entre otros, estén equipados con tecnología que permita su control desde dispositivos externos a la vez que están conectados a Internet. **Figura 1.1 Internet of Things.**

Figura 1.1 Internet of Things

Cada día es más común la presencia de la domótica y de sistemas de control de entorno tanto en los hogares como en los edificios públicos. Estos dispositivos permiten realizar acciones dentro del hogar de una forma más cómoda, al no tener

que activar directamente ningún mecanismo. También son útiles para mejorar la seguridad en edificios, pues pueden combinarse sensores y actuadores para programar avisos o acciones de protección. El desarrollo de los sistemas de control de entorno puede mejorar enormemente la calidad de vida y facilitar los quehaceres diarios de todos.

No obstante, estos desarrollos tecnológicos, junto con otros no menos importantes (como el entretenimiento digital, los electrodomésticos inteligentes o las telecomunicaciones) pueden suponer una barrera que aumente la distancia social entre las personas que sufren alguna discapacidad y las que no. Sin embargo, si este desarrollo tecnológico se realiza siguiendo criterios de accesibilidad y diseño universal, estas nuevas tecnologías pueden ofrecer grandes oportunidades para personas con discapacidades tanto físicas como intelectuales.

En cambio, los dispositivos móviles de última generación si están preparados para permitir formas de interacción no convencionales y que permiten el acceso a los mismos a las personas discapacitadas. Suelen incluir lectores de pantalla, ofrecen múltiples modos de interacción, personalizaciones en el modo de visualización y varias combinaciones posibles de realimentación ante las acciones del usuario.

1.1 Motivación

Tras finalizar el Máster Universitario en Ingeniería del Software, Métodos Formales y Sistemas de Información de la Universidad Politécnica de Valencia surgió la oportunidad de poder contribuir en un proyecto ambicioso en cuanto a tamaño e importancia, realizando el desarrollo del presente trabajo, el cual está enfocado al control mediante reconocimiento de voz de un sistema domótico.

La motivación no es otra que aprender nuevas tecnologías como las que engloba este proyecto, Android, REST, Reconocimiento de voz y además, poner en práctica lo aprendido en el Máster Universitario en Ingeniería del Software, Métodos Formales y Sistemas de Información.

1.2 Planteamiento del problema

Hoy en día los sistemas domóticos capaces de interactuar completamente mediante reconocimiento de voz no están muy extendidos, son todavía muy “jóvenes” y es posible que tarden algún tiempo en madurar lo suficiente para presentarse como una

alternativa de ayuda real para personas discapacitadas o de movilidad reducida que solo disponen de esta interfaz para controlar estos sistemas.

Además las investigaciones en estas áreas están encaminadas hacia formas de interacción más naturales, estando el reconocimiento de voz en el punto de mira de estos desarrollos. Dado que todavía no está implantado como una solución estable y queda mucho camino por recorrer, la presente Tesina de Máster pretende ofrecer una solución a este tipo de desarrollos.

1.3 Objetivos del trabajo

El objetivo de la presente tesina es desarrollar una infraestructura software, para poder gestionar una vivienda con elementos domóticos a través de la voz, por medio del uso de dispositivos móviles. Mediante este trabajo se pretende incorporar un nuevo mecanismo de interacción, "la voz", a la hora de manejar aplicaciones, sobre al ámbito de la domótica. Este trabajo permitirá una forma de controlar estos sistemas de una forma más natural, fácil y cómoda, y además ayudar a las personas con poca movilidad o con problemas de visión, el poder manejar una vivienda domótica más fácilmente.

La aplicación estará diseñada para dispositivos móviles, ya que estos son dispositivos que acompañan al usuario en todo momento. Lo que nos permite poder realizar acciones de control de entorno en cualquier ubicación. Estos dispositivos nos ofrecen una gran variedad de formas de interacción mediante pantallas táctiles, gestos, acelerómetros o control por voz, que los mandos tradicionales no pueden aportar.

Las nuevas tecnologías de voz son muy aplicables a casas inteligentes. Los comandos vocales son una manera más natural de comunicarnos con sistemas automáticos [1].

1.4 Contexto de la Tesina de Máster

Este trabajo de Fin de Máster se sitúa dentro de un marco de desarrollo de mayor envergadura dirigido por el Grupo de Inteligencia Ambiental del centro de investigación en métodos de producción de software (ProS) de la Universidad Politécnica de Valencia (UPV). Este grupo lleva trabajando desde hace más de una década en sistemas pervasivos, desarrollando métodos y aplicándolos en el contexto del hogar. La aplicación de estos métodos permite una utilización más eficiente de los

recursos energéticos disponibles, automatización de tareas, despliegue de servicios y aplicaciones inteligentes.

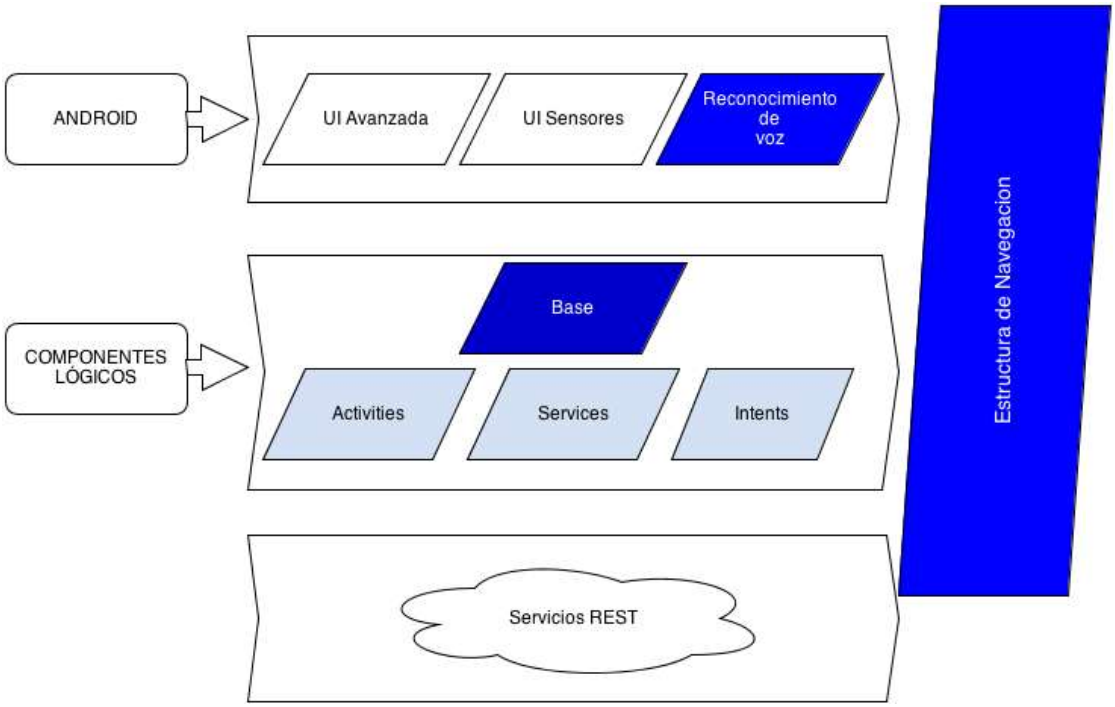


Figura 1.2 Estructura proyecto

Este trabajo de Fin de Máster ha consistido en el desarrollo de la estructura que se puede ver en la **Figura 1.2 Estructura proyecto**, con un color azul más intenso en los módulos en los que ha habido un desarrollo más importante y un color más claro en las que el desarrollo no ha sido tan específico.

La ***Estructura de Navegación*** otorga al sistema la funcionalidad necesaria para poder navegar entre las estancias de la vivienda o entre los dispositivos, saber nuestra ubicación actual o conocer el elemento domótico que queremos modificar.

Además esta estructura de navegación, permite escalar la aplicación fácilmente en lo que a funcionalidad se refiere, ya que es la encargada de toda la lógica de navegación.

Dentro de componentes lógicos vemos un componente llamado ***Base*** el cual se encarga de la funcionalidad tanto del reconocimiento como del motor de voz, evitando la duplicidad de código con los otros componentes, favoreciendo así la reutilización de código.

1.5 Estructura del documento

La Tesina está estructurada en seis grandes capítulos, incluido este capítulo introductorio. Además de contener dos anexos para mostrar en detalle la funcionalidad.

- *Capitulo 2* - Contexto tecnológico

Explica la situación del entorno que rodea el ámbito de este trabajo. Además sitúa el proyecto, la domótica, el desarrollo móvil, la interacción por voz y aplicaciones del mismo ámbito en un ambiente actual.

- *Capitulo 3* - Presentación de la propuesta

Presenta el planteamiento del trabajo a realizar, ofreciendo una vista tanto a nivel del problema como de la solución propuesta.

- *Capitulo 4* - Desarrollo de la propuesta

En este capítulo se explica con detalle el trabajo realizado. Se profundizará en cada una de las partes en las que se ha dividido el trabajo para alcanzar los objetivos finales.

- *Capitulo 5* - Caso de uso

A partir de un caso práctico se expondrá un ejemplo real de como un usuario puede usar y controlar la aplicación, para mostrar la funcionalidad del sistema desarrollado.

- *Capitulo* 6 - Conclusiones y trabajo futuro

En este capítulo se comentan las conclusiones alcanzadas y las decisiones tomadas durante el desarrollo. Además se presentan un par de propuestas para mejorar y seguir evolucionando la aplicación.

- *Anexo A. Api servicio REST*

En este anexo se muestra la representación de la interfaz de comunicación entre nuestro componente software. Se trata de un conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a determinados servicios.

- *Anexo B. Utilización servicio REST*

En este anexo se muestra como utilizar el componente software explicado en el Anexo A mediante un caso que se ha implementado en la aplicación desarrollada.

2 Contexto tecnológico

Este capítulo pretende introducir al lector en el contexto tecnológico en el que se desarrolla este Trabajo Fin de Máster. Primero, se presenta el ámbito de la aplicación compuesto por la domótica, el desarrollo móvil y como estos pueden interactuar por medio de la voz. En segundo lugar se exponen algunos proyectos en el desarrollo de sistemas domóticos.

2.1 Ámbito de la aplicación

Para presentar el ámbito de la aplicación primero se muestra la situación de estas tecnologías mediante los informes anuales de la reconocida firma de análisis Gartner.

En **Figura 2.1 Gartner's 2010 Hype Cycle**, podemos observar que tecnologías como “Speech to Speech translation”, “Internet TV”, se encuentran en fase de expectación y “Speech Recognition” en fase pendiente de despliegue.

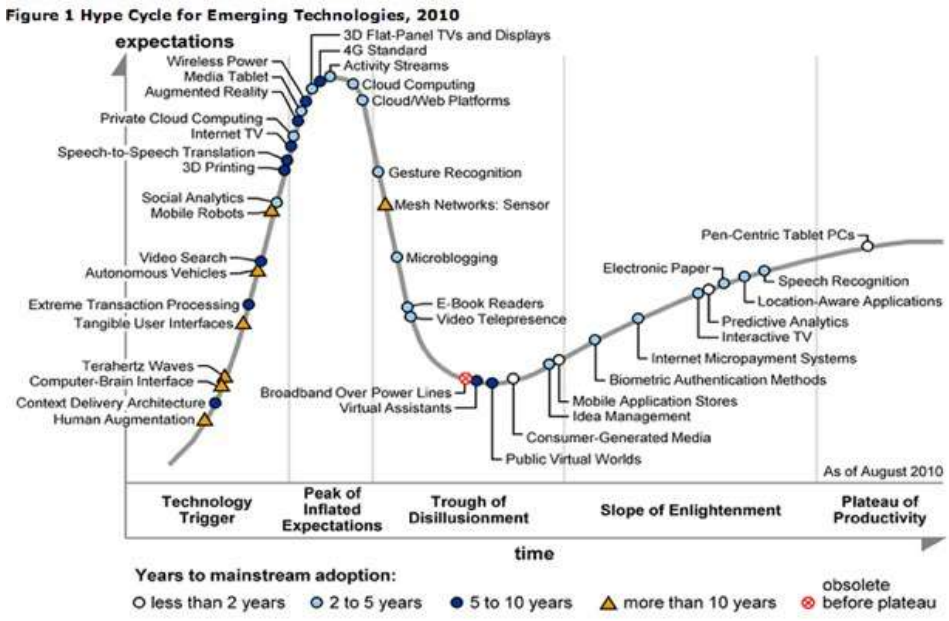


Figura 2.1 Gartner's 2010 Hype Cycle

En la **Figura 2.2 Gartner's 2012 Hype Cycle**, la tecnología más significativa que aparece es “Internet of Things” como una previsión de 10 años, mientras “Speech Recognition” sigue en un estado que aunque se utiliza, no termina de explotar.

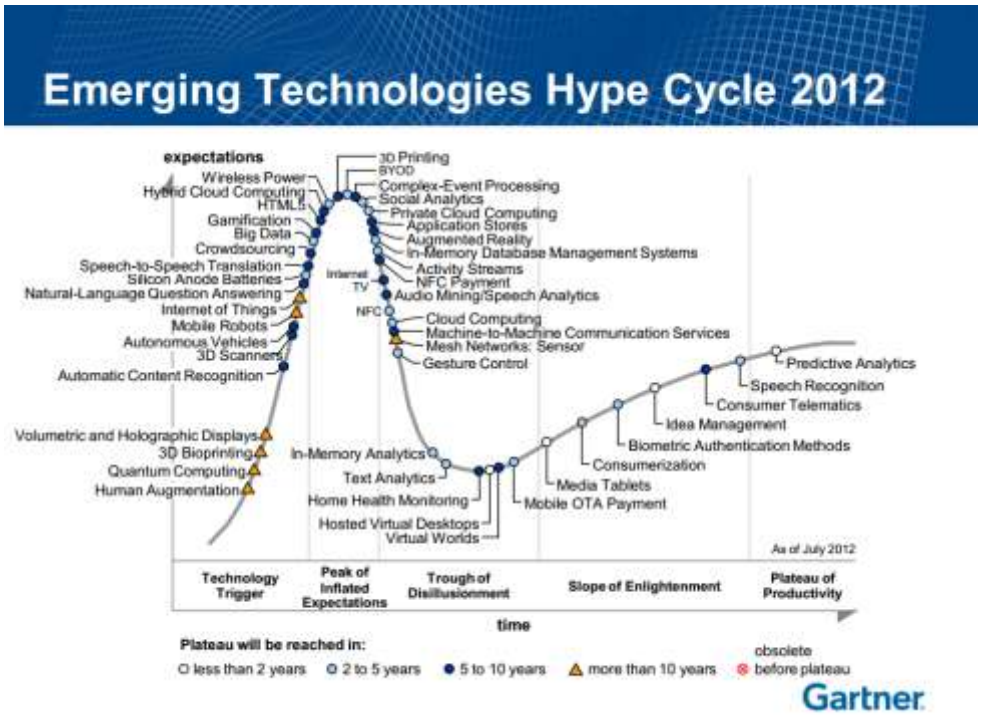


Figura 2.2 Gartner's 2012 Hype Cycle

Como conclusión a estas gráficas podemos obtener que las tecnologías con las que trabajamos en el presente proyecto no son tecnologías asentadas y que además, son tecnologías de las que queda mucho trabajo e investigaciones por desarrollar en universidades de todo el mundo.

2.1.1 Domótica

La palabra domótica proviene del latín “*domus*” añadiéndole el final de la palabra “*informática*” en griego y según explica la propia Real Academia Española de la Lengua

[2], es el conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda.

Los avances tecnológicos han ido apareciendo en nuestras vidas de una manera gradual, hasta el punto de acostumbrarnos a que estos formen parte de nuestro día a día. Actualmente no concebiríamos nuestras acciones diarias sin elementos como los teléfonos móviles o las tarjetas de crédito por citar algunos ejemplos. La rápida evolución tecnológica de la electrónica e informática ha inundado nuestro entorno con televisores, teléfonos móviles, equipos informáticos, etc.

Actualmente la relación con las nuevas tecnologías es constante por lo que es de esperar que si ya forman parte de otros muchos ámbitos de nuestra vida lo acaben haciendo en nuestra propia casa. Así es como nacen los conceptos de vivienda domótica y vivienda inteligente.

Figura 2.3 Ámbitos de la domótica

Estos conceptos están impulsados por tres factores principales: evolución tecnológica, cambios sociales y oportunidades de negocio. Es bien palpable que la evolución tecnológica y los cambios sociales están relacionados entre sí, es decir, son a la vez consecuencia y motor de la evolución de la sociedad.

Un sistema domótico es aquel sistema informático encargado de proporcionar servicios en el ámbito del hogar o, en general, los edificios [3]. Estos sistemas son capaces de recoger información proveniente de unos sensores o entradas, procesarla y emitir órdenes a unos actuadores o salidas. El sistema puede acceder a redes exteriores de comunicación o información.

La domótica permite dar respuesta a los requerimientos que plantean estos cambios sociales y las nuevas tendencias de nuestra forma de vida, facilitando el diseño de casas y hogares más humanos, más personales, polifuncionales y flexibles.

En cambio, comenzaríamos a hablar de vivienda inteligente en la medida en la que los dispositivos de la vivienda se integran bajo un mismo sistema, colaborando entre sí con el fin de automatizar, o al menos dar soporte informático a la realización de tareas domésticas y al control de electrodomésticos y equipamiento electrónico [4].

Esta diferencia podemos ilustrarla con un ejemplo: en una casa domótica podemos tener un riego automatizado programable, pero en una vivienda inteligente podríamos disponer, además del riego programable, de un sensor de humedad colocado en el jardín que avisará al sistema central para que éste anule el riego cuando llueva. Esto supone una gran ventaja para el propietario de la vivienda no solo en términos de confort sino también de ahorro energético y económico.

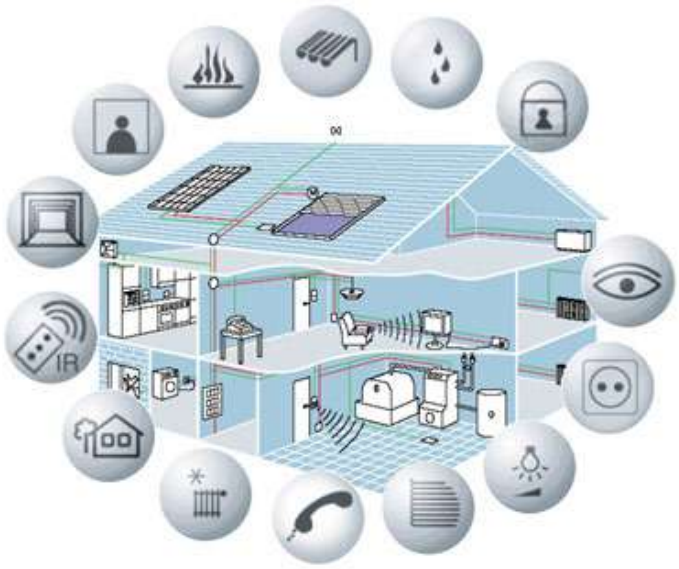


Figura 2.4 Vivienda domótica/inteligente

De un modo general una vivienda inteligente debe disponer de una red de comunicación que permita interconectar aquellos equipos que permiten obtener información acerca del entorno doméstico con aquellos que, basándose en esta información actúan sobre dicho entorno.

