Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Dra. Dª Macarena Espinillla Estévez

Y *Dr. D. Javier Medina Quero*, tutores del Proyecto Fin de Carrera titulado:

Desarrollo de servicios basados en sonido, voz y habla para interfaces Hombre-Máquina en ambientes inteligentes,

que presenta *Juan Carlos Navarrete Solana*, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2017

El alumno: Los tutores:

Juan Carlos Navarrete Solana

Macarena Espinilla Estévez

Javier Medina Quero

Agradecimientos:

La realización de este trabajo no hubiese posible sin la ayuda incondicional de algunas personas que han estado presente en todo momento de esta etapa ofreciendome su experiencia, cariño y apoyo.

En primer lugar, agradecer a mis tutores Javier Medina Quero y Macarena Espinilla Estévez y compañero Daniel Zafra por su dedicación y experiencia transmitida durante todo el proceso.

A mis padres y hermanos, los cuales me han apoyado y animado en todo momento siempre depositando confianza en mi en estos años de grado y a los que se lo debo todo. Y por ultimo también me gustaría mencionar a mis amigos, los que siempre han estado ahí cuando mas lo he necesitado . Gracias a todos.

Índice

Índice de Ilustraciones

Índice de tablas

1 INTRODUCCIÓN

1.1 Motivación

En la actualidad convivimos con una gran diversidad de capacidades tecnológicas, las cuales han sido creadas para facilitar y mejorar el trabajo de todos y cada uno de nosotros.

Existen muchas aplicaciones donde integrar elementos, tanto en el ámbito laboral como en el entorno doméstico existen desarrollos implementados para ayudarnos facilitar y automatizar la mayoría de aspectos que nos rodean.

Ilustración 1: Motivación

De forma reciente, se han realizado muchos esfuerzos de investigación para dotar de inteligencia a los dispositivos que nos rodean, algoritmos y procesos específicos de manera que dotan a estos dispositivos de una especie de inteligencia ambiental, por la cual son capaces de realizar tomas de decisiones de manera autónoma en base al contexto de aplicación e información recogida del ambiente en el que se encuentran.

La implementación de ambientes inteligentes está creciendo en empresas, oficinas, incluso en las propias viviendas dado que son capaces de reducir en consumo energético, vigilancia o seguridad. Los ambientes inteligentes aparecen con una intención de mejorar la calidad de vida de las personas con el propósito de

proporcionarle cierto bienestar e independencia. Por esta razón son creados entornos en los que los dispositivos son protagonistas y tienen la misión de asistir a los ocupantes en tareas cotidianas de manera inteligente y no invasiva. A estos entornos inteligentes se les ha proporcionado el nombre de *ambientes de vida asistida* (Ambient Assisted Living - AAL). [1][2]

En este trabajo de fin de grado se desarrollará un sistema de monitorización basado en el reconocimiento de voz y habla. Este proyecto estará formado por una arquitectura cliente-servidor, por la cual se podrán manejar de forma visual e intuitiva numerosos aspectos del hogar, como pueden ser dispositivos luminosos y electrodomésticos, aunque el desarrollo se ha realizado de forma modular y escalable para poder ser trasladado a muchos sensores y ambientes de forma sencilla. [1][2]

El proyecto constará de los siguientes elementos:

- **Sistema web de monitorización** de los diferentes componentes.
- **Reconocimiento de voz** es la interfaz de usuario, formada por componentes visuales e implementada mediante una API basada en java denominada Java Speech Recognition.
- **Persistencia** mediante un servidor de datos Java, para el registro de datos de los sensores, información de contexto para el reconocimiento (tokens, objetos y propiedades) utilizando para ello ficheros JSON.
- **La comunicación** entre los componentes anteriores se realiza mediante servicios REST, que pueden ser consumidos por cualquier plataforma. En este proyecto el back-end está realizado en java, y el cliente en Ajax-jquery.

Gracias a la interconexión de estos elementos podemos brindar la posibilidad de monitorizar a tiempo real la actividad de estos entornos de inteligencia ambiental.

1.2 Propuesta

El principal objetivo de este trabajo de fin de grado es la creación y desarrollo de un Sistema web para la monitorización de elementos de inteligencia ambiental combinado con Procesaminto de Lenguaje natural mediante el uso de un asistente de voz que permitirá el uso de los servicios mediante el uso del habla, así como la inclusión de un modulo de edición y configuración de elementos y valores de ajuste que permita una cierta escalabilidad para poder gestionar los elementos de forma fácil e intuitiva.

En el proyecto, pondremos en práctica un servicio de reconocimiento de voz, el cual nos permitirá la interacción del cliente con nuestro sistema mediante una API denominada Java Speech.

1.3 Objetivos

Los objetivos que aparecen al desglosar este trabajo de fin de grado son los siguientes:

1. Desarrollo de un prototipo de aplicación web que permita interactuar con los sensores actuadores del Entorno Inteligente, permitiendo cambiar su estado (encenderlos/apagarlos) desde una interfaz web.
2. Desarrollo de la funcionalidad Procesamiento de Lenguaje Natural (PLN). El PLN permite reconocer los objetos y propiedades del entorno y detectar las acciones que el usuario quiere realizar con los mismos.
3. Desarrollar los manuales de uso de dicho prototipo.
4. Implementación del prototipo en el Smart Lab de la Universidad de Jaén
5. Redactar una memoria que contenga el trabajo desarrollado.

1.4 Planificación

En este apartado mostraremos la estructura de trabajo propuesta en la figura 1.2 con la que podemos complementar junto con la tabla de tiempos reflejada en la tabla 1.1.

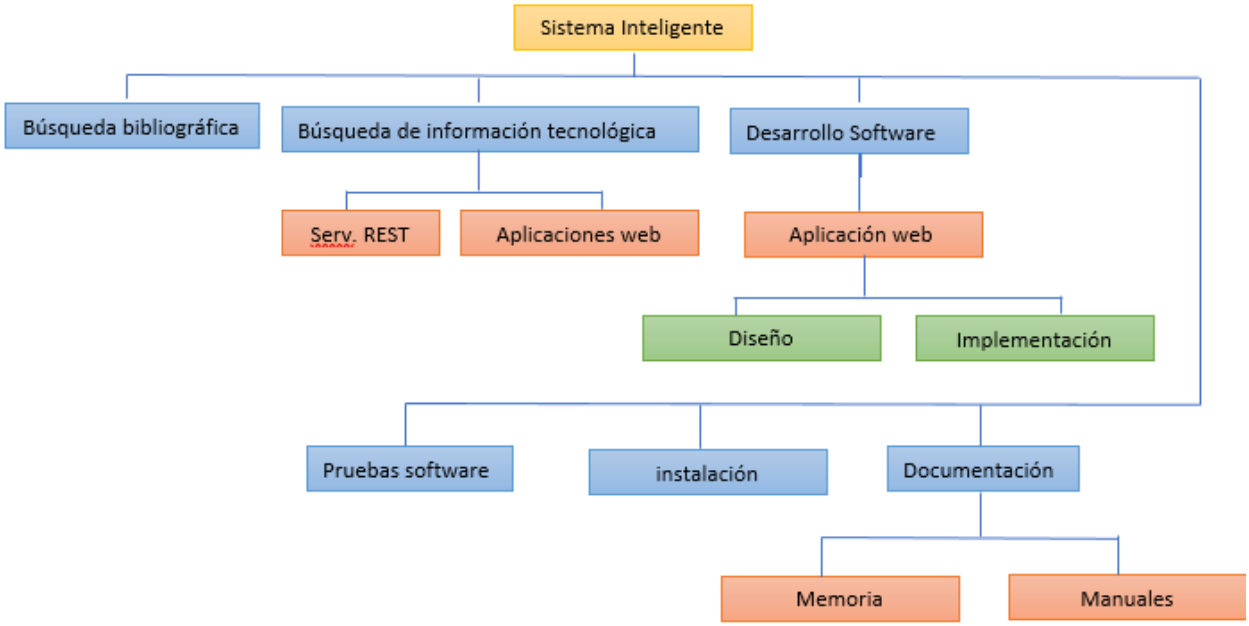

Figura 1.2: Estructura de trabajo

1.4.1 Estimación de tiempos

En este apartado reflejamos la estimación de tiempos realizada mediante una predicción pesimista de cada una de las tareas del trabajo de fin de grado. En la tabla 1 podemos observar con detalle cada una de las tareas acompañada de su duración. Como se puede apreciar, la suma de todas las tareas da lugar a unos 119 días dedicando en cada uno de ellos unas 5 horas por día.

Tarea	Estimación (días)
Búsqueda bibliográfica	15
Búsqueda de información tecnológica	
Serv. REST	3
Aplicaciones web	5
Desarrollo software	
Diseño	15
Implementación	46
Pruebas software	5
Instalación	1
Documentación	
Memoria	21
Manuales	8
Total:	119

Tabla 1:Planificación de tiempos

Para la realización de este proyecto, se han respetado y seguido minuciosamente los tiempos de trabajo establecidos sin sufrir desviaciones.

1.4.2. Diagrama de Gantt

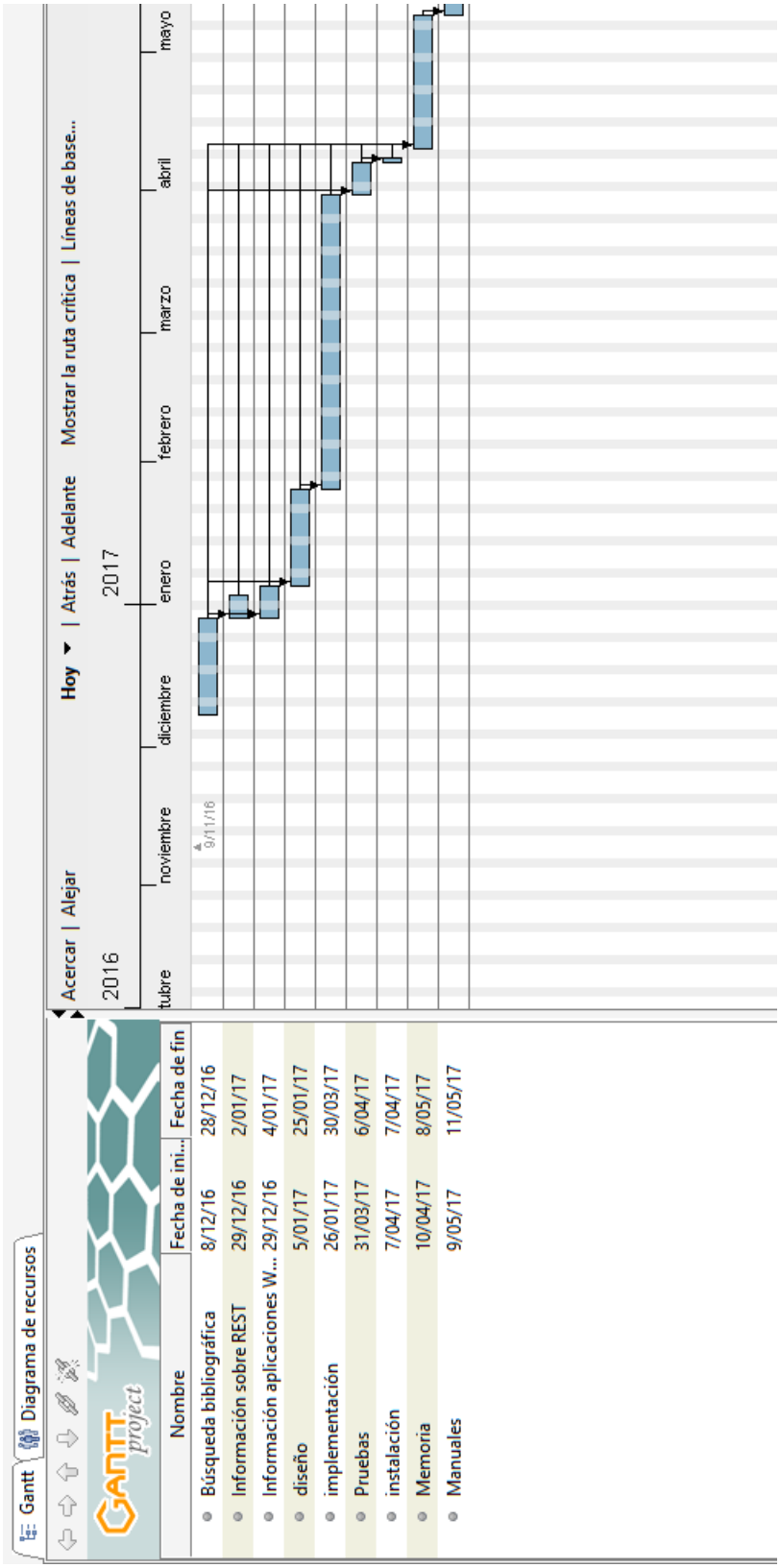

Ilustración 2: Diagrama de Gantt

En el diagrama de Gantt aparecen unas líneas de un color más blanquecino a lo largo de las líneas de vida de las tareas. Estas líneas representan los días festivos y fines de semana, que en la elaboración de esta estimación hemos considerado los fines de semana como no laborables.

1.5. Estructura de la memoria

En este apartado se va a describir la estructura de la memoria de este trabajo:

- **Apartado 1**. En este apartado se ha explicado la motivación, justificación de realización y objetivos dispuestos a alcanzar. Por último, se ha expuesto la planificación temporal del proyecto.

- **Apartado 2.** En este apartado se describirán con más detenimiento el concepto de ambientes inteligentes. Además, introducirnos un poco más el concepto de "*Compuación Ubicua*". Por último, describiremos un conjunto de laboratorios más relevantes que desarrollan este tipo de materias, en concreto nos centraremos en la explicación del Centro de estudios avanzados en tecnologías de la información y comulación (CEATIC) Ubicado en la Universidad de Jaén. Éste último ha sido el lugar en el que se ha desarrollado este proyecto.

- **Apartado 3.** En esta sección se explicarán todos los procesos de ingeniería de software aplicados para la realización del proyecto. En este apartado también se expondrán las tecnologías y herramientas utilizadas para el desarrollo del mismo.

- **Apartado 4.** Este apartado será dedicado a la exposición de concusiones extraídas de la realización del trabajo de fin de grado y se explicarán algunas posibles extensiones del proyecto.

- **Anexos**. En el Anexo A se incluye el contenido del CD-ROM adjunto. Se describirá detalladamente todo el contenido del mismo, así como también contendrá una copia de esta memoria. Por último, los siguientes anexos contendrán manuales de instalación, administración, y uso del proyecto elaborado en este trabajo

- **Bibliografía.** Éste será el apartado en el que plasmaremos toda la bibliografía consultada a lo largo de la realización de este trabajo.

2. Ambientes Inteligentes

La inteligencia ambiental estudia como proporcionar a las personas un entorno de convergencia tecnológica con interfaces fáciles de utilizar. Incluye un amplio abanico de tecnologías que tienen en todo momento en cuenta la presencia de la persona, la situación en la que esta se encuentra, adaptándose y respondiendo a sus necesidades.

La inteligencia ambiental se compone principalmente por tres características:

- **Ubicuidad.** Esta característica implica que los ambientes inteligentes permiten acompañar al usuario allá donde esté (hogar, colegio, medio de transporte…).
- **Invisibilidad.** Esta característica implica que suelen ser invisibles para el usuario, de manera que no son unos elementos ni sistemas intrusivos en el ambiente.
- **Inteligencia.** Este aspecto se refiere a la capacidad de adaptación a las preferencias de la persona.

Para aplicar estas características anteriormente citadas se necesita un sistema que monitorice y recaude información de manera permanente del ambiente en el que está instalado. Esta información puede ser ambiental, como por ejemplo puede ser la temperatura de una habitación, niveles de luz… O también puede registrar los cambios que se producen en el entorno como puede ser encender o apagar luces, televisión, puertas, electrodomésticos…

2.1 Computación Ubicua

La Computación Ubicua es un modelo de interacción en el que el procesamiento de información no se centra única y exclusivamente en procesamiento de un único dispositivo, sino que coordina de forma simultánea un conjunto de elementos que recogen y procesan el comportamiento del entorno de forma poco intrusiva y poco visible para el usuario.

Esta nueva metodología puede tener otro tipo de terminología, lo que también implica que pueda tener otro tipo de enfoque ligeramente distinto. Los términos más utilizados son: ubicomp, inteligencia ambiental, pervasive computing, internet of things, objetos inteligentes, spymes, everyware…

Ilustración 3: Interacción Hombre-Máquina

2.1.1. Aplicaciones de la Computación Ubicua

- **Seguridad.** Para la mejora de la seguridad se suelen utilizar hoy día multitud de elementos que facilitan el estudio de los datos de forma continua como pueden ser cámaras, sensores de movimiento, que te permiten recoger los datos de los usuarios y las acciones que realizan en el entorno.
- **Accesibilidad.** Por medio de esta característica los entornos pueden adaptarse de forma más fácil y rápida a las necesidades de cada usuario independientemente de cuál sea su minusvalía o discapacidad. Es por esto por lo que cada vez están siendo más utilizados en entornos donde conviven personas discapacitadas, ya que este tipo de computación les facilita bastante la realización de cualquier tipo de tarea doméstica.

- **Eficiencia energética.** Por último, un aspecto para el cual se utiliza mucho la computación ubicua es para realizar un ahorro en energía ya que, gracias a estos sensores antes mencionados, no será necesario mantener una bombilla encendida en una habitación, o mantener vigilancia sobre la temperatura de un espacio ya que todas estas tareas se pueden realizar de forma automática y sin necesidad de un control y supervisión del usuario que puede olvidarse de apagar un dispositivo o no puede preocuparse de planificar el encendido y apagado del mismo.

2.2 Laboratorios de Inteligencia Ambiental.

Como hemos mencionado anteriormente, existen multitud de centros que se dedican a la investigación de la inteligencia ambiental, intentando desarrollar continuamente nuevas formas de aplicarla a nuestra vida cotidiana.

Muchos centros de investigación están desarrollando sus laboratorios para realizar estudios y pruebas relacionadas con este ámbito. A continuación, hablaremos de algunos de ellos:

- **Amilab.** (Ambient Intelligence Laboratory) Situado en la Universidad Autónoma de Madrid. Éste, es un laboratorio en el que se encuentran coordinados esta universidad con su Escuela politécnica superior para el desarrollo e investigación de aplicaciones y utilidades dedicadas a personas con discapacidades. Uno de los proyectos que tienen creados ahora mismo es una aplicación llamada "*AssisT*", este software está dedicado exclusivamente a personas con discapacidades cognitivas para ayudarles a la realización de actividades cotidianas, navegación al aire libre y navegación de interior.

