



TEODORO PERSONAL ASSISTANT

Especificación de Requisitos

Equipo F

Álvaro Arellano Pardillo 14016
José María Barón Rufete 15031
Carlos Filiu Martínez 15149

Ingeniería del Software
2º MII 2020/2021

Índice

Introducción	1
Propósito	1
Audiencia.....	1
Alcance del Producto	1
Referencias.....	1
Descripción general del producto	1
Perspectiva de Producto	1
Funciones del Producto.....	2
Clases de Usuario y características	2
Entorno operacional	3
Limitaciones de diseño.....	3
Documentación de Usuario	3
Requisitos de interfaz externa	3
Interfaz de usuario	3
Interfaces hardware.....	3
Interfaces software	4
Interfaces de Comunicación.....	4
Características del Sistema	4
Características de <i>Engine</i>	7
<i>Descripción y Prioridad</i>	7
<i>Secuencias Estímulo / Respuesta</i>	7
<i>Requisitos funcionales</i>	11
Características de <i>System</i>	12
<i>Descripción y Prioridad</i>	12
<i>Secuencias Estímulo / Respuesta</i>	12
<i>Requisitos funcionales</i>	14
Características de <i>Applications</i>	14
<i>Descripción y Prioridad</i>	14

<i>Secuencias Estímulo / Respuesta</i>	14
<i>Requisitos funcionales</i>	20
Características de <i>Calendar</i>	21
<i>Descripción y Prioridad</i>	21
<i>Secuencias Estímulo / Respuesta</i>	21
<i>Requisitos funcionales</i>	25
Características de Teodoro	26
<i>Descripción y Prioridad</i>	26
<i>Secuencias Estímulo / Respuesta</i>	26
<i>Requisitos funcionales</i>	29
Requisitos No Funcionales	29
Requisitos de Desempeño-Rendimiento	29
Requisitos de Seguridad (producto) XXX Tema de Correo y usuario-.....	30
Atributos de Calidad.....	30
Apéndice A: Glosario	32
Apéndice B: Lista “To Be Determined”	32

Historial de Revisiones

Nombre	Fecha	Razón de los Cambios	Versión
Inicial	15/08/2021	Inicio del trabajo	1.0
	01/09/2021	Establecimiento de trabajo Ingeniería del Software	1.1
	03/10/2021	Presentación de la idea	1.2
	13/10/2021	Incorporación de Calendar	1.3
POO	02/11/2021	Reestructuración del código para POO	2.0
	05/11/2021	Creación de Base de Conocimiento	2.1
	09/11/2021	Incorporación de GUI	2.2
	14/11/2021	Establecimiento de Requisitos	2.3

Introducción

Propósito

Teodoro Personal Assistant, de aquí en adelante Teodoro, es el producto para el cual se define este documento. Esta documentación persigue describir de manera general el propósito y motivación de este sistema, además de ahondar en sus características y funciones principales, modos de uso, interacciones con sus diferentes subsistemas y con el usuario y la definición y detalle de los requisitos que Teodoro debe cumplir para poder validar su correcto funcionamiento.

Audiencia

Este documento se dirige a los desarrolladores de este producto, así como a todas aquellas partes interesadas en el mismo, como puedan ser el director del proyecto, testadores, usuarios o posibles clientes.

Alcance del Producto

Con Teodoro, el equipo desarrollador pretende poder ofrecer a la comunidad de usuarios de PC de una herramienta software para la ayuda de su día a día. Como se irá refiriendo a lo largo de este documento, el sistema en sus primeras versiones solo se podrá utilizar en el sistema operativo (S.O.) en el que habrá sido desarrollado, sin embargo, el objetivo es ir mejorando el sistema, haciéndolo más compatible con otros S.O., ampliando sus funcionalidades y bases de datos, implementando mejores interfaces de usuario para facilitar su uso, etc. Por tanto, este producto se enmarca como uno de tantos proyectos subido a repositorios de *Github* de acceso público, para, poco a poco, ir creciendo y mejorando para poder obtener un futuro producto final comercializable con empresas de software e informática para incorporarlo en sus propios S.O.

Referencias

Todos los documentos de instrucciones o guías para el usuario se encontrarán en el repositorio de acceso público de GitHub en el que se establece el presente proyecto: <https://github.com/filiu97/Teodoro>

Descripción general del producto

Perspectiva de Producto

La demanda de los asistentes personales ha ido en aumento en los últimos años, especialmente con la irrupción de los *smartphones* en el mercado global (*Google Assistant*, *Siri*...), pero también se han incorporado en la casa de muchas personas a través de sistemas de altavoces inteligentes y dispositivos compatibles con dicha tecnología (*Google Home*, *Alexa*, *Apple HomePod*...). Sin embargo, en el mundo de los

ordenadores, es un mercado que no se ha explotado aún, pues en Windows se tiene a *Cortana* y en macOS a *Siri*, sin embargo, son muchos los usuarios que no utilizan esta tecnología, bien porque no la conocen o porque no les ofrece una ayuda real en el uso de su ordenador en el día a día. Así, **Teodoro** surge de la idea de poder disponer de un asistente personal y altamente personalizado al alcance de cualquier usuario de ordenador, un asistente pensado para la ayuda en las tareas más comunes cuando se utiliza un PC, como el uso de *Spotify* para escuchar música, realizar búsquedas web con el buscador de *Google*, consultar el tiempo o poner una alarma.

Funciones del Producto

Las funciones o funcionalidades del sistema o producto son las que se presentan a continuación:

- **Escucha y reconocimiento de voz** (no visible).
- **Emisión de audio por el asistente** (no visible).
- **Proceso de repetición de escucha de petición no entendida** (no visible).
- **Creación de interfaces gráficas de usuario (GUI)** (no visible).
- **Control del equipo (apagado, reinicio y suspensión).**
- **Control de Spotify.**
- **Búsquedas web en Google, Wikipedia y YouTube.**
- **Creación de alarmas.**
- **Consulta de tiempo meteorológico.**
- **Conexión con Google Calendar (creación, mostrado y modificación de eventos).**
- **Interacción con el asistente (saludo, ánimos, consultas de hora o día actual, listado de nombres del asistente y despedida).**

Clases de Usuario y características

Las clases de usuario contempladas por las primeras versiones son de personas con cierto nivel de uso de sistemas Linux y uso de repositorios GitHub. En cuanto a conocimientos de programación, usuarios privilegiados son aquellos que dominen el lenguaje de programación Python y que, por tanto, pueda readaptar su sistema a sus propias necesidades. También el sistema está pensado para usuarios sin previa experiencia en programación, ya que toda la modificación o mejora se realiza a través de la base de datos de la que se nutre el sistema, que, como se detallará más adelante, se tratará en las primeras versiones de archivos de texto fácilmente modificables.

Para versiones más avanzadas, la accesibilidad del código será cada vez menor, para asegurar la consistencia y robustez del mismo, para acabar con un producto comercializable en forma únicamente de archivo ejecutable.

Entorno operacional

El sistema se ha programado enteramente en el lenguaje de programación Python en la distribución Ubuntu 18.04 de Linux. Eso ha sido así por la facilidad y versatilidad de este S.O. para la creación de productos de software. La primera versión operativa del producto estará, por tanto, destinada a este sistema operativo, para, posteriormente, ir compatibilizando su uso para el resto de las distribuciones de Linux más importantes, y, finalmente, conseguir una portabilidad con otros S.O. cómo Windows y macOS.

Limitaciones de diseño

En cuando a limitaciones que el producto puede tener, lo primero es citar lo comentado en el punto anterior, la restricción del uso de la primera versión en el sistema compatible en el que se ha desarrollado. Por otro lado, también citar la limitación del sistema en cuanto a las bases de datos de las que se nutre, ya que, para esta primera versión, se ha planteado un sistema de conocimiento sencillo y apto para poder modificarse y ampliarse por un usuario sin conocimientos de programación, pero este sistema es escaso y simple, por lo que para posteriores versiones se plantearía la ampliación y mejora de esta base de datos, utilizando herramientas más profesionales como *SQL*, *TinyDB* o *MongoDB*.

Documentación de Usuario

Este producto estará disponible a través de un repositorio online de *Github*, por lo que la documentación necesaria, como un README, un documento de ejemplos, una guía de usuario y las instrucciones de instalación y requerimientos estarán disponibles en el mismo.

Requisitos de interfaz externa

Interfaz de usuario


El producto (al menos en sus primera versiones) no dispondrá de una interfaz gráfica convencional, estática en el tiempo de ejecución, ya que el objetivo del producto es que interfiera lo mínimo necesario con el uso del PC por parte del usuario, por lo que su interacción con este será preferentemente la voz. Sin embargo, existirán interfaces gráficas de usuario (GUI) **específicas** para la visualización de ciertas funcionalidades o procesos del programa, además de otra interfaces que se requieran por la necesidad de una mayor interacción con el usuario que por voz resultaría tediosa. Así, estas interfaces serán todas relativamente simples, de un estilo similar y creadas a partir de la librería de Python “Tkinter”.




Interfaces hardware

El usuario debe disponer de un ordenador en el S.O. especificado en los requisitos del entorno operacional. Además, es imprescindible el uso de periféricos de un PC convencional (pantalla, teclado, ratón y, especialmente, micrófono y sistema de audio).

Interfaces software

La interfaz software principal será la  escucha, reconocimiento y emisión de audio. Los mensajes se recogen a través del micrófono, son procesados y se realiza (o no) la funcionalidad pertinente. Además, se contará, como se ha citado anteriormente, de GUIs específicas para determinadas funcionalidades, por lo que también habría una interacción visual con el usuario.

Como se comentará más adelante, existirá una “**Base de Conocimiento**” (BdC) sencilla, basada en archivos de texto, para establecer toda la información que el asistente debe conocer de antemano de cualquier ejecución del mismo, lo que representará la base de datos de Teodoro. 