2.1.1.1 Estándar EIB/KNX

A finales de 1992, varias empresas españolas plantean la idea de crear una Asociación de empresas que se dedican a fabricar productos relacionados con la tecnología EIB. A comienzos de 1993 se formaliza oficialmente la Asociación EIBA España. Una de las primeras tareas es la cuantificación del mercado nacional, lo que debido a la escasez de datos se presenta difícil. Otra prioridad es la estrecha colaboración con el órgano central EIBA internacional.

En 1999, las Asociaciones internacionales de los tres sistemas domóticos líderes europeos EIB, BatiBus y EHS deciden unir fuerzas creando la KNX Association, que mantiene su sede en Bruselas.

Así, por ser el campo de directa aplicación de este estudio, se dedicará a continuación un apartado para tratar el sistema EIB/KNX de forma extensa.

El Bus de Instalación Europeo (EIB, European Installation Bus) surgió con la idea de introducir en el mercado un sistema unificado para la gestión de edificios, creado por el consorcio europeo EIBA (European Installation Bus Association) en 1990 por más de setenta compañías (ABB, Siemens, Jung...). [5]

En la actualidad la asociación tiene más de cien miembros, existiendo unas veinte empresas que suministran productos, siendo las más importantes; Siemens, ABB, Temper, Grasslin y Niessen. También existen miembros científicos que colaboran en el desarrollo de actividades de I+D, especialmente universidades y centros de investigación. Las funciones de la asociación son básicamente el soporte para la preparación de normas unificadas y la definición de las pruebas y requisitos de homologación que garanticen la calidad y compatibilidad de los productos. En la actualidad, EIBA se ha integrado dentro de la asociación Konnex (KNX) para la convergencia de sistemas europeos.

Figura 2.5 Estándar KNX

Se trata, además, de un sistema abierto sobre las mismas premisas que otros sistemas de comunicación como los buses de campo abiertos. Tanto las especificaciones del protocolo como los procedimientos de verificación y certificación están disponibles, así como los componentes críticos del sistema (microprocesadores específicos con la pila del protocolo y electrónica de acoplamiento al bus).

KNX/EIB está recogido en normas europeas e internacionales (EN 50090 e ISO/IEC 14543), y es la tecnología domótica abierta más instalada en España y a nivel europeo.

2.1.1.1.1 Aspectos tecnológicos.

El KNX/EIB es un sistema distribuido (no requiere de un controlador central de la instalación), en el que todos los dispositivos que se conectan al bus de comunicación de datos tienen su propio microprocesador y electrónica de acceso al medio.

En una red EIB podemos encontrar cuatro tipos de componentes: módulos de alimentación de la red, acopladores de línea para interconectar diferentes segmentos de red, y elementos sensores y actuadores.

2.1.1.1.2 Ventajas derivadas de la utilización de EIB/KNX

Se pueden destacar las siguientes ventajas:

- Gran cantidad de fabricantes desarrollan componentes domóticos que utilizan esta tecnología.

- Estándar internacional que garantiza su continuidad en el futuro:

    - ISO/IEC: Aprobó la tecnología KNX como el Estándar Internacional ISO/IEC14543-3 en 2006.

    - CENELEC: Aprobó la tecnología KNX como el Estándar Europeo EN 50090 en 2003.

- Interoperabilidad & Interworking de productos: El proceso de certificación KNX asegura que funcionarán y se comunicarán diferentes productos de diferentes fabricantes usados en diferentes aplicaciones. Esto asegura un alto grado de flexibilidad en la extensión y modificaciones de las instalaciones. Laboratorios neutrales (terceras compañías) son quienes analizan la conformidad del producto. KNX es el único estándar para el control de casas y edificios que lleva a cabo un plan de certificación para productos, centros de formación (instituciones profesionales y privadas) e incluso personas (electricistas, proyectistas).

- Software unificado: Mediante una única herramienta se pueden diseñar y configurar todos los productos certificados KNX. Dicha herramienta es además independiente del fabricante: el integrador de sistemas podrá combinar los productos de varios fabricantes en una instalación.

- Sistema completo: KNX puede ser usado para el control de todas las posibles funciones/aplicaciones en casas y edificios desde iluminación, contraventanas, control de seguridad y alarmas, calefacción, ventilación, aire acondicionado, control de agua y dirección de energía, medición, hasta aplicaciones para el hogar, audio, etc.

- Adaptación a cualquier tipo de construcción: KNX puede ser usado tanto en nuevas construcciones como en las ya existentes. Las instalaciones KNX pueden ser fácilmente extendidas y adaptadas a las nuevas necesidades, con una pequeña inversión de tiempo y dinero. Igualmente KNX puede ser instalado tanto en pequeñas casas como en grandes edificios.

2.1.2 Desarrollo móvil

En los últimos años se ha podido observar la aparición de nuevos tipos de dispositivos móviles de pequeño y mediano tamaño, ofreciendo mecanismos de interacción novedosos. Por ejemplo, las interfaces multi-táctiles se han consolidado como una solución más natural en el ámbito móvil, impulsando la penetración de estos dispositivos [6].

El desarrollo móvil se ha convertido en el entorno objetivo en el que construir nuestras aplicaciones y es que los Smartphone y las Tablets han dejado atrás a los PCs.

La principal ventaja es el paso de lo estático a lo móvil, los Smartphone y las Tablets ofrecen la posibilidad de poder conectarnos y comunicarnos con los demás a través de la red móvil desde lugares impensables hace algunos años. Por este motivo, los aspectos más valorados por los usuarios con respecto a la formalidad inherente del PC incluso de un ordenador portátil, es la inmediatez y la espontaneidad de los dispositivos actuales.

Figura 2.6 Control domótico desde dispositivo móvil

Las características que engloban a estos dispositivos han hecho que rápidamente se abran líneas de investigación, fusionando el desarrollo móvil y la domótica. La posibilidad de conocer el estado de todos los elementos electrónicos del hogar, video

vigilancia en tiempo real, control remoto y como característica innata la movilidad han propulsado que estos dos paradigmas estén fuertemente ligados.

Los comienzos de la domótica situaban un panel de control central en algún sitio accesible para controlar nuestros dispositivos, después aparecieron las pasarelas que ofrecían una interfaz a través del televisor, todas ellas obligándonos a estar en la propia vivienda. Ahora mediante aplicaciones móviles el dispositivo móvil se convierte en ese panel avanzado de control

2.1.3 Interacción por voz

El Reconocimiento Automático de Voz es un campo de investigación de creciente relevancia que día a día se gana más adeptos. El desarrollo de mejores algoritmos y de modelados más precisos, junto con la aparición de sistemas informáticos más potentes y asequibles, posibilita la integración de los sistemas de dialogo hombre-máquina a través de la voz en numerosos ámbitos de la sociedad actual. Estos sistemas de dialogo permiten el acceso a una gran cantidad de información a través de una forma de comunicación tan natural como es el habla, facilitando un elevado número de servicios interactivos utilizando el teléfono, la televisión o el ordenador como elementos de acceso.

Podríamos afirmar que, genéricamente, el principal objetivo que el Reconocimiento de Habla persigue es proporcionar una "apropiada" interacción hombre-máquina a través de órdenes habladas.

Figura 2.7 Comunicación hombre-máquina

Las principales características que diferencian a los sistemas basados en Reconocimiento del Habla frente a otras alternativas son: la naturalidad que supone utilizar el habla en las operaciones de comando y control, y la precisión y robustez en la comunicación para diferentes usuarios y diferentes entornos.

El estado actual de la investigación en Reconocimiento del Habla nos muestra excelentes resultados de sistemas trabajando en entornos controlados de laboratorio. Sin embargo, una aplicación real de esta tecnología exige un funcionamiento en el mundo real donde el grado de dificultad de los problemas es un orden de magnitud mayor.

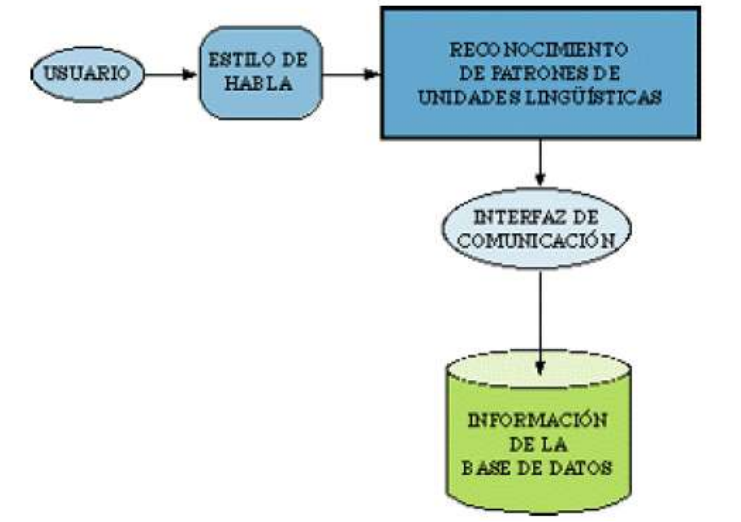


Figura 2.8 Modelo genérico de comunicación para Reconocimiento de Habla

Bajo esa premisa de buscar una aplicación real, el modelo genérico de comunicación que el Reconocimiento del Habla propone para el diálogo hombre-máquina puede representarse, de forma simplificada, tal y como muestra el diagrama de la **Figura 2.8 Modelo genérico de comunicación para Reconocimiento de Habla,** para un caso de acceso a una base de datos [7].

2.1.4 Buscador de Google mediante Reconocimiento de Voz

La búsqueda de información en la web a través del móvil es un área de creciente interés. El acceso a internet a través de dispositivos inteligentes ha derivado en un

aumento de las ventas de éstos en todo el mundo y la mayoría de los modelos de Smartphone ofrecen una experiencia de navegación web que rivaliza con los ordenadores tradicionales. Los usuarios usan cada vez más sus dispositivos para hacer búsquedas en la web, por lo que se están conduciendo los esfuerzos para mejorar la usabilidad de los dispositivos sobre las búsquedas en la web. Y aunque la usabilidad de los dispositivos móviles ha mejorado, teclear las búsquedas puede llegar a ser engorroso, provocar errores tipográficos incluso peligrosos en algunos escenarios. [8]

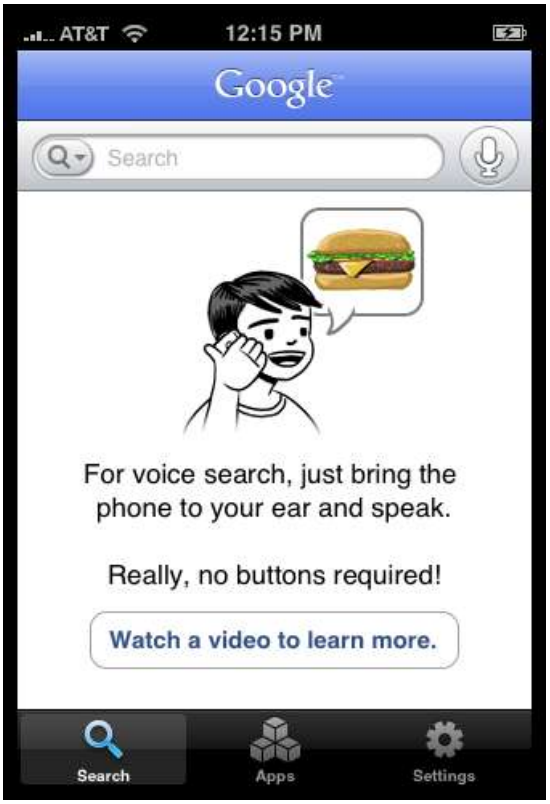


Figura 2.9 Buscador Google por Voz para iPhone

El objetivo del Buscador por Voz de Google es reconocer cualquier consulta dictada mediante la voz, es decir, que sea capaz de controlar cualquier consulta que el buscador de Google pueda manejar. Este hecho es considerado uno de los retos del reconocimiento de voz, ya que el vocabulario y la dificultad de las consultas son demasiado largos.

En la **Figura 2.10 Diagrama básico de reconocimiento de voz** se representa la arquitectura básica del sistema de reconocimiento de voz de Google.

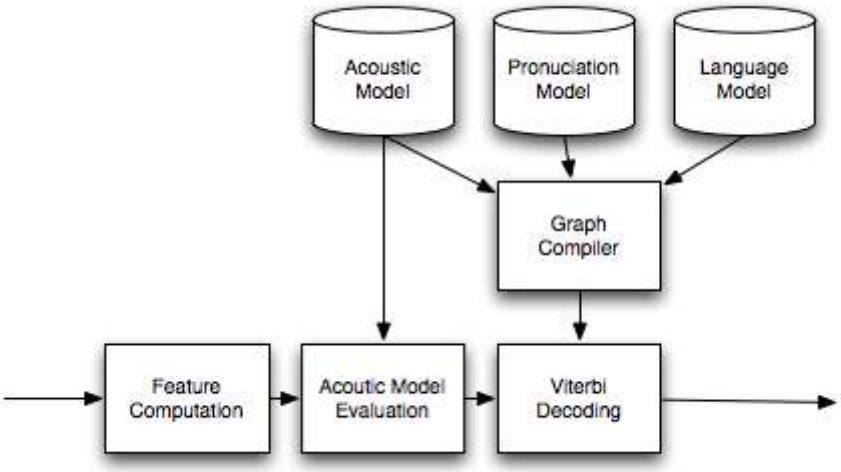


Figura 2.10 Diagrama básico de reconocimiento de voz

Para evaluar la calidad del sistema de reconocimiento de voz, Google utiliza unas métricas, que son las que marcan el camino de las investigaciones.

Las métricas que usan son entre otras:

- Word Error Rate(WER)

Esta métrica compara la palabra introducida por el sistema de reconocimiento de voz y las que el usuario realmente pronunció.

$$WER = \frac{\text{Number of Substitution + Insertions + Deletions}}{\text{Total number of words}}$$

- Semantic Quality (WebScore)

Para el sistema de Google de reconocimiento de voz, los errores individuales no afectan necesariamente al resultado final. Por lo que es métrica realiza un seguimiento de la calidad a nivel semántico.

$$WebScore = \frac{\text{Number of correct search results}}{\text{Total number of spoken queries}}$$

Por otro lado, los modelos acústicos proporcionan una estimación de la probabilidad de las características observadas en un marco de contexto fonético de una oración dada. Estas características se relacionan generalmente con las mediciones espectrales en un intervalo de tiempo. Un ejemplo de estos modelos se puede estudiar sobre el Modelo oculto de Márkov[1], utilizado por los algoritmos de Google, ya que este modelo es especialmente aplicable al reconocimiento del habla. [8]

En los últimos años el modelado del lenguaje ha experimentado cambios enfocados a la escalabilidad. El responsable de estos cambios es la capacidad de aumentar significativamente las cantidades de datos de entrenamiento que son relevantes para los problemas de reconocimiento de voz automático.

2.1.5 Manejo de una vivienda domótica mediante el uso de la voz

Los entornos inteligentes enfatizan en interfaces más amigables para el usuario, servicios más eficientes, más control por parte del usuario y soporte para interacciones humanas, como el reconocimiento de voz.

En los últimos años, hemos escuchado en numerosas ocasiones estudios que prevén en un futuro no muy lejano, poder controlar nuestros hogares con la voz a través de la domótica. Esta técnica lleva mucho tiempo desarrollándose y hasta ahora, los desarrollos de estas soluciones de voz han tenido la necesidad de un algoritmo complejo que reconozca nuestro lenguaje.

En definitiva, el futuro de la domótica se está definiendo, pero con la introducción de estos sistemas en un teléfono inteligente, podemos hablar de un auténtico impulso para esta ingeniería que representa la domótica.

Este tipo de software puede hacer que nuestra voz se convierta en nuestro propio sistema de entrada y salida a un mundo más cómodo y sencillo de manejar.

[1] http://es.wikipedia.org/wiki/Modelo_oculto_de_M%C3%A1rkov

2.2 Aplicaciones relacionadas

A continuación se comentan algunas de las aplicaciones relacionas con el ámbito de trabajo de este proyecto.

2.2.1 Domóticas

En el ámbito de la domótica las aplicaciones relacionadas que vamos a comentar son:

- Open source for internet of things.
- Sistemas de control de entorno de BjAdaptaciones.
- Kora.
- Insteon for Hub.

2.2.1.1 OPEN REMOTE

OpenRemote [9] es un proyecto libre que desarrolla una distribución GNU/Linux enfocada al control de dispositivos domóticos en el hogar, ofreciendo una capa intermedia que provee acceso uniforme a alto nivel a los dispositivos domóticos de distintos fabricantes y estándares. También comercializan soluciones hardware con el software preinstalado y configurado para utilizar en multitud de ámbitos. Es un proyecto muy grande, potente y versátil, además de libre. No obstante, la instalación manual puede ser muy compleja, el sistema no se considera estable todavía, no hay casi documentación para el código y apenas hay manuales para configurar el sistema. Las aplicaciones para acceder al servidor domótico mediante dispositivos móviles que los autores del proyecto proveen están diseña das para usuarios con grandes conocimientos de domótica, pues requieren configuración desde el propio dispositivo. Hay aplicaciones para iPhone y Android ambas gratuitas y libres,

OpenRemote admite diversas plataformas, ARM9, ARM11, Cortex-A8, Cortex-A9, Intel Atom.

Se puede usar para desarrollar aplicaciones en muchas ámbitos diferentes.

- Alumbrado y Energía.
- Asistencia.
- Entretenimiento.
- Seguridad.
- Comunicación.
- Combinando todos estos elementos en un sistema inteligente.

Figura 2.11 Open Remote

Este proyecto ofrece comunicación con diferentes dispositivos y protocolos pero para la navegación y el uso de la aplicación no ofrece reconocimiento de voz. Sería una capa superior que tendríamos que implementar nosotros.

2.2.1.2 SISTEMAS DE CONTROL DE ENTORNO DE BJ ADAPTACIONES

BJ Adaptaciones [10] es una empresa dedicada a comercializar y producir software y dispositivos de soporte a la autonomía de personas con discapacidad. Es una empresa fundada en 2002 por dos hermanos, uno de los cuales tiene esclerosis múltiple, y su primer proyecto consistió en ofrecer soluciones a personas en situaciones similares. En

su catálogo ofrecen múltiples dispositivos, y entre ellos se encuentran mandos para control de entorno. Estos mandos están específicamente adaptados para una instalación concreta que tienen que hacer los técnicos de la empresa y si se hacen cambios en la instalación domótica hay que reprogramar el mando completo, modificando la funcionalidad de los botones.

Utilizan un protocolo propio para comunicarse con los receptores y únicamente son útiles a personas que, pese a tener problemas de movilidad, siguen pudiendo mover los dedos de la mano.