Ilustración 4: AssisT

- **Centro de estudios avanzados en tecnologías de la información y la comunicación (CEATIC).** Para este trabajo de fin de grado se ha necesitado la disposición de este centro para la realización de pruebas y desarrollo del sistema inteligente, por lo que por ello nos vamos a centrar de manera más especial a este centro.

2.3 Procesamiento del lenguaje natural

Actualmente, estos ambientes inteligentes están siendo complementados con un nuevo campo que combina las tecnologías de ciencia computacional con la lingüística aplicada. Su nombre técnico se denomina **Procesamiento del lenguaje natural (PLN)**, y su objetivo es la comprensión y el procesamiento asistidos por ordenador de la información expresada por un ser humano en lenguaje natural.

Este campo puede ser aplicado en multitud de aplicaciones, a continuación, mostraremos una serie de las más importantes de ellas: [3]

- **Análisis de opiniones y sentimientos**

  Éste es uno de los ámbitos más importantes del PLN se ocupa de analizar computacionalmente textos que han producido seres humanos y deducir los sentimientos u opiniones que les han causado, dependiendo de la forma en la que este escrito (forma, expresiones, palabras utilizadas…).

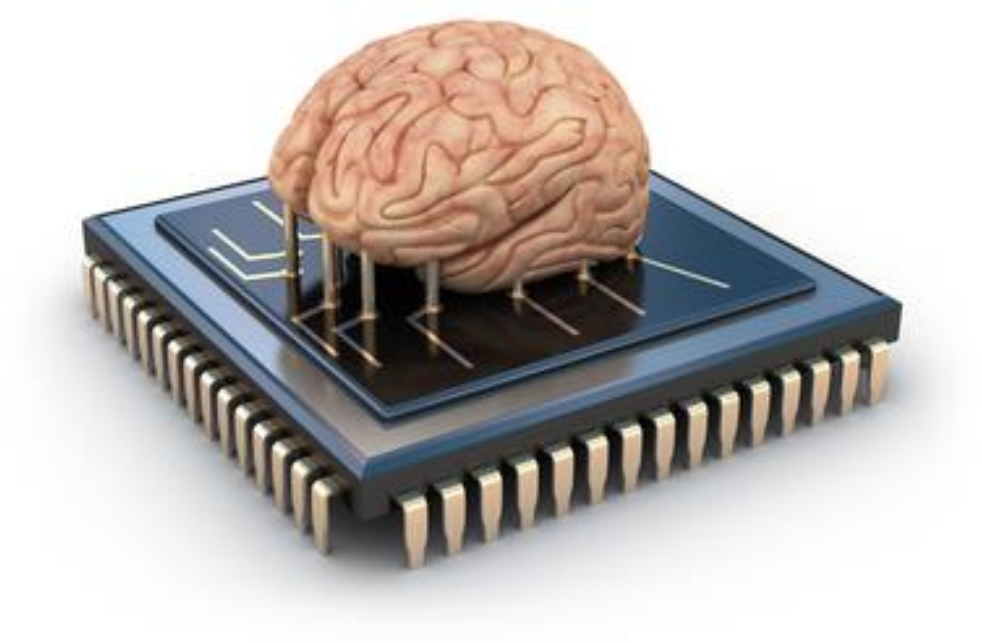

Ilustración 5: Procesamiento del lenguaje natural

- **Traducción automática**

  Esta técnica se basa en la traducción automática de textos escritos en algún lenguaje humano a otro idioma, por lo que existen muchas técnicas de traducción automática dependiendo del contexto en el que este escrito el texto, del planteamiento, etc.

- **Reconocimiento y clasificación de entidades nombradas**

  Este ámbito es muy importante en el ámbito de la extracción de información. Se basa en la identificación y la clasificación de elementos dentro de un texto, estos elementos se denominan “Entidades Nombradas” como pueden ser sustantivos, verbos, etc. Esta técnica es la que vamos desarrollar en nuestro trabajo de fin de grado y que explicaremos más adelante en esta memoria.

2.4 Speech to text

Hoy día, las aplicaciones relacionadas con la inteligencia ambiental están comenzando a implementar servicios en los que los clientes pueden mejorar la accesibilidad la aplicación incluyendo la posibilidad de interacción mediante la voz con el sistema.

En este sistema web implementaremos un servicio de interacción por voz, que nos permitirá la interacción del cliente con todos y cada uno de los sensores instalados en el servidor. Este servicio será implementado mediante una API proporcionada por la compañía Google denominada Java Speech Recognition.

2.4.1 Java Speech

Java Speech API fue desplegado por vez primera en 1998, definiendo dos tipos de paquetes, uno dedicado al reconocimiento de voz, otro especialmente implicado con la Síntesis de voz. [4]

Este tipo de API mantiene los siguientes objetivos:

- Proveer soporte para sintetizadores de voz y para los sistemas de reconocimiento de habla.
- Conseguir una interfaz multiplataforma robusta para la síntesis y reconocimiento del habla, permitiendo a los desarrolladores diseñar sistemas que no contengan características propias de un motor o plataforma de búsqueda.

2.5 Centro de Estudios Avanzados en Tecnologías de la Información y la Comunicación (CEATIC).

El Centro de Estudios Avanzados en Tecnologías de la Información y Comunicación (CEATIC) está situado en la dependencia 109 del edificio C6 del campus universitario de la Universidad de Jaén.

Se trata de una simulación de una vivienda cotidiana formada por una cocina, un salón y un dormitorio distribuidos en una superficie de 25 metros cuadrados donde se pueden encontrar todo tipo de electrodomésticos que podemos encontrar en una vivienda normal.

Este centro está orientado a la investigación de nuevas utilidades y usos de la inteligencia ambiental dedicado a la vida de las personas con una edad elevada o con algún tipo de minusvalía, consiguiendo de esta forma que dichos usuarios ganen tanto en comodidad, calidad de vida y seguridad.

Ilustración 6: Plano CEATIC

Ilustración 7: Imagenes CEATIC

3. Análisis y Diseño

En este apartado se explica de forma detallada la arquitectura de componentes y el desarrollo de los mismos para la realización detallada del proceso de ingeniería del software realizado para llegar exitosamente a la realización de este trabajo de fin de grado que consiste en un sistema de monitorización de entornos de inteligencia ambiental apoyados en un sistema de reconocimiento de voz y habla.

Como se indicó en el primer apartado, este trabajo consta de las siguientes partes:

- **Sistema web de monitorización** basado en una interfaz web implementada en html5 y javascript mediante la cual, el cliente puede interactuar con los ambientes inteligentes de un hogar.
- **Reconocimiento de voz** basada en Procesamiento de Lenguaje Natural, mediante el cual se realizará el procesado de las entradas recibidas por la interfaz, apoyado por una API denominada JavaSpeech.
- **Persistencia,** que será implementada mediante un servidor de datos realizado en Java, y donde realizaremos el almacenamiento de toda la información relevante de los elementos, y del contexto del sistema.
- **La conexión** de todos los elementos de nuestro sistema, será implementada mediante el uso de servicios REST, de manera que la comunicación se realizará de forma segura y eficiente y con utilizando unos servicios que pueden ser utilizados por cualquier plataforma.

3.1. Fases de Ingeniería de Software

Las fases de las que consta este proceso de ingeniería de software son las siguientes:

- **Especificación de Requerimientos:** Requisitos y funcionalidades que tendrá el sistema.

- **Análisis del Sistema:** Especificación formal de los requerimientos del sistema.
- **Diseño del Sistema:** Especificación del funcionamiento para satisfacer los requisitos analizados.
- **Implementación del sistema:** Desarrollo del software del sistema, cumpliendo todos los requerimientos bajo el diseño anteriormente establecido.
- **Pruebas:** Verificar y validar que cumple con los requerimientos y estándares del sistema.

3.2. Especificación de requisitos

3.2.1 Requisitos del Sistema web

Ésta es la primera fase de nuestro proceso, en la que debemos extraer cuales son las claves para que nuestro sistema web funcione de la mejor manera posible.

A continuación, vamos a realizar una descripción completa del sistema web dividiendo esta información en dos grupos:

Requisitos funcionales: Son los requisitos que definen alguna función del sistema, centrándose en los parámetros de entrada y respuestas esperados en sus casos de uso.

Requisitos no funcionales: Dentro de este grupo formarían parte los requisitos que especifican criterios para estudiar comportamientos específicos. Estos abarcan desde tipos de licencias de uso, elección de lenguaje de programación, etc.

A continuación, procederemos a describir con detalle cada uno de estos requerimientos del sistema.

3.2.1.1. Requerimientos Funcionales

- **Requerimiento 1: gestión de entornos**

El sistema podrá gestionar los entornos, es decir, el usuario podrá filtrar, editar, borrar y crear nuevos objetos a través de nuestro sistema de forma intuitiva. El sistema agrupará la información en los siguientes campos:

- **Información de los objetos**: El sistema guardará la información de los objetos, para poder ser identificados y filtrados a la hora de realizar su búsqueda.
- **Información de las propiedades de los objetos**: El sistema almacenará las propiedades que tendrá asociadas cada objeto para su posterior localización.
- **Tokens:** El sistema almacenará en un fichero JSON todos los tokens que nos ayudará a la hora de realizar la búsqueda y filtrado de objetos.

- **Requerimiento 2: Gestionar objetos**

El sistema permitirá a nuestro usuario la gestión de objetos creados de forma manual, de manera que el usuario podrá tanto dar de alta, modificar o eliminar un objeto y a su vez asociar a dicho objeto las acciones o propiedades que vea convenientes.

- **Requerimiento 3: Procesamiento del Lenguaje Natural**
  - **Requerimiento 3.1: Conversión de voz a texto (Speech to Text)**

    El sistema deberá tener la funcionalidad de procesamiento de voz, captada mediante un dispositivo de entrada, y posteriormente, este mensaje será convertido en texto para su posterior procesado.

- **Requerimiento 3.1: Algoritmo de reconocimiento de objetos, propiedades y acciones.**

  El sistema deberá ser capaz de reconocer los objetos, sus propiedades y acciones para poder descifrar correctamente la orden que debe realizar sobre el ambiente o entorno.

- **Requerimiento 4: visualización de entornos.**

El sistema permitirá al usuario la visualización y listado de todos los objetos que están actualmente dados de alta en la base de datos. Estos objetos aparecerán identificados mediante un identificador único, y con una sublista donde se mostrarán todas las acciones o propiedades que tiene asociadas dicho objeto con una descripción de la propiedad y un botón para ejecutar la misma.

- **Requerimiento 5: Estado de los objetos**

El sistema permitirá la visualización del estado (Encendido, apagado, etc.) de todos los objetos mediante un icono que lo refleje.

- **Requerimiento 6: gestión de tokens**

El sistema permitirá a nuestro usuario la gestión de tokens creados de forma manual, de manera que el usuario podrá tanto dar de alta, modificar o eliminar un conjunto de tokens.

3.2.1.2 Requerimientos no funcionales.

Como hemos comentado en el apartado anterior, los requisitos no funcionales nos mostraban las limitaciones o restricciones que hay que tener en cuenta para que nuestro proyecto pueda ser utilizado de forma correcta y funcional.

A continuación, se mostrarán los siguientes requerimientos:

- **Requisitos físicos**

Al tratarse de una arquitectura Cliente-Servidor, es necesario un equipo informático que ofrezca los servicios. En este apartado se detallan los requisitos mínimos que debe tener este equipo para que pueda funcionar correctamente ejecutando los servicios:

- Procesador: 1,30 GHz
- Memoria RAM: 1.5 GB
- Memoria ROM: 12 GB

Respecto al cliente solamente bastará con la conexión a internet, pero se recomienda que cumpla los requisitos que se mostrarán a continuación:

- Procesador: 1,30 GHz
- Memoria RAM: 2GB
- Memoria ROM: 40GB
- Dispositivo de entrada de sonido (Micrófono)

- **Requisitos Software**

Como hemos realizado con los requisitos físicos, estos requisitos también se dividirán en dos apartados, uno para el cliente y otro para el servidor.

Para la instalación del servidor en el equipo, éste deberá disponer de una distribución linux y java 1.7 o superior.

Por otro lado, para nuestro cliente solamente será necesario que tenga un sistema operativo basado en Windows, linux o Mac OS, un navegador web como Mocilla Firexox o Chrome y que tenga soporte para JavaScript.

- **Interfaz**

Los requerimientos necesarios para la nuestra interfaz deben contener las siguientes características:

- **Flexibiliad.** La aplicación web podrá ser visualizada desde cualquier dispositivo.
- **Robustez.** La aplicación deberá de poseer la cualidad de soportar fallos que pueda producir el usuario.
- **Sencillez.** La aplicación deberá ser intuitiva para el usuario.

3.3 Análisis del sistema

3.3.1. Casos de uso de la aplicación web

Los diagramas de casos de uso reflejan el comportamiento que debe tener el sistema desde el punto de vista del cliente y representando todas las funciones que el sistema debe ejecutar.

Un caso de uso está compuesto por los siguientes componentes:

- **Actor:** Representa al cliente, que será el usurario final que utilizará el sistema.
- **Condiciones de entrada:** Condiciones que se deben dar antes de que una acción pueda ser realizada.
- **Eventos:** Eventos que ocurrirán al realizar la acción.

Primeramente, vamos a proceder a la creación de un diagrama frontera con el que mostraremos todos los requerimientos mostrados en el anterior apartado. (Ver figura 3.1).

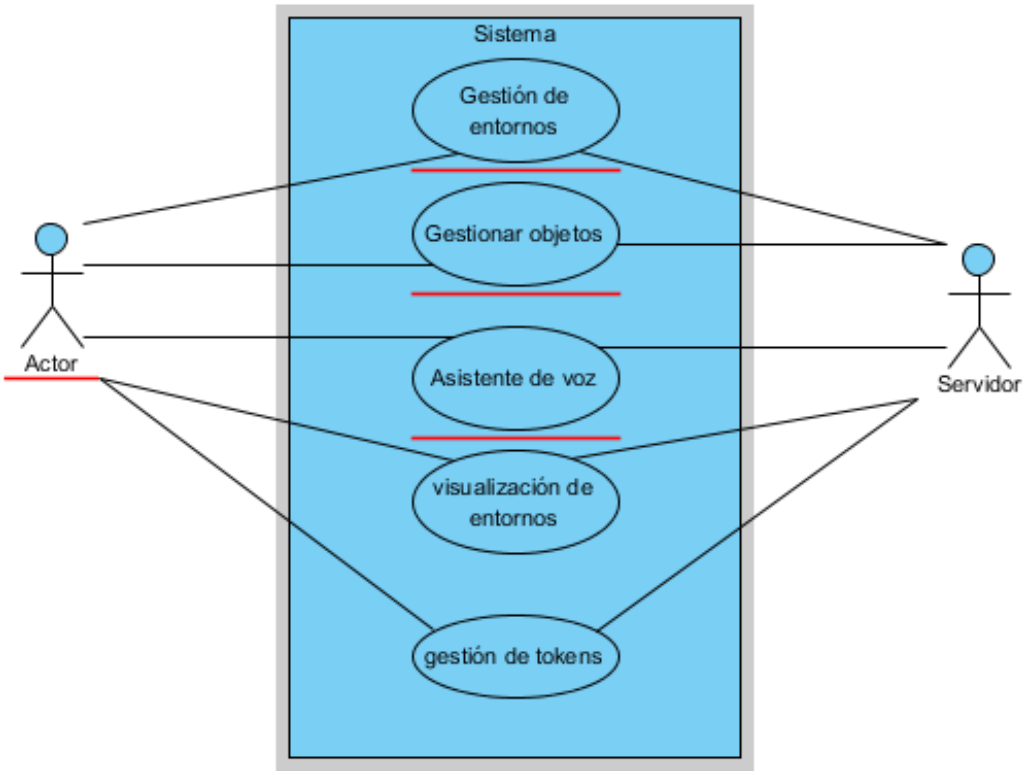

Ilustración 8: Diagrama Frontera

A continuación, mostraremos los casos de uso para cada uno de los requerimientos funcionales mostrados anteriormente. Dentro de los casos de uso, se pueden usar dos tipos de relaciones, las relaciones *extend*, con la que se suelen mostrar alternativas en el caso de uso, y la relación *include* con la que se representarían acciones de uso común.

- **Caso de uso 1: Gestión de objetos.**

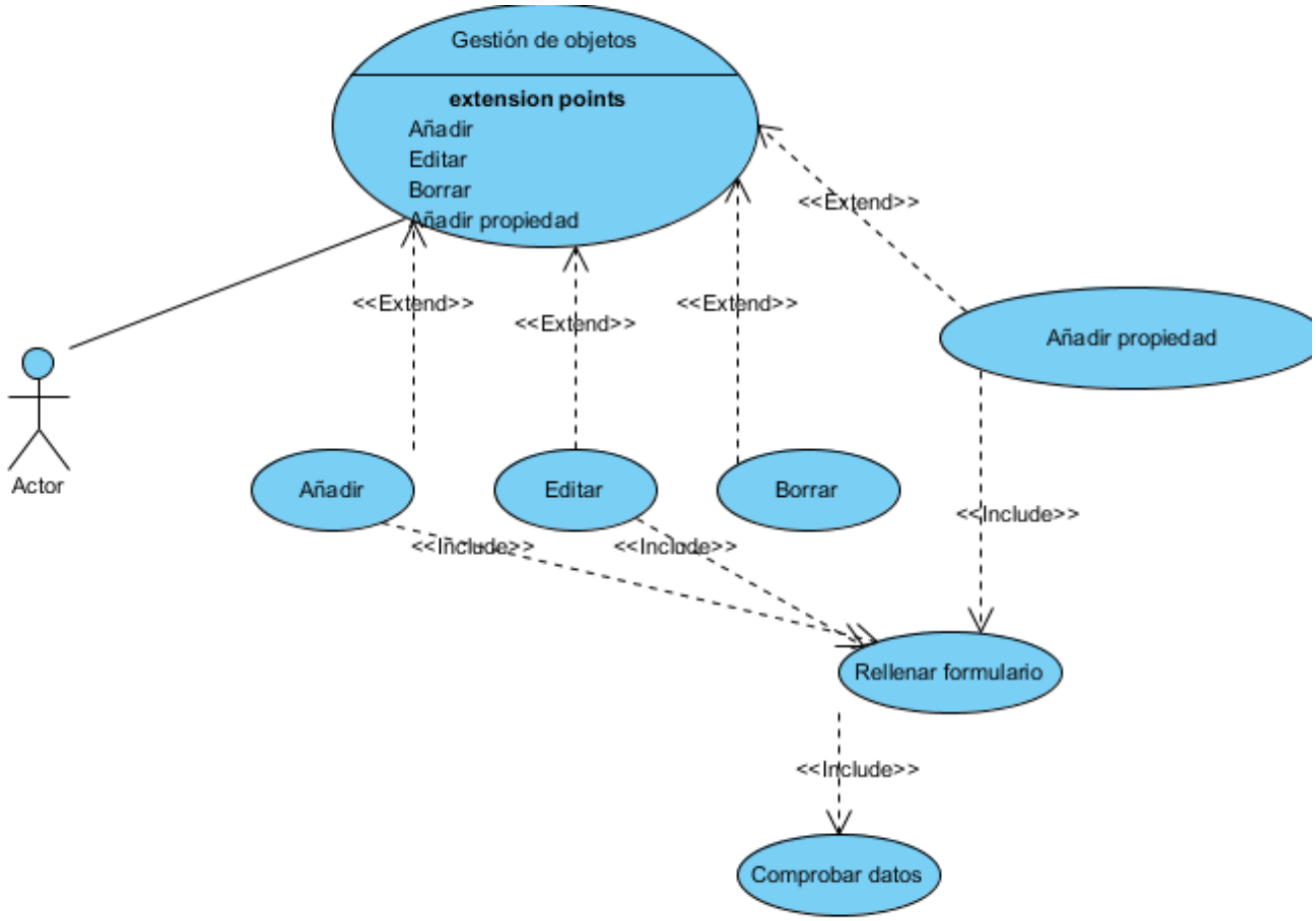

Ilustración 9: Caso de uso: gestión de objetos

- **Actores:** Usuario
- **Eventos:**
    - El sistema muestra todos los objetos.
    - El usuario puede crear un nuevo objeto.

a) El sistema mostrará el formulario a rellenar con los datos y características del objeto.
b) El usuario rellena los campos del formulario.
c) El sistema pasa a comprobar si todos los datos introducidos son correctos.
d) Si todos los campos son correctos, el sistema crea el objeto y muestra una lista con todos los que hay existentes incluyendo el recién creado.