Por último, existirán numerosas librerías de Python para poder ejecutar cada una de las funcionalidades previstas por el sistema, que se añadirán a un documento de “*Requirements*”, para que el usuario conozca e instale cada una de estas librerías.

Interfaces de Comunicación

El idioma de la comunicación, en estas primera versiones, será el Castellano. Además, se requerirán códigos de acceso para la funcionalidad de *Calendar*, como se comentará más adelante. También se necesitará de conexión a Internet para aquellas funcionalidades que así lo requieran (búsquedas en Google, Wikipedia, YouTube, consulta meteorológica, etc.)

Características del Sistema

El sistema de asistente por voz Teodoro busca proporcionar al usuario las funcionalidades propias de un asistente convencional incorporando el control de los eventos de las aplicaciones Google Calendar y Trello dentro del sistema operativo Linux. Es importante resaltar que el desarrollo de la aplicación se está realizando en el entorno de programación Python orientado a objetos, donde las cinco clases que componen el software se observan en la Ilustración 1.

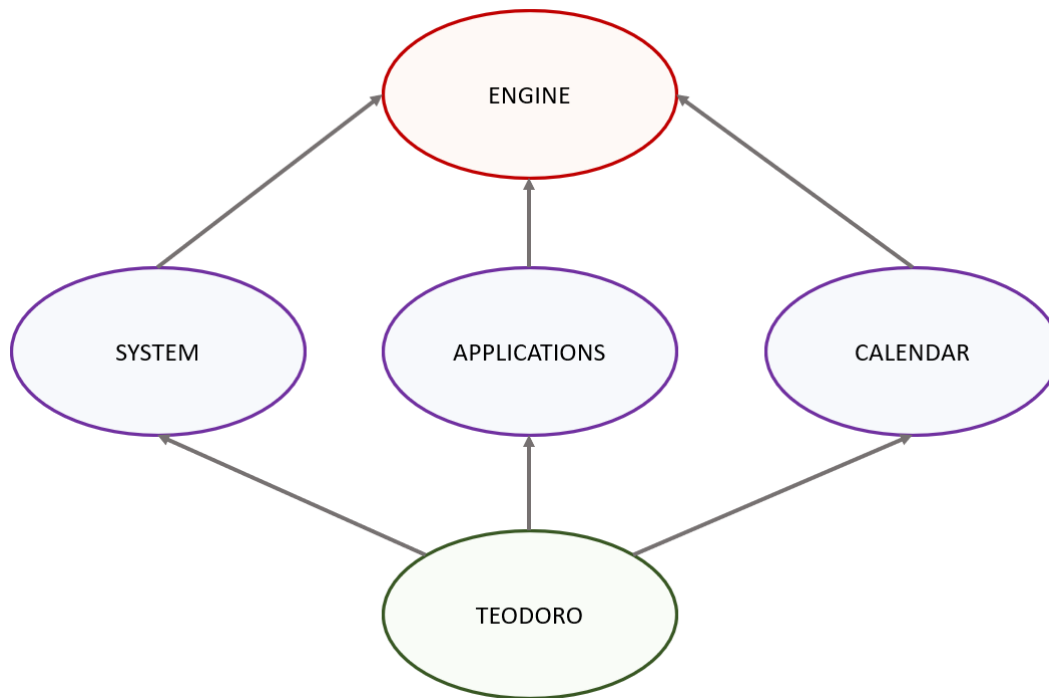


ILUSTRACIÓN 1: DEPENDENCIAS ENTRE CLASES EN TEODORO.

Observando la Ilustración 1, la clase Teodoro, como clase base, hereda todos los atributos y métodos asociados a las clases *System*, *Applications* y *Calendar*, que serán llamadas clases intermedias, que ejercen de contenedoras de aquellas funcionalidades que el usuario “ve” y que tienen algo de complejidad. A su vez dichas clases heredan las características y métodos de la clase *Engine*, que ejerce de “clase madre” o “superclase”, la clase que contendrá todo lo relacionado con la escucha, reconocimiento de audio y habla del asistente, además de ser la encargada de realizar la interfaz visual de usuario para algunas funcionalidad que así lo requieran. Por tanto, Teodoro estará compuesto por todas la funcionalidades, atributos y métodos asociadas a las propiedades de las anteriores clases, las cuales son expuestas a continuación:

- **Engine:** Compuesta por todas las funcionalidades que dotan al sistema de la capacidad de interactuar con los periféricos de audio, tanto micrófono como altavoz, incorporando a su vez la lógica del tono de voz de Teodoro, la interpretación del mensaje de voz transmitido por el usuario y la lógica relacionada con el proceso de “Repeat”, en el que el sistema no logra identificar la acción que desea el usuario o el mensaje transmitido por el usuario.

Además, la clase *Engine* incorpora un método especial llamado “GUI” que albergará la lógica para la creación de las interfaces de usuario específicas para aquellas funcionalidades del sistema que requieran de una visualización textual o de una interacción con el usuario más allá de la propia voz.

- **System:** Incorpora a *Engine* los atributos y métodos necesarios para interactuar con el sistema operativo y realizar las acciones apagar, reiniciar y suspender.
- **Applications:** Incorpora a *Engine* los atributos y métodos necesarios para interactuar con Spotify, el navegador, crear alarmas y consultar el tiempo, y así realizar las siguientes acciones:

- Pausar la canción, reanudar la canción, pasar a la siguiente canción, volver a la anterior canción y parar la reproducción
- Realizar búsquedas en Google, Wikipedia y YouTube.
- Poner una alarma y consultar el tiempo de una zona geográfica.
- **Calendar:** Incorpora a *Engine* los atributos y métodos necesarios para interactuar con Google Calendar y Trello. Es importante resaltar que al dotar al sistema de esta funcionalidad incorpora información sensible del usuario que permite la autenticación de este y el uso de los servicios de Google Calendar y Trello. Con dicha conectividad con las APIs online de Trello y Google Calendar, es capaz de realizar las siguientes acciones:
 - Mostrar los eventos pendientes del usuario para diferentes rangos de tiempo como se expondrá más adelante en el documento.
 - Modificar eventos ya existentes en Google Calendar.
 - Crear nuevos eventos.

A su vez, la clase **Teodoro** que hereda todos los métodos y los atributos del resto de clases, incorpora funcionalidades relacionadas con la interfaz Usuario-Teodoro, dotando al sistema de más naturalidad al ser capaz de interactuar con el usuario frente a preguntas casuales relacionadas con el “estado de ánimo de Teodoro” y un surtido acciones de interacción como informar del día actual, la hora e, incluso, animar al usuario. Además, incorporará del método “*getAction*”, el cual ejecuta la lógica de ejecución de la funcionalidad requerida en función de la petición del usuario. Por otro lado, el archivo contenedor de esta clase se encontrará el *main* del programa.

A parte de esto, el asistente personal dispondrá de lo que se denominado “**Base de Conocimiento**” (BdC), que, para la primera versión consistirá en simples archivos de texto que albergarán diferentes conocimientos previos con los que el asistente debe partir. Entre estos archivos se encontrarán archivos de definición de los posibles nombres a los que el asistente debe ser capaz de responder; de definición de los comandos o palabras u oraciones clave para el desencadenamiento de las funcionalidades del sistema; diccionarios al estilo Python (clave-valor) para ciertos conocimientos que debe incorporar el asistente; etc... Esta BdC se incorporará al asistente al principio de la ejecución del mismo, en la que cada clase leerá estos archivos de texto o recibirá de la clase que la inicialice estos parámetros que guardará como atributos para su completo acceso durante el progreso del sistema.

Evalutando el sistema Teodoro en su conjunto, únicamente se puede exponer un modelo de operación, en el que el usuario interactúa con el sistema Teodoro directamente dirigiéndose al mismo por nota de voz bajo la siguiente cadena de eventos:

- El usuario nombra directamente al sistema Teodoro bajo la palabra clave “Teodoro” o “Teo” (o cualquier otro nombre puesto por el usuario en la base de nombres externa).
- El sistema Teodoro responde de forma inmediata al *input* realizado por el usuario respondiendo con la oración “¿Sí?” y quedándose a la espera de que el usuario transmita su oración. En el supuesto caso de que el usuario no transmita ninguna instrucción en el periodo de tiempo 10 segundos, Teodoro volverá a su estado inicial.

- El usuario transmite mediante una oración la petición y Teodoro la identifica correctamente.

En base a la petición realizada por el usuario, el sistema Teodoro podrá responder de forma diferente en base a dos escenarios:

- Teodoro es capaz de interpretar e identificar correctamente una serie de palabras y oraciones claves en el mensaje transmitido por el usuario, lo que le permite responder adecuadamente frente al estímulo y generar la respuesta.
- Teodoro no identifica adecuadamente ninguna palabra u oración clave dentro del mensaje transmitido por el usuario y, a continuación, se ejecuta el proceso “Repeat” por el cual se vuelve a preguntar al usuario por el mensaje. Si este proceso no resulta exitoso, se vuelve al estado inicial de escucha activa.

Esta lógica se verá más en detalle en el siguiente apartado, reservado a detallar la clase *Engine*.

Características de *Engine*

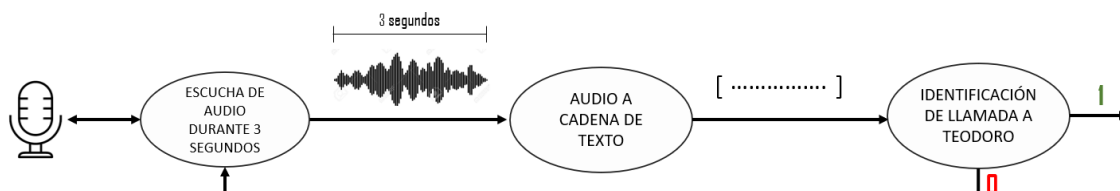
Descripción y Prioridad

La prioridad de *Engine* dentro del software de Teodoro es la clase de la que heredan todos los demás subsistemas, por tanto, se puede decir que su prioridad es muy elevada. Como se ha mencionado, es la clase encargada de realizar todos los procesos relacionados con la escucha, reconocimiento y habla del asistente personal Teodoro. Además, incorpora la funcionalidad de creador de las interfaces de usuario específicas para aquellas funcionalidades que lo requieran.