El objetivo de los sistemas de control de entorno de Bj Adaptaciones es ofrecer una capa de interfaz de usuario adaptable a las necesidades, capacidades y preferencias del mismo. A través de esta interfaz se accede a los dispositivos domóticos para operar con ellos o consultar su estado. La comunicación con el sistema que controla los dispositivos se realiza a través del middleware de comunicación BlueRose, también desarrollado en la Universidad de Granada. La conexión a bajo nivel con los dispositivos se realiza sobre bibliotecas para KNX (principalmente, Calimero). Está previsto el desarrollo de una biblioteca que permita conectarse con varios estándares como ZigBee, Lonworks o X-Bee de forma genérica.

En este caso sería muy ventajoso poder utilizar el reconocimiento de voz.

2.2.1.3 KORA

Kora [11] consiste en una aplicación para Android que pretende facilitar la interacción con el entorno a personas con algún tipo de discapacidad motora, sensorial o cognitiva. Kora permite controlar elementos domóticos con un dispositivo móvil, además de mostrar el estado de los mismos, así como interactuar con otras aplicaciones. Está siendo desarrollada dentro del marco del proyecto Sc@ut del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada, un proyecto que pretende facilitar la integración de las personas con necesidades especiales.

Un concepto básico en el funcionamiento de Kora es el de perfil. Debido a que la aplicación debe dar soporte a varios usuarios que utilizarán el mismo dispositivo en momentos distintos, Kora soporta la gestión de varios usuarios. Cada usuario tiene asociados un perfil de uso (que determina cómo va a interactuar con la aplicación y

cuál va a ser su aspecto) y un perfil de dispositivos que indica qué dispositivos le está permitido usar al usuario y qué acciones podrá realizar sobre los mismos.

Figura 2.12 App KORA

Su objetivo principal es ofrecer una capa de interfaz de usuario adaptable a las necesidades, capacidades y preferencias del mismo. A través de esta interfaz se accede a los dispositivos domóticos para operar con ellos o consultar su estado. La comunicación con el sistema que controla los dispositivos se realiza a través del middleware de comunicación BlueRose, también desarrollado por la Universidad de Granada. La conexión a bajo nivel con los dispositivos se realiza sobre bibliotecas KNX (principalmente, Calimero). Tienen previsto el desarrollo de una biblioteca que permita conectarse con varios estándares como ZigBee, Lonworks o X-Bee de forma genérica.

Entre las ampliaciones futuras que encontramos para este proyecto la más importante relacionada con nuestro tema de estudio es el control por voz.

2.2.1.4 INSTEON for hub

Insteon hub es un controlador central simple y sencillo que se conecta a tu vivienda desde cualquier Smartphone o Tablet. El hub trabaja junto con el router y Smartphone

o Tablets, para controlar dispositivos Insteon en su casa, cuando está de vacaciones o en el trabajo. [12]

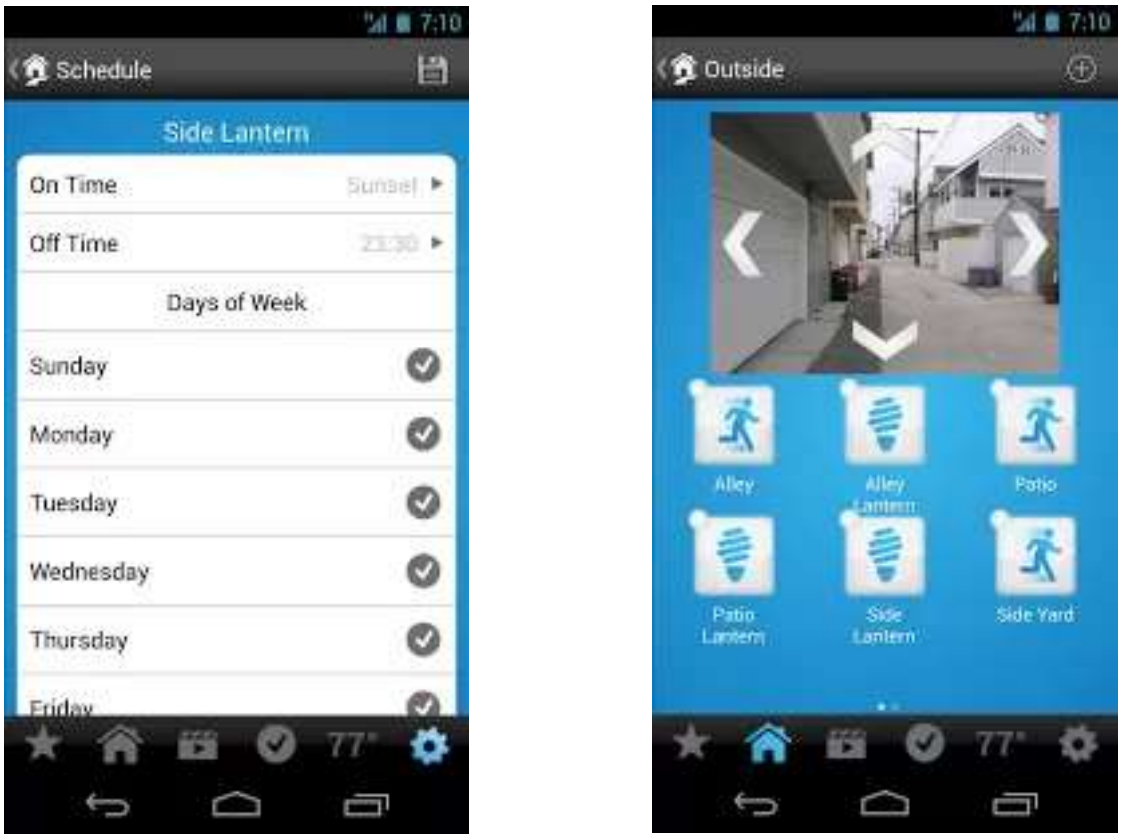


Figura 2.13 Insteon ejemplo interfaz 1

Insteon es una tecnología de control de redes domesticas inalámbricas fiable. Ofrece escalabilidad y flexibilidad. Insteon utiliza simultáneamente RF inalámbrico y las líneas eléctricas de un edificio.

No hay límite práctico para el tamaño de una red Insteon. Las instalaciones con más de 400 nodos son bastantes comunes.

Utilizando una tecnología única, Statelink, una señal Insteon puede activar simultáneamente cientos o incluso miles de dispositivos.

Insteon además proporciona almacenamiento en la nube. Se puede almacenar la configuración de sus hogares para acceder en cualquier momento desde un Smartphone o Tablet.

Figura 2.14 Insteon ejemplo interfaz 2

2.2.2 Control de voz

A continuación se describen dos de los sistemas de reconocimiento de voz más usados a día de hoy. El primero Sherpa, desarrollado para el sistema operativo Android y el segundo Siri, es la versión presentada por Apple para teléfonos Iphone.

2.2.2.1 SHERPA

Sherpa [13] con más de 10 años de experiencia en I+D en tecnologías de procesamiento del lenguaje, el equipo de Sherpa ha utilizado la tecnología más avanzada de reconocimiento de voz, Procesamiento del Lenguaje Natural y Semántica

capaz de realizar un análisis cubriendo 5 niveles: morfológico, sintáctico, semántico, pragmático y funcional. El resultado es un dialogo casi natural entre humano y maquina. Además, Sherpa tiene capacidad de "aprender" y recordar las conversaciones e interacciones realizadas con el usuario.

Figura 2.15 Sherpa app

Algunas de las características que ofrece Sherpa son:

- Información y compra de entradas para conciertos en una ciudad.
- Traductor y definiciones de palabras.
- Tecnología "Intelligent Proactive System Shertab", aprende de tus hábitos para darte información que te interesa.
- El modulo informacional "Multiknowledge System" exclusivo de Sherpa incorpora un nuevo algoritmo con más fuentes de información que es capaz de ofrecer resultados más exactos.
- Cuenta con soporte idiomático para:
    - Español (España)

- Español (Chile)
- Español (Argentina)
- Español (Colombia)
- Español (Venezuela)

2.2.2.2 SIRI

Siri [14] es un asistente virtual, desarrollado por Apple, que puede realizar numerosas funciones por medio de nuestra propia voz: enviar mensajes, programar reuniones, hacer llamadas, etc. Se puede interactuar con Siri con el mismo tono que pueden utilizar dos personas en una conversación.

Para conocer el inicio de Siri, hay que remontarse a la presentación del Iphone 4S a finales de 2011, cuando en la quinta generación de móviles de Apple incluyó esta nueva funcionalidad software, con la que intentaban cambiar drásticamente nuestro modo de usar nuestro Smartphone.

Siri tiene un manejo sencillo, se basa en hablar con un dispositivo sin tener que dar órdenes prefijadas, con el fin de que este realice diferentes funciones. Las conversaciones con Siri pueden ser perfectamente fluidas y ofrece gran cantidad de respuestas a las preguntas planteadas. La transcripción del mensaje no se procesa en el dispositivo, sino que se manda a los servidores de Apple y entonces se recibe la respuesta. Gracias a ello, la inclusión de Siri en Smartphones no implica una reducción del almacenamiento interno, aunque esto pueda causar a veces, algo de retraso.

Figura 2.16 SIRI

Siri no solo reconoce cada palabra, sino que es lo bastante inteligente como para comprender el sentido global. Si por ejemplo se le hace una pregunta: ¿Hay sitios de hamburguesas por aquí cerca?, Siri podría responder: He encontrado nueve hamburgueserías, hay 8 que están bastante cerca de ti. Además si Siri no reconoce que exactamente lo que se busca, realiza preguntas para tenerlo más claro.

Apple está colaborando con fabricantes de coches para integrar Siri en muchos sistemas de control por voz. Pulsando un botón del volante se podrá hacer preguntas a Siri sin apartar la vista de la carretera. Para reducir aún más las distracciones, la pantalla del dispositivo IOS no se iluminará.

2.2.3 Android @Home

Android @home es una iniciativa de Google, iniciada en una conferencia en 2011, que lograría acercar a Android a las tareas de casa, permitiendo mejorar toda clase de aparatos con teléfonos basados en Android. La empresa anunció que uno de los primeros dispositivos en llegar al mercado sería una bombilla LED, la cual sería fabricada por Lighting Science y llegaría al mercado a finales de 2011. Esto nunca ocurrió y no ha habido ninguna palabra oficial del estado del proyecto desde entonces.

Sin embargo, Android @Home parece no haber desaparecido, investigaciones sobre Android revelan restos de este sistema en la actualización de Android 4.2.2 y algunas búsquedas en Linkedln revelan que la empresa mantiene el equipo y por lo tanto el proyecto. [15]

Una de las piezas clave de este proyecto podría ser el reconocimiento de voz de Google. Las funciones de reconocimiento de voz basadas en la nube de la compañía han avanzado muchísimo en los últimos años, hasta el punto donde la voz se ha convertido en uno de los principales métodos de entrada de Google TV.

Figura 2.17 Android @Home

2.3 Impacto industrial tecnológico

El progreso tecnológico, con la informática, la robótica, el diseño gráfico, la inteligencia artificial, la fabricación asistida por ordenador, supone una profunda transformación del sistema productivo de las empresas. Este progreso provoca, que la innovación tecnológica escape al dominio de la mayor parte de las empresas y se transforme en un imperativo para el crecimiento y la supervivencia.

Ahora mismo cientos de millones de personas están conectadas a internet, la tecnología informática no ha parado de aumentar durante cuarenta años a pasos agigantado. La combinación de nuevos avances en muchas aéreas de la tecnología continuará revolucionando el mundo de los negocios.

Algunos de los campos más importantes en los que se centrarán en el futuro los desarrollos tecnológicos son: las fuentes alternativas de energías no renovables, la robótica, la informática, la inteligencia artificial y el estudiado en esta tesina de máster; el reconocimiento de voz y el procesamiento del lenguaje natural.

2.4 El reconocimiento de voz en la industria

La primera industria que utilizó el software de reconocimiento de voz, como una aplicación comercial, fue la industria de la salud. Al inicio, los doctores, pensaron que con esta tecnología, podrían reemplazar a las transcripciones médicas tradicionales. Pero esta idea, no fue muy exitosa.

Actualmente se está realizando grandes avances en el ámbito de la automatización del hogar, especialmente el manejo de las viviendas inteligentes mediante la voz.

Con el desarrollo de este trabajo, se considera que se puede entrar en el mundo industrial integrando alguno o varios de los componentes. Ya sea por incorporar a las aplicaciones existentes nuestro módulo de reconocimiento de voz, de navegación o por ofrecer un producto completo de control de una vivienda inteligente a las empresas que carezcan de desarrolladores propios.

Varias de las aplicaciones estudiadas en el punto 2.2.1 se encuentran en esta situación; disponen de un sistema de control para una vivienda inteligente pero carecen del factor que les produciría un valor añadido, como es el reconocimiento de voz. Así,

todas estas aplicaciones, comentan como trabajo futuro incorporar esta característica a su proyecto. Por lo que incorporar una tecnología como la desarrollada en esta Tesina de Máster a su proyecto les ahorraría costes, en tiempo y dinero, ya que el camino que se ha recorrido para alcanzar el objetivo de este producto no ha sido trivial.

Otro camino industrial en el cual este producto se podría integrar dentro de otros proyectos, o desarrollar proyectos nuevos con las características que se proponen en esta Tesina, es el ámbito de la tecnología adaptativa para discapacitados.

La tecnología adaptativa puede llegar a reducir el impacto de la discapacidad y satisfacer el derecho de la calidad de vida de las personas con necesidades especiales.

Las mejoras tecnológicas y sobre todo la creciente competitividad entre fabricantes hace que cada día se tengan más en cuenta las necesidades de los usuarios desde el mismo momento de la concepción de un producto. Pese a que esas necesidades, en principio sean las de la población en general, las soluciones adoptadas acaban beneficiando a sectores de población con discapacidad.

3 Presentación de la propuesta

3.1 Planteamiento del problema

La evolución tan rápida de la tecnología a la que estamos sometidos, nos genera grandes dificultades para usarla y es que las personas no somos capaces de asimilar esos cambios rápidamente, en especial, personas no familiarizadas con estas tecnologías. Cada elemento que forma parte de esta evolución proviene de una raíz distinta, soporta una manera de interactuar diferente, tiene nuevas funcionalidades, nuevas versiones y un sinfín de características distintas. Las personas temen tener que afrontar nuevos retos para poder disfrutar de, a veces, una simple particularidad.

Cada vez más, todas estas tecnologías convergen hacia un punto en común, hacia una forma natural de interactuar con las maquinas. Los humanos, la forma más natural con la que nos comunicamos es mediante la voz Ésta, es una línea de investigación importante hoy en día. Prácticamente todos los dispositivos permiten este medio de interacción pero es el software el que todavía tiene que evolucionar para poder ofrecer una mayor naturalidad. Mediante la voz da igual con que dispositivo estemos tratando si entiende nuestra lengua. Nos ofrecerá ayuda en momentos que no sepamos continuar, ampliará las opciones de personas discapacitadas o que no puedan manejar estos dispositivos a través de otra interfaz.

Es por esto que la necesidad de controlar un sistema domótico mediante la voz resulta indispensable. Necesitamos que nuestro sistema sea útil para todo tipo de personas y además de fácil manejo. Tan fácil como el manejo de cualquier interruptor de nuestra vivienda.

El problema propuesto trata de abarcar todo el control de los dispositivos de la vivienda, desde un interruptor hasta elementos electrónicos más avanzados. Los puntos necesarios a cubrir son:

- Navegación por estancias de la vivienda.
- Navegación por tipos de dispositivos de la vivienda.

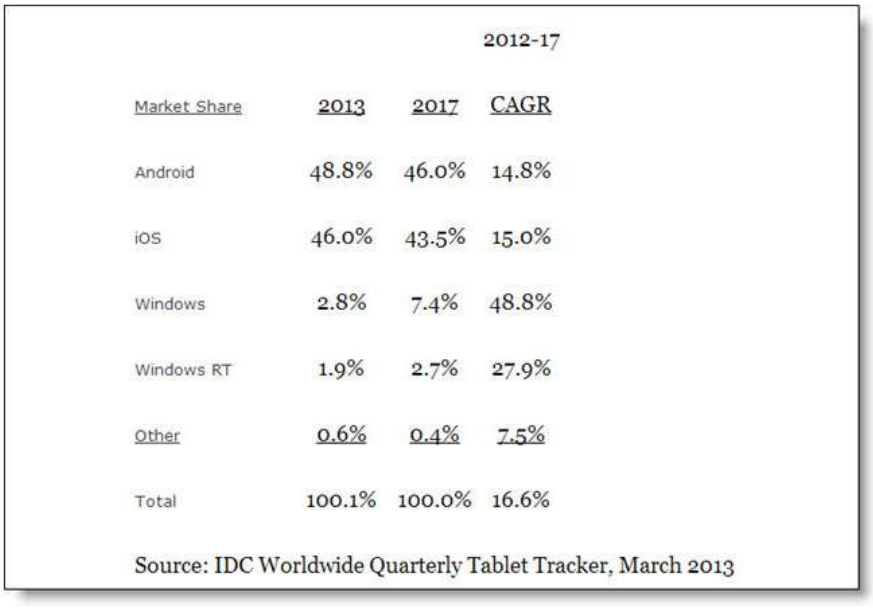
- Comprobación del estado de los dispositivos.
- Reportes de información provenientes de los sensores.
- Detalles de los dispositivos.
- Programación de los actuadores.
- Control local.
- Control remoto.

3.2 Planteamiento de la solución

Para poder dar solución al planteamiento definido en el punto anterior nos hemos basado en el desarrollo de una aplicación Android.

Android está presente en cientos de millones de dispositivos móviles en más de 190 países en todo el mundo. Es la plataforma móvil con más amplitud y la que más rápidamente ha crecido. Todos los días millones de usuarios se inician con un dispositivo Android, en busca de apps, juegos y todo tipo de contenido digital.

Como se muestra en la **Figura 3.1 Gráfica crecimiento previsto Android**, el crecimiento de dispositivos Android que se espera para los próximos años es mayor a cualquier otro [16].



		2012-17	
Market Share	2013	2017	CAGR
Android	48.8%	46.0%	14.8%
iOS	46.0%	43.5%	15.0%
Windows	2.8%	7.4%	48.8%
Windows RT	1.9%	2.7%	27.9%
Other	0.6%	0.4%	7.5%
Total	100.1%	100.0%	16.6%

Source: IDC Worldwide Quarterly Tablet Tracker, March 2013

Figura 3.1 Gráfica crecimiento previsto Android

La aplicación que vamos a desarrollar tiene como objetivo el manejo de un sistema domótico desde un dispositivo móvil. No hay opción al manejo desde un ordenador porque necesitamos poder configurar la vivienda desde cualquier sitio. Todos los dispositivos móviles disponen de micrófono y altavoz por lo que no es necesario ningún accesorio complementario, como sería necesario en un ordenador de sobremesa convencional.

El uso de dispositivos Android para este propósito también tiene beneficios técnicos para el procesamiento de audio que es procesado por los servidores de Google. Desarrollar en Android implica aprender todas las tecnologías necesarias para ello, conocer a fondo las capacidades de los terminales. Al tratarse de un sistema operativo de código abierto cualquier persona puede aportar sus conocimientos lo que le añade un potencial muy elevado.