- El usuario puede editar un objeto

a) El sistema mostrará el formulario a rellenar con los datos y características del objeto.
b) El usuario rellena los campos del formulario.
c) El sistema pasa a comprobar si todos los datos introducidos son correctos.
d) Si todos los campos son correctos, el sistema modifica el objeto y muestra una lista con todos los que hay existentes incluyendo el recién editado.

- El usuario puede borrar un objeto.

a) El usuario busca en la lista el objeto al que esta dispuesto a borrar.
b) Pulsa el botón borrar.
c) El sistema borra el objeto seleccionado.
d) El sistema muestra todos los objetos existentes.

- El usuario puede añadir una propiedad a un objeto

e) El sistema mostrará el formulario a rellenar con los datos y características de la propiedad del objeto.
f) El usuario rellena los campos del formulario.

g) El sistema pasa a comprobar si todos los datos introducidos son correctos.
h) Si todos los campos son correctos, el sistema modifica el objeto y muestra una lista con todos los que hay existentes incluyendo el recién editado.

- **Excepciones:** Si algunos de los datos introducidos en el formulario no son correctos, el sistema muestra un mensaje y solicita al usuario que los vuelva a meter

- **Caso de uso 2: Asistente de voz**

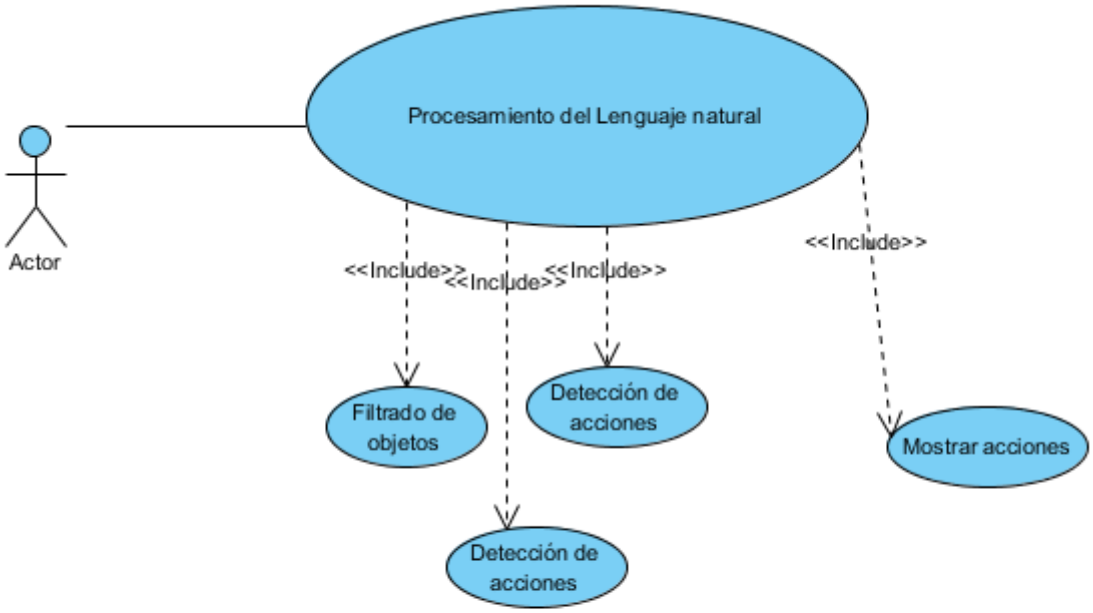

Ilustración 10: Caso de uso: Asistente de voz

- **Actores:** Usuario
- **Eventos:**

    - El usuario decide usar el asistente como método de búsqueda.
    - Pulsa un botón para activar el micrófono.
    - Introduce la orden mediante el habla.
    - El sistema reconoce el mensaje introducido por el usuario.
    - El sistema realiza el filtrado de objetos teniendo en cuenta la frase introducida.
    - Posteriormente filtra las propiedades y acciones de dicho objeto teniendo en cuenta el mensaje de entrada.
    - Finalmente muestra al cliente los objetos obtenidos de realizar los filtrados anteriores.

- **Caso de uso 3: gestión de tokens.**

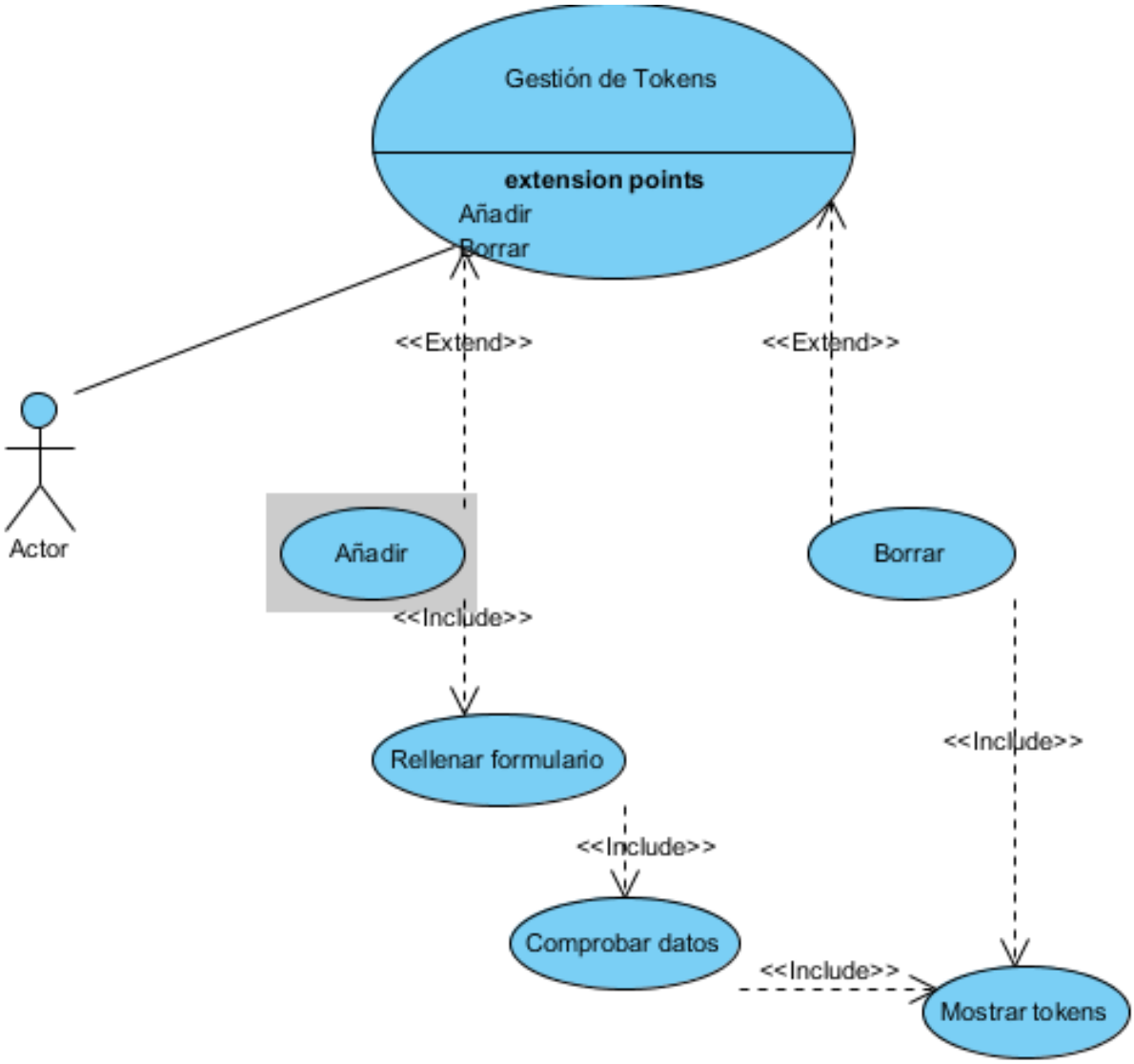

Ilustración 11: Caso de uso: Gestión de tokens

- **Actores:** Usuario
- **Eventos:**

    - El sistema muestra todos los tokens.
    - El usuario puede crear un nuevo token.

        e) El sistema mostrará el formulario a rellenar con los datos y características del objeto.
        f) El usuario rellena los campos del formulario.
        g) El sistema pasa a comprobar si todos los datos introducidos son correctos.
        h) Si todos los campos son correctos, el sistema crea el token y muestra una lista con todos los que hay existentes incluyendo el recién creado.

    - El usuario puede borrar un token.

        e) El usuario busca en la lista el token al que esta dispuesto a borrar.
        f) Pulsa el botón borrar.
        g) El sistema borra el token seleccionado.
        h) El sistema muestra todos los tokens existentes.

- **Excepciones:** Si algunos de los datos introducidos en el formulario no son correctos, el sistema muestra un mensaje y solicita al usuario que los vuelva a meter

3.4 Diseño del sistema

3.4.1. Diseño de clases

3.4.1.1. Patrón MVC

Para la realización y desarrollo de este proyecto se ha seguido un esquema de diseño MVC (Modelo-vista-controlador).

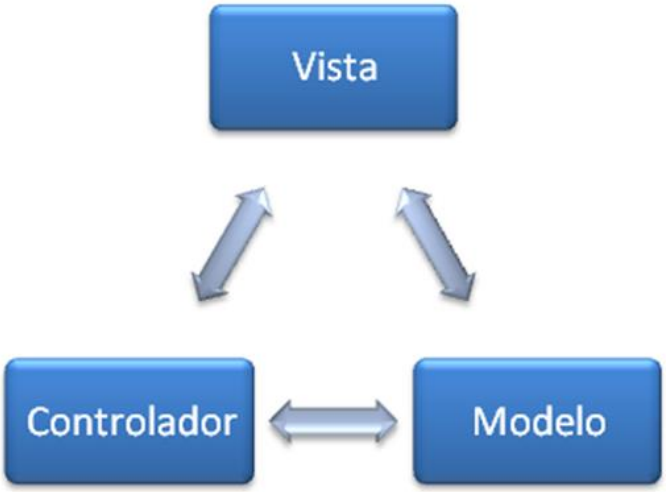

Ilustración 12: Modelo-vista-controlador

El patrón MVC (Modelo-Vista-Controlador) fue construido para separar la interfaz gráfica del código que controla una aplicación. Esta idea fue inventada para el contexto de Smalltalk, que se trata de un contexto de programación orientado a objetos y de tipo dinámico. [4]

En este modelo, las entradas del cliente, los modelos y la parte visual de la aplicación se encuentran totalmente separados mediante la creación de tres objetos que están especializados cada uno para tareas concretas.

3.4.1.2. Ventajas del patrón MVC

A continuación, pasaremos a mostrar las numerosas ventajas de este tipo de arquitectura:

- Consta de una implementación modular, por lo que se encuentra estructurada en módulos que cada uno realiza una función concreta.
- El programador no debe preocuparse de que las vistas se encuentren actualizadas puesto que de esa tarea se encarga el controlador.
- Cualquier modificación, como puede ser el aumento de métodos en el dominio, no tiene porque afectar al funcionamiento de todo el mecanismo de comunicación y actualización de modelos.
- MVC está siendo un patrón de arquitectura bien elaborado ya que las aplicaciones que lo desarrollan son más extensibles y sostenibles en comparación con otras aplicaciones que se basan en otros patrones.

3.4.1.3. Partes del patrón MVC

- **Modelo:** En este caso el modelo lo formaran el conjunto de datos proporcionado por el servidor y estará estructurado en formato JSON.
- **Vista:** Como se tratará de una aplicación web las vistas serán totas y cada una de las paginas HTML.
- **Controlador:** Esta parte del proyecto es la parte primordial ya que se encarga de la comunicación de la aplicación de lado del cliente con el servidor, es la parte que realiza todas las peticiones. Para la realización de este controlador se ha utilizado lenguaje Javascript.

3.4.1.3 Modelo del sistema web

A continuación, se procederá a mostrar un diagrama de clases en el que se representa la distribución y la organización del modelo de nuestro sistema web.

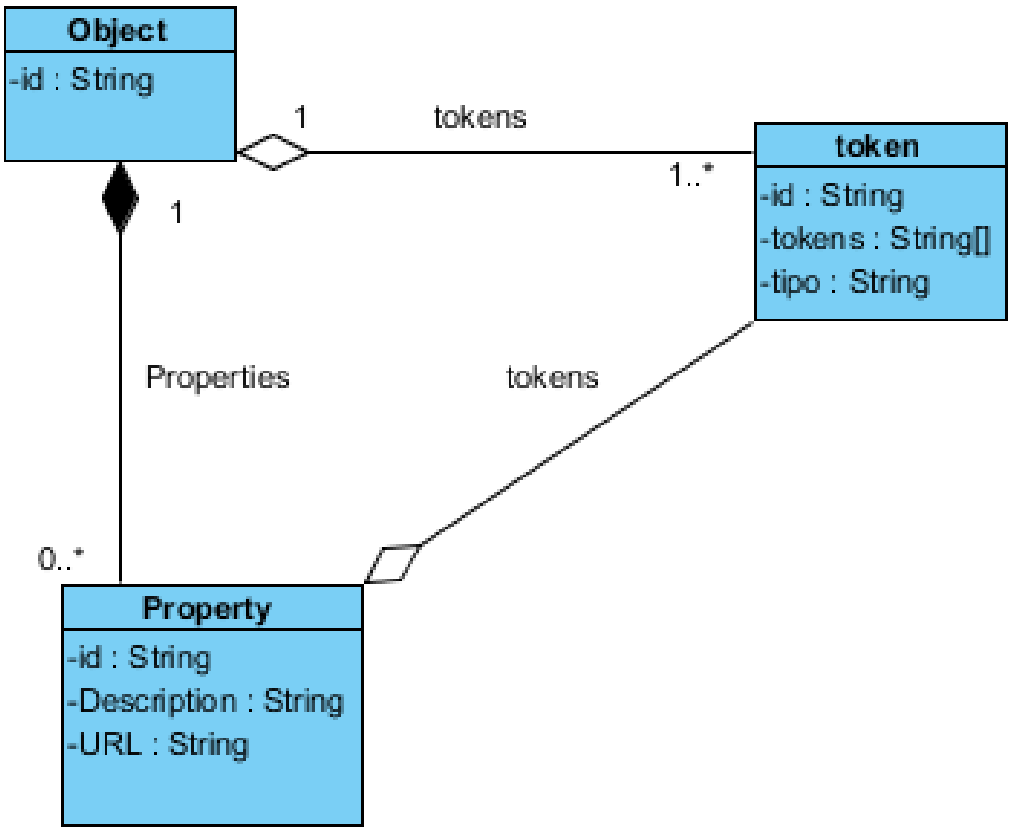

Ilustración 13: Diagrama de clases Sistema Web

3.4.2. Diseño de Datos

El propósito de este apartado será la explicación y determinación de cuál será la estructura de persistencia que utilizaremos para nuestro sistema web.

La persistencia de nuestro sistema web la realizaremos mediante la implementación de un simple servidor montado en lenguaje java. Éste nos proporcionara los ficheros JSON que mostraremos a continuación la estructura de cada uno de ellos y la información que nos proporciona.

- **Fichero Tokens.json. (**Ver Ilustración 11**)**

Este fichero nos proporcionará todos los tokens creados anteriormente por el sistema web. Este fichero será leído y procesado justamente con el arranque del sistema y la estructura de los datos que contiene siguen la siguiente distribución.

-**Id:** Cada token dispone de una palabra identificadora y que se utilizara posteriormente para la identificación a la hora de procesar el mensaje con el asistente de voz.

-**Tipo:** Cada token solamente puede ser de dos tipos: acción para los tokens que forman parte de acciones (Encender, apagar, abrir…) y objeto para los tokens que forman parte de entornos de la vivienda (puerta, salón, cocina…). Esta cualidad nos ayudará para el procesamiento del mensaje de búsqueda para poder sacar una orden única y poder detectar cuales son las acciones a realizar y los objetos con los que operar en el entorno.

-**Tokens:** Cada token contiene un conjunto de subtokens, estos, son palabras relacionadas con el identificador anteriormente mencionado. Este vector se suele componer de variantes de palabras que tienen relación con el identificador. Por ejemplo, el vector Tokens del identificador "encender" seria encender, encienda, enciéndanos, enciéndeme… Estas variantes son las que utilizaremos para procesar el mensaje inicial de búsqueda y poder obtener el mensaje imperativo para mandar al servidor de nuestra vivienda.


```
{"id":"salon","tipo":"objeto","tokens":["sala"," salon"," salón"," sala de estar"]},
{"id":"cocina","tipo":"objeto","tokens":["cocina","cocinita"]},
```


Ilustración 14: Distribución fichero Tokens.json

- **Fichero objetcts.json(**Ver ilustración 12**)**

Este fichero contendrá todos los objetos pertenecientes a los entornos de la vivienda. Este fichero será leído y procesado justamente con el arranque del sistema y la estructura de los datos que contiene siguen la siguiente distribución.

- **Id:** Cada objeto tendrá un atributo identificador que se corresponderá con una serie de letras y números. Cada identificador deberá ser único, ya que este identificador lo será enviado al servidor situado en la vivienda para realizar la acción correspondiente, por lo que no puede haber dos identificadores iguales.