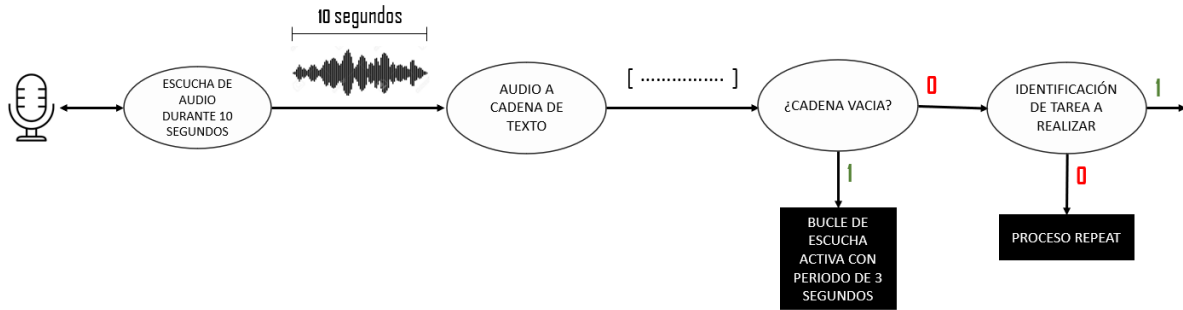
Secuencias Estímulo / Respuesta

Dentro de las funcionalidades que aporta esta clase al sistema Teodoro, pueden resaltar tres procesos principalmente:

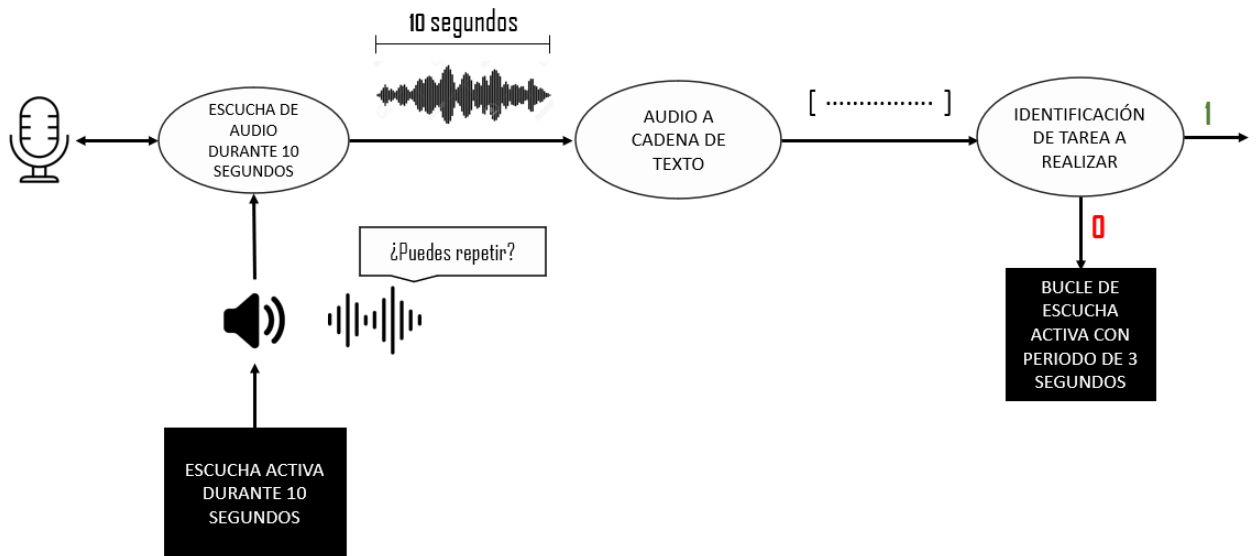
- Bucle de escucha activa con periodo de 3 segundos:



- Escucha activa durante 10 segundos:

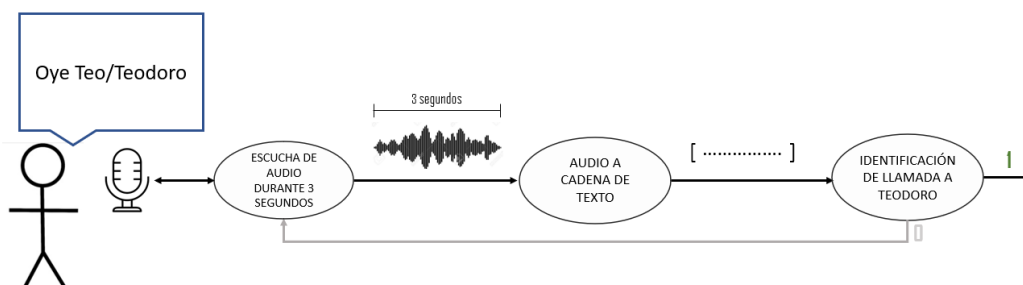


- Proceso Repeat:

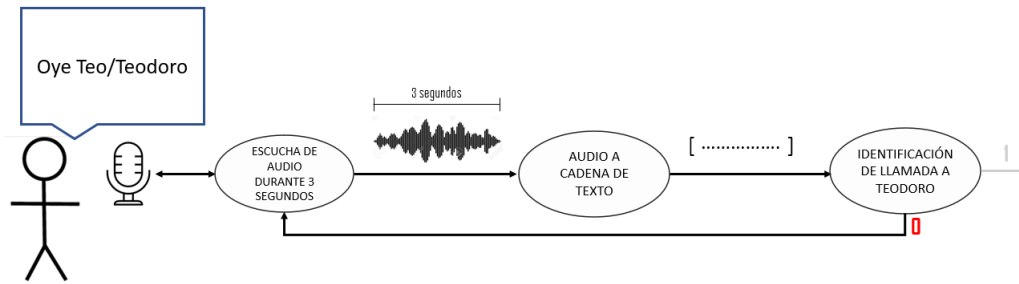


Respecto a los casos de secuencia estímulo respuesta se puede resaltar los siguientes casos:

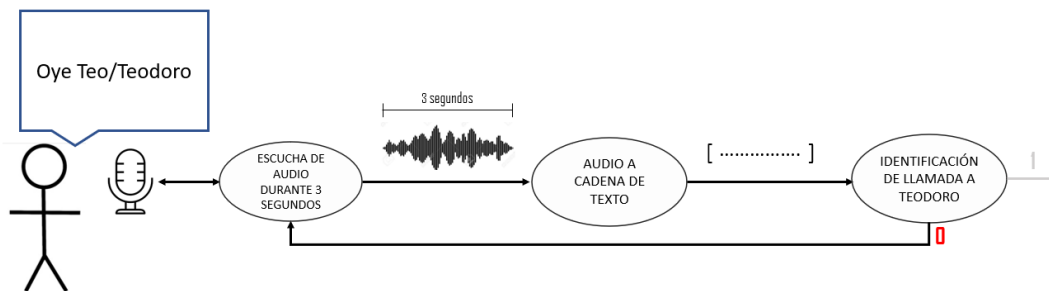
- Usuario nombra correctamente al sistema:



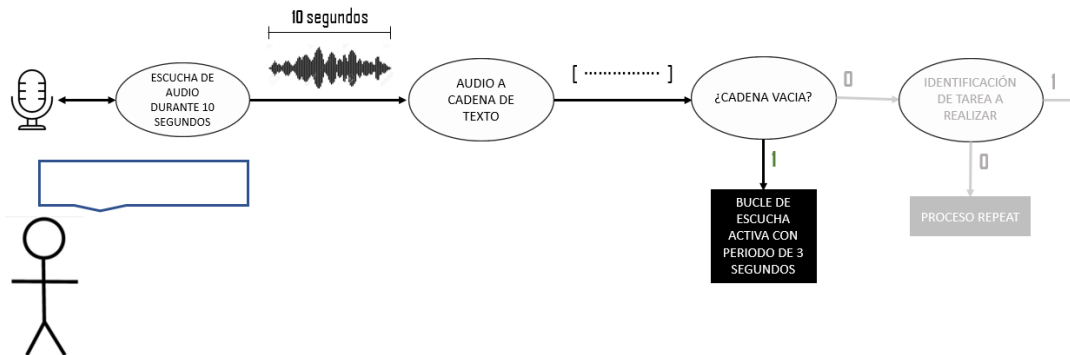
- Usuario nombra correctamente al sistema, pero el sistema no identifica bien el audio:



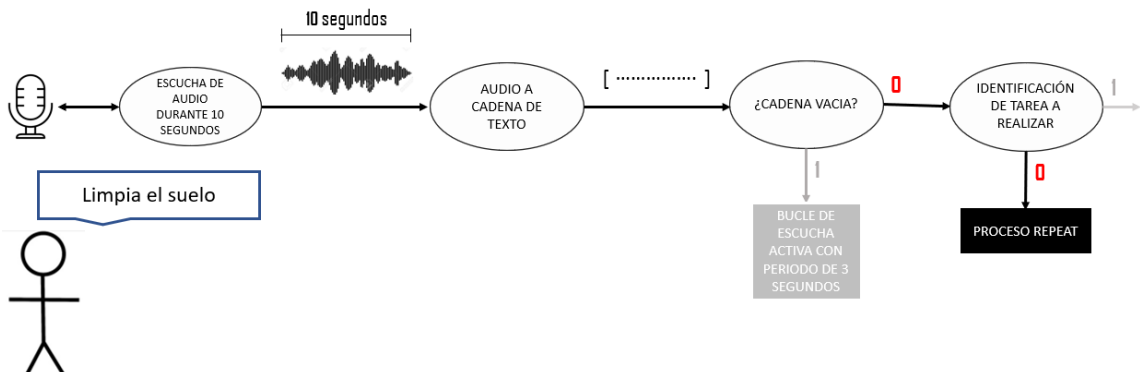
- Usuario **NO** nombra correctamente al sistema:



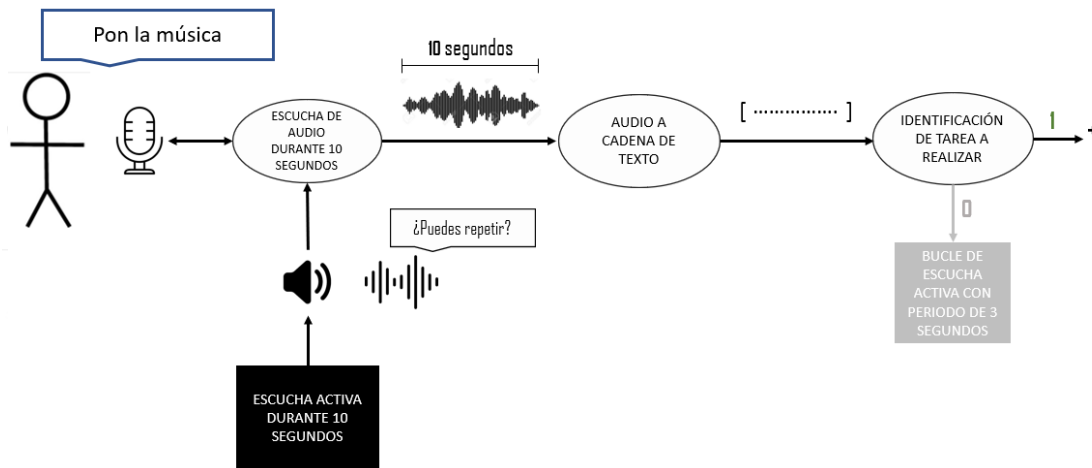
- Usuario nombra correctamente al sistema pero luego no dice nada:



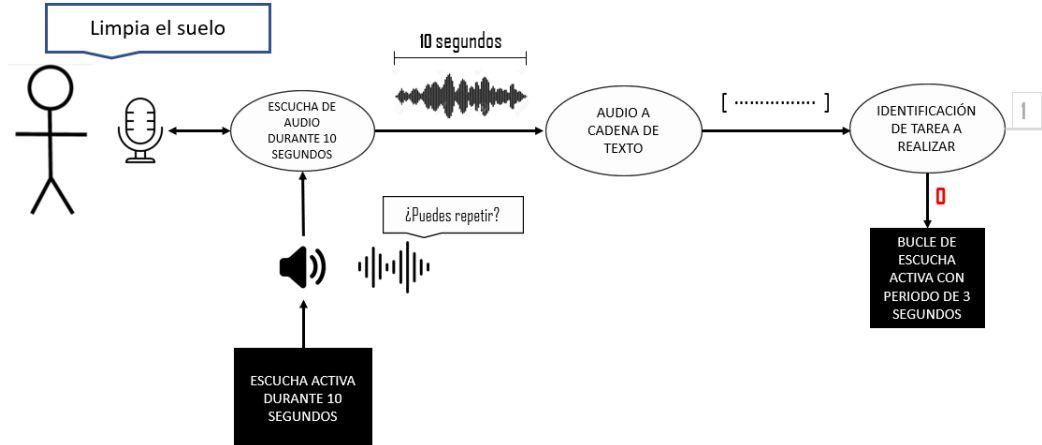
- Usuario nombra correctamente al sistema y transmite una tarea pero Teodoro **NO** reconoce dicha Tarea:



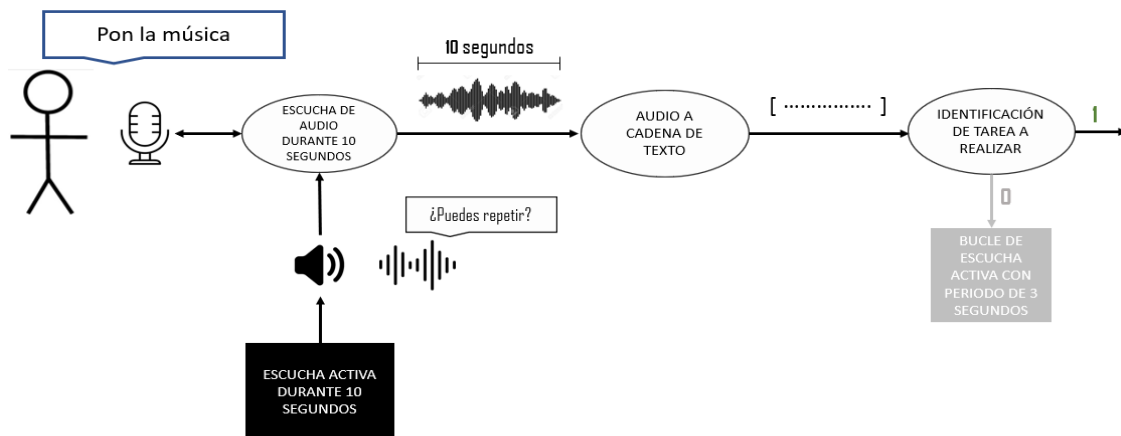
- Usuario nombra correctamente al sistema y transmite una tarea que Teodoro es capaz de identificar:



- Usuario nombra correctamente al sistema, transmite una tarea pero Teodoro no es capaz de identificarla y al pedirle que la repita, vuelve a **NO** identificarla:






- Usuario nombra al sistema, transmite una tarea pero Teodoro no es capaz de identificarla y al pedirle que la repita, finalmente logra identificarla:



En cuanto a la funcionalidad de creación de **interfaces de usuario**, como se ha mencionado anteriormente en los Requisitos de Interfaz Externa, Teodoro no dispondrá de una interfaz gráfica al uso, sino que se habla de GUIs específicas para ciertas funcionalidades y visualizaciones de resultados. La clase *Engine* contendrá el método por el que se cree las diferentes GUIs, con una lógica similar a un “switch-case”, en el que el primer parámetro de la llamada establezca el tipo de interfaz gráfica que se requiere. Este método podrá ser llamado desde cualquier punto del programa, desde cualquier clase en la que se necesite la creación de una GUI. Además, estas interfaces podrán ser bloqueantes o no del proceso según la funcionalidad que acompañe su uso. A continuación, se lista las principales funcionalidades y procesos que requerirán de una GUI específica:

- **Status.** Se muestra por pantalla una ventana con el estado actual de Teodoro. Se utilizará para el bucle de escucha activa, para indicar al usuario que Teodoro está escuchando y procesando su petición. Es una GUI no bloqueante, se destruirá automáticamente con la ejecución o la no ejecución de la funcionalidad requerida.
- **Show.** Se muestra por pantalla una ventana con la respuesta o resultado de Teodoro. Esta GUI será muy utilizada para todas aquellas funcionalidad sencillas cuya salida sea una cadena de texto con información, como el día actual, la hora, etc. Será una GUI bloqueante, es decir, el usuario debe cerrarla para continuar con la ejecución, indicando que ya ha sido informado.
- **Image.** Se muestra por pantalla un imagen. Esta GUI estará destinada a aquellas funcionalidades cuya salida pueda visualizarse en formato imagen. Será una GUI bloqueante, es decir, el usuario debe cerrarla para continuar con la ejecución, indicando que ya ha sido informado.
- **CountDown.** Se muestra por pantalla información relativa a la alarma puesta por el usuario. Una GUI no bloqueante que se vaya actualizando con el tiempo restante de una alarma. Es una GUI no bloqueante, se destruirá automáticamente al terminar al sonar la alarma.
- **GetCalendar.** Se muestra por pantalla una información extensa. Esta GUI estará diseñada para la funcionalidad de mostrar los eventos del calendario del usuario, ya que pueden ser muchos dependiendo del periodo temporal especificado por este. Por ello, será una ventana más grande que las demás y contará con un *scroll bar* para poder desplazarse cómodamente por la ventana. Será una GUI bloqueante, es decir, el usuario debe cerrarla para continuar con la ejecución, indicando que ya ha sido informado.
- **SetCalendar.** Se muestra por pantalla una ventana que pregunta al usuario si desea establecer un descripción y una localización del evento que se quiere crear, en la funcionalidad de crear evento de la clase *Calendar*. Será una GUI bloqueante, es decir, el usuario debe cerrarla para continuar con la ejecución, indicando los parámetros de descripción y localización del evento, que puede dejar como campos vacíos para indicar que no desea añadirlos a dicho evento.

Requisitos funcionales

- **ENG-1.** Teodoro debe realizar la conexión con periféricos de entrada y salida de audio y hacer uso de ellos. Para realizar la escucha al usuario, el reconocimiento y el habla del asistente. 
- **ENG-2.** Teodoro debe incorporar la lógica del tono de voz de Teodoro. Se deben configurar las características de la voz del asistente. 
- **ENG-3.** Teodoro debe tener capacidad de creación de interfaces de usuario. Dependiendo de la funcionalidad que el usuario active se mostrará por pantalla las diferentes GUI. 

- **ENG-4.** Teodoro debe configurar la escucha activa para llamada a Teodoro. Realización de un bucle de escucha activa de 3 segundos.
- **ENG-5.** Teodoro debe realizar la identificación de llamada a Teodoro. Se debe reconocer cuando el usuario llama al asistente mediante las palabras claves de nombres especificados en la BdC.
- **ENG-6.** Teodoro debe configurar la escucha activa para la petición a Teodoro. Realización de un bucle de escucha activa de 10 segundos
- **ENG-7.** Teodoro debe poder activar el proceso *Repeat*. Cuando el usuario hace una petición que no es identificada por el sistema, el sistema debe responder con un “¿Puedes repetir?” y volver al bucle de escucha activa de 10 segundos hasta que consiga identificar una palabra clave y realizar una funcionalidad.