Se han estudiado varias alternativas para desarrollar esta aplicación con las características necesarias. Este estudio comprende la diferencia entre una aplicación nativa y una aplicación con Html5, como se muestra en la **Figura 3.2 Aplicaciones nativa vs html5**. Desarrollar en una tecnología u otra implica una serie de ventajas e inconvenientes que describimos a continuación.

En un mundo cada vez más social y abierto, las aplicaciones móviles juegan un papel vital. A la hora de desarrollar una aplicación surge la duda de desarrollar una aplicación móvil nativa o una aplicación móvil web optimizada para dispositivos móviles.

Hay muchos factores que juegan un papel importante en la estrategia de desarrollo, las habilidades del equipo de producción, funcionalidades requeridas del dispositivo, seguridad, interoperabilidad.

Al final no es una cuestión de qué debería hacer tu aplicación, sino como lo debería hacer.

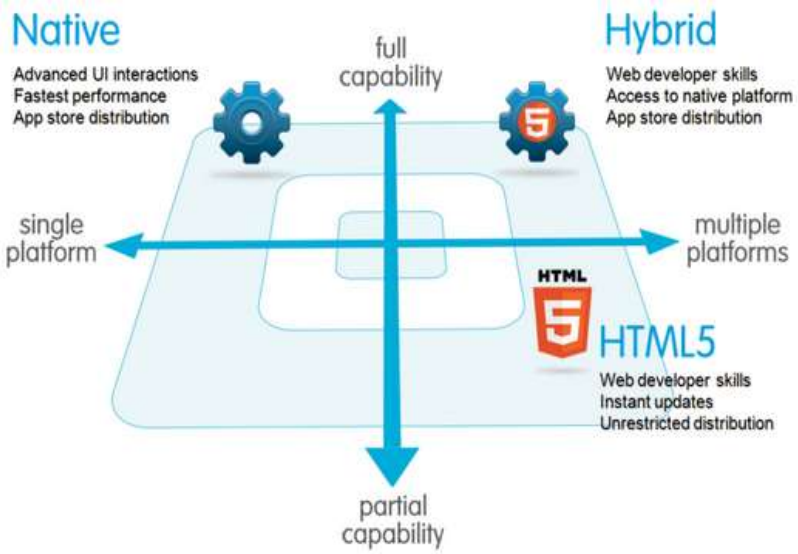


Figura 3.2 Aplicaciones nativa vs html5

Android es un sistema operativo y una plataforma software basada en Linux para teléfonos móviles. Tiene apenas 5 años y en ese tiempo lo hemos visto evolucionar muy rápidamente. Actualmente es el sistema operativo más utilizado en Smartphone lo que invita a desarrollar aplicaciones con esta tecnología.

3.2.1 ¿Por qué desarrollar aplicaciones nativas?

- Amplio mercado

Los teléfonos inteligentes equipados con el sistema operativo Android cuentan con más de mil millones de usuarios en todo el mundo. Dispone de una cuota de mercado del 80%.

- Se ofrece una buena experiencia de usuario.

Sobre todo, a partir de la versión 4.0, Android dispone de interfaces eficaces, bonitas y potentes. Dejando atrás a sus competidores.

- Acceso más amplio al hardware del dispositivo.

Esta característica está centrada sobre todo para desarrolladores, ya que Android ofrece mediante APIs toda la funcionalidad que se necesita para crear aplicaciones en este sistema operativo.

- La aplicación esta optimizada para el hardware sobre el que está corriendo.

La aplicaciones nativas en Android están dentro de un contexto gestionado por el sistema operativo, lo que hace que aumente la eficiencia y seguridad entre otras característica.

Las aplicaciones móviles nativas ofrecen una mayor rapidez y una mejor experiencia de usuario, permiten usar muchas funcionalidades del sistema operativo, notificaciones push, check-ins, códigos QR.

El dispositivo móvil es sinónimo de innovación, nuevos sensores, cámaras 3D, servicios en segundo plano, reconocimiento de voz y proporciona API’s sin tener que esperar a la aparición de estándares que retrasan la incorporación de nuevas funcionalidades.

Además las aplicaciones nativas son más potentes que las aplicaciones web, el ecosistemas en que se encuentran las aplicaciones nativas no están encapsuladas únicamente en el navegador y forman parten del dispositivo. Pueden escuchar eventos, como una llamada de teléfono.

Soporte completo offline; no es necesario que estén en constante sincronización, lo que hace que se ahorre batería.

3.2.2 ¿Por qué desarrollar aplicación móvil en HTML5?

Las aplicaciones web basadas en **HTML5** permiten ser consultadas desde cualquier dispositivo sin necesidad de cargar ninguna aplicación, ofrecen un acceso inmediato, funcionan a través de todas las plataformas y no son dependientes del dispositivo.

HTML5 está presente en todos los dispositivos, teléfonos, Tablets, Tv. Es flexible, puedes reutilizar, aplicar estilos propios para cada dispositivo y no necesitas aprender una nueva tecnología cada vez que quieras rehacer la aplicación para otro dispositivo. Las aplicaciones web usan tecnologías web estándar, como HTML5, CSS y JavaScript lo que permiten ser usadas en múltiples plataformas.

Existen multitud de frameworks multiplataforma de desarrollo para aplicaciones móviles que hacen uso de estas tecnologías:

- **PhoneGap**: Uso de HTML, CSS y JavaScript junto con librerías propias para el desarrollo multiplataforma de aplicaciones nativas. Soporta las siguientes plataformas: iOS, Android, BlackBerry, Symbian, Windows Phone 7.

- **Rhomobile**: Hace uso de Ruby para la lógica de negocio y utiliza HTML, CSS y JavaScript para la interfaz. Uso de Rhosync para la sincronización de datos cliente-servidor. Soporta las siguientes plataformas: iOS, Android, BlackBerry, Symbian, Windows Phone 7.

- **Titanium**: Uso de JavaScript junto a librerías propias para el diseño de aplicaciones nativas. Es un Framework de código o abierto. Soporta las siguientes plataformas: iOS, Android, BlackBerry.

3.2.3 Conclusiones

El problema que tratamos de abordar es el de una aplicación de reconocimiento de voz para el control domótico de una vivienda. Después de estudiar las diferencias entre una aplicación nativa y una aplicación en html5, decidimos desarrollarla en código nativo para Android. Desarrollar una aplicación de reconocimiento de voz con html5

implica tener una conexión a internet continuamente, mientras que con una aplicación nativa puedes trabajar con tu red local para interactuar con los demás dispositivos.

Otra de las necesidades que nos planteamos para desarrollar en Android nativo son las notificaciones, el teléfono estará preparado para recibir notificaciones del control domótico de la vivienda.

Para el desarrollo de este trabajo, se ha decidido que el manejo de la aplicación este controlado por una serie de comandos y de órdenes que se han definido previamente. Lo ideal sería que se desarrollara bajo un sistema de Reconocimiento de Voz semántico, pero la complejidad es demasiado grande y a fecha de hoy todavía se está trabajando mucho a nivel de investigación en esta faceta. Por todo esto y porque la funcionalidad requerida por la aplicación es limitada se ha optado por desarrollar la aplicación con comandos ya definidos. La aplicación en todo momento muestra sugerencias de los comandos posibles para evitar tener que memorizar comandos.

A continuación se muestra una tabla comparativa entre una aplicación Android y una aplicación desarrollada en HTML5.

	Android Nativo	HTML5
Rendimiento	✓	✗
Trabajo offline reconocimiento de voz	✓	✗
Compatibilidad	✓	✗
Portabilidad	✗	✓
Conectividad hardware	✓	✗

Rapidez de desarrollo		

Figura 3.3 Android vs HTML5

Podemos ver que las características estudiadas en esta comparativa y que son necesarias para el desarrollo de este trabajo, favorecen a la elección de un desarrollo sobre el sistema operativo Android nativo.

4 Desarrollo de la propuesta

En este capítulo se describen los requisitos relativos al desarrollo de la aplicación. Y las funcionalidades que debe cumplir.

4.1 Requisitos

Los que se pretende desarrollar en esta Tesina de Máster es una interfaz para una aplicación de control domótico mediante la voz.

4.1.1 Requisitos funcionales

Los puntos a desarrollar respecto a la funcionalidad de la aplicación son:

- Navegabilidad por estancias y por dispositivos.
- Reconocimiento de comandos.
- Interacción con los dispositivos.
- Reportes de estado de los dispositivos.
- Configuración de escenas de confort.

La aplicación debe de disponer de una navegabilidad sencilla por todas las zonas de la vivienda, para poder situarnos en una zona u otra rápidamente. El sistema es capaz de reconocer un conjunto de órdenes acotados previamente las cuales se detallan en el punto 4.5 Resumen de comandos, y dependiendo de en qué punto de la aplicación se encuentre el usuario, estas podrán variar para realizar una funcionalidad de navegación u otra.

Una vez situados en una zona de la vivienda podremos interactuar con los dispositivos, con comandos como, “listar dispositivo”, “mostrar dispositivos” o “dispositivos actuales”, el sistema nos reproducirá los dispositivos agrupados por tipos, iluminación, seguridad, confort. Diciendo el nombre del dispositivo accederemos a ver su estado y las opciones de configuración que tiene, podremos ejecutar acciones sobre el dispositivo con comandos como; “encender”, “apagar, “bajar” o “subir temperatura”.

La aplicación además de controlar los dispositivos accediendo a cada zona de la vivienda podremos controlar los dispositivos de un modo general accediendo a los grupos de cada tipo. Por ejemplo “listar todas las luces”, “mostrar todas las persiana”, y ejecutar comandos generales sobre ellos, “apagar todas las luces”, “subir todas las persianas”.

El sistema también incorpora la funcionalidad de escenas de confort, podremos adaptar toda la vivienda mediante una escena ya programada. Por ejemplo si queremos ver cine, podremos indicar al sistema que se adapte a la escena cine, lo que llevará asociado una serie de acciones:

1. Bajar las persianas al 60%

2. Bajar la intensidad de la luz al 10 %

3. Poner la temperatura a 25%

Otra escena de confort que podríamos programar es escena lluvia y las acciones asociadas serian:

1. Bajar persianas al 100%

2. Subir intensidad de la luz al 80%

Estas escenas se podrán indicar al sistema mediante comandos como “activar escena lluvia”, “escena cine”, “escena confort deportes”, “escena seguridad”.

En la aplicación en todo momento debe de estar disponible el comando de “ayuda”, para que cuando el usuario no sepa que hacer, siempre tenga asistencia por parte del sistema disponible. El comando ayuda se podrá solicitar con comandos como “mostrar ayuda”, “ayuda”, “que hacer”, etc.

4.1.2 Requisitos no funcionales

La funcionalidad de reconocimiento de voz en el sistema operativo Android se encuentra a partir de la versión 1.5 que está descatalogada y con una cuota de mercado muy reducida. Hoy en día las versiones del sistema operativo que disponen los dispositivos se encuentran entre las versiones 3.x y 4.x por lo que no debe de haber ningún problema en la implantación con dispositivos actuales.

- Android.speech : este paquete requiere Api level 3 o superior
- Android.speech.tts: este paquete requiere Api level 4 o superior.

El dispositivo debe disponer de Wifi y de Conexión a internet, para poder controlar el sistema tanto en local como en remoto.

En esta versión de la aplicación, no está desarrollado ningún módulo de seguridad, por lo que no es necesario autentificación para utilizar la aplicación.

El diseño y el desarrollo de la aplicación deben cubrir las necesidades de escalabilidad que puedan surgir en un futuro.

El código debe de estar lo más desacoplado posible unas clases de otras, y definir correctamente la funcionalidad de cada objeto para conseguir una alta cohesión.

4.2 Arquitectura general

La arquitectura general en la que está centrado nuestro proyecto es la que se muestra en la **Figura 4.1 Gráfico general de contexto**, desarrollamos el cliente como una aplicación Android con una interfaz de reconocimiento de voz que mediante servicios REST se conectará con los servicios domóticos de la vivienda inteligente.

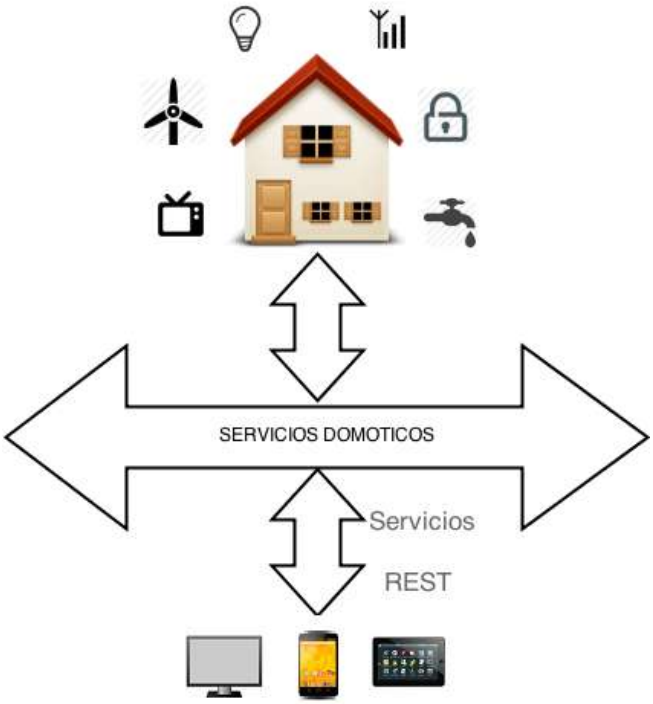


Figura 4.1 Gráfico general de contexto

Para el desarrollo de esta aplicación se ha aplicado una arquitectura basada en capas, ya que la funcionalidad desarrollada se integra en un proyecto de mayor envergadura, y así facilitar la integración y poder hacerlo de un modo casi transparente.

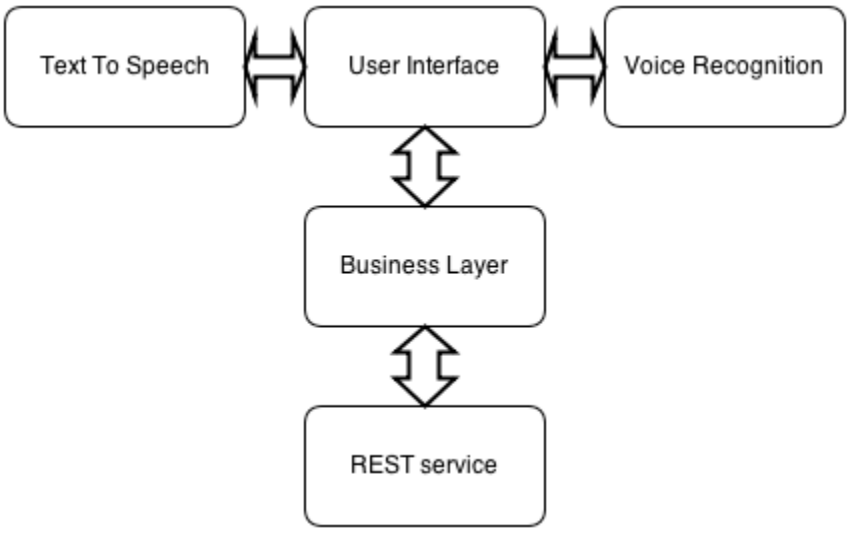


Figura 4.2 Estructura en capas

Como se muestra en la **Figura 4.2 Estructura en capas** la arquitectura consta de un bloque principal de tres capas:

- User Interface
- Business Layer
- Servicios REST

La primera capa de interfaz de usuario está ligada al reconocimiento de voz, capaz de interpretar los comandos del usuario mediante la voz, y que dicha información la transmite hacia la capa de lógica que se encarga de realizar la funcionalidad necesaria. La capa de interfaz de usuario también está conectada al sintetizador de voz, para ofrecer al usuario un feedback mediante voz.

4.2.1 Estructura de módulos

La estructura de módulos que se ha descrito para el desarrollo de la aplicación**, Figura 4.3 Estructura de módulos**, se basa en dos bloques principales; el primer bloque se centra en un conjunto de Activities que se encargan de mostrar cada tipo de interfaz de usuario. Cada una de estas Activities está relacionada con un Layout que describe en formato XML como se compone la UI de la aplicación. La interface de usuario representada conceptualmente por la clase Activity está relacionada con el servicio de sintetizador de voz. El segundo bloque principal es la clase Navigator que representa la navegabilidad del sistema y es la encargada de soportar el peso de la lógica. Las clases de Localizaciones y Dispositivos componen la vivienda a controlar mediante la aplicación.

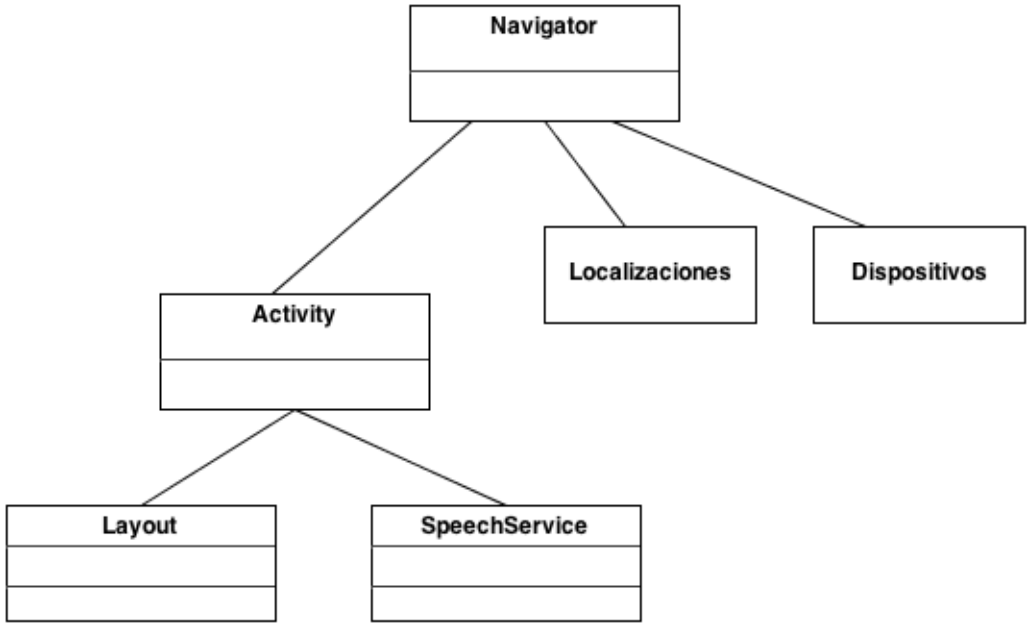


Figura 4.3 Estructura de módulos

4.2.2 Arquitectura de la vivienda en XML

La arquitectura de la vivienda se encuentra representada en dos ficheros XML, tal y como se describen a continuación:

configV5.1.xml: Este fichero contiene las estancias de la vivienda y los dispositivos que se encuentran en ella. Las estancias y los dispositivos están representados con un identificador con el que podremos relacionar las estancias con los dispositivos y las estancias con otras estancias para representar una jerarquía. En la siguiente figura podemos ver cómo está representada una estancia:

```
<Location id="HAUS_LOCATION_e13F4w0FSrRRIHn" idDB="" db_status="NEW"
name="Casa" type="HAUS_LOCATION_TYPE_HOME" root="1">

    <ChildLocations>

        <ChildLocation idRef="HAUS_LOCATION_OYcogrfyvSfPJzS"/>

    </ChildLocations>

    <LocationDevices>

     <LocationDevice idRef="HAUS_DEVICE_TEbRUXw6QXXEe8O"/>

     <LocationDevice idRef="HAUS_DEVICE_xgRTExMkhkrQvKt"/>

     <LocationDevice idRef="HAUS_DEVICE_vdPRxL03VzuCNIb"/>

     <LocationDevice idRef="HAUS_DEVICE_TRaRVYw6QTTEe91"/>

    </LocationDevices>

</Location>
```

Figura 4.4 Representación XML de una estancia

La etiqueta <Location> abre la configuración de la estancia, en ella podemos encontrar el atributo ID, para su identificación. Después vemos la etiqueta <ChildLocations> para representar la jerarquía dentro de la distribución de las estancias de la vivienda. A continuación con la etiqueta <LocationDevices> asociamos a esta estancia una serie de dispositivos.