- **Tokens:** Cada objeto contendrá un conjunto de identificadores de tokens, de esta manera podremos identificar de que objeto estaríamos hablando a la hora de realizar la búsqueda. Por ejemplo, si estamos hablando de la luz que se encuentra situada en el salón, el identificador podría ser Luz1 y el conjunto de tokens relacionados con este objeto serían: bombilla, salón.

- **Properties:** Este sería un vector de objetos de tipo Property en el cual contendrá todas las propiedades que están asociadas a cada objeto, de manera que recorriendo este vector podremos saber cuántas propiedades hay creadas en dicho objeto. Por ejemplo, el objeto Luz1 podría tener tres propiedades que podrían ser: Encender, Apagar, y Tenue, esta última sería una propiedad específica para poner dicha luz con una intensidad baja sin tener que ser apagada ni encendida por completo.

-**Property:** Este tipo de objetos son los que estarían formando parte del vector Properties nombrado en el guión anterior. Estos objetos tendrían cuatro atributos que son:

- id: Este id será una palabra clave que formará parte de la URL que será enviada al servidor de la vivienda.
- Description: Este atributo será una breve descripción de la funcionalidad que va a desempeñar esta propiedad.
- Tokens: Este atributo está formado por un identificador de el fichero Tokens.json que deberán ser de tipología *accion.*
- URL: Este atributo será la dirección URL que será enviada al servidor en el caso que se decida realizar dicha acción.

A continuación, se mostrará una imagen para que se pueda apreciar el aspecto en formato JSON de estos objetos.


```
{"properties":[{"url":"http://smart-lab/set:tv:value:on","description":"encender la tele","id":"on","tokens":["encender"]},
{"url":"http://smart-lab/set:tv:value:off","description":"apagar la tele","id":"off","tokens":["apagar"]}]
,"id":"tv0",
"tokens":["television","salon"]},
```


Ilustración 15: Distribución fichero Objects.json

3.4.3. Diseño de la Interfaz

Una interfaz web es un conjunto gráfico que permite a los usuarios tener un control completo de todas las funcionalidades disponibles en el sitio web.

Dos de las cualidades más importantes a la hora de implementar una interfaz es que esta sea lo más eficiente posible para que su funcionamiento sea correcto, pero, por otro lado, también es necesario la creación de una interfaz gráfica lo suficientemente intuitiva y fácil de usar ya que, si el cliente no se encuentra cómodo a la hora de utilizarla, nuestra aplicación tiene un gran riesgo de caer en el fracaso. [5]

3.4.3.1. Estilo

En multitud de sitios web hemos visto que han llegado al éxito incluso siento menos potentes o eficientes que otras páginas. Sin embargo, no nos hemos parado a pensar en que es lo que ha podido suceder o realizar para que los clientes se decanten por una aplicación web u otra. Es en estos ámbitos donde entran en juego los estilos de las páginas dado que el uso correcto de una tipografía, una elección de una paleta de colores adecuada y otros elementos que veremos a continuación pueden ser cruciales para el éxito de nuestro sitio web. [6]

La creación de un documento donde se estandarizan todos los aspectos visuales que puede tener nuestra página o aplicación web se denomina guía de estilo.

A continuación, procederemos a realizar nuestra propia guía de estilo indicando todos los factores que hemos elegido para la creación de nuestra aplicación web.

- **Tipografía:** La tipografía es la elección del tipo de escritura (Tipo de letra, tamaño…) de manera que el texto le parezca compacto y cómodo de leer al cliente.

  Para el cuerpo de nuestra aplicación hemos utilizado una tipografía con las siguientes características.

  - **Fuente:** Helvética Neue
  - **Tamaño de letra:** 16px
  - **Interlineado:** 1,428

- **Colores:** Para este proceso de elección de color hemos decidido utilizar una gama de colores ayudándonos de *Adobe color cc y brandColors.net.* Ambas herramientas nos ayudan a crear gamas cromáticas aceptables por el cliente pudiendo elegir entre gamas de colores análogos, monocromáticos, complementarios, etc.

  La gama elegida será la siguiente:

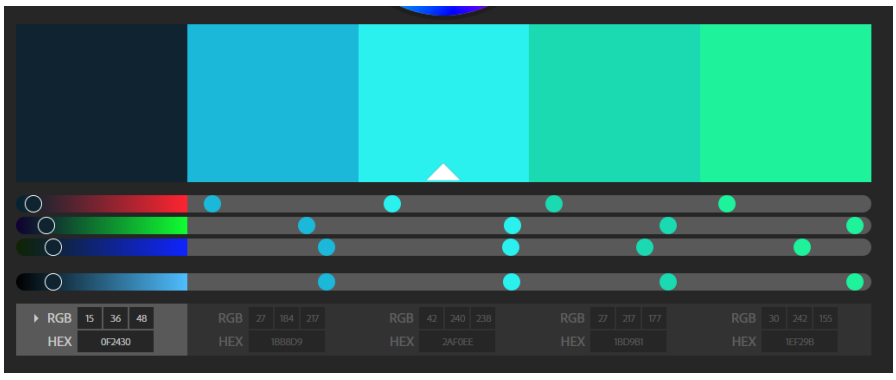

Ilustración 16: Gama Cromática

- **Botones:** Para el uso de los botones nos hemos ayudado del framework que hemos utilizado para la realización de nuestro trabajo, por lo que serían los botones pertenecientes al framework *bootstrap.*

3.4.3.2. Mensajes

Los mensajes en una aplicación web se utilizan principalmente para advertir de algún error ocurrido durante la ejecución de la aplicación. Estos mensajes normalmente van dirigidos a los clientes que causan algún error al introducir algún dato o provocan alguna situación anómala en el sistema.

Estos mensajes deben de ser breves, normalmente no suelen tener más de dos líneas de extensión y el mensaje deberá ser breve y directo para evitar la posible confusión del cliente.

A continuación, procedemos a mostrar todos los tipos de mensajes de error que pueden aparecer en nuestro sistema.

- **ID de objeto repetido o existente**

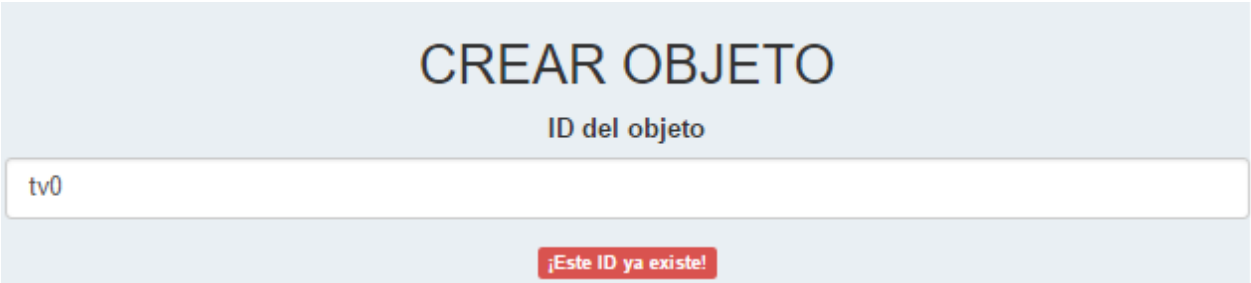

Ilustración 17: Excepción id objeto

- **Id de una propiedad de un objeto existente**

Ilustración 18: Excepción: Propiedad existente

- **Dejamos algún campo de un formulario vacío**

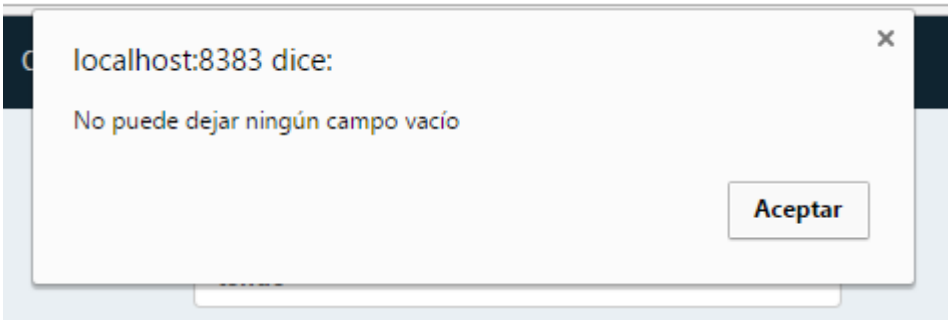

Ilustración 19: Excepción campo vacío

- **Id de token ya existente**

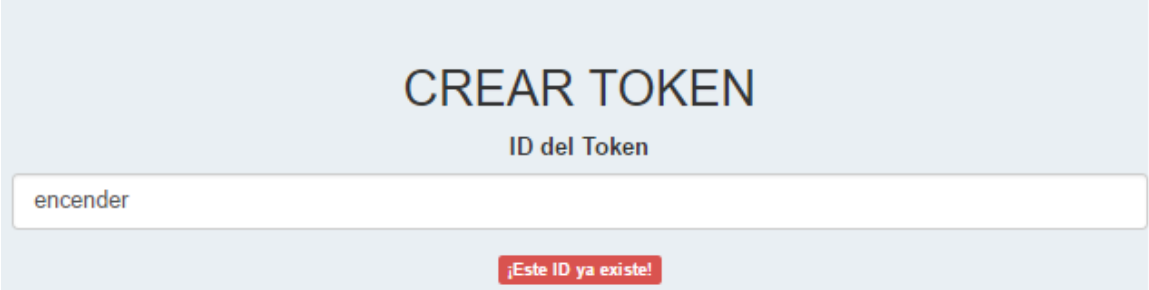

Ilustración 20: Excepción: token ya existente

3.4.3.3 Storyboard del sistema

Un storyboard es una secuencia de imágenes que describen de forma rápida y sencilla el funcionamiento de una web, aplicación, o cualquier programa informático. [7]

Este tipo de representaciones pueden ser de varios tipos que mostraremos a continuación:

- **Storyboard de tipo 1**

  Este tipo de storyboards tienen un elevado parecido a la apariencia de un comic y narran todos los diferentes momentos en los que el cliente interactúa con el producto.

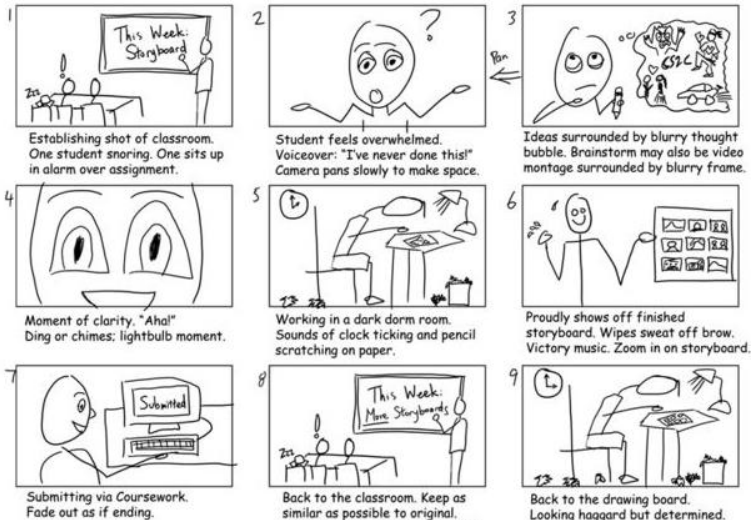

Ilustración 21:Ejemplo deStoryboard de tipo 1

- **Storyboard de tipo 2**

  En este tipo de reproducciones la información proporcionada por las imágenes y descripciones es más detallada que la anterior, por lo que nos proporciona mayor información acerca del producto y de cómo el cliente puede interaccionar con nuestra aplicación.

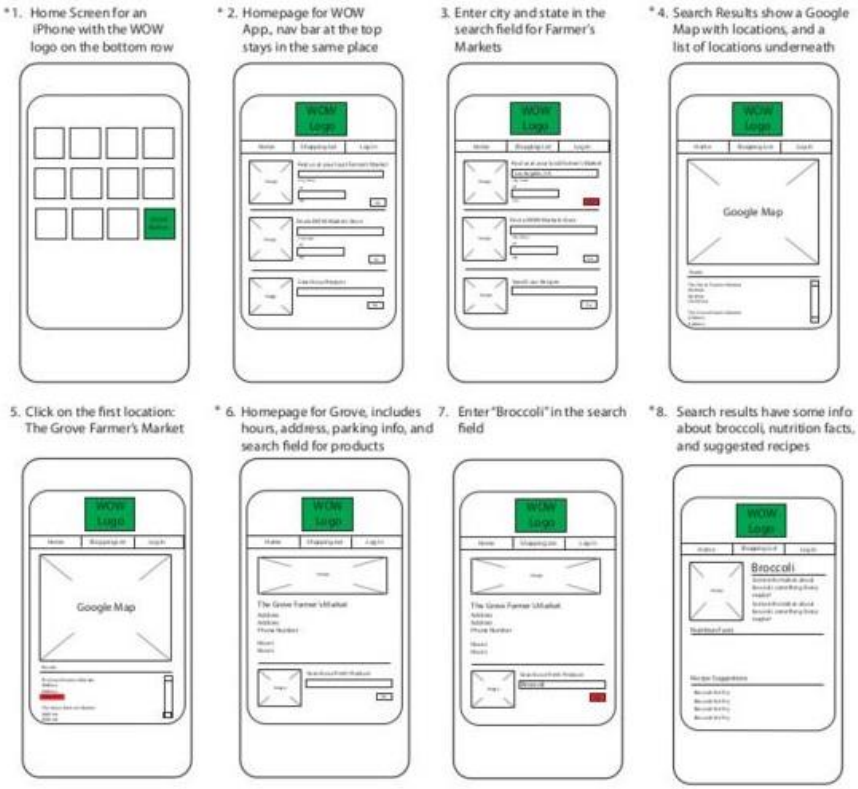

Ilustración 22: Ejemplo de storyboard tipo 2

A continuación, procederemos a la exposición del storyboard de nuestro proyecto. Para la realización de las ilustraciones nos hemos apoyado en una herramienta web llamada Mokingbird.com la cual nos ha permitido realizar el trabajo de manera fácil e intuitiva.

- **Panel principal**

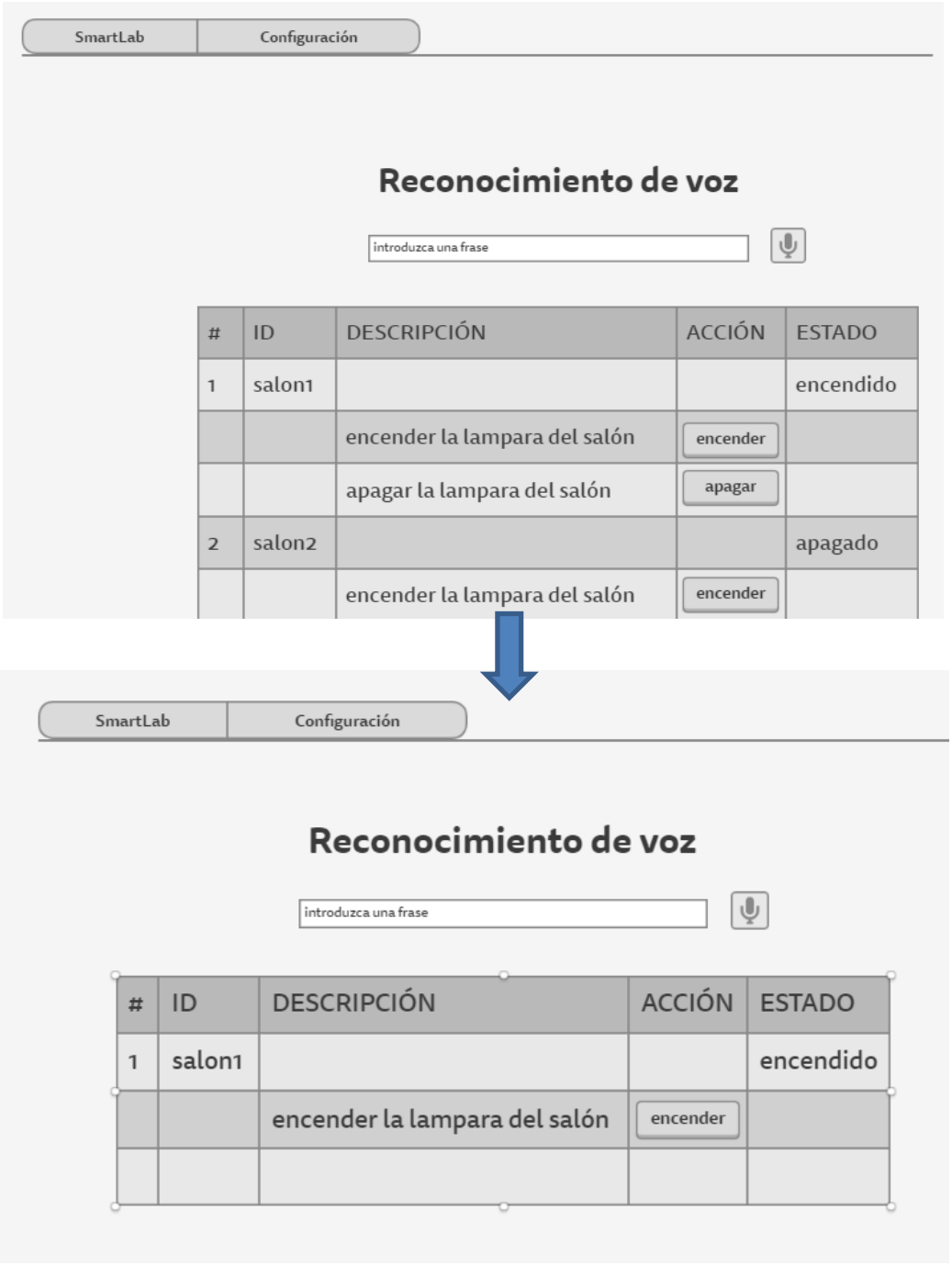

Ilustración 23: Storyboard Escenario principal

- **Panel de configuración**
  - **Listado de comandos**

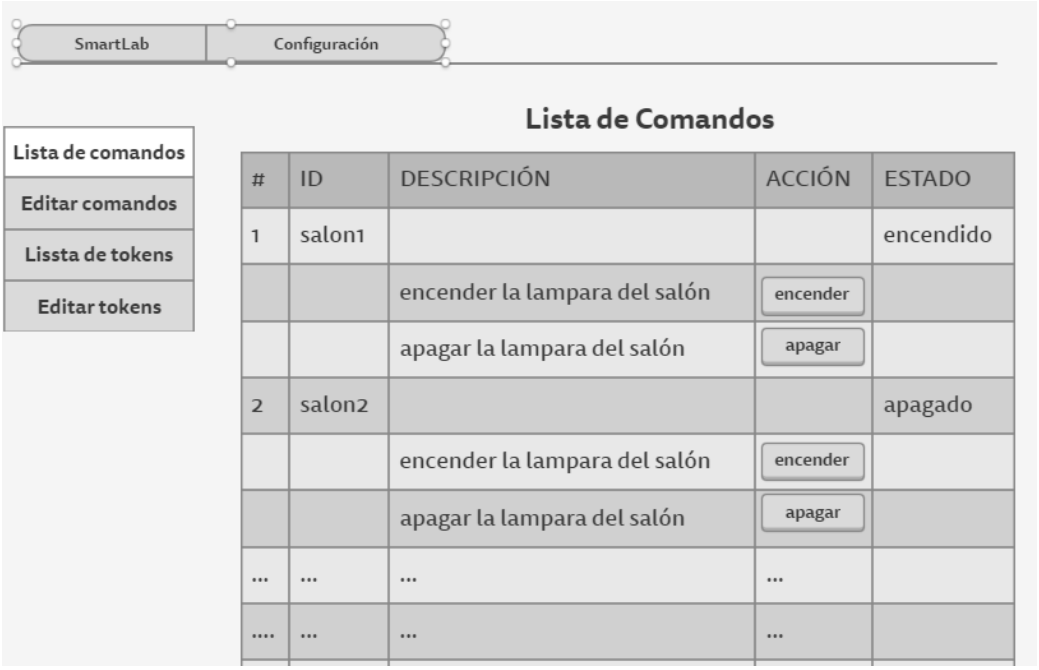

Ilustración 24: Storyboard: Listado de comandos

  - **Editar comandos**

Ilustración 25: Storyboard: Edición de comandos

- **Editar Tokens**

Ilustración 26:Storyboard: Edición de tokens

3.5 Implementación del sistema.

3.5.1 Arquitectura del sistema

Esta etapa es el último proceso de desarrollo antes de comenzar con las pruebas de software. Por otro lado, es la etapa más costosa dado que es donde hay que poner en marcha el desarrollo de nuestro sistema, para ello debemos seleccionar cual será el lenguaje de programación que utilizaremos para la implementación por lo que demos de realizar una pequeña valoración antes de comenzar a programar.