Características de System

Descripción y Prioridad

El subsistema o subclase *System* es una de las clases más simples de las que se compone el sistema Teodoro. En ella se encuentran aquellas funcionalidad del asistente íntimamente relacionadas con la interacción con el equipo. Por el momento se ha prevista la existencia de tres acciones: apagar, suspender y reiniciar el equipo. Su prioridad es baja, por ser sus métodos simples en comparación con otras funcionalidades.

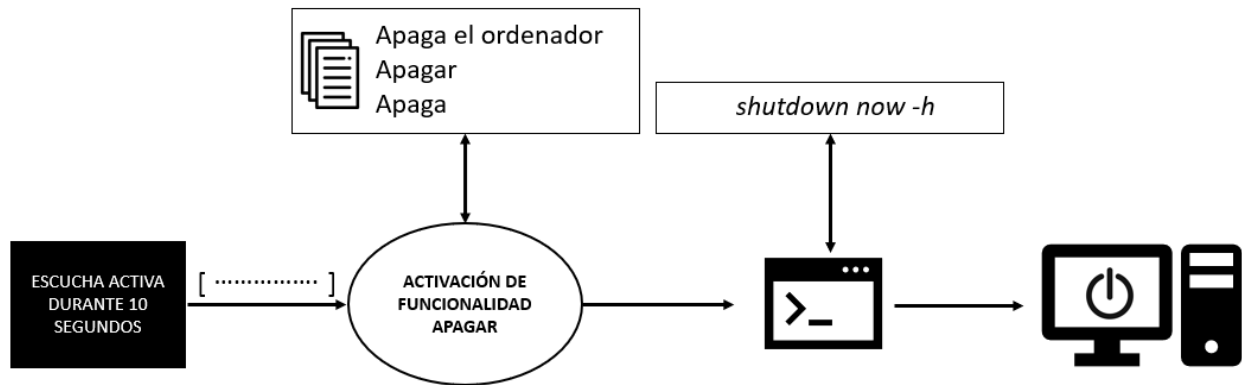
Secuencias Estímulo / Respuesta

1. Apagado del ordenador

Descripción: Teodoro apaga el equipo.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario quiere realizar la acción de apagar el equipo por lo que formula una frase con las palabras clave "apaga el ordenador", "apagado" o "apagar".
3. Teodoro responde por voz con frases como: "En seguida", "ahora mismo" y similares.
4. Teodoro ejecuta "*shutdown now -h*" en el terminal.
5. El equipo se apaga.

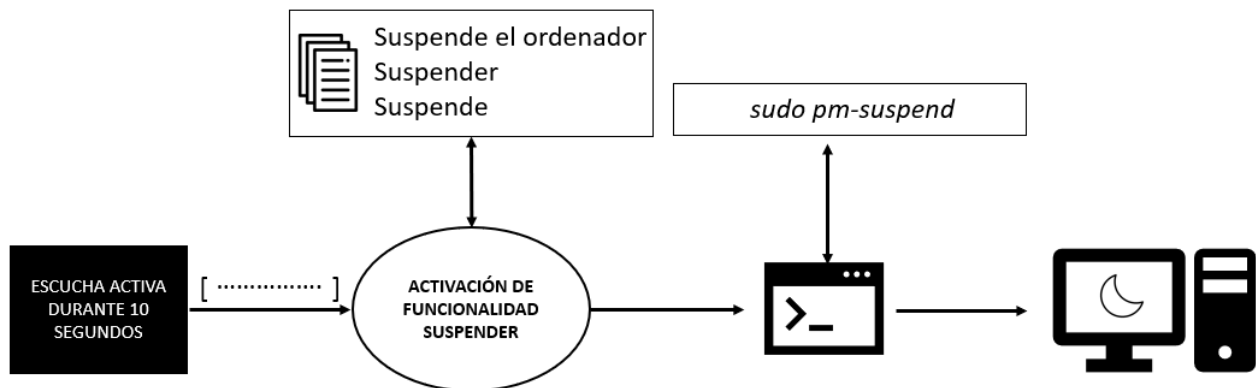


2. Suspensión del ordenador

Descripción: Teodoro suspende el equipo.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario quiere realizar la acción de suspender el equipo por lo que formula una frase con la palabra clave "suspende el ordenador", "suspensión" o "suspender".
3. Teodoro responde por voz con frases como: "En seguida", "ahora mismo" y similares.
4. Teodoro ejecuta "*sudo pm-suspend*" en el terminal.
5. El equipo se suspende.



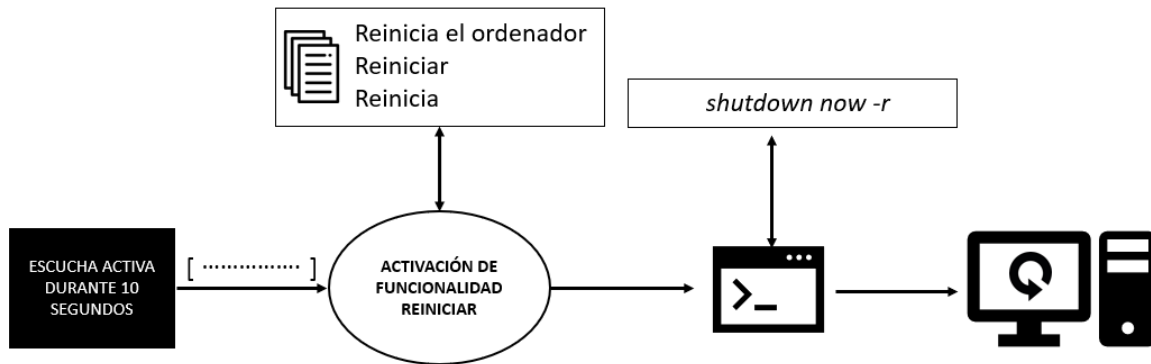
3. Reinicio del ordenador

Descripción: Teodoro reinicia el equipo.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario quiere realizar la acción de reiniciar el equipo por lo que formula una frase con la palabra clave "reinicia el ordenador", "reinicio" o "reiniciar".

3. Teodoro responde por voz con frases como: "En seguida", "ahora mismo" y similares.
4. Teodoro ejecuta "*shutdown now -r*" en el terminal.
5. El equipo se suspende.



Requisitos funcionales

- **SYS-1.** Teodoro ha de tener acceso privilegiado a funciones como editar el terminal.
- **SYS-2.** Apagado del equipo en menos de 10 segundos. Tras realizarse la secuencia de apagado y tras la ejecución del comando, el equipo debe apagarse en menos de 10 segundos.
- **SYS-3.** Suspensión del equipo en menos de 10 segundos. Tras realizarse la secuencia de suspensión y tras la ejecución del comando, el equipo debe suspenderse en menos de 10 segundos.
- **SYS-4.** Reinicio del equipo en menos de 10 segundos. Tras realizarse la secuencia de suspensión y tras la ejecución del comando, el equipo debe reiniciarse en menos de 10 segundos.

Características de Applications

Descripción y Prioridad

La clase *Applications* aúna todas aquellas funcionalidades de Teodoro que no son control del ordenador (*System*) ni relacionadas con el calendario del usuario (*Calendar*). Por tener una gran cantidad de acciones muy diversas e íntimamente relacionadas con el propósito del proyecto (ayuda al usuario a realizar acciones cotidianas en su PC), su prioridad se puede decir que moderada o elevada.

Secuencias Estímulo / Respuesta

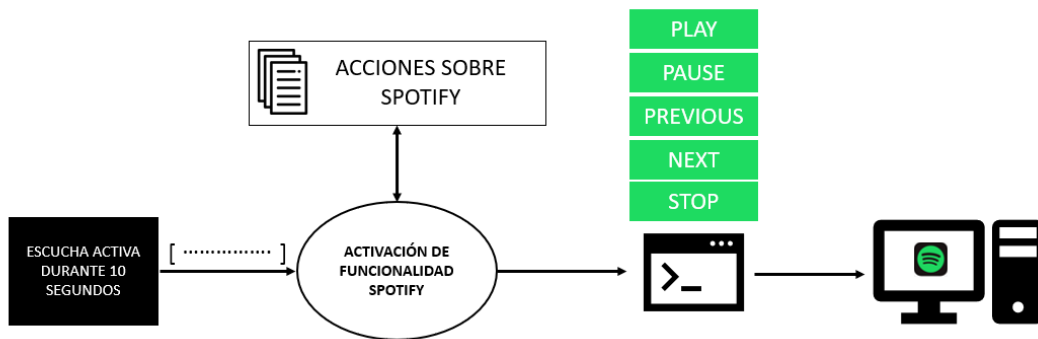
1. Conexión con Spotify.

Descripción: Teodoro conecta con Spotify para poder usar sus funciones básicas. Estas son:

- “Play”. Reanuda la reproducción.
- “Pause”. Pausa la reproducción.
- “Next”. Pasa a la siguiente canción.
- “Previous”. Vuelve a la canción anterior.
- “Stop”. Para la reproducción.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario formulará frases con palabras y oraciones clave como "pon la música", "para la música", "pasa de canción"... Cada una de estas acciones llevará asociada una serie de palabras clave y un comando que se ejecutará en la terminal.
3. Teodoro ejecutará un comando que realice la acción con Spotify.
4. Spotify reaccionará según dicho comando.