En este fichero además encontramos la configuración de los dispositivos como podemos ver en la figura siguiente.

```
<Device id="HAUS_DEVICE_vdPRxL03VzuCNIb" idDB="" db_status="NEW"
name="ElectroValve" type="HAUS_DEVICE_TYPE_SENSOR_OTHER"
technology="HAUS_TECHNOLOGY_KNX" manufacturer="" manufacturerURL=""
modelName="" serialNumber="">

    <DeviceInterfaces>

     <DeviceInterface idRef="HAUS_INTERFACE_KNX_ELECTRO_VALVE_1.0.1"
idDB="" db_status="NEW">

      <ConfigParameter idRef="HAUS_IIA_KNX_ELECTRO_VALVE_1.0.1_ADDRESS"
idDB="" value="0/0/10" db_status="NEW"/>

     </DeviceInterface>

    </DeviceInterfaces>
```

Figura 4.5 Representación XML de un dispositivo

La configuración comienza con la etiqueta <Device>, y en ella podemos encontrar el atributo ID para representar su identificación. Otra de las etiquetas importantes de este fragmento es <DeviceInterface> que nos indica la interfaz que utiliza este dispositivo para su control domótico. <ConfigParameter> indica entre otras cosas el tipo y una descripción de los parámetros de configuración que necesita.

HausInterfaces.xml: Este fichero representa las interfaces de los elementos domóticos que los dispositivos pueden utilizar. Podemos añadir nuevos dispositivos a la vivienda para ser manejados con nuestro sistema domótico siempre y cuando cumpla con alguna de las interfaces implementadas.

En la siguiente figura vemos un ejemplo de la implementación de una interfaz.

```
<Interface id="HAUS_INTERFACE_KNX_ELECTRO_VALVE_1.0.1"
name="ElectroValve" version="1.0.1" technology="HAUS_TECHNOLOGY_KNX">

        <ConfigParameters>

                <ConfigParameter
        id="HAUS_IIA_KNX_ELECTRO_VALVE_1.0.1_ADDRESS" name="address"
        description="The KNX Group Address (Format: X.X.X)" basicType="STRING"
        mandatory="TRUE"/>

                </ConfigParameters>

                <InterfaceProperties>

                        <InterfaceProperty
id="HAUS_INTERFACE_PROPERTY_KNX_ELECTRO_VALVE_1.0.1_IS_OPEN"
name="isOpen" description="" basicType="BOOLEAN">

                                <States>

                                        <State name="OPEN" junctor="EQUAL"
                                value="TRUE"/>

                                        <State name="CLOSED" junctor="EQUAL"
                                value="FALSE"/>

                                </States>

                        </InterfaceProperty>

                </InterfaceProperties>
```

Figura 4.6 Representación XML interfaces

De la figura anterior podemos destacar la etiqueta <Interface> que contiene un identificador para que sea referenciado desde un dispositivo correctamente, y una etiqueta <InterfaceProperties> que incluye dentro de ella la etiqueta <State> para controlar el estado del dispositivo.

4.3 Tecnologías usadas

A continuación se explican las tecnologías utilizadas en este proyecto, con el fin de ofrecer unos conocimientos básicos a todo aquel que lea esta Tesina, describiendo cada una de ellas.

Estas tecnologías han sido utilizas por ser las mejores opciones para ofrece la solución adecuada después de conocer el problema planteado.

4.3.1 Android TTS Speech

Text-To-Speech (TTS) [17], también llamado "speech synthesis", proporciona la capacidad al dispositivo de hablar en diferentes lenguajes.

El motor TTS integrado en la plataforma Android soporta varios lenguajes: Inglés, Francés, Alemán, Italiano y Español. El motor TTS necesita conocer el lenguaje a hablar. La voz y el diccionario son recursos específicos de cada lenguaje por lo que tienen que ser cargados antes de que el motor pueda empezar a hablar.

Aunque todos los dispositivos soportan esta funcionalidad, algunos dispositivos tienen limitaciones de almacenamiento y pueden carecer de los ficheros de un lenguaje especifico. Así que en la creación de una Activity, un buen primer paso, es comprobar si los recursos de TTS están disponibles con el correspondiente Intent.

```
Intent checkIntent=newIntent();

checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);

startActivityForResult(checkIntent, MY_DATA_CHECK_CODE);
```

Si la comprobación es correcta, devolverá un resultado CHECK_VOICE_DATA_PASS, indicando que el dispositivo está preparado para hablar, después de la instanciación del objeto Android.speech.tts.TextToSpeech. Si no, hay que informar al usuario la necesidad de instalar los componentes para obtener un dispositivo multi lenguaje.

La necesidad de descargar e instalar los componentes viene informada por ACTION_INSTALL_TTS_DATA Intent, el usuario deberá acceder a Android Market e iniciar la descarga, una vez descargado el proceso de instalación, el proceso de instalación se realizara automáticamente. A continuación se muestra un ejemplo de la implementación de onActivityResult():

```
private TextToSpeech mTts;

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

   if (requestCode == MY_DATA_CHECK_CODE) {

      if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {

         // success, create the TTS instance

         mTts = new TextToSpeech(this, this);

      } else {

         // missing data, install it

         Intent installIntent = new Intent();

         installIntent.setAction(

            TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);

         startActivity(installIntent);

      }

   }

}
```

En el constructor de TextToSpeech pasamos la referencia del *Contexto* en el que se encuentra, en el ejemplo anterior la Activity, y el método onInitListener que se llamará cuando el motor TextToSpeech se haya inicializado. Este listener proporciona la capacidad a nuestra aplicación de notificar cuando el motor Text-To-Speech está totalmente cargado, es decir, cuando podemos empezar a configurarlo y a usarlo.

Una vez hayamos configurado la instancia de TextToSpeech, ya podemos empezar a hacer que nuestra aplicación hable. Una manera sencilla de hacerlo es usando el método speak().

El motor TTS maneja una cola global de todas las entradas para sintetizar. Cada instancia de TextToSpeech, puede controlar su propia cola con el fin de controlar que expresión interrumpirá la actual y la que simplemente se pone en cola. En el siguiente ejemplo vemos como la primera llamada al método speak() interrumpe cualquiera que estuviese siendo sintetizada, la cola es "fluseada" y la nueva expresión es encolada y se colocará en cabeza de la cola. La segunda expresión es encolada y se ejecutará después de que myText1 se haya completado.

```
String myText1 = "Did you sleep well?";

String myText2 = "I hope so, because it's time to wake up.";

mTts.speak(myText1, TextToSpeech.QUEUE_FLUSH, null);

mTts.speak(myText2, TextToSpeech.QUEUE_ADD, null);
```

El motor de TextToSpeech depende de un delicado servicio que es compartido por todas las aplicaciones que usen esta característica. Cuando se termine de usar TTS, una buena práctica es indicar que no lo vamos a necesitar más llamando a su método shutdown().

4.3.2 Servicios REST

REST define un conjunto de principios arquitectónicos por los cuales se diseñan servicios web, haciendo énfasis en los recursos del sistema, incluyendo, cómo se accede al estado de dichos recursos, y cómo se transfieren por *HTTP* hacia clientes escritos en diversos lenguajes. *REST* ha emergido en los últimos años como el modelo predominante para el diseño de servicios. De hecho, *REST* ha logrado un impacto tan grande en la web, que prácticamente ha desplazado a *SOAP* y las interfaces basadas en *WSDL*, por tener un estilo bastante más simple de usar.

REST no tuvo mucha atención, cuando Roy Fielding lo presentó por primera vez en el año 2000, en la Universidad de California, durante la charla académica "*Estilos de Arquitectura y el Diseño de Arquitecturas de Software basadas en Redes*", la cual analizaba un conjunto de principios arquitectónicos de software, para usar la Web como una plataforma de procesamiento distribuido. Ahora, años después de su presentación, comienzan a aparecer varios frameworks *REST* y se ha convertido en una parte integral de Java 6 a través de JSR-311. Los cuatro principios de *REST* son:

1. Utiliza los métodos HTTP de manera explícita.

Una de las características claves de los servicios web *REST*, es el uso explícito de los métodos *HTTP*, siguiendo el protocolo definido por *RFC 2616*. Por ejemplo, *HTTP GET* se define como un método productor de datos, cuyo uso está pensado para que las aplicaciones cliente obtengan recursos, busquen datos de un servidor web, o ejecuten una consulta, esperando que el servidor web la realice y devuelva un conjunto de recursos.

REST hace que los desarrolladores usen los métodos *HTTP* explícitamente, de manera que resulte consistente con la definición del protocolo. Este principio de diseño básico, establece una asociación *uno-a-uno*, entre las operaciones de crear, leer, actualizar y borrar, y los métodos *HTTP*. De acuerdo a esta asociación:

- Se usa *POST* para crear un recurso en el servidor
- Se usa *GET* para obtener un recurso
- Se usa *PUT* para cambiar el estado de un recurso o actualizarlo

- Se usa *DELETE* para eliminar un recurso

2. No mantiene estado.

Los servicios web *REST* necesitan escalar, para poder satisfacer una demanda en constante crecimiento. Se usan *clusters* de servidores con balanceadores de carga y alta disponibilidad, *proxies*, y *gateways* para conformar una topología útil/práctica, que permita transferir peticiones de un equipo a otro, para disminuir el tiempo total de respuesta de una invocación al *servicio web*. El uso de servidores intermedios para mejorar la escalabilidad, hace necesario que los clientes de servicios web *REST*, envíen peticiones completas e independientes, es decir, se deben enviar peticiones que incluyan todos los datos necesarios, para cumplir el pedido, de manera que los componentes en los servidores intermedios, puedan re-direccionar y gestionar la carga, sin mantener el estado localmente entre las peticiones.

Una petición completa e independiente, hace que el servidor no tenga que recuperar ninguna información de contexto o estado, al procesar la petición. Una aplicación o cliente de servicio web *REST* debe incluir, dentro del encabezado y del cuerpo *HTTP* de la petición, todos los parámetros, contexto y datos que necesita el servidor, para generar la respuesta. De esta manera, el no mantener estado, mejora el rendimiento de los servicios web y simplifica el diseño e implementación de los componentes del servidor, ya que la ausencia de estado en el servidor, elimina la necesidad de sincronizar los datos de la sesión, con una aplicación externa.

3. Despliega URIs con forma de directorios

Desde el punto de vista del cliente de la aplicación que accede a un recurso, la *URI* determina qué tan intuitivo va a ser el servicio web *REST*, y si éste va a ser utilizado tal como fue pensado al momento de diseñarlo. La tercera característica de los servicios web *REST* es justamente sobre las *URIs*.

Las *URIs* de los servicios web *REST* deben ser intuitivas, hasta el punto de que sea fácil adivinarlas. Se ha de pensar en las *URI* como una interfaz auto-documentada, que necesita de muy poca o ninguna, explicación o referencia, para que un desarrollador pueda comprender a lo que apunta, y a los recursos derivados relacionados.

Una forma de lograr este nivel de usabilidad, es definir *URIs* con una estructura al estilo de los directorios. Este tipo de *URIs* es jerárquica, con una única ruta raíz, y va abriendo ramas a través de las sub-rutas, para exponer las áreas principales del

servicio. De acuerdo a esta definición, una *URI* no es solamente una cadena de caracteres delimitada por barras, sino más bien, un árbol con padres e hijos organizados como nodos, por ejemplo:

http://www.miservicio.com/webservice/JSONApi/retrieveNotifications.php

4. Transfiere XML, JSON o ambos.

Estos son los dos formatos en los que un servicio web *REST*, puede mostrar los datos devueltos por el servidor.

4.3.3 Patrón Singleton

El patrón Singleton es uno de los patrones más sencillos que se presentan en el libro patrones de diseño [18]. Es también uno de los patrones más conocidos y utilizados. Su propósito es asegurar que solo exista una instancia de una clase y proporciona un punto de acceso global a esta instancia.

El patrón Singleton se puede utilizar en los siguientes casos:

- Debe haber exactamente una instancia de la clase.
- La única instancia de la clase debe ser accesible para todos los clientes de esa clase.

El patrón Singleton es relativamente simple, ya que sólo involucra una clase. Una clase Singleton tiene una variable *static* que se refiere a la única instancia de la clase que quieres usar. Esta instancia es creada cuando la clase es cargada en memoria. La clase se debería de implementar de tal forma que se prevenga que otras clases puedan crear instancias adicionales de una clase Singleton. Esto significa que se debe asegurar de que todos los constructores de la clase son privados.

Para acceder a la única instancia de una clase Singleton, la clase proporciona un método *static*, normalmente llamado getInstance, el cual retorna una referencia a la única instancia de la clase.

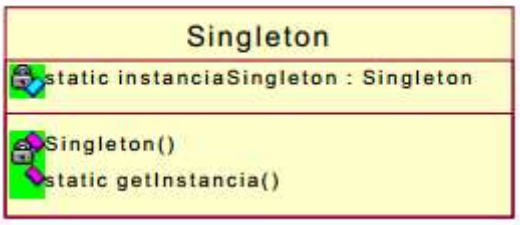


Figura 4.7 Patrón Software Singleton

4.3.4 Json

JSON (JavaScript Object Notation) es un ligero formato de intercambio de datos. Es fácil de leer y escribir para los humanos y además es fácil para las computadoras parsear y generar. Está basado en un subconjunto de JavaScript Programming Language, Standard ECMA-262 3rd Edition December 1999. JSON es un formato de texto que es un lenguaje completamente independiente pero usa convenciones que son familiares para los programadores. Lo que hace que JSON sea un lenguaje ideal para intercambio de datos.

JSON está construido sobre dos estructuras:

- Una colección de tuplas nombre/valor. En varios lenguajes, esto se implementa como objetos, estructuras, diccionarios, tablas hash, array asociativos, etc.

- Un lista de valores ordenador. Esto se puede realizar como un array, un vector, una lista o una secuencia.

Estas estructuras de datos son universales, todos los lenguajes de programación soportan estas estructuras.

Un JSON puede tomar alguna de estas formas.

1. Un objeto es un conjunto no ordenado de tuplas nombre/valor. Un objeto comienza con “{“ y termina con “}”. Cada nombre esta seguido de “:” y la tupla nombre/valor separado con una “,”.

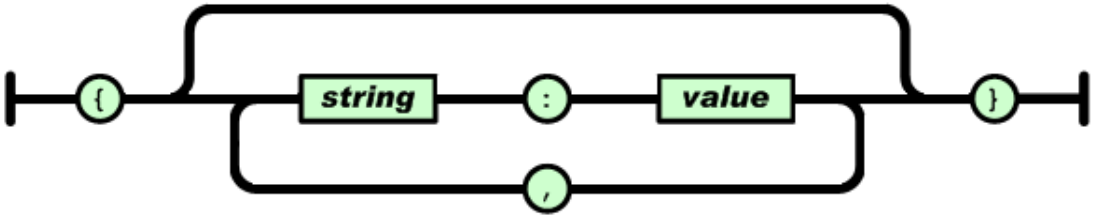


Figura 4.8 Json objeto

2. Un array es una colección ordenada de valores. Un array comienza con "[" y termina con "]". Los valores están separados por una coma.

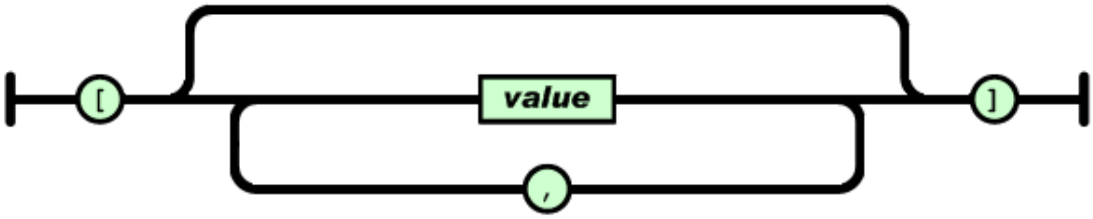


Figura 4.9 Json array

3. Un valor puede ser un string, un numero, true o false o null, un objeto o un array. Estas estructuras pueden ser anidadas.

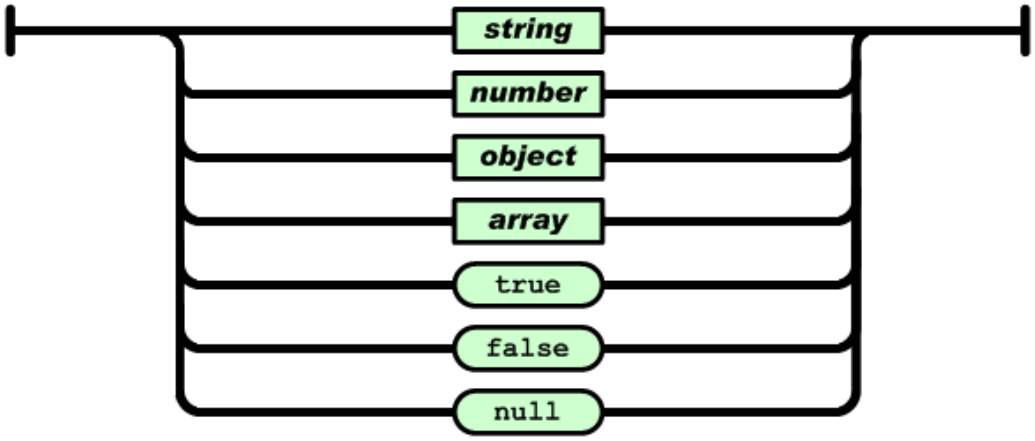


Figura 4.10 Json value

4. Un string es una secuencia de cero o más caracteres, envueltas en dobles comillas y una barra invertida para los caracteres especiales. Un carácter está representado como una cadena de caracteres única, como en C o Java.