Para la realización del proyecto se va a utilizar una arquitectura Cliente/Servidor. Para una mejor comprensión de esta arquitectura nos acompañamos de la siguiente imagen: (ver ilustración 27)

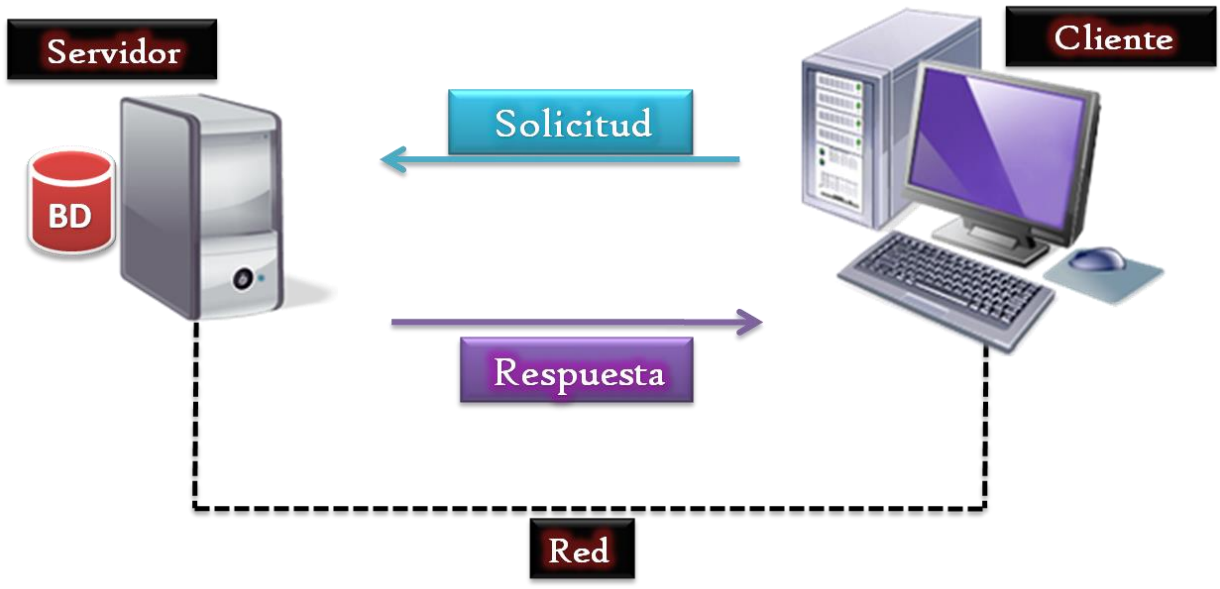

Ilustración 27: Esquema Cliente/Servidor

3.5.1.1 REST

Para realizar la comunicación con el servidor hemos utilizado la metodología REST (Representational State Transfer), este término se refiere a un estilo arquitectónico que se basa a menudo en el desarrollo de servicios web. Esta arquitectura de desarrollo web se apoya principalmente en el estándar HTTP.

REST está capacitado para crear servicios y aplicaciones las cuales se pueden usar desde cualquier dispositivo que atienda HTTP.

Esta comunicación entre cliente y servidor se realiza mediante el envío de un mensaje en formato JSON por medio del protocolo HTTP, utilizando uno de los métodos que se encuentran disponibles en este protocolo (GET, POST, PUT y DELETE).[8]

- **GET:** Se utiliza para realizar lecturas
- **POST:** Se utiliza para realizar escrituras
- **PUT:** Se utiliza para realizar actualizaciones
- **DELETE:** Se utiliza para realizar borrados

3.5.2 Procesamiento del Lenguaje Natural

En este apartado, explicaremos todos los procesos implementados para la realización de nuestro asistente de voz. Para su realización hemos utilizado numerosas técnicas de Procesamiento del Lenguaje Natural.

3.5.2.1 Técnicas de PLN

3.5.2.1.1 Tokenización

La Tokenización forma parte de uno de los procedimientos más importantes del lenguaje natural. Su objetivo es producir la división de un texto en elementos más pequeños como pueden ser palabras, frases u otros elementos. Estos fragmentos del texto se denominan *tokens.*

Para realizar estos métodos de Tokenización, se puede realizar detectando los espacios en blanco del texto, y así detectar todas y cada una de las palabras, o por otro lado podríamos tener en cuenta los signos de puntuación como pueden ser los puntos, que nos ayudarán a separar cada una de las frases del texto.

Una vez tenemos una estructura con todos los tokens, podemos comenzar el procesamiento del lenguaje y utilizar algunas de las técnicas que mostraremos a continuación. [9]

3.5.2.1.2 Lematización

Este proceso consiste en la obtención de un lema (palabra que engloba todas conjugaciones, variedades verbales, género, numero…) de una palabra. Por lo tanto, un lema es una palabra que se computacionalmente se utiliza como una etiqueta informática que identifica a una palabra y a todas sus derivaciones, por ejemplo: *encender* sería el lema de *Enciende, Encienda, Enciéndanos, Enciéndeme, etc.*

Por lo tanto, un lematizador es un programa que se basa en numerosos algoritmos, y su función sería convertir el cuerpo de un texto en una serie de términos simplificados o lemas. [10]

3.5.2.1.3. Stopwords

Las Stopwords, también conocidas como palabras vacías, son palabras que se encuentran en un texto y son extraídas mediante el proceso de Tokenización, pero no contienen realmente un significado concreto, por lo que los buscadores suelen desecharlas y solamente seleccionar las palabras que realmente poseen algún tipo de significado o información relevante para ellos que les pueda ayudar a realizar la búsqueda.

3.5.2.2 Implementación del asistente de voz

Para la implementación de nuestro asistente, hemos utilizado las tres técnicas que han sido explicadas en el apartado anterior. De esta forma hemos logrado conseguir un asistente de voz que analice una frase inicial recogida mediante un micrófono y procesada mediante la API de reconocimiento de voz Java Speech Recognition API posteriormente la separe en tokens, estos tokens son comparados con un fichero de lemas, y por último se obtiene una orden que será enviada al servidor de la vivienda mediante peticiones REST.

Para realizar la búsqueda, nuestro asistente utiliza la orden extraída de la frase inicial, y comienza una búsqueda en profundidad sobre los objetos y sus correspondientes acciones. Dado que el mensaje final tendrá la estructura de una acción, por ejemplo, si la frase inicial es "*Hola asistente, quiero que enciendas la luz de la cocina"*, el algoritmo de preprocesado de nuestro asistente de voz convertirá la frase en "encender luz cocina", de modo que la frase tendrá una estructura imperativa de manera que facilitará la búsqueda y la comparación con nuestra base de datos de tokens.

Los resultados que este asistente puede proporcionarnos serán los siguientes:

- **Obtiene varias coincidencias como resultado**

  Si nuestro buscador recibe varias coincidencias en el resultado, este mostrará las acciones que tienen el mismo peso referente a la búsqueda realizada.

- **No obtiene ningún resultado**

  Si nuestro buscador no obtiene ninguna coincidencia con la búsqueda, mostrará un mensaje automáticamente informando de la situación.

- **Obtiene un único resultado**

  Por último, si el buscador encuentra un único resultado en su búsqueda, procederá de forma automática a la realización de la acción solicitada.

3.5.3 Algoritmo de reconocimiento de objetos, propiedades y acciones.

Este algoritmo será el responsable de realizar el procesado de los datos de entrada y realizar la solicitud de operación al servidor remoto.

A continuación, serán descritos las etapas que constituye este algoritmo de Procesamiento del Lenguaje Natural:

- **Primera etapa. Separación de elementos**

  En esta primera etapa del algoritmo se tratará el mensaje de entrada, de manera que será procesado y dividido en palabras, que facilitarán el tratamiento del mensaje en etapas posteriores.

- **Segunda etapa. Stopwords**

  Una vez el mensaje a superado el primer proceso, esta estructura de palabras es expuesta a un filtrado en el que se eliminan los stopwords (palabras vacías) pertenecientes a este mensaje. De manera que el mensaje quedaría reducido a una estructura de tokens.

- **Tercera etapa. Tokens**

  En esta etapa, el proceso se basa en la extracción y la identificación de la orden a ejecutar a partir de la estructura de tokens resultante de las etapas anteriores. En esta etapa existe un proceso de filtrado realizando una comparación con el servidor de datos en el que se encuentran alojados los tokens relevantes con los que realizará la comparación. Los tokens de nuestro servidor de datos poseen un atributo denominado “tipo”, este atributo es el que nos indica si el token que estamos tratando se trata de

un objeto (puerta, luz, etc.), o una acción (abrir, encender, etc.). Una vez identificados los tokens, pasamos a trabajar con los lemas pertenecientes a estos tokens. De esta forma, conseguimos estandarizar la orden que buscamos clasificando el mensaje en acciones y objetos.

- **Cuarta etapa. Score**

  Una vez identificado el tipo de cada token, procedemos a la identificación de los objetos, realizando una búsqueda iterativa en el servidor de objetos que, a su vez, son comparados con los tokens de nuestro mensaje. En el proceso de comparación se va a necesitar el apoyo de un sistema de puntuación denominado “Score” el cual utilizaremos para determinar los objetos con los que se detecta alguna coincidencia al comparar los tokens de nuestro mensaje, con los asociados a cada uno de nuestros objetos.

  Una vez completado el proceso, este sistema Score determinara cuál o cuáles de los objetos existentes en el sistema poseen más similitud.

- **Quinta etapa. Determinación de acciones**

  Una vez que los objetos han sido identificados, pasamos al proceso de identificación de acciones. Para completar este proceso se ha realizado una búsqueda primero en profundidad sobre las propiedades de los objetos coincidentes en la cuarta etapa, de esta forma se pretende comparar cada una de las propiedades asociadas a cada objeto con nuestro token de tipo “accion” analizado en la tercera etapa.

- **Sexta etapa. Mostrar datos**

  Por último, esta sexta etapa se centra en mostrar las acciones (acción+objeto) resultantes de manera ilustrativa e intuitiva para el usuario como se puede observar en la siguiente imagen.

Ilustración 28: Listado de acción resultante

3.5.3. Servicios

Para la implementación de estos servicios se ha utilizado OpenHAB, un software para la integración de los diferentes sistemas y tecnologías de automatización y domótica dentro del hogar el cual, permite la creación de reglas para la automatización de estas operaciones.

Para proporcionar la posibilidad de reglas de automatización, este software ofrece una serie de interfaces, tanto web como móvil para que el usuario pueda interactuar con cada uno de los dispositivos, de forma independiente del software que estemos utilizando para llamar a los servicios.

A continuación, nombraremos algunos los servicios de los que disponemos en el servidor de la vivienda y de los que podemos hacer uso:

- Lista de eventos por ambiente:
  - **URL:** /event/environment/ambiente
  - **Parámetros:** idAmbiente.
  - **Tipo:** GET

- Consultar un sensor:
  - **URL:** /event/evento/environment/ambiente
  - **Parámetros:** idAmbiente,ObjetoEvent
  - **Tipo:** post

- Listar el estado actual de un ambiente:
  - **URL:** /object/environment/idEnvironment/currentstatus
  - **Parámetros:** idAmbiente,
  - **Tipo:** GET

En este apartado procederemos a mostrar algunos ejemplos que pueden servir de gran utilidad a la hora de comprender el funcionamiento de estas peticiones.

- **Ejemplo de petición GET:**

URL	http://192.168.83.108:8080/rest/items/IluminacionI
Tipo	GET
Información	Devuelve la información del estado de la iluminación del laboratorio

Ilustración 29: ejemplo servicios GET

URL	http://192.168.83.108:8080/rest/items/IluminacionColor
Tipo	GET
Información	Devuelve la información del color (RGB) de las luces

Ilustración 30: ejemplo servicios GET

- **Ejemplo de petición POST**

URL	http://192.168.83.108:8080/rest/items/IluminacionI
Tipo	POST
Información	Enciende/Apaga la iluminación
Parámetros (Payload RAW)	ON OFF

Ilustración 31: Ejemplo de petición Post

URL	http://192.168.83.108:8080/rest/items/IluminacionS
Tipo	POST
Información	Establece la intensidad de la iluminación, acepta valores de 0 a 100 ya que indica que el porcentaje total de iluminación
Parámetros (Payload RAW)	50

Ilustración 32:Ejemplo de petición post 1

Por último, vamos comentar las características del servidor "Servidor_Objetos". Éste se trata de un servidor bastante ligero, su desarrollo está preparado para poder ser embebido en otras aplicaciones, por lo que se decidió utilizar Jetty, un servidor de aplicaciones bastante ligero. Al ser tan ligero, su tamaño lo hace ideal para ofrecer sus servicios en aplicaciones independientes.

Ilustración 33: Jetty

Otro aspecto que debemos destacar de este servidor es la estructura de persistencia que hemos utilizado. Para su implementación hemos utilizado ficheros en formato JSON en los que almacenábamos todos los objetos y tokens creados como hemos mostrado anteriormente (Ver apartado 3.4.2 Diseño de datos). Estos ficheros se encargan de mantener la persistencia de los datos de la aplicación mediante el uso de servicios REST los cuales mostraremos a continuación.

A continuación, expondremos una lista con los servicios que tenemos disponibles en nuestra aplicación, y los cuales pueden ser utilizados en cualquier momento.

- **Obtener todos los objetos creados en el sistema**

Url	http://localhost:8080/?operation=getObjects
Tipo	GET
Información	Solicita al servidor el envío de todos los objetos que se encuentran creados en el sistema.

- **Crear un objeto**

Url	http://localhost:8080/?operation=addObject&data={"id":"luz3","tokens":["bombilla", "cocina"],"properties":[]}
Tipo	GET
Información	Crear un nuevo objeto con identificador “Luz3”, donde se le pasa toda la información relevante por la Url.

- **Eliminar un objeto**

Url	http://localhost:8080/?operation=removeObject&data={"id":"luz3","tokens":[],"properties":[]}
Tipo	GET
Información	Eliminar un objeto que posee como id “Luz3”, donde se le pasa toda la información relevante por la Url.

- **Añadir propiedades a un objeto**

Url	localhost:8080/?operation=addProperty&data={"id":"luz3","tokens":["bombilla", "cocina"],"properties":[{"id":"on","tokens":["enceder"],"url":"http://smart-lab/set:luz3:value:on","description":"encender la bombilla de la cocina"},{"id":"off","tokens":["apagar"],"url":"http://smart-lab/set:luz3:value:off","description":"apagar la bombilla de la cocina"}]}
Tipo	GET
Información	Añadirle propiedades (ON y OFF) al objeto con id ”luz3”.

- **Obtener todos los tokens creados en el sistema**

Url	http://localhost:8080/?operation=getTokens
Tipo	GET
Información	Obtener todos los tokens creados en el sistema

- **Crear token**

Url	http://localhost:8080/?operation=addToken&data={"id":"encender","tokens":["encender","dar"]}
Tipo	GET
Información	Crear un nuevo token con id “encender”

- **Eliminar token**

Url	http://localhost:8080/?operation=removeToken&data={"id":"apagar", "tokens":["apagar","quitar"]}
Tipo	GET
Información	Eliminar token con id “apagar”

3.5.4 Tecnología en el cliente de nuestro sistema web

- **Html5:** (Hyper Text Markup Language) Es un lenguaje utilizado para dar estructura y presentar el contenido para la web. Este lenguaje tiene como objetivo facilitar el desarrollo web. Html5 posee herramientas para la administración efectiva de datos, audio y video. Por otro lado, también nos facilita la creación de sitios web adaptativos, es decir capaces de poder reproducirse en multitud de dispositivos, incluyendo los portátiles. Para nuestro proyecto, hemos elegido este lenguaje para armar la vista de nuestra aplicación ya que se trata de un lenguaje puntero y nos ofrecía numerosas ventajas respecto a otros lenguajes.[11]
- **Css3:** (Cascading Style Sheets) Para definir la estructura de la web, HTML utiliza etiquetas en las que definir un estilo concreto para cada apartado de nuestra página y así estandarizar la apariencia de nuestro sitio web.[12]

- **Bootstrap**: Bootstrap es un framework para crear páginas web. Este entorno esta basado en Html5, css3 y Jquery, lo que significa que ahorra mucho tiempo y simplifica mucho el trabajo de los desarrolladores ya que posee ya parte del trabajo hecho.[13]
  Las ventajas que ofrece Bootstrap son las siguientes:

  - Rapidez ya que gran cantidad del trabajo ya está realizado.
  - Posee una gran cantidad de desarrolladores en Git Hub.
  - Tiene integrados Html5 y css3 por lo que resulta mas poderoso y ligero a la hora de ponerlo en marcha en los navegadores
  - Integra 12 librerías de Jquery.