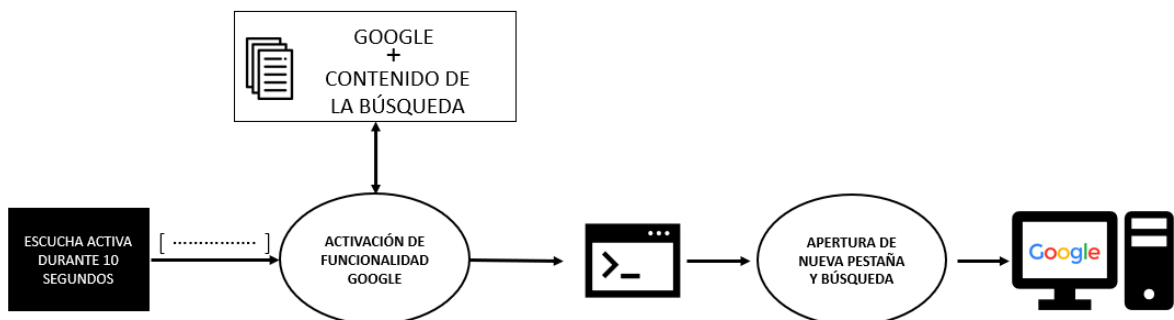


2. Conexión con el buscador de Google.

Descripción: Teodoro conecta con el buscador de Google para realizar una búsqueda web.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario formulará la palabra clave ("google") y el contenido de su búsqueda.
3. Teodoro abrirá una pestaña en el navegador y realizará la búsqueda en "www.google.es".

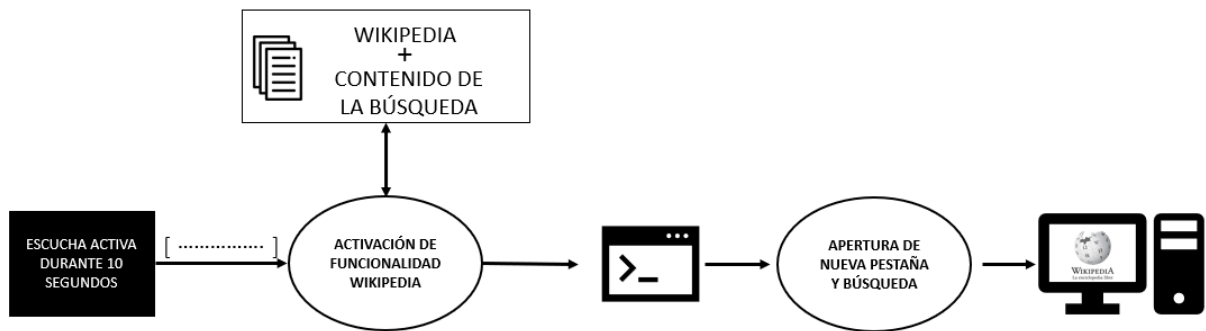


3. Conexión con la Wikipedia.

Descripción: Teodoro conecta con la Wikipedia para realizar una búsqueda web.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario formulará la palabra clave (“wikipedia”) y el contenido de su búsqueda.
3. Teodoro abrirá una pestaña en el navegador y realizará la búsqueda en “www.wikipedia.es”.
4. Si existe más de una página de Wikipedia para la misma búsqueda, el sistema debe ser capaz de hacer que el usuario pueda decidir por voz o a través de una GUI cual es la opción que le interesa abrir. (**TbD**)

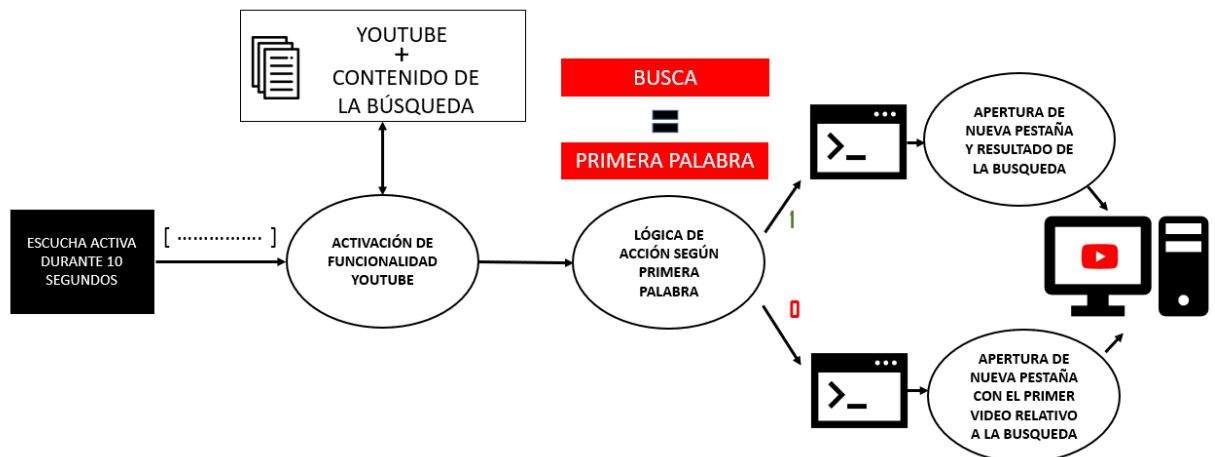


4. Conexión con YouTube.

Descripción: Teodoro conecta con YouTube para realizar una búsqueda web.

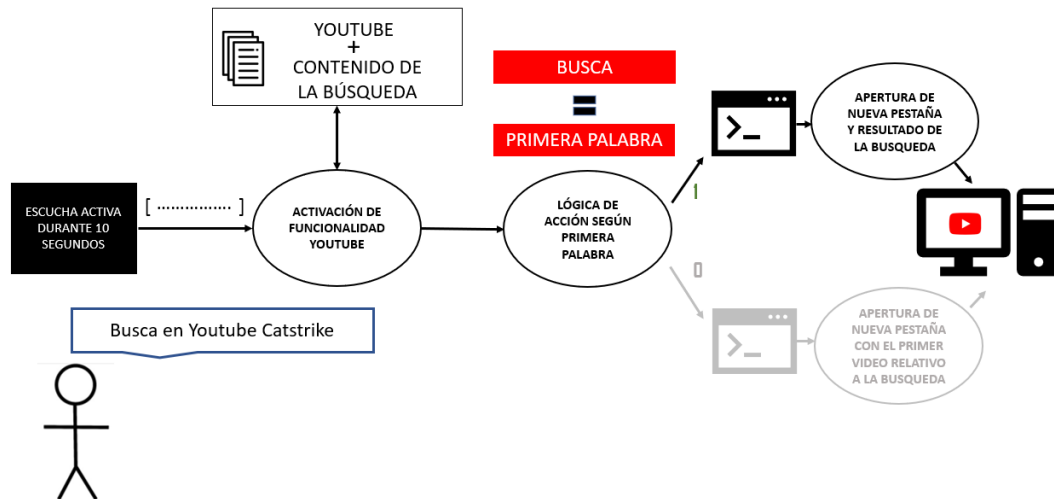
Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario formulará la palabra clave (“youtube”) y el contenido de su búsqueda.
3. Teodoro realizará una de las dos acciones siguientes en función de la primera palabra de la petición del usuario:
 - Primera palabra es “busca”. Se abrirá una pestaña con el resultado de la búsqueda en YouTube.
 - Primera palabra no es “busca”, podría ser “pon” o “reproduce”. Se abrirá una pestaña con el primer resultado (video) de la búsqueda deseada.

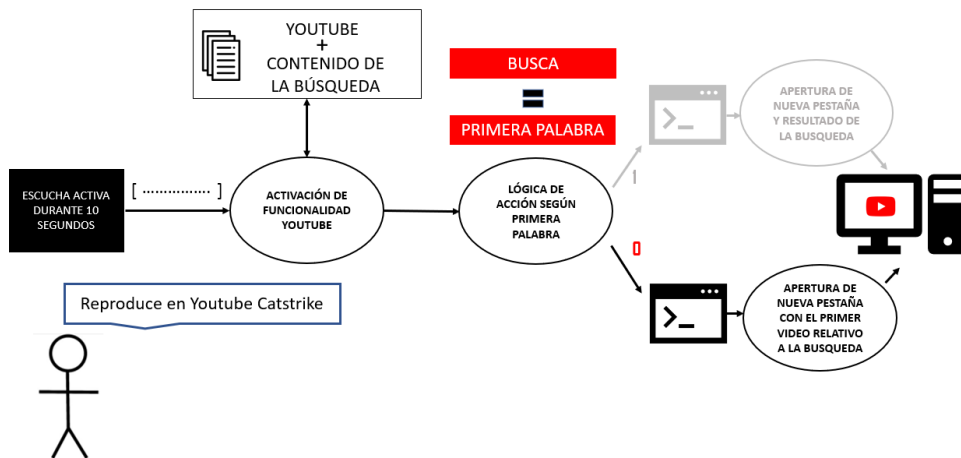


Respecto a los casos de secuencia estímulo respuesta se puede resaltar los siguientes casos:

- Usuario utiliza una frase con la palabra clave buscar:



- Usuario utiliza una frase con una palabra clave diferente a buscar:



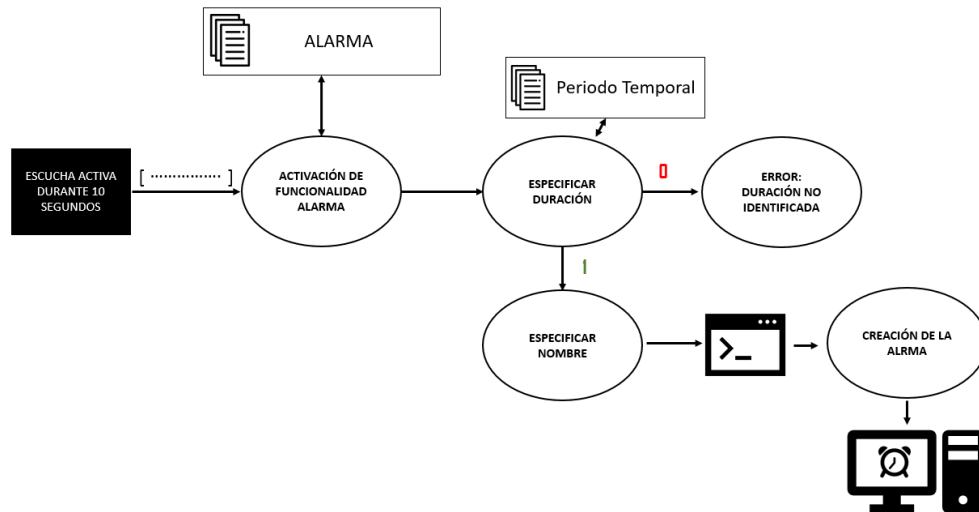
5. Creación de alarma.