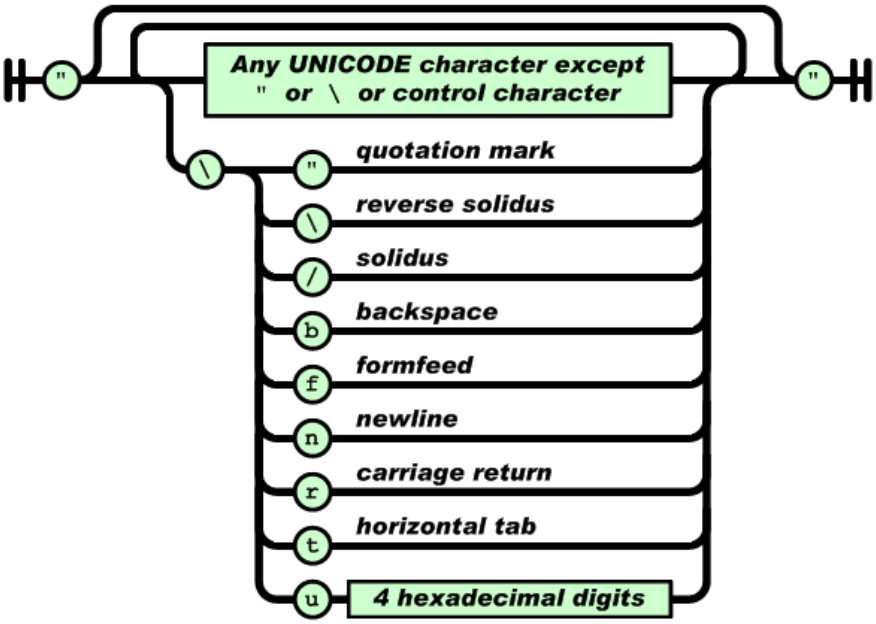


Figura 4.11 Json String

5. Un número es muy parecido a C o Java, excepto que los formatos octal y hexadecimal no son usados.

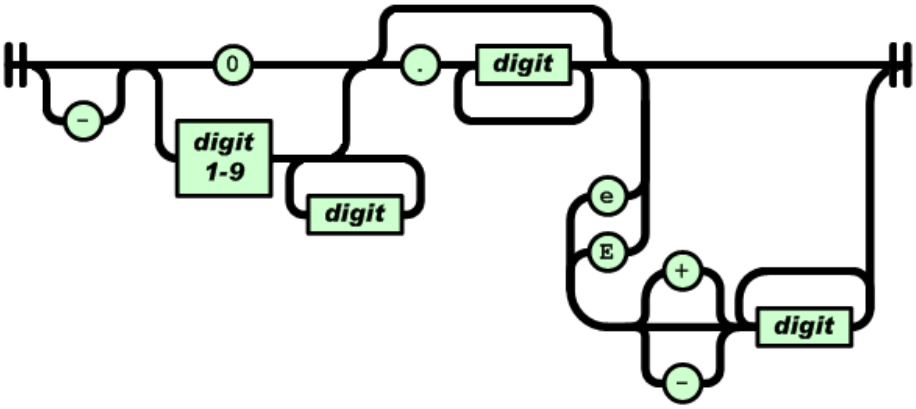


Figura 4.12 Json número

4.3.5 XML

XML, siglas en ingles eXtensible Markup Language, es un lenguaje de marcas desarrollado por el W3C. XML es una tecnología sencilla que tiene a su alrededor otra

que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Actualmente tiene un papel importante ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Estas son algunas de las ventajas de XML:

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.

- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones.

- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.

- Transformamos datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.

Android hace un fuerte uso de XML y recomienda usar XmlPullParser, el cual es una eficiente manera de parsear XML en Android. Históricamente Android ha tenido dos implementaciones de esta interface.

- KXmlParser via XmlPullParserFactory.newPullParser().

- ExpatPullParser, via Xml.newPUllParser()

En la documentación oficial podemos encontrar ejemplos de uso. [19]

4.3.6 Conceptos fundamentales de una aplicación Android

Toda aplicación en Android tiene que hacer uso, en mayor o menor cantidad, de los siguientes conceptos fundamentales.

4.3.6.1 Activities

Una Activity representa una pantalla para la interfaz de usuario, está compuesta por un conjunto de ficheros, entre los que se encuentra la definición de una clase que representa el corazón de una Activity. Esta clase es una subclase de *Activity,* una clase que en el SDK está definida con muchos atributos y con métodos que controlan el ciclo de vida de dicha Activity.

El ciclo de vida de una Activity está compuesto por un flujo de ejecución y unos estados que le otorgan un comportamiento, en la siguiente **Figura 4.13 Ciclo de vida Activity** se puede ver un esquema de su funcionamiento.

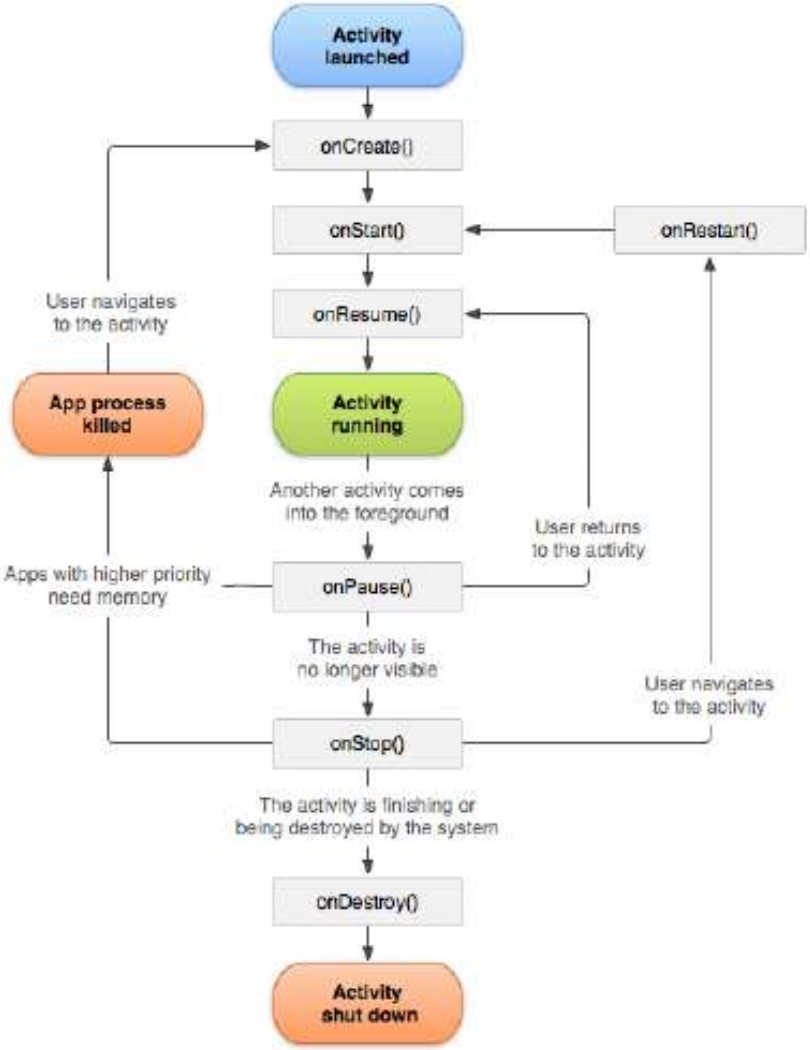


Figura 4.13 Ciclo de vida Activity

Los métodos que controlan el comportamiento de este ciclo de vida son:

onCreate(): Este método es llamado cuando una Activity se ejecuta por primera vez.

onRestart(): Se llama después de que una Activity haya sido parada, justo antes de que se inicialice de nuevo, y siempre seguido por onStart().

onStart(): Este método es llamado después de onCreate() o después de onRestart() cuando la Activity ha sido parada pero va a ser otra vez mostrada al usuario.

onResume(): Se llama antes de que la Activity interactúe con el usuario. En este momento la Activity está en el punto más alto de la pila de Activities. Siempre precede a onPause().

onPause(): Se llama cuando el sistema va a reanudar otra Activity. Este método se utiliza habitualmente para hacer los cambios no guardados en los sistemas de persistencia de datos, parar animaciones y otras cosas que puedan consumir CPU, etc. Estas acciones se deben de crear rápidamente porque la siguiente Activity no será creada hasta que esta regrese. En situaciones donde el sistema necesite más memoria, podrá eliminar los procesos pausados para liberar recursos.

onStop(): Se llama a este método cuando la Activity no es visible al usuario durante mucho tiempo. Precede a onRestart() si la Activity vuelve para interactuar con el usuario on onDestroy si la Activity es eliminada.

onDestroy(): Este método se llama antes de que la Activity sea destruida. Esta es la última llamada que se realiza para una Activity.

4.3.6.2 Services

Un servicio es un componente que mejora el rendimiento de operaciones largas en segundo plano y no provee una interfaz de usuario. Otro componente puede arrancar un servicio y este continuara en segundo plano incluso si el usuario cambia a otra aplicación. Además un componente puede enlazar con un servicio para interactuar con el incluso realizar interprocesos de comunicación. Por ejemplo un servicio puede manejar transacciones de red, reproducir música, todo desde el backgroud.

Un servicio puede adoptar dos formas:

Started: Un servicio esta “started” cuando un componente lo lanza ejecutando una llamada a “startService”. Una vez lanzado, un servicio puede correr en segundo plano indefinidamente, incluso si el componente que lo ha lanzado se destruye. Normalmente, un servicio en ejecución realiza operaciones simples y no devuelve ningún resultado al componente que lo ha inicializado.

Bound: Un servicio esta “bound”, limitado, cuando un componente se une a él haciendo una llamada al método bindService(). Un servicio en este estado ofrece una

interfaz cliente servidor que permite a los componentes interactuar con el servicio, enviar peticiones, obtener resultados. Un servicio "bound" corre solo mientras otro componente está ligado a él. Muchos componentes pueden unirse o enlazarse al servicio una vez, pero cuando todos los componentes se han desligado, el servicio será destruido.

En la documentación oficial podemos encontrar estas definiciones por separado aunque realmente se puede trabajar con las dos a la vez, simplemente depende de si implementamos o no los dos métodos; "onStartCommand()" que permite a los demás componentes lanzar este servicio y "bind()" que permite enlazarlos.

Independientemente de si la aplicación es started , bound o ambas, un componente puede usar el servicio (incluso desde una aplicación independiente) de la misma manera que un componente puede usar una activity, iniciándola con un Intent. Sin embargo puedes declarar un servicio como privado, en el fichero manifest, y bloquear el acceso desde otra aplicación.

4.3.6.2.1 Conceptos básicos.

Para crear un servicio es necesario crear una subclase de "Service" y sobrescribir algunos métodos que manejan aspectos claves del ciclo de vida del servicio y proveen un mecanismo para unir componentes al servicio si es necesario. Los métodos más importantes que debemos sobrescribir son:

- onStartCommand():

El sistema llama a este método cuando otro componente, como una actividad, pide que comience el servicio, llamando "startService()". Una vez se ejecuta este método, el servicio comienza y puede correr en background indefinidamente. Si tu implementas onStartCommand() es tu responsabilidad parar el servicio cuando se haya llevado a cabo su labor, llamando a stopSelf() or stopService().

- onBind():

El sistema llama a este método cuando otro componente se quiere enlazar con el servicio, llamando a bindService(). En la implementación de este método, debes proveer una interfaz que los clientes utilizan para comunicarse con el servicio,

devolviendo una instancia de la interfaz IBinder. Se debe implementar siempre este método, pero si no quieres permitir enlazado, entonces debes devolver un null.

- onCreate():

El sistema llama a este método cuando el servicio es creado por primera vez, para realizar el proceso de configuración, antes de las llamadas a los métodos onStartCommand() y onBind(). Si el servicio ya se está ejecutando, este método no ejecuta.

- onDestroy()

El sistema llama a este método cuando el servicio ya no se ejecuta y está siendo destruido. Tu servicio debería implementar este método para limpiar cualquier recurso. Esta es la última llamada que recibe el servicio.

Si un componente lanza el servicio llamando al método startService(), este estará en ejecución hasta que el mismo ejecute la llamada a stopSelf() u otro componente lo pare ejecutando la llamada a stopService().

Si un componente realiza una llamada al método bindService() para crear el servicio, entonces este solo se ejecutará siempre que el componente esté enlazado a él. Una vez el servicio se desenlace de todos los clientes, el servicio se destruirá.

4.3.6.2.2 Declarar un servicio en el manifest

Como las Activities y otros componentes, debes declarar todos los servicios en fichero manifest de tu aplicación.

```
<manifest ... >
  <application ... >
     <service android:name=".ExampleService" />
  </application>
</manifest>
```

El único atributo obligatorio es “Android:name”, el cual especifica el nombre de la clase del servicio.

4.4 Funcionalidad

En este capítulo se detalla la funcionalidad de la aplicación explicando que se puede hacer en cada una de las situaciones que nos podemos encontrar en el sistema.

4.4.1 Navegación entre estancias

La primera funcionalidad principal con la que nos encontramos es la navegación entre estancias, esta nos permite recorrer todos los lugares de la casa usado los comandos especificados para ello en la tabla del apartado 4.5. Podemos recorrer la casa de un modo lineal, navegando de una habitación a otra, solamente diciendo el nombre de la habitación. También podemos recorrer la vivienda en un modo de jerarquía, como por ejemplo, pedir al sistema que nos lleve al primer piso y una vez allí mediante el comando de ayuda pedir al sistema que nos muestre que lugares son accesibles desde el primer piso. O cogiendo como otro ejemplo, estar en una habitación y mediante el comando especificado para ello pedir al sistema que salga de la habitación el cual nos llevará a su estancia “padre” de donde nos encontramos.

En la **Figura 4.14 Pantalla principal** nos encontramos en la pantalla principal y le indicamos a la aplicación mediante el comando **"ir a casa"** que se sitúe mediante navegación en la vista principal de la casa.

Figura 4.14 Pantalla principal

El sistema recoge la orden y nos muestra las ubicaciones colindantes para poder navegar mediante jerarquía, **Figura 4.15 Vista estancia casa**.

Figura 4.15 Vista estancia casa

Y además en esta misma interfaz los dispositivos que se encuentran en esta estancia para poder interactuar con ellos **Figura 4.16 Dispositivos disponibles**.

Figura 4.16 Dispositivos disponibles

Como ejemplo indicamos al sistema mediante el comando **"ir a Cocina"** que navegue hacia ese punto de la vivienda. En este punto la aplicación nos muestra la situación de la estancia indicada, un mensaje indicando que no tenemos más estancias a las que navegar: **"Esta estancia no tiene subestancias"** mediante jerarquía y si que nos muestra los dispositivos que se encuentran en esta estancia **Figura 4.17 Estancia cocina**

Figura 4.17 Estancia cocina

Una vez llegados a este punto podemos profundizar, como se muestra en el capítulo 4.4.2, en los dispositivos que contiene esta estancia.

4.4.2 Listado de dispositivos de una estancia

Cuando nos encontramos en una estancia podemos indicar al sistema mediante el comando **"listar dispositivos"** que muestre los dispositivos que se encuentran en esta estancia, **Figura 4.18 Listado de dispositivos de una estancia**. Esta interfaz nos muestra en que estancia nos encontramos, el nombre del dispositivo y las acciones que podemos realizar sobre ella misma. Dado que esta interfaz nos muestra un listado de todos los dispositivos para poder interactuar específicamente con uno de ellos, debemos acceder al detalle que este nos ofrece, como se explica en el punto 4.4.4.

Figura 4.18 Listado de dispositivos de una estancia

Para acceder al detalle de un dispositivo tenemos que indicar el nombre del dispositivo que al que queremos acceder. **"luz cocina".**

4.4.3 Navegación por tipos de dispositivos

Para poder manejar todos los dispositivos de un mismo tipo a la vez, por ejemplo las luces, se ha desarrollado la funcionalidad de poder controlar estos dispositivos rápidamente. Como se puede ver en la **Figura 4.19 Navegación por tipo de dispositivos**, mediante el comando **"listar luces"** nos muestra todos los dispositivos de tipo luces que hay en la vivienda y en qué estado están, encendidas o apagadas.

Figura 4.19 Navegación por tipo de dispositivos

En este punto de la aplicación podemos apagar todas las luces con un solo comando, **"apagar luces"** como muestra la **Figura 4.20 Todos los dispositivos luces apagados.**

Figura 4.20 Todos los dispositivos luces apagados.

4.4.4 Detalle de un dispositivo

La tercera funcionalidad se centra en los dispositivos, mostrando los detalles que este nos ofrece. Podemos pedir al sistema que nos indique que acciones podemos realizar con este dispositivo o ver en qué estado se encuentra, indicando el nombre del dispositivo, por ejemplo **"luz cocina"**. En la **Figura 4.21 Detalle dispositivo luz apagada**, vemos las características en detalle de este dispositivo, que acciones se pueden realizar sobre él y en qué estado se encuentra.

Figura 4.21 Detalle dispositivo luz apagada

Podemos cambiar el estado de este dispositivo con el comando **"luz cocina encender"** y como vemos en la **Figura 4.22 Detalle dispositivo luz encendida**, nos muestra el estado actual del dispositivo y las acciones que podemos realizar.

Figura 4.22 Detalle dispositivo luz encendida

4.4.5 Escenas

Las escenas son conjuntos de acciones definidas por el usuario que se pueden ejecutar definiendo un único comando, o de forma automática cuando se produce una determinada condición.

En el desarrollo de la aplicación se ha definido el comportamiento de dos escenas:

- Escena cine.
- Escena detección de presencia.

Estas escenas ya están definidas en la aplicación y en esta versión, el usuario no tiene posibilidad de modificarlas. Únicamente puede activarlas o desactivarlas.

4.4.5.1 Escena cine

La escena cine conlleva una configuración predefinida previamente, para así evitar, en uno de los casos más comunes, que tengamos que recorrer toda la vivienda o todos los dispositivos que queramos ajustar.

Esta escena realiza una serie de acciones habituales que normalmente se desarrollan para ver cine cómodamente en el salón de la casa.

La primera acción consiste en bajar la intensidad de la luz, poniendo esta intensidad adecuadamente para ver la televisión. Las persianas se bajan hasta un 90% para así evitar todo tipo de reflejos. Y por último ajustamos la temperatura a 25 grados.

Podemos ver que mediante un solo comando podemos ajustar varios dispositivos acorde a nuestras necesidades.

La imagen que nos aparece en la aplicación es la que se muestra en la **Figura 4.23 Escena cine**.

Figura 4.23 Escena cine

Mientras vemos esta imagen el sistema nos avisa de los cambios que se van a realizar.

- Las luces se quedan encendidas al 20%.
- Las persianas se bajan un 90%.
- La temperatura se ajusta a 25 grados.

4.4.5.2 Escena detección de presencia

La escena detección de presencia pone en funcionamiento todos los dispositivos relacionados con la seguridad que se han definido previamente.

Esta escena consiste en la simulación de personas dentro de la vivienda. Un ejemplo de una situación real que podríamos poner sería: al detectar una presencia en los alrededores de la vivienda por ejemplo mediante el sensor de movimiento ubicado en el exterior. Para evitar un posible intento de robo, el sistema encendería las luces, bajaría las persianas y conectaría el sonido de la alarma.