- **Javascript:** Al igual que HTML, Javascript es un lenguaje de scripts o rutinas utilizado para la construcción de páginas web, aunque en su funcionamiento se asimila mucho al lenguaje Java, ha sido creado explícitamente para un uso web. [14]
  Javascript fue creado por Netscape Corporation, y hoy en día sigue teniendo éxito y resultando muy útil para la creación de páginas web más dinámicas.

Este lenguaje mejora la gestión cliente/servidor y en el cliente puede realiar tareas tales como:

- Comprobación de validez de campos.
- Manejo de ventanas.
- Tratamiento de cadenas de texto.

3.5.5. Conexión del sistema web

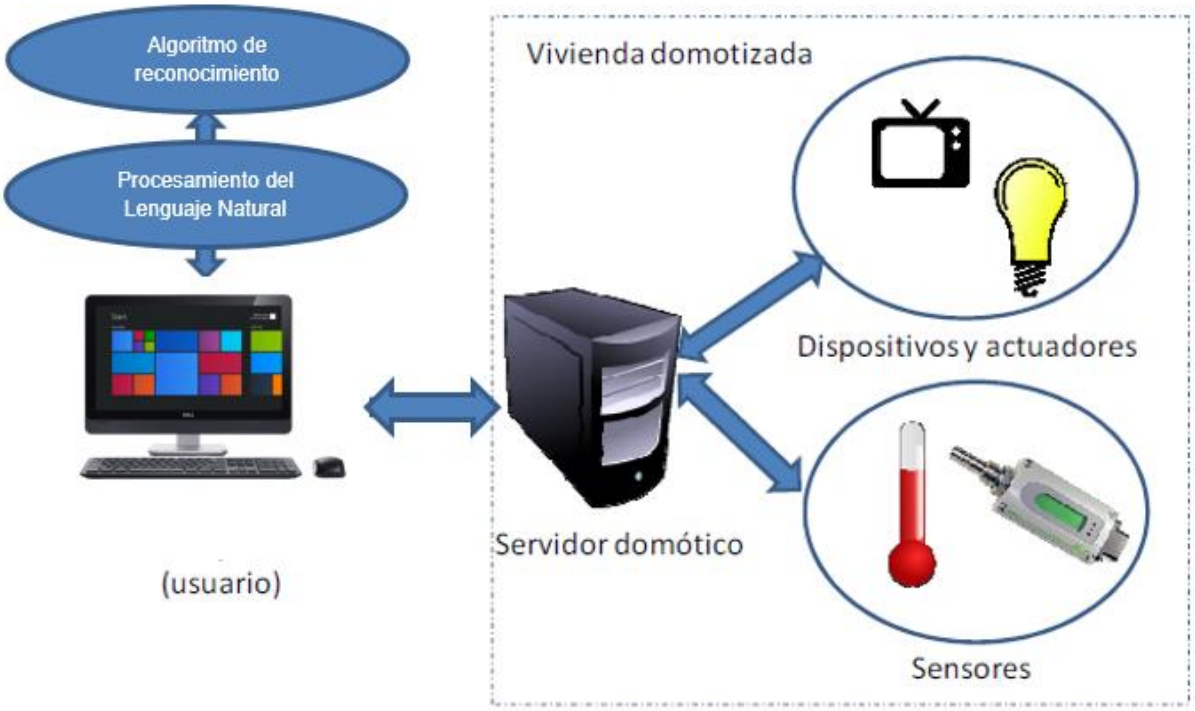

Ilustración 34: conexión de sistema web

La imagen anterior muestra la estructura que podría tener nuestro sistema web de reconocimiento por voz, de manera que nuestro cliente web conecte mediante servicios REST con el servidor domótico alojado en el ambiente inteligente, que será responsable de realizar todas las operaciones indicadas o solicitadas por el cliente.

3.5.6. Herramientas utilizadas para el desarrollo de nuestro sistema.

A continuación, después de la descripción de todos los lenguajes y framework utilizados para la creación y desarrollo de nuestro proyecto, procederemos a la exposición de las herramientas utilizadas en el desarrollo de nuestro proyecto.

- **NetBeans 8.2:** Esta herramienta es un entorno de programación, desarrollado principalmente para el lenguaje Java, pero a pesar de esto integra la posibilidad de trabajar con un amplio rango de tecnologías de desarrollo tanto de escritorio, como aplicaciones web. Mediante este entorno hemos realizado el desarrollo de nuestro sistema web. [15]

3.6 Pruebas

Como hemos comentado anteriormente, el proceso de pruebas es el último proceso de ingeniería de software. Mediante esta fase el desarrollador puede comprobar y evaluar y mejorar la calidad del producto final. Por lo tanto, el objetivo de estas pruebas seria maximizar la cantidad de errores o defectos con respecto al desarrollo o funcionamiento del producto, y una vez realizado esto, permite a los desarrolladores del sistema corregir y aumentar la confiabilidad del sistema.

Por las razones que hemos comentado anteriormente, creemos que es una de las fases más importantes de la ingeniería del software. Las Personas encargadas de realizar las pruebas pueden ser o los desarrolladores del sistema o un examinador independiente y que se encuentre excluido totalmente del sistema, ya que, a la hora de realizar las pruebas de calidad y funcionalidad, éste será más objetivo e imparcial a la hora de detectar los errores.

A continuación, mostraremos una serie de tablas (Ver tablas desde Tabla 2 a Tabla 15) en las que se mostrará el funcionamiento basándonos en los requisitos funcionales de nuestro sistema:

3.6.1 Test

Tabla 2: Test 1: Puesta en marcha del sistema de búsqueda por voz

Condiciones	El usuario debe estar dentro del sistema
Acción	El usuario debe pulsar el activador del micrófono (icono de micrófono) he introducir el mensaje de de la acción a realizar mediante voz.
CheckPoint	El sistema procesa los datos introducidos pone en marcha el filtrado.

Tabla 3: Test 2: Localización de comandos fallida

Condiciones	El usuario debe haber introducido el mensaje
Acción	El sistema pone en marcha el sistema de búsqueda con los datos introducidos y no encuentra ningún comando coincidente.
CheckPoint	El sistema muestra los un mensaje al usuario de alerta, indicando que no existen comandos con coincidencias en la búsqueda.

Tabla 4: Test 3: Localización múltiple de comandos sin acciones

Condiciones	El usuario debe haber introducido el mensaje, pero no ha introducido la acción a realizar.
Acción	El sistema pone en marcha el sistema de búsqueda con los datos introducidos y encuentra varios comandos que mantienen coincidencias con el mensaje introducido.
CheckPoint	El sistema muestra todos los comandos con los que han aparecido coincidencias con todas sus correspondientes acciones.

Tabla 5: Test 4: Localización múltiple de comandos con acciones

Condiciones	El usuario debe haber introducido el mensaje y a su vez, ha introducido la acción a realizar.
Acción	El sistema pone en marcha el sistema de búsqueda con los datos introducidos y encuentra varios comandos que mantienen coincidencias con el mensaje introducido.
CheckPoint	El sistema muestra todos los comandos con los que han aparecido coincidencias y para cada coando encontrado mustra solo la acción solicitada.

Tabla 6: Test 5: Localización única de comando con acciones

Condiciones	El usuario debe haber introducido el mensaje y a su vez, ha introducido la acción a realizar.
Acción	El sistema pone en marcha el sistema de búsqueda con los datos introducidos y encuentra un único comando que mantiene coincidencia con el mensaje introducido.
CheckPoint	El sistema muestra el comando con el que han aparecido coincidencias y de forma automática realiza la acción solicitada.

Tabla 7: Test 6: Creación correcta de un Objeto

Condiciones	El usuario debe estar dentro de nuestro sistema.
Acción	El usuario rellena los formularios del nuevo objeto, y una vez rellenados todos los campos, pulsa el botón “Crear Objeto”.
CheckPoint	El sistema valida todos los campos y datos introducidos, procede a la creación del objeto y muestra todos los objetos instalados incluyendo el creado recientemente.

Tabla 8: Test 7: Creación incorrecta de un objeto

Condiciones	El usuario debe estar dentro de nuestro sistema.
Acción	El usuario rellena los formularios correspondientes a la creación del objeto, pero deja alguno de estos sin rellenar y pulsa el botón “Crear Objeto”.
CheckPoint	El sistema valida todos los campos y detecta que hay un campo que se encuentra vacío, por lo que manda un mensaje emergente al usuario informando que no puede haber ningún campo del formulario vacío.

Tabla 9: Test 8: Adición de propiedades a objetos

Condiciones	El usuario debe estar dentro de nuestro sistema.
Acción	El usuario rellena los formularios correspondientes a la adición de propiedades y pulsa el botón “Añadir Propiedad”
CheckPoint	El sistema valida todos los campos y datos introducidos, procede a la creación y adición de la propiedad al objeto correspondiente. Posteriormente, muestra todos los objetos instalados incluyendo sus propiedades.

Tabla 10: Test 9: Eliminar propiedad de Objeto

Condiciones	El usuario debe estar dentro de nuestro sistema.
Acción	El usuario busca la propiedad que quiere eliminar en el apartado de editar comandos y pulsa el botón “Eliminar Propiedad”.
CheckPoint	El sistema recibe la petición de eliminación de la propiedad y lanza una petición de confirmación al usuario, si éste la confirma, entonces el sistema procederá a la eliminación.

Tabla 11: Test 10: Eliminación de objetos

Condiciones	El usuario debe estar dentro de nuestro sistema.
Acción	El usuario busca el objeto que quiere eliminar en el apartado de editar comandos y pulsa el botón “Eliminar Objeto”
CheckPoint	El sistema recibe la petición de eliminación del objeto y lanza una petición de confirmación al usuario, si éste la confirma, entonces el sistema procederá a la eliminación.

Tabla 12: Test 11: Listado de comandos

Condiciones	El usuario debe estar dentro de nuestro sistema.
Acción	El usuario solicita la acción de listar los comandos existentes.
CheckPoint	El sistema recoge la petición, recoge del servidor todos los comandos creados, y los muestra mediante una lista junto con todas sus acciones correspondientes.

Tabla 13:Test 12: Creación correcta de tokens

Condiciones	El usuario debe estar dentro de nuestro sistema.
Acción	El usuario rellena los formularios del nuevo Token, y una vez rellenados todos los campos, pulsa el botón “Crear Token”.
CheckPoint	El sistema valida todos los campos y datos introducidos, procede a la creación del token y posteriormente muestra todos los tokens creados incluyendo el creado recientemente.

Tabla 14: Test 13: Creación incorrecta de tokens

Condiciones	El usuario debe estar dentro de nuestro sistema.
Acción	El usuario rellena los formularios correspondientes a la creación del token, pero deja alguno de estos sin rellenar y pulsa el botón “Crear Token”.
CheckPoint	El sistema valida todos los campos y detecta que hay un campo que se encuentra vacío, por lo que manda un mensaje emergente al usuario informando que no puede haber ningún campo del formulario vacío.

Tabla 15: Test 14: Listado de tokens

Condiciones	El usuario debe estar dentro de nuestro sistema.
Acción	El usuario solicita la acción de listar los tokens existentes
CheckPoint	El sistema recoge la petición, recoge del servidor todos los tokens creados y los muestra mediante una lista.

3.6.2 Resultados

En este apartado mostraremos la tabla de resultados en la cual reflejamos en que test se han producido errores durante las pruebas realizadas.

Test	Resultado	Problemas detectados	Final
Test 1	No superado	Fallo en el reconocimiento de voz	Corregido
Test 2	Superado		Superado
Test 3	Superado		Superado
Test 4	No Superado	Problema al realizar la petición POST.	Superado
Test 5	Superado		Superado
Test 6	Superado		Superado
Test 7	Superado		Superado
Test 8	Superado		Superado
Test 9	Superado		Superado
Test 10	Superado		Superado
Test 11	Superado		Superado
Test 12	Superado		Superado
Test 13	Superado		Superado
Test 14	Superado		Superado
Test 15	Superado		Superado

Como se puede apreciar en la tabla, durante el periodo de pruebas se han detectado principalmente dos errores importantes:

- **Error en Test 1**

Durante la realización de este test descubrimos un error en el procesamiento de las palabras del buscador, dado que la estructura en la que transformaba el mensaje para su procesamiento era incorrecta y causaba fallos de entendimiento al sistema. Por esta razón decidimos cambiar la estructura y de esta forma conseguir paliar el problema

- **Error en Test 4**

Esta anomalía fue delatada al realizar una búsqueda en la que se introdujese una acción junto con un objeto. Al realizar esta búsqueda, el sistema mostraba varias coincidencias por lo que debes seleccionar una explícitamente, y en este punto era donde se encontraba el problema dado que el array de objetos que mostraba no era correcto, por lo que al realizar la petición POST el sistema causaba errores.

3.6.3 Validación Reconocimiento de voz

Este apartado de pruebas estará destinado a la comprobación de funcionamiento del reconocedor de voz implementado en nuestra aplicación.

A continuación, mostraremos una tabla en la que mostraremos numerosas entradas de usuario, junto con el reconocimiento obtenido mediante nuestro sistema de procesamiento de lenguaje natural.

Tabla 16: Pruebas reconocimiento de voz

Prueba	Entrada de usuario	Tokens	Mensaje procesado	Objetos y acciones obtenidas
1	**“enciéndeme la luz salon1”**	Enciéndeme, luz, salon1	Encender salon1	**-Acción**: Encender **Objeto:** salón 1
2	**“enciéndeme la luz del salón”**	Enciéndeme, luz, salón	Encender luz salón	- **Acción:** Encender **Objeto:** salon1 - **Acción:** Encender **Objeto:** salon2
3	**“Apaga el hall”**	Apaga, hall	Apagar hall	- **Acción:** Apagar **Objeto:** hall
4	**“Apágame la luz del dormitorio”**	Apágame, luz, dormitorio	Apagar dormitorio	- **Acción:** Apagar **Objeto:** Dormitorio
5	**“enciende la luz”**	Enciende luz	Encender luz	- **Acción:** Encender **Objeto:** salon1 - **Acción:** Encender **Objeto:** salon2 - **Acción:** Encender **Objeto:** hall - **Acción:** Encender **Objeto:** dormitorio - **Acción:** Encender **Objeto:** cocina
6	**abre**	abre	abrir	- **Acción:** Abrir **Objeto:** puerta
7	**Enciende una escena**	Enciende, escena	Encender escena	- **Acción:** Encender **Objeto:** escena_dormir

				- **Acción:** Encender **Objeto:** escena_despertar
8	**"enciende la escena de dormir"**	Enciende, escena, dormir	Encender escena dormir	- **Acción:** Encender **Objeto:** escena_dormir
9	**"Cierra la puerta"**	Cierra, puerta	Cerrar puerta	- **Acción:** Cerrar **Objeto:** puerta
10	**"Qué hay en la cocina"**	Hay, cocina	cocina	- **Acción:** Encender **Objeto:** cocina - **Acción:** Apagar **Objeto:** cocina
11	**"apágame salon2"**	Apágame, salon2	Apagar salon2	- **Acción:** Apagar **Objeto:** salon2
12	**"apágame la cocina"**	Apágame, cocina	Apagar cocina	- **Acción:**Apagar **Objeto:**Cocina
13	**"muéstrame todos los comandos"**	Muéstrame, todos, comandos	Mostrar Comandos	- **Acción:** (todas) **Objeto:** (todos)

3.6.3.1 Análisis de los resultados

Como hemos podido apreciar en la tabla anterior (ver tabla 16), dependiendo del mensaje que es introducido por el cliente, el sistema de reconocimiento nos devuelve unos resultados u otros.

A continuación, se procederá a describir algunos de los comportamientos que hemos podido apreciar en dicha tabla:

- **Prueba 1**

  En este caso, el usuario introduce un mensaje objetivo, ya que nombre explícitamente el id específico del objeto en sí. En este caso, el sistema procede a realizar la orden ya que únicamente se ha encontrado un resultado y una acción (encender).

- **Prueba 2**

  En esta prueba, dado que disponemos de dos objetos, los cuales poseen tokens relacionados con “salón”, el sistema no logra reconocer a cuál se refiere el cliente exactamente, por lo que decide mostrar los dos objetos relevantes junto con la acción detectada, “encender” en este caso.

- **Prueba 5**

  En este caso, el sistema no logra concluir la acción que el cliente desea realizar, dado que en el mensaje no proporciona una información detallada por lo que el sistema decide mostrar todos los objetos que coinciden con la operación y el token detectados (“encender” y “luz”).

- **Prueba 6**

  En esta prueba, el sistema detecta una sola acción (“abrir”), por lo que compara con todos los objetos de nuestra base de datos y busca cuál de ellos posee una operación que coincida con abrir, en este caso solamente encuentra el objeto “puerta” con la propiedad “abrir”, por lo que, al haber solamente una alternativa, el sistema procede a la realización de la operación de forma automática.

- **Prueba 10**

  En este caso, el sistema únicamente detecta un token (“cocina”), por lo que en su búsqueda solamente encuentra un objeto relacionado con cocina, en este caso es una luz, por lo que procede a mostrar dicho objeto con todas las acciones pertenecientes a éste.

- **Prueba 13**

  Por último, en esta prueba el sistema detecta la operación “mostrar_todos”, por lo que procede al listado de todos los comandos juntos con sus operaciones.

4. Conclusiones y posibles líneas de trabajo

Actualmente en nuestro país dispone de un censo considerable de personas mayores de 65 años, concretamente se sitúa en un 18,2% de la población española, es decir aproximadamente unos nueve millones de personas. [16][17][18]

Según los estudios y estadísticas del Instituto nacional de estadística, la proyección de esta población va en importante aumento por lo que se calcula que ese 18,2% de población mayor de 65 años, pasará a convertirse en un 24,9% sobre el año 2029 por lo que el número de personas ancianas sería mucho mayor que actualmente.