Descripción: Teodoro crea una alarma para el usuario. Este avisará a la finalización de la misma.

Secuencia inicial paso a paso:

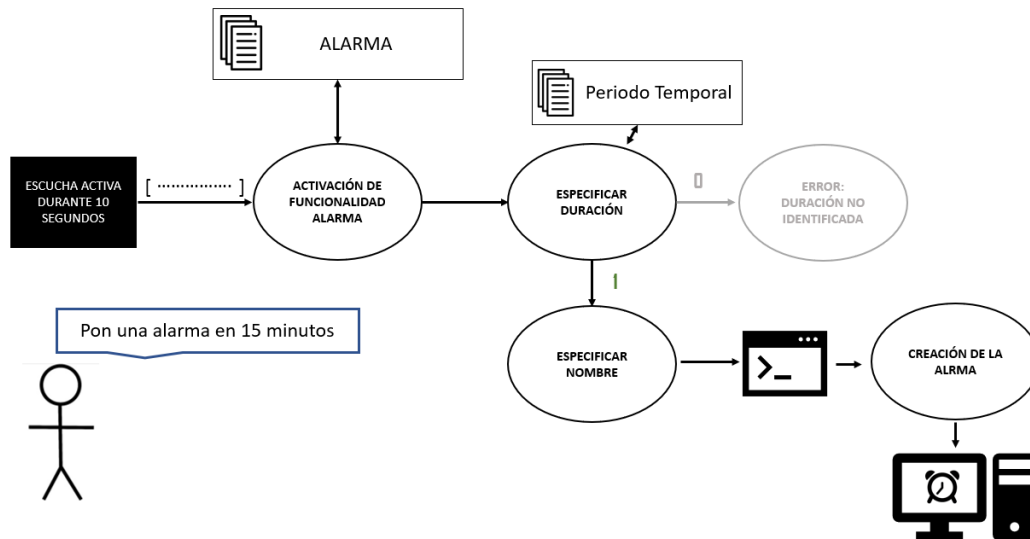
1. El usuario llama por voz a Teodoro.
2. El usuario formula una frase con la palabra clave "alarma" seguido de la duración de la misma u hora del día que se desea que finalice, y, además, de manera opcional, un nombre para esta alarma.
3. Teodoro ejecutará la funcionalidad sin bloquear el proceso del programa, para asegurar que se pueda seguir utilizando el asistente mientras la alarma esté activa. (**TbD**)
4. Cuando llegue la hora de finalización de la alarma, Teodoro realizará las siguientes acciones:
 - Parar la música (si no hay música sonando, no afectará).

- Avisar al usuario de la finalización de la alarma.
 - Reanudar la música (si no había música, no afectará).
5. Muestra del tiempo restante de la alarma a través de la opción *Countdown* de la GUI. (TbD)

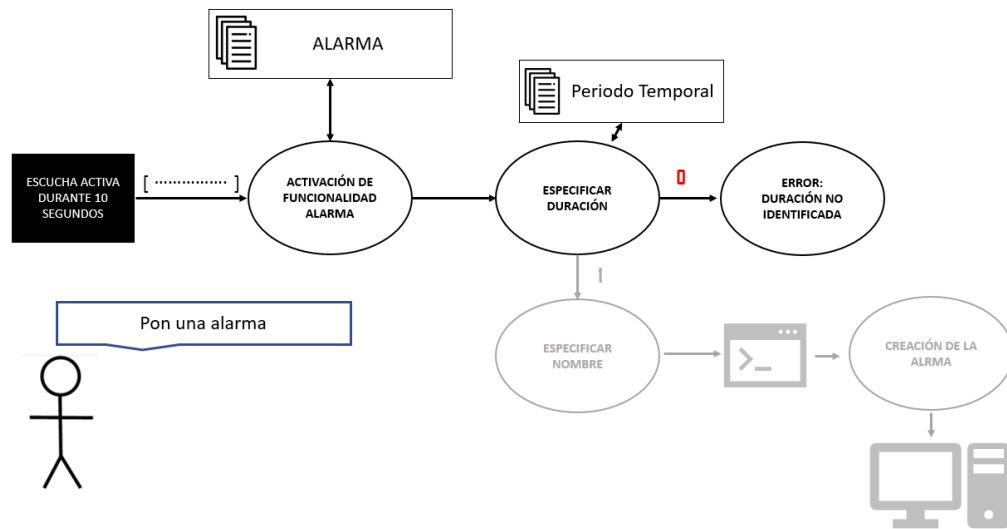


Respecto a los casos de secuencia estímulo respuesta se puede resaltar los siguientes casos:

- Usuario **NO** especifica la duración de la alarma:



- Usuario especifica la duración de la alarma:



6. Consulta del tiempo en una zona geográfica.

Descripción: Teodoro informa del tiempo meteorológico al usuario a través de una API en la red con soporte para el terminal.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario formula una frase con la oración clave “tiempo hace en”, seguido de la zona geográfica deseo de la consulta.
3. Teodoro accederá a los datos de clima de la zona determinada.
4. Teodoro guardará un archivo “.png” con el resultado de la consulta.
5. Teodoro abrirá el archivo para que el usuario pueda visualizar los resultados de su consulta.
6. Teodoro informará de los datos más relevantes del resultado por voz.



Respecto a los casos de secuencia estímulo respuesta se puede resaltar los siguientes casos:




- Usuario **NO** especifica la localización:




- Usuario especifica la localización:



Requisitos funcionales

- **APP-1.** Teodoro debe tener acceso a Spotify. El usuario debe acceder a una cuenta de Spotify ya registrada para que Teodoro pueda realizar las funcionalidades descritas. 
- **APP-2.** Teodoro debe tener acceso al navegador del equipo para realizar búsquedas en Google, Wikipedia y YouTube. 
- **APP-3.** Teodoro debe ser capaz de reproducir el primer video de YouTube resultado de la búsqueda solicitada del usuario. 

- **APP-4.** Teodoro debe ser capaz de avisar si existe una desambiguación del término buscado en la Wikipedia, y pedir al usuario que especifique qué opción de las posibles quiere consultar.
- **APP-5.** Teodoro debe tener acceso a una API en la red con soporte para el terminal para consulta del tiempo. Según la localización que el usuario consulte por voz, se accederá a los datos del clima de dicha localización.
- **APP-6.** Teodoro debe poder guardar la imagen proporcionada por la API de consulta del tiempo. Debe mostrar por pantalla el tiempo de la localización consultada mediante la GUI *Image*. 
- **APP-7.** Teodoro debe realizar un informe por voz del tiempo consultado con las características más importantes.
- **APP-8.** Teodoro debe crear alarmas. El usuario especificará la duración y el nombre de la alarma (opcional) y Teodoro creará una alarma.
- **APP-9.** La duración de la alarma debe poder establecerse mediante intervalo de tiempo (“5 minutos”) u hora de finalización de la misma (“a las 21:30”).
- **APP-10.** Teodoro debe vincular las alarmas a la GUI *Countdown*. Esta GUI mostrará el tiempo que falta hasta que suene la alarma por pantalla.
- **APP-11.** Teodoro no debe quedarse bloqueado por el proceso de alarma. Durante su transcurso, Teodoro debe ser capaz de responder de forma normal a las peticiones del usuario.

Características de *Calendar*

Descripción y Prioridad

La clase *Calendar* recoge todos métodos y atributos relacionados con la gestión del calendario de Google del usuario. Esta clase requiere una serie de pasos previos que el usuario ha de seguir con el objetivo de dotar al programa de los permisos necesarios para poder acceder a los datos del calendario. Por esta razón, y debido a la complejidad de sus funcionalidades, su prioridad es elevada en el proyecto. Además, no solo puede visualizar los eventos del usuario en el calendario asociado a su cuenta de Gmail, sino que, si este utiliza la aplicación Trello, y, previamente ha establecido la conectividad necesaria entre esta herramienta y Google Calendar, Teodoro será capaz de mostrar las tareas del tablero o tableros que el usuario especifique en el archivo de texto asociado en la BdC.

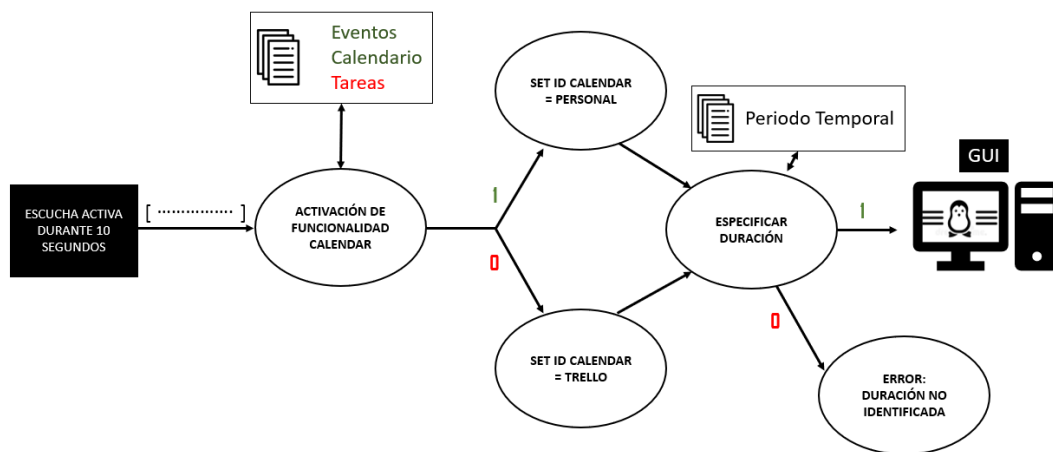
Secuencias Estímulo / Respuesta

1. Muestra de eventos o tareas.

Descripción: Teodoro conecta con Google Calendar para poder acceder a sus datos de eventos.