Con esto estaríamos simulando que dentro de la vivienda hay gente, con el fin de ahuyentar a las personas que estén intentando entrar.

La **Figura 4.24 Escena detección de presencia**, muestra la interfaz mientras se activa esta escena.

Figura 4.24 Escena detección de presencia

La configuración que se ha realizado para esta escena es:

- Luces encendidas al 100 %.
- Persianas bajadas al 100%.

- Sonido de alarma conectado.
- Luz alarma activada.

4.5 Resumen de comandos

En este capítulo vamos a mostrar los comandos que podemos utilizar para manejar la aplicación en una tabla, agrupados por funcionalidades.

Navegación	
Comando	***Descripción***
Ir a *+ estancia* **Acceder** **Entrar**	Este comando permite navegar hacia una estancia específica.
Volver	Permite volver a una estancia "padre", es decir, la estancia de la vivienda colindante que precede a esta.

Ayuda	
Comando	***Descripción***

Ayuda **Comandos posibles** **Posibles comandos**	Estos comando los podemos utilizar en cualquier parte de la aplicación y nos dice los comandos que podemos utilizar para realizar las diferentes acciones.

Listados	
Comando	***Descripción***
Listar localizaciones **Listar estancias** **Listas habitaciones**	Este comando permite al sistema listar todas las localizaciones de la vivienda
Listar dispositivos	Con este comando podemos listar todos los dispositivos que contiene una estancia.

Escenarios	
Comando	***Descripción***
Escena + *tipo de escena*	Con este comando podemos activar las escenas que tengamos predefinidas en el sistema. Ejemplo: Escena cine.

Comandos dispositivos	
Comando	***Descripción***
Luz + localización + acción	Ejemplo: luz cocina encender, luz cocina apagar.
Encender + dispositivo	Este comando se usa para encender un dispositivo.
Apagar + dispositivo	Con este comando podemos apagar un dispositivo.
Aumentar + dispositivo	Este comando se usa para aumentar dispositivos graduales. Ejemplo: Luz gradual
Disminuir + dispositivo	Este comando se usa para disminuir dispositivos graduales. Ejemplo: Luz gradual

4.6 Subsistemas desarrollados

En este capítulo se describen en detalle cada uno de los subsistemas desarrollados de este trabajo.

4.6.1 Activities

A continuación se muestra como se ha desarrollado la implementación de la clase (Activity) que soporta la lógica del sistema de reconocimiento de voz. Abstrayendo los algoritmos de reconocimiento de voz a un nivel superior, conseguimos el objetivo de que nuestra aplicación sea reutilizable y adaptable a otras aplicaciones. Bien por qué solo necesiten el sistema de reconocimiento de voz o por qué además del reconocimiento de voz necesitan el sistema de navegación.

4.6.1.1 Base Activity

En esta Activity es en la que recae el peso del reconocimiento de voz, eliminando la responsabilidad a las Activities que hereden de ella pero a la vez proporcionándoles la capacidad de Reconocimiento de voz. Esta clase evita la duplicidad de código haciéndola más mantenible y desacoplada.

```
public class BaseActivity extends Activity
```

El método onCreate de la clase BaseActivity obtiene la instancia del objeto singleton Navigator.

```
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

       nav = Navigator.getInstance();

}
```

Este método comprobamos que el reconocimiento de voz esté disponible para evitar fallos. Este método está descrito en el capitulo Voice Recognition

```
public void checkVoiceRecognition() {}
```

El método speak() lanza el intent configurado para el reconocimiento de voz , que lo usaran todas las Activities que requieran esta funcionalidad desarrollado en capítulo Speech engine *4.6.2*

```
public void speak(View view) {}
```

onActivityResult() es el encargado de recoger la información procedente del intent de reconocimiento de voz y transmitirla.

```
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
     if (requestCode == VOICE_RECOGNITION_REQUEST_CODE)

      //If Voice recognition is successful then it returns RESULT_OK
      if(resultCode == RESULT_OK) {

       ArrayList<String> textMatchList =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);

       if (!textMatchList.isEmpty()) {

             if (textMatchList.get(0).contains("search")) {
                String searchQuery = textMatchList.get(0);
                searchQuery = searchQuery.replace("search","");
                Intent search = new
Intent(Intent.ACTION_WEB_SEARCH);
                search.putExtra(SearchManager.QUERY, searchQuery);
                startActivity(search);
             }
             else
             {
                   String primerMatch = textMatchList.get(0);
                   textOperation(primerMatch);

             }

       }
      }else if(resultCode == RecognizerIntent.RESULT_AUDIO_ERROR){
       [...]
      }
     super.onActivityResult(requestCode, resultCode, data);
    }
```

Se ha creado una plantilla para mostrar el botón speak que lanza el reconocimiento de voz en un fichero xml el cual lo incluirán las demás Activities en sus layouts.

```
<Button
    xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/btSpeak"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="speak"
    android:text="@string/btSpeak"
    android:typeface="serif"
    tools:context=".VoiceRecognitionActivity" />
```

Con esto evitamos tener que duplicar código en todas las Activities y en los layouts nos queda un código más limpio.

4.6.2 Speech engine

Este motor es el que proporciona la capacidad a una aplicación Android de reproducir vocalmente, es una característica que está disponible desde casi el comienzo de este sistema operativo, versión 1.2.

Este componente se ha desarrollado mediante un Service ya que debía estar disponible en todos los puntos de la aplicación y no requiere de una interfaz visual.

```
public class SpeechService extends Service implements
OnInitListener{
```

SpeechService es el Service que se ha creado para otorgar esta capacidad al sistema, debemos implementar la interfaz OnInitListener y rescribir el método onInit() que indica que el motor TextToSpeech se ha completado y está disponible para usarse.

```
      public void onInit(int arg0) {

        Locale loc = new Locale("es", "","");

            if(speakEngine.isLanguageAvailable(loc) >=
TextToSpeech.LANG_AVAILABLE){
                  speakEngine.setLanguage(loc);
                  speak();
              }
      }
```

Al tratarse de un servicio, el primer método que se ejecuta es el método onStartCommand el cual aprovechamos para recoger los datos que provienen del Intent, analizarlos y prepararlos para pasárselos al motor de voz.

```
      public int onStartCommand(Intent intent, int flags, int
      startId){
            mns = intent.getStringExtra("mensaje");
            if(mns == null){
                  mnsArray =
intent.getStringArrayListExtra("mensaje");
                  siArray = true;
            }
            speakEngine = new TextToSpeech(this, this);

            return START_NOT_STICKY;
      }
```

Y finalmente el método el método que lanza este motor es el método speak() el cual llama a otro método speech de la clase TextToSpeech.

```
private void speak() {

            if(siArray)
            {
                  for(String sms:mnsArray)
                  {
                        speakEngine.speak(sms,
                  TextToSpeech.QUEUE_ADD, null);
                  }
            }
            speakEngine.speak(mns, TextToSpeech.QUEUE_ADD, null);


      }
```

4.6.3 Voice Recognition

Voice Recognition es capaz de reconocer nuestra voz y convertirla en texto o realizar las acciones apropiadas. Lo primero que hacemos es comprobar que el dispositivo tiene esta capacidad disponible para evitar fallos en la aplicación.

```
  public void checkVoiceRecognition() {
        // Check if voice recognition is present
        PackageManager pm = getPackageManager();
        List<ResolveInfo> activities = pm.queryIntentActivities(new
Intent(
          RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);
        if (activities.size() == 0) {
         Toast.makeText(this, "Voice recognizer not present",
           Toast.LENGTH_SHORT).show();
        }
   }
```

Podemos acceder a Voice Recognition mediante un RecognizerIntent, creando un Intent de este tipo, pasándole los parámetros necesarios y llamando al método startActivityForResult() con el Intent anteriormente creado.

```
Intent intent = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);


        // Specify the calling package to identify your application
        intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
getClass()
          .getPackage().getName());


        //There are two form of language model available
        //1.LANGUAGE_MODEL_WEB_SEARCH : For short phrases
        //2.LANGUAGE_MODEL_FREE_FORM  : If not sure about the words
or phrases and its domain.
```

Internamente voice recognition se comunica con el servidor y obtiene los resultados. Hasta la versión de Android Jelly Bean (Api level 16) es necesario el acceso a internet para realizar el reconocimiento de voz. Una vez realizado el reconocimiento de voz se recoge el resultado en el método onActivityResult().

Es necesario otorgar permisos de INTERNET a nuestra aplicación en el Fichero Manifest.

4.6.4 Interfaz Navigation

Para hacer la aplicación más mantenible y escalable se ha definido la interfaz Navigation, la cual recoge la funcionalidad de qué debe de hacer esta clase. El aspecto fundamental de la interfaz es separar la especificación de qué tiene que hacer esta clase respecto del cómo se desarrolle.

```
public interface Navigation {}
```

4.6.5 Clase Navigator

La clase Navigator implementa la interfaz Navigation es la que nos dará la capacidad de navegar por la vivienda y de saber en todo momento donde nos encontramos.

Para la implementación de esta clase se ha utilizado el Patrón Singleton.

La clase Navigator está compuesta por los siguientes atributos:

- Una referencia a la instalación de la vivienda.
- Una referencia a las interfaces de la vivienda.
- Una referencia a la estancia actual en la que nos encontramos.
- Una referencia al dispositivo actual en que nos encontramos.

```
protected Installation installation;
protected ArrayList<Interface> interfaxes = null;
protected Location actual;
protected Device deviceActual;
```

4.6.6 Desarrollo e implementación del Patrón Singleton.

En el desarrollo de nuestra aplicación se ha implementado este patrón mediante la clase “Navigator”:

```
private static Navigator INSTANACE = null;

private synchronized static void createInstance() {
      if(INSTANCE == null)
            INSTANCE = new Navigator();
}

public static Navigator getInstance() {
      if(INSTANCE == null)
            createInstance();
      return INSTANCE;
}
```

Esta clase nos proporciona la capacidad de navegar por la vivienda y de saber en cada momento donde nos encontramos. Garantizando que esta clase solo tenga una instancia, todos los dispositivos podrán compartir nuestra ubicación actual.

Para que nuestra clase *Navigator* adapte el comportamiento de este patrón se han seguido los siguientes conceptos:

1. Ocultar el constructor de la clase *Navigator*, para que los clientes no puedan crear instancias.

2. Declarar en la clase *Navigator* una variable privada que contenga la referencia a la instancia única que queremos gestionar.

3. Proveer en la clase *Navigator* una función o propiedad que brinde acceso a la única instancia gestionada por el Singleton. Los clientes acceden a la instancia a través de esta función o propiedad.

Se podría pensar que pueden utilizarse clases con miembros estáticos para el mismo fin, pero los resultados no son los mismos, ya que en este caso, la responsabilidad de tener una única instancia recaería en el cliente de la clase. El patrón Singleton hace

que la clase sea responsable de su única instancia, quitando así este problema a los clientes.

4.6.7 Utilización de un servicio REST

A continuación se muestra como desde la aplicación utilizamos un servicio Rest que actúa sobre un dispositivo de la vivienda.

4.6.8 Fichero Manifest

El fichero manifest presenta la información esencial sobre la aplicación, describiendo el nombre del paquete Java para la aplicación, los componentes del sistema; Activities, services, también los permisos de la aplicación y los permisos para interactuar con otras aplicaciones.

En la primera parte del fichero vemos el nombre del paquete de nuestra aplicación y las versiones que soporta.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.appv2"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="4"
        android:targetSdkVersion="15" />
```

Una de las partes más importantes es la asignación de permisos a la aplicación. Para poder realizar la función de reconocimiento de voz, es indispensable otorgar permisos de acceso a internet.

```
    <!-- Permissions -->
    <uses-permission android:name="android.permission.INTERNET" />
```

Para la utilización del servicio del motor de voz que se ha desarrollado, se ha declarado el servicio en el fichero manifest como se ve a continuación.