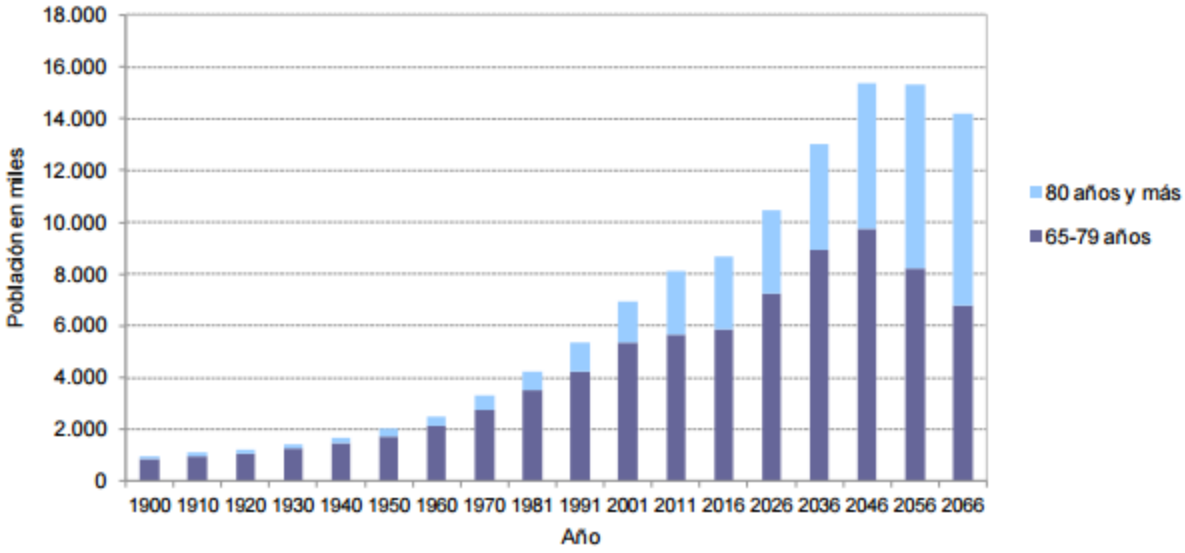

Ilustración 35: Gráfica de población en España

Como podemos apreciar en el gráfico anterior, la población está envejeciendo por lo que cada vez las personas pasarán más tiempo en sus hogares. Por este motivo, la inteligencia ambiental en hogares está tomando cada vez más protagonismo, y cada día más empresas españolas se esfuerzan por conservar al máximo la independencia de los mayores sin renunciar a su calidad de vida.

Actualmente, el nivel de desarrollo tecnológico de este tipo de aplicaciones es realmente básico, como comenta el director del Gabinete Técnico del colegio de Aparejadores de Madrid:

> "*En la actualidad, los sistemas más básicos de domótica orientados al campo de la salud consisten en alarmas accionadas por el propio usuario que avisan de las urgencias a empresas y centros de asistencia, mientras que los más sofisticados incluyen sensores capaces de detectar el movimiento, la respiración o el ritmo cardíaco. Los sensores de última generación pueden incluso detectar una caída*"

Este proyecto se centra en este sector de la población, estas personas cada vez necesitarán más ayuda a la hora de realizar las tareas domésticas como por ejemplo abrir o cerrar una puerta.

El objetivo de este trabajo ha sido realizar un sistema de control por voz y habla en el que, mediante un sistema de comandos persistentes, el cliente pueda realizar actividades cotidianas como encender y apagar luces, abrir y cerrar puertas, encender electrodomésticos, de manera que estas actividades le resulten más fáciles, intuitivas y menos físicas que si las realizan de la forma cotidiana.

El sistema software ha sido creado y desarrollado con la ayuda del apartamento "SmartLab" del departamento de inteligencia ambiental del Centro de Estudios Avanzados de Tecnologías de la Información y Comunicación (CEATIC).

Para concluir, quiero expresar mi sensación satisfactoria al realizar este proyecto y poder dar uso a todo el conocimiento recibido en estos años de grado, y el adquirido durante la realización de este trabajo ya que he aprendido nuevas aplicaciones de la informática, abriéndome puertas a nuevas áreas de esta materia.

5. Anexos

Anexo 1. Contenido CD-ROOM

En el Anexo 1 mostraremos todos los archivos que contendrá el CD-ROOM adjunto con la memoria de este trabajo.

- **Aplicaciones finales**

    - **Aplicación cliente Web:** Aplicación HTML
    - **Servidor Json:** Servidor Java que se encarga de la persistencia de los tokens y objetos.

- **Código fuente**

En el interior de esta carpeta, encontraremos todo el código desarrollado a lo largo de este trabajo. Para poder abrirlos necesitaremos la herramienta de desarrollo NetBeans 7.5 o superior.

- **Memoria.pdf**

Este fichero se trata de un documento en formato .pdf que contiene toda la documentación del trabajo de fin de grado.

- **Video-Review.mp4**

Este archivo contiene el vídeo en el que mostramos el funcionamiento del sistema web en el laboratorio.

Anexo 2. Manual de instalación

En este anexo mostraremos el manual de instalación a seguir para poder desplegar el sistema web para el uso y manejo de los entornos inteligentes.

Nuestra aplicación se apoya de un servidor auxiliar el cual se encarga de la gestión de objetos y tokens, por lo que cuando arranca nuestro cliente web realiza una petición al servidor para que éste, mande toda la información persistente y una vez cargada, poder comenzar a trabajar con nuestro sistema web.

Este servidor es el denominado "Servidor_Objetos". Para demostrar el funcionamiento de esta aplicación lo haremos mediante el IDE Netbeans 8.2 en el que instalaremos primeramente el servidor alojado en la carpeta llamada "servidor_objetos". Para ello, en Netbeans, desplegamos el servidor como podemos observar en la siguiente imagen (ver ilustración 30).

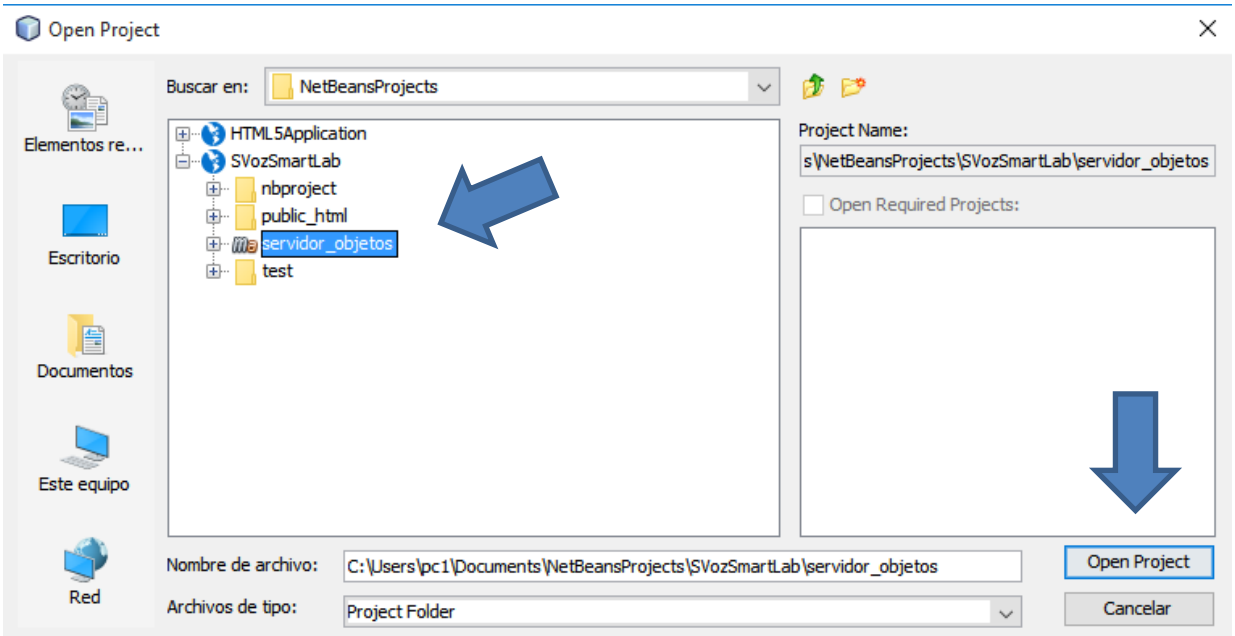

Ilustración 36: Desplegar Servidor_Objetos

Una vez tenemos el servidor cargado en nuestro IDE procedemos a cargar nuestro sistema web llamado “SVozSmartLab”. Para ello, deberíamos realizar los mismos pasos que hemos realizado para desplegar nuestro servidor (Ver ilustración 31).

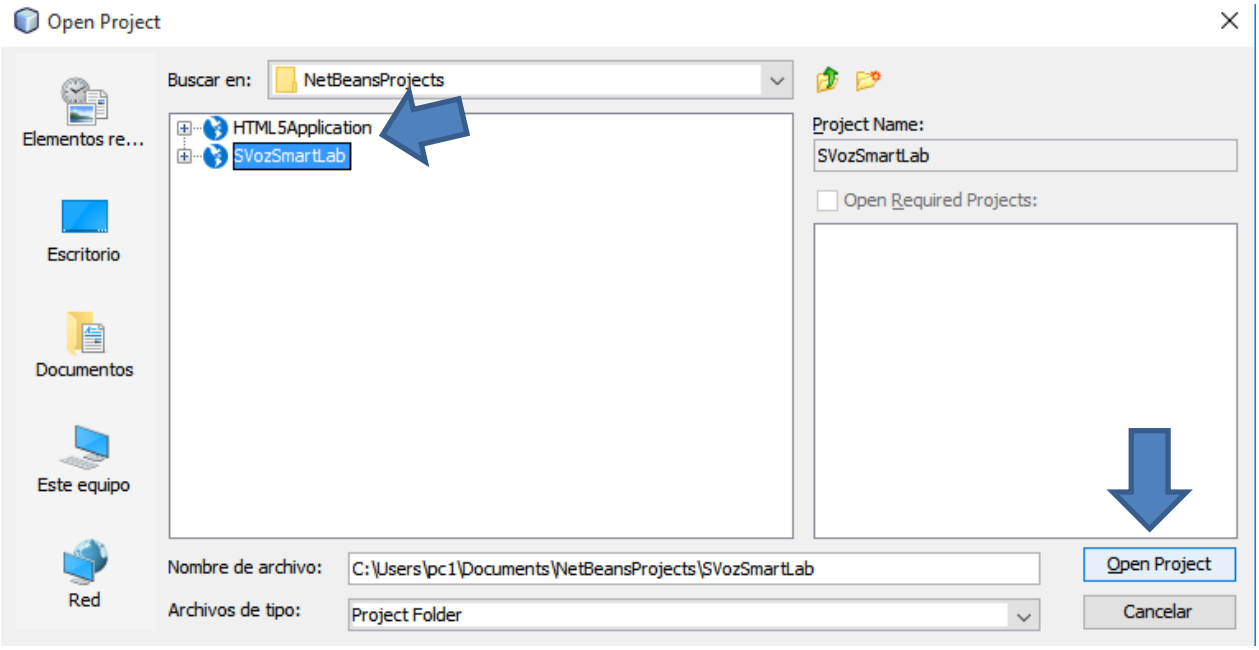

Ilustración 37: Desplegar SVozSmartLab

A continuación, una vez tenemos cargados los dos proyectos (Servidor_Objetos y SVozSmartLab), procedemos a su lanzamiento, pero para ello debemos seguir un orden, ya que el proyecto “SVozSmartLab” realiza peticiones Rest a nuestro servidor para poder cargar todos los datos necesarios, por lo que en primer lugar debemos ejecutar “Servidor_objetos” (Ver ilustración 32).

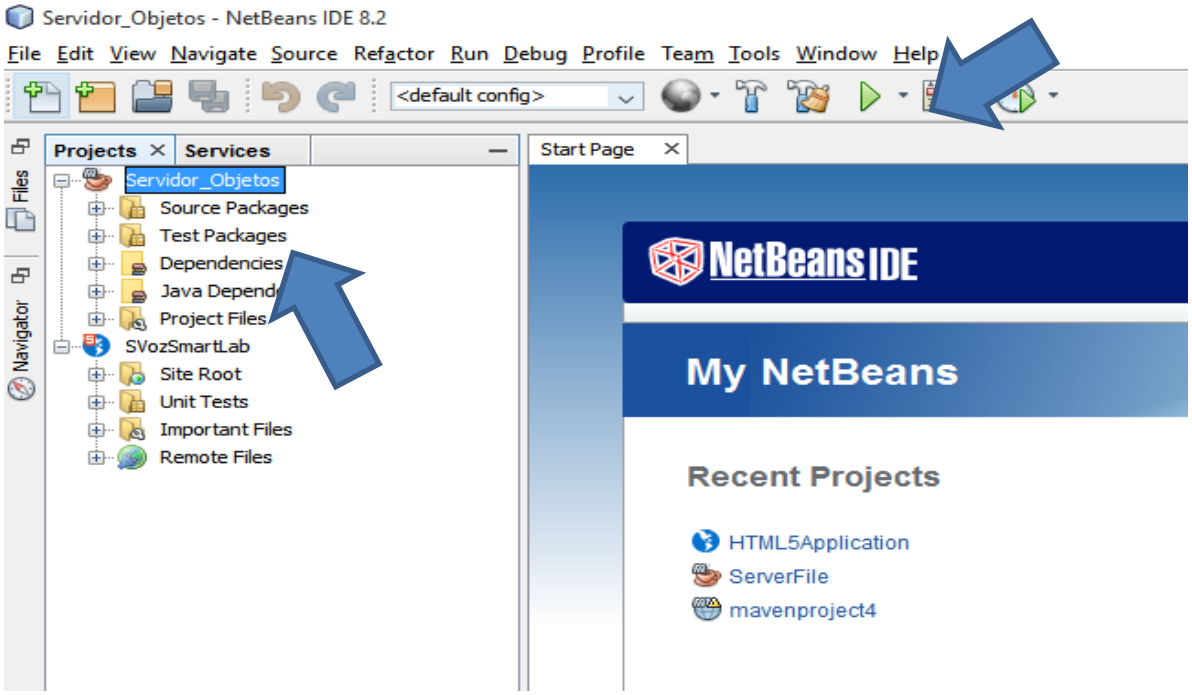

Ilustración 38: Ejecución Servidor_objetos

Una vez lanzado el servidor, procedemos al despliegue de “SVozSmartLab”. Una vez desplegado, podemos comenzar a utilizar en nuestro propio navegador la aplicación web (Ver Ilustración 33).

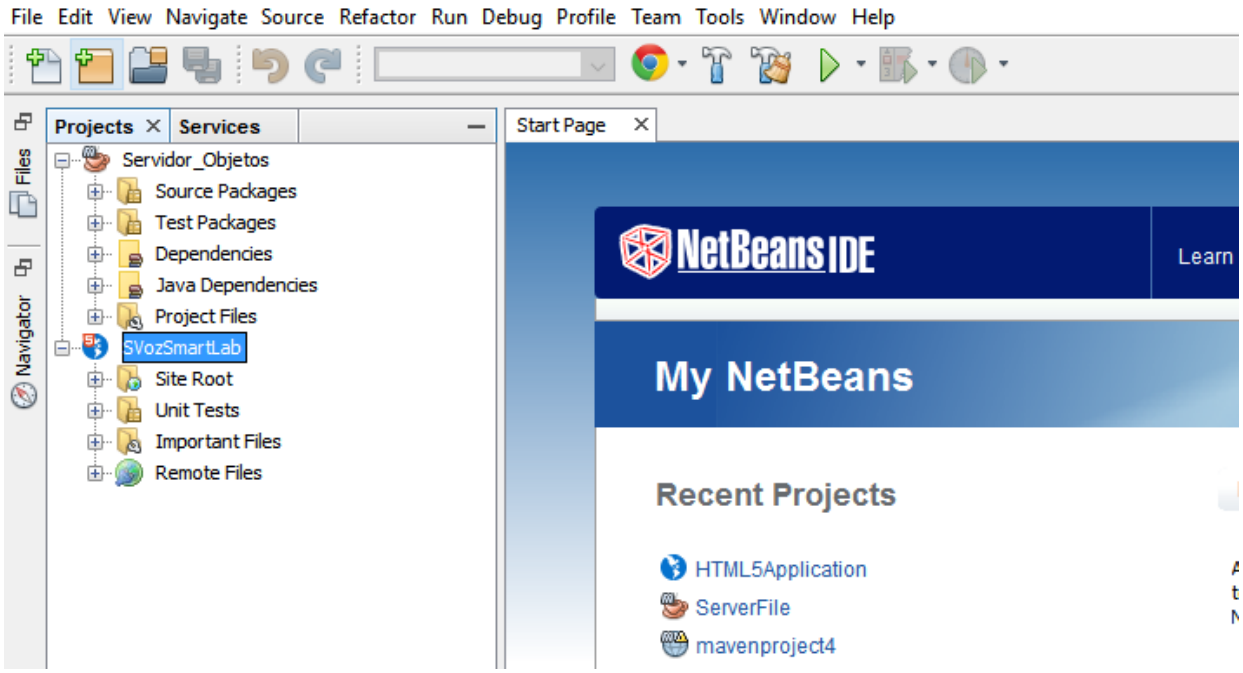

Ilustración 39: Ejecución SVozSmartLab

Anexo 3. Manual de uso

A3.1 Página principal

Como primer paso, antes de ejecutar nuestra aplicación web, nos tenemos que asegurar de que tenemos lanzado nuestro servidor de tokens, ya que la aplicación web la primera función que realiza justamente cuando arranca el sistema, es la comprobación de todos los datos enviando varias peticiones Rest a nuestro servidor para de esta forma poder mostrar todos los objetos y tokens de manera correcta para poder trabajar con ellos posteriormente.

Ilustración 40: Página principal 1

Ilustración 41: Página principal 2

Como podemos apreciar en las ilustraciones 30 y 31, éste sería el aspecto de nuestra página principal. En ella disponemos de una barra de navegación donde podemos dirigirnos a la configuración de nuestro sistema y al apartado de contacto donde mostraremos alguna información relevante con relación a nuestro proyecto.

En nuestra página principal se dispone de la función principal de esta aplicación, que es la de reconocimiento de voz y habla, por el cual el usuario puede realizar una búsqueda exhaustiva de los comandos que desea activar en cada momento. Para poder hacer uso de esta utilidad solamente debemos pulsar el icono de la grabación (micrófono) y posteriormente comenzar a mencionar nuestra frase de búsqueda, durante ese momento, el sistema realizará la escucha hasta que el usuario realice un silencio considerable, lo cual significará que ha finalizado su frase. Una vez el sistema a procesado los datos relevantes, pasará a mostrar el mensaje recibido por pantalla y comenzará realización de la acción solicitada, pero en el caso de que el sistema tenga varias coincidencias en el proceso de búsqueda, lanzará un mensaje alterante de que se han encontrado varios resultados y mostrará los resultados en pantalla, para que aquí el usuario pueda realizar la operación deseada.

A3.2 Menú de configuración

En este menú, el usuario podrá realizar y administrar toda la configuración referente al sistema y a los datos relevantes para él. En él encontraremos un menú lateral en la parte izquierda donde tendremos las numerosas tareas que podemos realizar. (Ver ilustración 32).

Ilustración 42: Menú de configuración

Como podemos observar en la imagen anterior, en el menú de que aparece en pantalla podremos elegir entre cuatro opciones, dos de ellas serán para ver las listas de objetos y tokens existentes y otras dos de configuración y edición de los datos del sistema como veremos en las siguientes ilustraciones.

A3.2.1 Lista de comandos

Mediante esta tarea, que se activará al pulsar el botón de Listar comandos, situado en la primera posición de nuestro menú, nos permitirá visualizar todos y cada uno de los objetos que tenemos creados en el sistema (Ver ilustración 33).

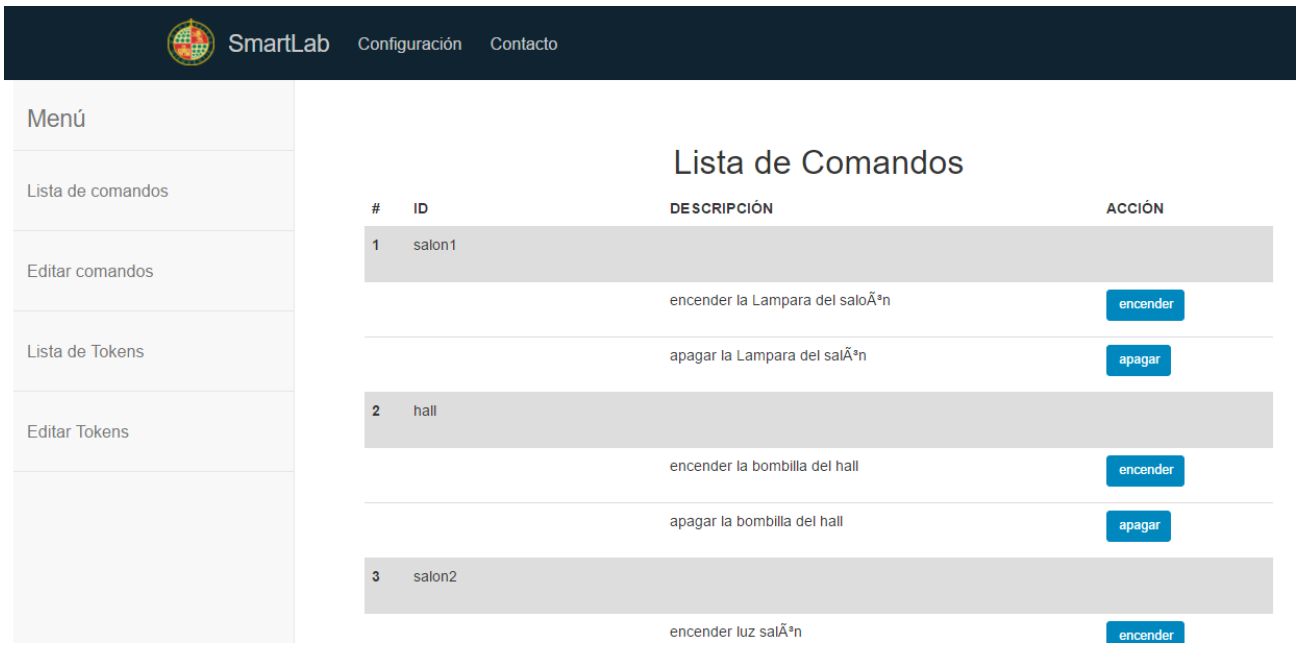

Ilustración 43: Lista de Comandos

Como podemos apreciar en la imagen, cada objeto perteneciente a la lista aparece acompañado de sus correspondientes propiedades o acciones que tienen disponibles junto con un botón de activación para realizar la petición directamente desde esta lista.

A3.2.2 Editar comandos

En esta sección, podremos manipular de total forma los objetos de nuestro sistema, de manera que podremos crear y añadir nuevos objetos o incluso eliminarlos. A su vez, también podremos modificar estos objetos añadiendo o eliminando sus propiedades asociadas. (Ver ilustración 34,35 y 36).

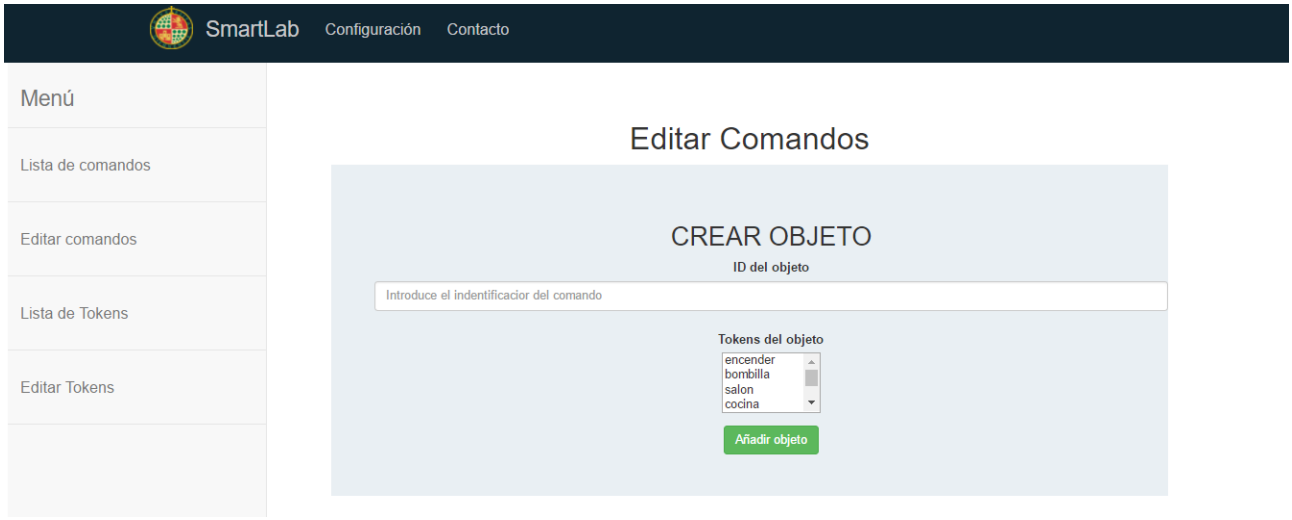

Ilustración 44: Editar comandos 1

Ilustración 45:Editar comandos 2

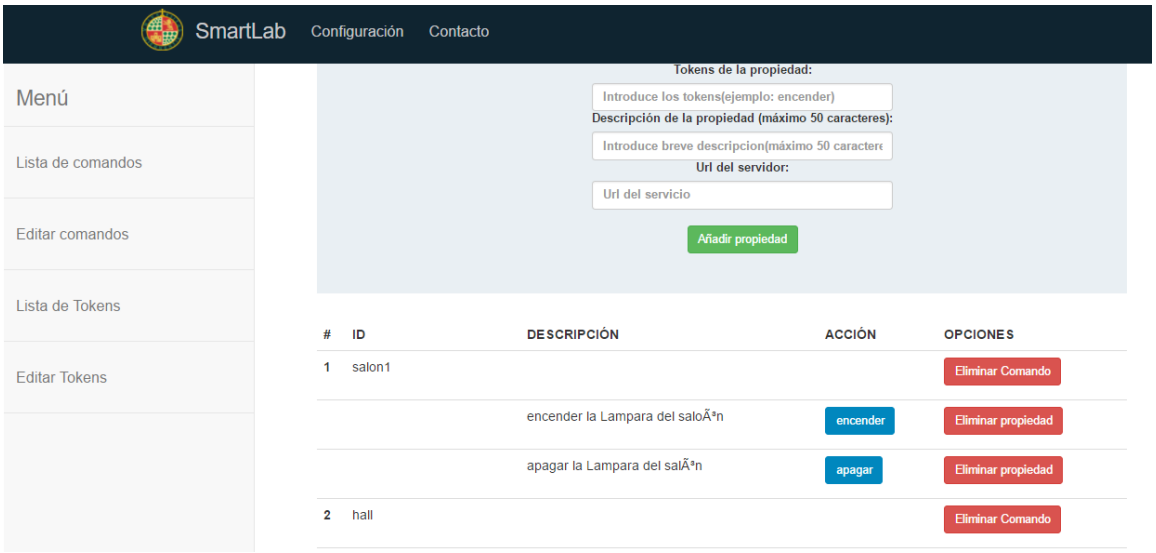

Ilustración 46: Editar comandos 3

A3.2.3 Lista de comandos

Como podeos apreciar en la Ilustración 37, esta operación lista los tokens que el sistema utiliza para realizar la búsqueda y el procesado de la información que nosotros le proporcionamos mediante el asistente de voz. De esta manera podremos apreciar la estructura que tiene cada uno de los tokens del sistema (Ver ilustración 37).

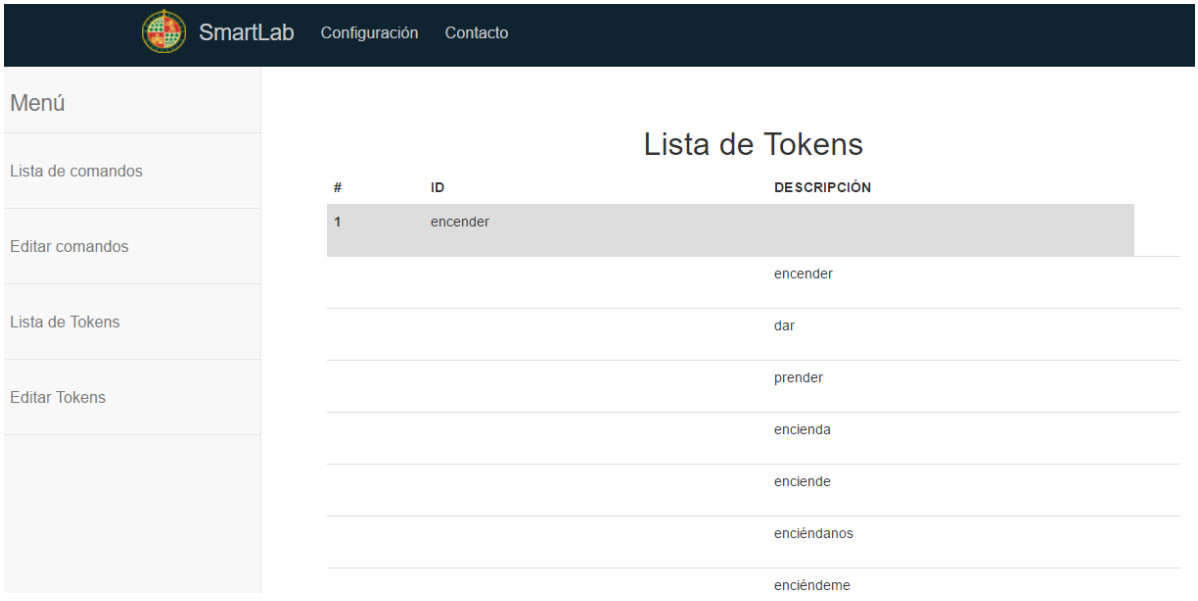

Ilustración 47: Lista de tokens

A3.2.4 Editar Tokens

Por último, el sistema posee la funcionalidad de poder personalizar estos tokens dependiendo de las necesidades de cada usuario, de manera que podrá modificar estos tokens dependiendo de los objetos que tena instalados en su hogar y a su vez estará modificando el sistema de búsqueda por reconocimiento de voz, ya que, como hemos comentado anteriormente, nuestro asistente por voz se apoya en una estructura de tokens proporcionada por nuestro servidor. (ver ilustraciones 38 y 39).

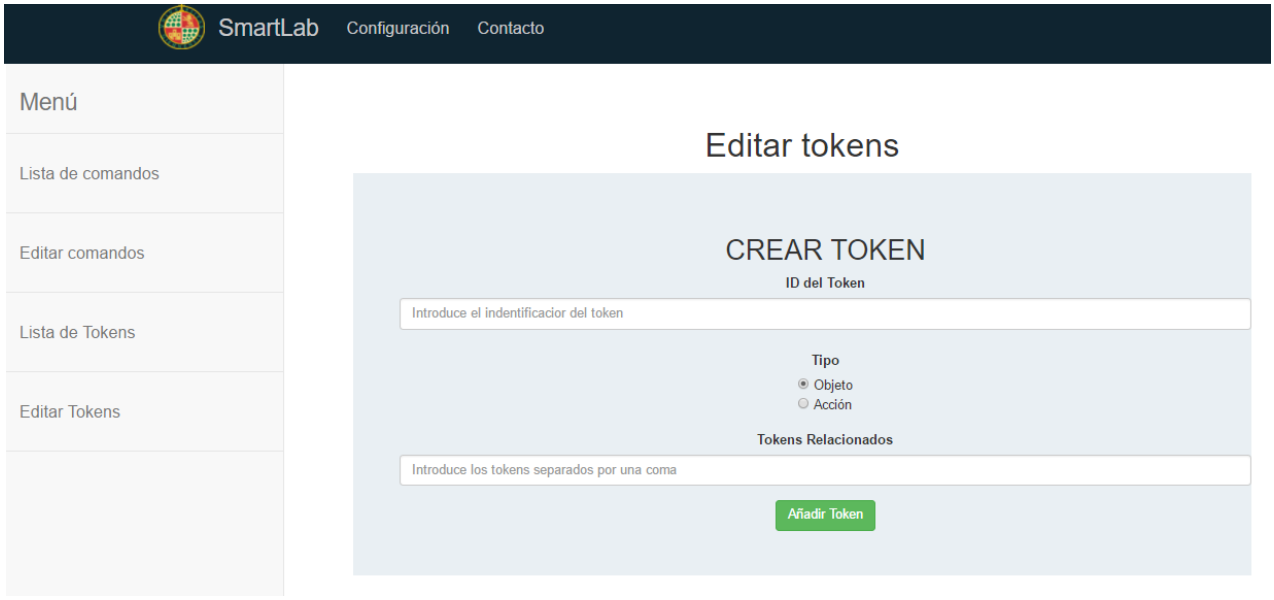

Ilustración 48: Editar Tokens 1

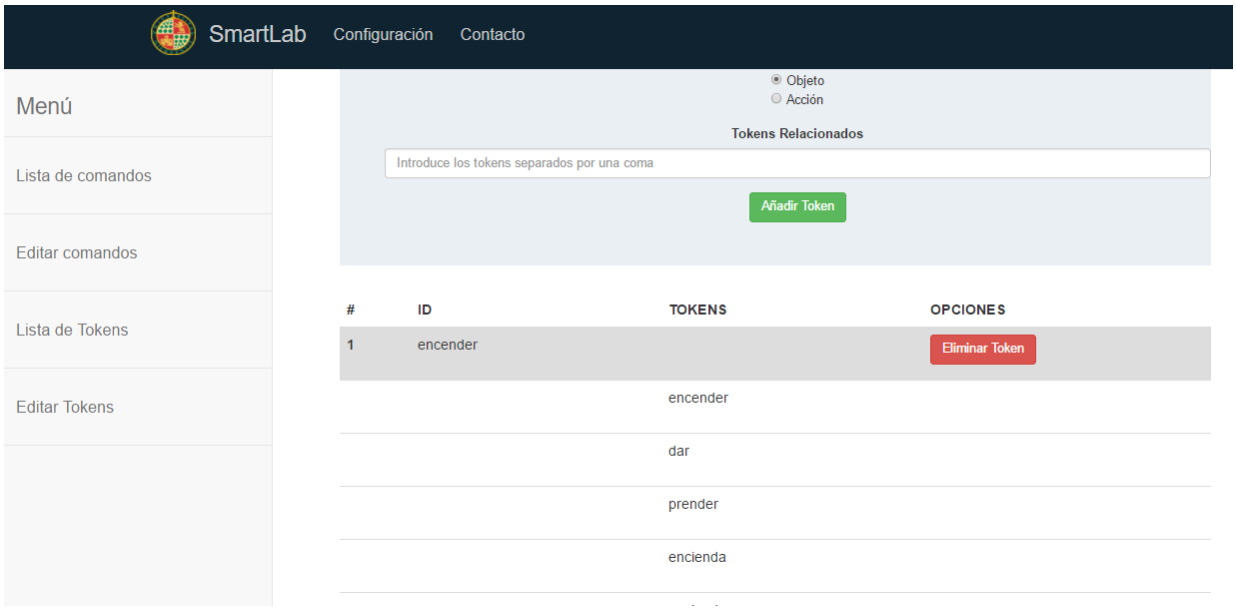

Ilustración 49: Editar Tokens 2

6. Bibliografía

- [1] Computación Ubicua www.um.es Disponible: http://www.um.es/aulasenior/saavedrafajardo/apuntes/doc/vivienda-inteligente.pdf

- [2] "Computación Ubicua". www.biko2.com Disponible: http://www.biko2.com/ubicomp/la-computacion-ubicua-y-por-que-nos-interesa/

- [3] "Procesamiento del Lenguaje Natural". www.vicomtech.org Disponible: http://www.vicomtech.org/t4/e11/procesamiento-del-lenguaje-natural

- [4] "Documentación sobre Java Speech" es.slideshare.net Disponible: https://es.slideshare.net/diegoerodriguezv/presentacion-rv

- [4] "Información sobre Patrón MVC". www.juntadeandalucia.es Disponible: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/122

- [5] ”Diseño de la Interfaz” www.lawebera.es Disponible: http://www.lawebera.es/diseno-web/el-diseno-de-la-interfaz.php#

- [6] “Estilo de la Interfaz” www.wix.com Disponible: https://es.wix.com/blog/2011/01/como-crear-una-guia-de-estilo/

- [7] ”Información sobre los Storyboard” www.antonionavas.info Disponible: http://www.antonionavas.info/blog/softwareproductmanagement/requerimientos-en-el-software-viii-storyboards/

- [8] ”Peticiones REST” asiermarques.com Disponible: http://asiermarques.com/2013/conceptos-sobre-apis-rest/

- [9] “Tokenización” urclmr.github.io Disponible: http://uclmr.github.io/stat-nlp-book-scala/01_tasks/00_tokenization.html

- [10] “Stopwords” googleseo.marketing Disponible: http://googleseo.marketing/seo-que-son-stop-words-palabras-vacias/

- [11] ”HyperText Markup Language (HTML5)” www.ibm.com Disponible: https://www.ibm.com/developerworks/ssa/web/library/wa-html5fundamentals/index.html

- [12] “Cascading Style Sheets (Css)” developer.mozilla.org Disponible: https://developer.mozilla.org/es/docs/Web/CSS

- [13] ”Bootstrap” www.negocioscaninos.com Disponible: http://www.negocioscaninos.com/que-es-bootstrap-bootstrap-framework-front-end/

- [14] ”Javascript” www.masadelante.com Disponible: http://www.masadelante.com/faqs/javascript

- [15] ”Javascript” www.gestiopolis.com Disponible: https://www.gestiopolis.com/definicion-javascript/

- [16] ”NetBeans” www.genbetadev.com Disponble: https://www.genbetadev.com/herramientas/netbeans-1

- [17] “Estadisticas INE” www.ine.es Disponible: http://www.ine.es/prensa/np980.pdf

- [18] ”Estadisticas INE” www.ine.es Disponible: https://www.fayerwayer.com/2010/10/espana-avances-en-viviendas-inteligentes-para-mayores/