```
<service android:name="voice.SpeechService" />
```

5 Caso de estudio

En este capítulo, se presentan dos casos de estudio para ver la funcionalidad del sistema de reconocimiento de voz diseñado en esta Tesina de Máster en un sistema domótico aplicado a un entorno real. Se desarrollan como casos prácticos dos escenarios que un usuario se puede encontrar en una situación común, para que así el lector, pueda entender mejor como este trabajo está siendo aplicado.

5.1 Introducción

Alberto, tiene instalado en su hogar un sistema domótico correctamente configurado como el presentado en este trabajo y el cual dispone de los siguientes elementos domóticos:

Nombre	Tipo	Ubicación
AirConditioning	ACTUATOR	Casa
AlarmLighting	ACTUATOR	Casa
IndoorThermometer	SENSOR	Oficina
IndoorAlarmSound	ACTUATOR	Casa
GardenThermometer	SENSOR	Jardín
GradualLighting	ACTUATOR	Habitación
Blind	ACTUATOR	Habitación

ElectroValve	SENSOR	Casa
KitchenLighting	ACTUATOR	Cocina
HallLighting	ACTUATOR	Recibidor
BrightnessMeter	SENSOR	Jardín
IndoorMovementDetector	SENSOR	Oficina
Fan	ACTUATOR	Oficina
ParkingDoor	ACTUATOR	Garaje
PresenceDetector	ACTUATOR	Casa
Heating	ACTUATOR	Casa

Figura 5.1 Dispositivos de la vivienda

En su dispositivo, Alberto tiene instalado el software necesario, dispone del sistema operativo Android 4.1.2. Además dispone de la arquitectura necesaria para poder usar este sistema tanto en un entorno local como en un entorno remoto fuera de casa.

5.2 Caso de estudio "Salir de vacaciones"

Alberto se dispone a salir de vacaciones con su familia y durante este tiempo la casa permanecerá vacía. Por lo que decide configurar los elementos domóticos necesarios de la vivienda para asegurarse que no tendrá ningún problema en su casa mientras está fuera.

En el momento de abandonar la casa Alberto decide cerrar las válvulas de agua, para evitar posibles fugas y con ello posibles inundaciones. También quiere apagar todas las

luces de la vivienda que pudieran estar encendidas y además bajar las persianas completamente.

A continuación en la **Figura 5.2 Situación inicial caso de estudio** podemos ver el estado en el que se encuentran los elementos domóticos con los que se interactúa en este caso de estudio antes de abandonar la casa.

Nombre	Estado inicial
KitchenLighting	 Encendida
ElectroValve	 Abierta
HallLighting	 Encendida
Blind	 40% Cerrada

Figura 5.2 Situación inicial caso de estudio

Se muestra como por un descuido la luz de la cocina y la luz del recibidor se han quedado encendidas. La persiana de la habitación se encuentra un 40% cerrada, y la electroválvula que controla el paso de agua se encuentra abierta.

Alberto, se dispone a configurar su sistema domótico, para ello coge su Smartphone y realiza la siguiente acción **Figura 5.3 Acción 1 apagar electroválvulas**:

Figura 5.3 Acción 1 apagar electroválvulas

Envía la orden al dispositivo de apagar la electroválvula, este recoge la acción y la realiza, el resultado es cierre total del sistema de agua de la vivienda.

En la siguiente tabla podemos ver el estado final en el que se encuentra este dispositivo domótico, mientras que el resto de elementos mantienen su estado.

Nombre	Estado Previo	Estado final
ElectroValve	Abierta	Cerrada

Figura 5.4 Estado previo y final electroválvula

Para continuar con la configuración, Alberto tiene la posibilidad de poder recorrer toda la vivienda e ir apagando luces en cada estancia, o utilizar un manejo general de navegación por tipo de dispositivos que también dispone la aplicación.

Alberto lo que necesita es apagar todas las luces, por lo que decide apagar todos los dispositivos de este tipo con un solo comando.

Figura 5.5 Acción 2 apagar luces

El sistema muestra el resultado de la acción realizada, todos los dispositivos de tipo luz que hay en la vivienda y en el estado final que están **Figura 5.6 Estado previo y final luces.**

Nombre	Estado Previo	Estado final
KitchenLighting	Encendida	Apagada
HallLighting	Encendida	Apagada

Figura 5.6 Estado previo y final luces

Por último solo falta bajar la persiana de la habitación que se ha quedado subida. Alberto se da cuenta de este detalle y decide bajar la persiana completamente **Figura 5.7 Acción 3 cerrar persiana.**

Figura 5.7 Acción 3 cerrar persiana

La siguiente tabla muestra el detalle del estado previo de la persiana y el estado final, después de realizar la acción.

Nombre	Estado Previo	Estado final
Blind	40% Cerrada	100% cerrada

Figura 5.8 Estado previo y final persiana

Una vez que Alberto se ha asegurado que todos los elementos de la casa están bien cerrados y/o apagados, puede comenzar tranquilamente las vacaciones.

El estado final en el que quedarían los elementos domóticos de la vivienda lo podemos ver en la ¡Error! No se encuentra el origen de la referencia. siguiente.

Nombre	Estado Final
KitchenLighting	 Apagada
ElectroValve	 Cerrada
HallLighting	 Apagada
Blind	 100% cerrada

Figura 5.9 Situación final caso de estudio "Salir de vacaciones"

5.3 Caso de estudio "Al salir del trabajo"

Como segundo caso de estudio se muestra las acciones que realiza Alberto al salir del trabajo para que al llegar a casa se encuentre con la configuración y las comodidades de confort y seguridad que desea; un ajuste adecuado de la temperatura interior de la vivienda, desactivación del sistema de alarma. También está automatizada la apertura de la puerta del garaje para cuando se acerque con el coche mandar la acción necesaria para abrir la puerta.

A continuación, **Figura 5.10 Estado inicial "Al salir del trabajo"** se muestra una tabla con el estado inicial de los dispositivos.

Nombre	Estado inicial
Heating	 Apagada

PresenceDetector	 Activo
IndoorMovementDetector	 Activo
ParkingDoor	 Cerrada

Figura 5.10 Estado inicial "Al salir del trabajo"

Alberto, al salir del trabajo quiere que al llegar a casa la calefacción esté encendida, para conseguir una temperatura adecuada en la vivienda. Alberto coge su Smartphone, ejecuta la aplicación desarrollada, permanentemente conectada a su vivienda, y envía la orden de encender la calefacción.

Figura 5.11 Acción 1 encender calefacción

Tras realizar la acción, los elementos domóticos afectados, entran en funcionamiento para adecuar la casa a la temperatura adecuada. El estado final de los elementos

domóticos los podemos ver en la Figura 5.11 Acción 1 encender calefacción **Figura 5.12 Estado final encender calefacción**

Nombre	Estado Previo	Estado final
Heating	 Apagada	 Encendida

Figura 5.12 Estado final encender calefacción

Alberto está llegado a casa, y para una mayor comodidad quiere tener la puerta del garaje abierta así no tendrá que esperar mientras se esté abriendo.

Coge su dispositivo y envía la orden al sistema domótico para abrir la puerta del garaje.

Figura 5.13 Acción 2 abrir puerta garaje

La acción que acaba de realizar tiene la repercusión que se muestra en la **Figura 5.14 Estado final abrir puerta garaje.**

Nombre	Estado Previo	Estado final
ParkingDoor	 Cerrada	 Abierta

Figura 5.14 Estado final abrir puerta garaje

Una vez Alberto se encuentra en casa, justo antes de entrar necesita desconectar la alarma, que ha estado conectada durante todo el día. Como última acción Alberto, mediante su Smartphone manda la siguiente acción a su sistema domótico.

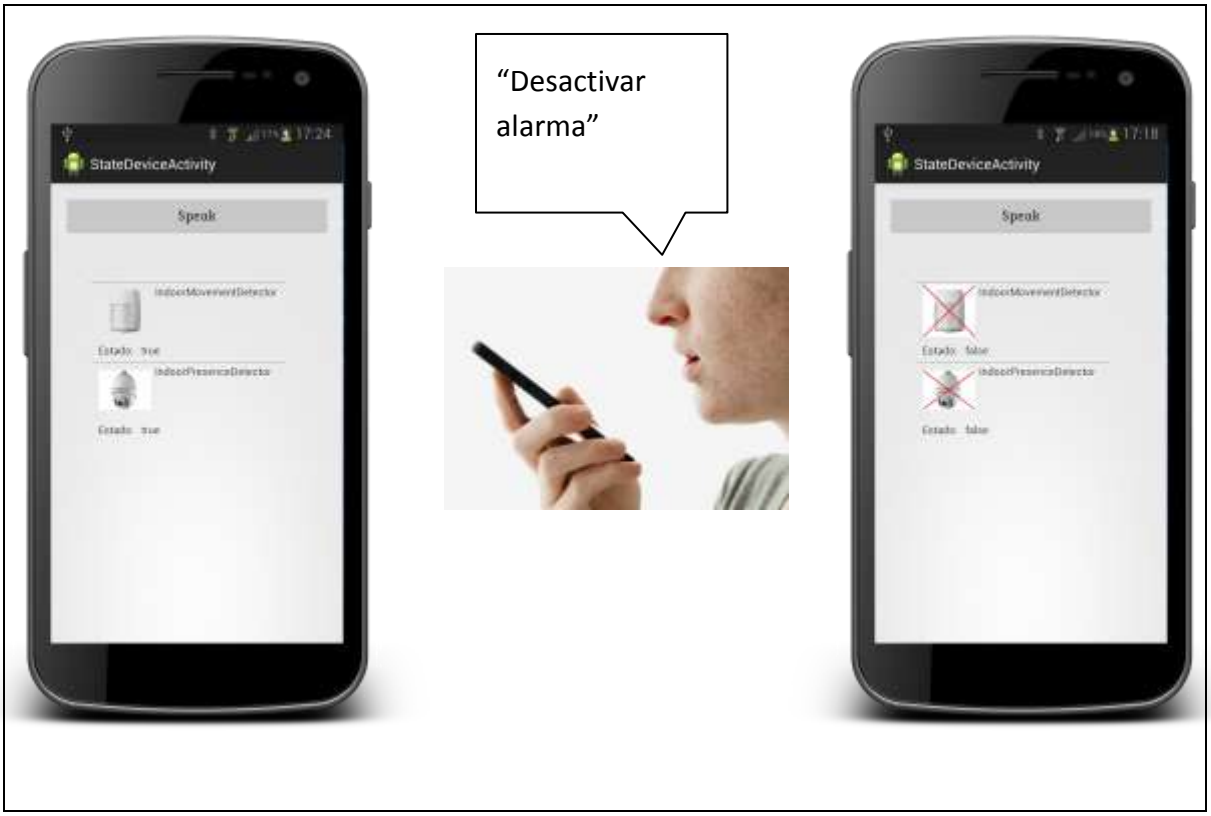


Figura 5.15 Acción 3 desactivar alarma

La acción que acaba de realizar lleva implícita la modificación de varios elementos domóticos, la modificación de estos se detallan en la siguiente tabla **Figura 5.16 Estado final apagar alarma.**

Nombre	Estado Previo	Estado final
PresenceDetector	 Activada	 Desactivada
IndoorMovementDetector	 Activada	 Desactivado

Figura 5.16 Estado final apagar alarma

Con estas acciones puede dar por concluida las acciones realizadas en una situación común de la vuelta del trabajo. Se puede comprobar la comodidad que ofrece un sistema domótico de estas características y el fácil manejo de la aplicación al tener unas interfaces sencillas y lo más importante un control total por reconocimiento de voz.

La situación final de los elementos domóticos sería la que se muestra en la siguiente tabla:

Nombre	Estado Final
Heating	 Encendida
PresenceDetector	 Desactivado
IndoorMovementDetector	 Desactivado
ParkingDoor	 Abierta

Figura 5.17 Situación final caso de estudio "Al salir del trabajo"

6 Conclusiones y trabajo futuro

Para finalizar el presente trabajo, el cual ha consistido en el desarrollo de un sistema de reconocimiento de voz para una aplicación de viviendas domóticas, se introduce las conclusiones del trabajo desarrollado y se proponen trabajos futuros para seguir avanzando en este proyecto.

6.1 Conclusiones

El reconocimiento de voz es, sin duda, una tecnología, que promete cambiar la forma en cómo interactuamos con las computadoras. Sin embargo, la tecnología aún no ha entrado en la etapa de ser una herramienta que pueda ser usada de forma dinámica por la gente. Todavía falta un poco de tiempo para que esta sustituya al teclado y al ratón, o al menos, para que modifique nuestras prácticas de computación. El reconocimiento de voz todavía tiene muchos defectos y limitaciones.

Estas limitaciones se basan en las deficiencias de la inteligencia artificial. Esta tecnología sirve como un traductor de comandos determinados, ya que las computadoras actualmente, la información que pueden recibir del contexto es limitada. Al mismo tiempo, el procesamiento del lenguaje es una disciplina bastante compleja. La realidad es que a una computadora, le es difícil procesar múltiples frases y reconocer los comandos fácilmente. La mayoría del software de reconocimiento de voz tiene que ser entrenado antes de ser usado para funcionar correctamente, para que pueda reconocer las órdenes y comandos que se le dicta. Sin embargo, se espera que en un futuro el software de reconocimiento de voz, sea una parte integral de las computadoras, no sólo en las industrias, sino también dentro de nuestros hogares.

El desarrollo de este trabajo ha supuesto un esfuerzo importante para familiarizarme con todo el ámbito de la tecnología de Android y que al final se ha obtenido el resultado que se planteó en un principio, la aplicación desarrollada es capaz de realizar

acciones dictadas por la voz y controlar todo el sistema domótico instalado en una vivienda.

Este trabajo no solo ha servido para aprender las tecnologías de Android, sino que además se ha diseñado una arquitectura de la aplicación que hace que esta se escalable con muy pocas modificaciones, y se pueda añadir funcionalidad de una forma fácil. También esta arquitectura posibilita un sistema de navegación fluido y eficaz sobre toda la vivienda.

Podemos decir, que esta Tesina Final de Máster, ha sido un trabajo completo en muchas facetas del Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información cursado en la Universidad Politécnica de Valencia.

6.2 Trabajo futuro

Como trabajo futuro se plantean tres líneas de mejora y nuevos ámbitos de adaptación usando el desarrollo de esta tesina.

La primera de ellas, consiste en la configuración de las escenas por parte del usuario. Actualmente las escenas están predefinidas y el usuario no tiene la posibilidad de cambiar los elementos que interactúan con cada escena. Sería interesante ofrecer al usuario una interfaz para poder crear, modificar o borrar las escenas. Esto permitiría que el usuario definiese por una parte nuevas escenas, que le permitiesen configurar el estado de un conjunto de dispositivos; o bien, personalizar las escenas ya predefinidas, como la de cine. Es importante destacar que la activación de este tipo de escenas continuaría requiriendo la activación por parte del usuario.

Un paso más en el tema de las escenas, sería configurar su ejecución en base a la información contextual del entorno. De este modo, se podrían configurar escenas dependientes de los sensores instalados en la vivienda: presencia, movimiento, humedad…, así como de información contextual como la fecha, hora, temperatura… Un ejemplo de este tipo de escena podría consistir en definir una escena *Lluvia* que bajara las persianas de la casa al recibir la detección de lluvia por parte del sensor correspondiente. Como vemos este tipo de escenas no requiere la intervención directa del usuario para su activación. La información contextual sería le responsable de lanzar su ejecución.

La definición de escenas sensibles al contexto conlleva mucha más dificultad que la simple activación de dispositivos en base a condiciones del entorno. ¿Qué comportamiento deberíamos tomar cuando las condiciones de activación son contrapuestas? Es decir, si una escena indica que debe realizarse una determinada acción y en el mismo momento se activa otra escena que implica la acción contraria. Podemos ilustrarlo fácilmente con un ejemplo, por una parte tenemos una escena muy simple llamada *Despertar* que implica únicamente levantar las persianas a las 8.00 de la mañana. Por otra parte, tenemos la anteriormente mencionada escena *Lluvia,* que implica bajar las persianas al detectar humedad. ¿Cuál debería ser la acción a realizar por el sistema si son las 8.00 de la mañana y además está lloviendo? Como vemos las escenas sensibles al contexto suponen una dificultad añadida que requieren de un estudio profundo para su definición, quedando propuesto como futuro trabajo de este proyecto.

Como última propuesta, se plantea poder usar el reconocimiento de voz en modo offline de Android, esta característica está disponible para desarrolladores a partir de la versión Android 4.1, Jelly Bean o superior. Durante el desarrollo de esta Tesina de Fin de Máster, Android pone a disposición de los desarrolladores el reconocimiento de voz offline. Con esta mejora no solo funcionaria más rápido ya que se evitaría el envío y recepción de datos, sino que además implica un ahorro en el tráfico de datos a nivel mensual. El reconocimiento de voz offline también ayudaría a la privacidad ya que los datos no saldrían de nuestro dispositivo.

Para ofrecer una visión más amplia e interesante del desarrollo del presente trabajo, se propone para un trabajo futuro, nuevos dominios de aplicación en los que poder adaptar los artefactos software relacionados con el reconocimiento de voz que componen esta tesina final de máster.

Un dominio de trabajo en el que se podría incorporar el reconocimiento de voz desarrollado, sería el sector de la hostelería. En este sector podemos incorporar aplicaciones para ofrecer el servicio de atención al cliente automático y además controlado por reconocimiento de voz. El cliente obtendría la comodidad y la rapidez para solicitar cualquier servicio y además a través de un medio tan natural como pedírselo a un camarero.

La adaptación del reconocimiento de voz desarrollado en este trabajo no sería muy difícil gracias al desarrollo por módulos seguido en esta tesina.

Otro dominio de trabajo en el cual el reconocimiento de voz se vuelve más complicado pero no menos interesante, es el mundo de los video juegos. Adaptar el reconocimiento de voz desarrollado, no sería una tarea trivial, dada la cantidad de órdenes y acciones que se podría manejar en una aplicación de este tipo.

Cuando podamos manejar eficazmente el reconocimiento de voz podremos, realizar acciones a la velocidad de los pensamientos.

7 Bibliografía

[1] Joint Application of Speech and Speaker Recognition for Automation and Security in Smart Home. Kong Aik Lee, Anthony Larcher, Helen Thai, Bin Ma, and Haizhou Li

[2] Definición domótica Real academia Española:
http://lema.rae.es/drae/?val=domotica

[3] Hacia el Modelado Conceptual de Sistemas Domóticos. J.Muñoz, J.Fons, V.Pelechano, O.Pastor

[4] Control del hogar digital desde dispositivos móviles. Jesús Friginal López, Juan Luis Posadas Yagüe, Juan Carlos Ruiz García, Jesús Camacho Villanueva

[5] KNX Association. [En línea] [Citado el: 02 de 11 de 2008.] http://www.knx.org/es/knx-estandar/estandardizacion/

[6] GeMMINi: Prototipado de interafces de usuario sobre múltiples dispositivos. Una estrategia basada en Líneas de Producto y MDD. Ignacio Mansanet, Joan Fons, Ismael Torres, Vicente Pelechano

[7] Estado del arte en el reconocimiento Automático de voz. Deiby Alexander Fandiño Rodríguez. Universidad Nacional de Colombia. 2005.

[8] Publicaciones oficiales de investigación de Google.http://research.google.com/

[9] Página oficial: http://www.openremote.org/display/HOME/OpenRemote

[10] Página oficial empresa BJ Adaptaciones: http://www.bj-adaptaciones.com/

[11] Kora, Control de entorno adaptable mediante dispositivos móviles, José Alcalá Correa, Ingeniería Informática, Universidad de Granada.

[12] Control domótico de vivienda INSTEON: http://www.insteon.com/

[13] Página oficial: http://sher.pa/

[14] Página oficial Apple: http://www.apple.com

[15] Página actualidad tecnológica:
http://gigaom.com/2013/05/07/why-the-time-has-come-for-android-home-to-finally-make-a-splash/

[16] IDC Worldwide Quarterly Tablet Tracker, March 2013

[17] Artículo oficial introductorio para desarrolladores sobre Text-To-Speech Android: http://android-developers.blogspot.com.es/2009/09/introduction-to-text-to-speech-in.html

[18] Gamma E., Helm, R., Johnson, R., Vlissides J.: Design Patterns: Elements of Reusable Object Oriented Software, Addison Wesley, 1995.

[19] Documentación oficial Android XML
http://developer.android.com/training/basics/network-ops/xml.html#choose

8 Referencias

- Android Programming Guide For Beginners: http://eduonix.com
- Android Speech Interface to a Home Robot July 2012: http://people.cs.missouri.edu/~reu/REU12/AndroidSpeechInterfacetoaHomeRobot/AndroidRobot/AndroidRobotPaper.pdf
- HTML5 versus Android: Apps or Web for Mobile Development? : Google I/O 2011: May 11
- Android developer: http://developer.android.com/index.html
- Android developer: http://developer.android.com/training/index.html
- La gestión integrada de servicios del hogar digital: el Sistema de Información del Hogar Digital (SIHD): Antonio Pereira Rama, Julián Chaparro Peláez Grupo de Ingeniería de Organización. E.T.S.I. de Telecomunicación. Universidad Politécnica de Madrid. apereira@gio.etsit.upm.es, chaparro@gio.etsit.upm.es
- Domótica y Hogar digital; Stefan Junestrand, Xavier Passaret, Daniel Vázquez.
- Asociación Española de Domótica http://www.cedom.es:
- Patron singleton msdn
  http://msdn.microsoft.com/es-es/library/bb972272.aspx
- Articulo reconocimiento de voz: http://www.articulosinformativos.com/Reconocimiento_de_Voz-a963743.html
- Artículo sobre KORA: http://www.gskbyte.net/kora/

- Reconocimiento de voz offline Android : http://sobreandroid.com/9785/llega-el-reconocimiento-de-voz-en-modo-offline-para-todas-las-apps-android/

- Tutorial Reconocimiento de voz Android http://code4reference.com/2012/07/tutorial-android-voice-recognition/

-

9 Anexo A: API servicio REST

9.1 Interacción con dispositivos tipo SWITCH

Switch on: http://localhost:8182/switch/X.X.X/SWITCHON

Switch off: http://localhost:8182/switch/X.X.X/SWITCHOFF

Ejemplo: http://localhost:8182/switch/1.0.1/SWITCHON

9.2 Interacción con dispositivos tipo Pulse:

Send pulse: http://localhost:8182/pulse/X.X.X

Ejemplo: http://localhost:8182/pulse/1.0.3/

9.3 Interacción con dispositivos tipo Movimiento Vertical (Persianas):

Move up: http://localhost:8182/vm/X.X.X/MOVEUP

Move down: http://localhost:8182/vm/X.X.X/MOVEDOWN

Stop: http://localhost:8182/vm/X.X.X/STOP

Ejemplo: http://localhost:8182/vm/0.0.10/MOVEUP

9.4 Interacción con dispositivos tipo Dimmer (Reguladores):

Regulador Porcentual: http://localhost:8182/dimmer/PERCENTUAL/X.X.X/VALUE_TO_SET {0-100}

Regulador Angular: http://localhost:8182/dimmer/ANGULAR/X.X.X/VALUE_TO_SET {0-360}

Regulador Hexadecimal: http://localhost:8182/dimmer/HEX/X.X.X/VALUE_TO_SET {0-255}

Ejemplo: http://localhost:8182/dimmer/HEX/1.0.5/217

9.5 Lectura de valor de un sensor:

http://localhost:8182/sensor/X.X.X/DATATYPE{INT, STRING, FLOAT, BOOL}

Ejemplo: http://localhost:8182/sensor/1.3.4/INT

10 Anexo B: Utilización de un servicio REST

```
private void runOperation(String operation) {

            List<InterfaceActionArgument> arguments = null;
            for (int op = 0; op <
interfaxe.getInterfaceActions().size(); op++) {

                  if
(interfaxe.getInterfaceActions().get(op).getName()
                              .compareTo(operation) == 0) {
```

```
                        arguments =
interfaxe.getInterfaceActions().get(op)
                                        .getArguments();
                        break;
                }
            }

            String deviceUrl = Server.getInstance().getUrlServer() +
interfaxe.getName()
                        + "/" + deviceName;

            HttpClient httpclient = new DefaultHttpClient();
            HttpResponse response;


            HttpEntity entity = null;

            HttpPut put = new HttpPut(deviceUrl);
            StringEntity entityString;
            try {
                  JSONObject jsonOb = new JSONObject();
                  JSONObject jsonOpParams = new JSONObject();
                  JSONArray jsonArrayArgs = new JSONArray();

                  for (int args = 0; args < arguments.size();
args++) {
                  //    JSONObject jsonObArgs = new JSONObject();
                  //    jsonObArgs.put("name",
arguments.get(args).getName());
                  //    jsonObArgs.put("value", newValue.getText());
                  //    jsonArrayArgs.put(jsonObArgs);
                  }

                  jsonOpParams.put("arguments", jsonArrayArgs);
                  jsonOpParams.put("name", operation);
```

```
                  jsonOb.put("operation", jsonOpParams);


                  Log.v("JSON-OBJ", jsonOb.toString());
                  entityString = new
StringEntity(jsonOb.toString());

      entityString.setContentType("application/json;charset=UTF-
8");// text/plain;charset=UTF-8
                  entityString.setContentEncoding(new
BasicHeader(HTTP.CONTENT_TYPE,
                              "application/json;charset=UTF-8"));
                  put.setEntity(entityString);
            } catch (Exception e1) {
                  e1.printStackTrace();
            }


            put.setHeader("content-type", "aplication/json");
            try {
                  response = httpclient.execute(put);
                  entity = response.getEntity();


                  if (entity != null) {
                        String respStr =
EntityUtils.toString(entity);
                        Log.v("Response", respStr);
                        ResponseDeviceType responseDT = new
ResponseDeviceType();
                        JSONObject jsonObject = new
JSONObject(respStr);

      responseDT.setResult(jsonObject.getString("result"));
                        responseDT.setResponseValues(jsonObject
                                    .getJSONArray("responseValues"));
```

```
                        LinearLayout linearLocations =
(LinearLayout) findViewById(R.id.ly);

                        TextView textView = new
TextView(currentContext);
                        textView.setText(respStr);
                        textView.setTextSize(18);
                        linearLocations.addView(textView);
                  }

            } catch (Exception e) {

                  e.printStackTrace();
            }

      }
```