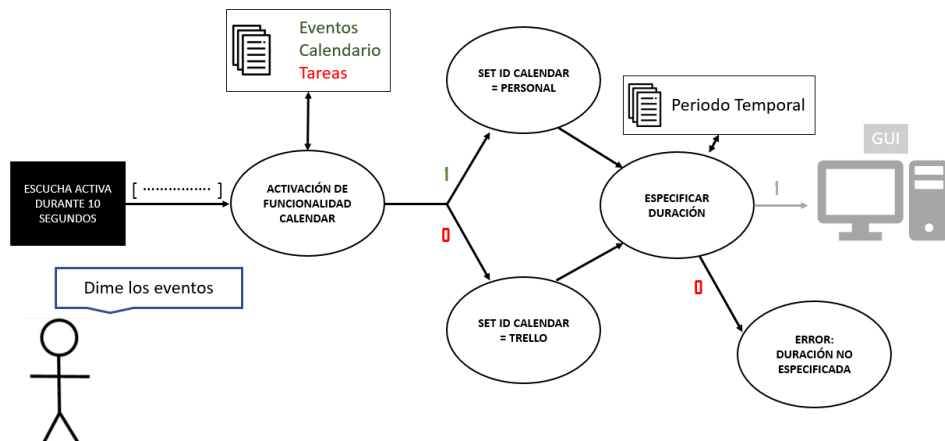
Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario realiza la petición con las palabras clave “eventos”, “calendario” (para acceder al calendario personal de su cuenta de Gmail) o “tareas” (para acceder a las tareas asociadas a Trello).
3. El usuario, además, debe establecer el periodo temporal de su petición, que podrá ir desde el mismo día (“hoy”), hasta “X” meses en adelante.
4. Teodoro procesa la información y accede a los datos solicitados.
5. Muestra de resultados a través de la opción *GetCalendar* de la GUI.

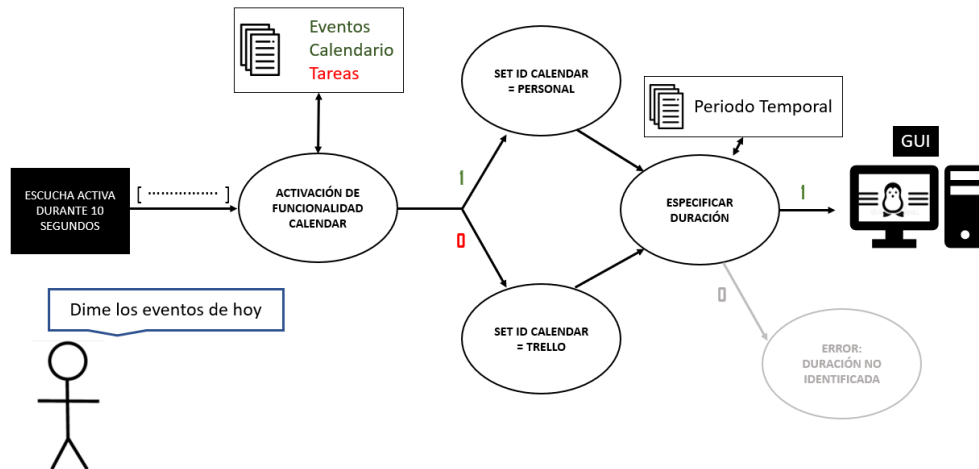


Respecto a los casos de secuencia estímulo respuesta, se puede resaltar los siguientes casos:

- Usuario **NO** especifica el periodo temporal del evento:



- Usuario especifica el periodo temporal del evento:

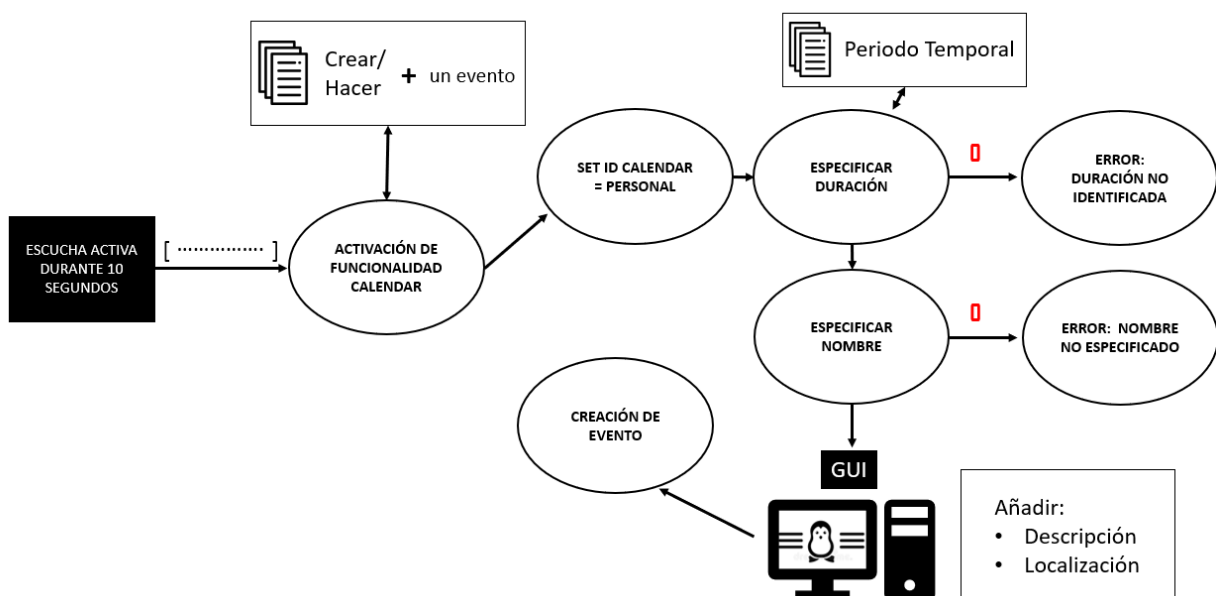


2. Creación de evento.

Descripción: Teodoro conecta con Google Calendar para poder crear un evento. Solo se podrá crear eventos en la cuenta personal del usuario, ya que las “tarea” de Trello se deberán crear en la propia aplicación.

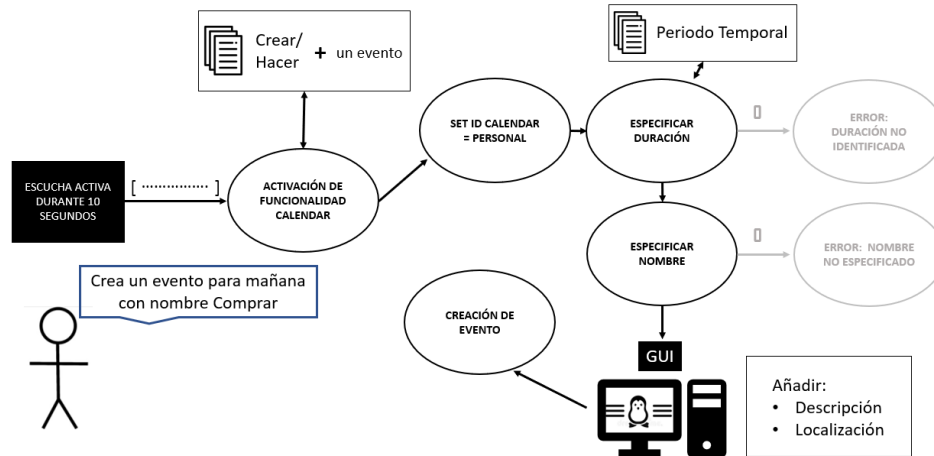
Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario realiza la petición con la oración clave “crear evento” o similares.
3. El usuario, además, debe establecer el periodo temporal de su petición, que podrá ir desde el mismo día (“hoy”), el día siguiente (“mañana”), en dos días (“pasado mañana”) o una fecha concreta.
4. Teodoro procesa la información.
5. Petición de descripción y localización del evento a través de la opción *SetCalendar* de la GUI.
6. Creación del evento.

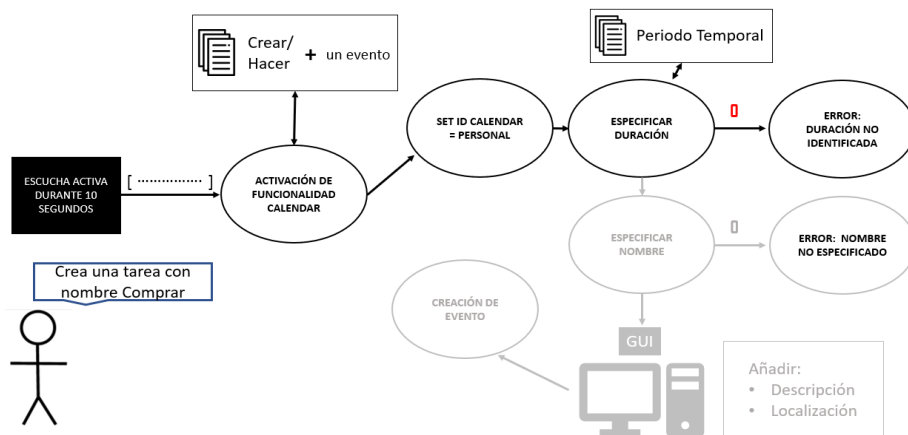


Respecto a los casos de secuencia estímulo respuesta se puede resaltar los siguientes casos:

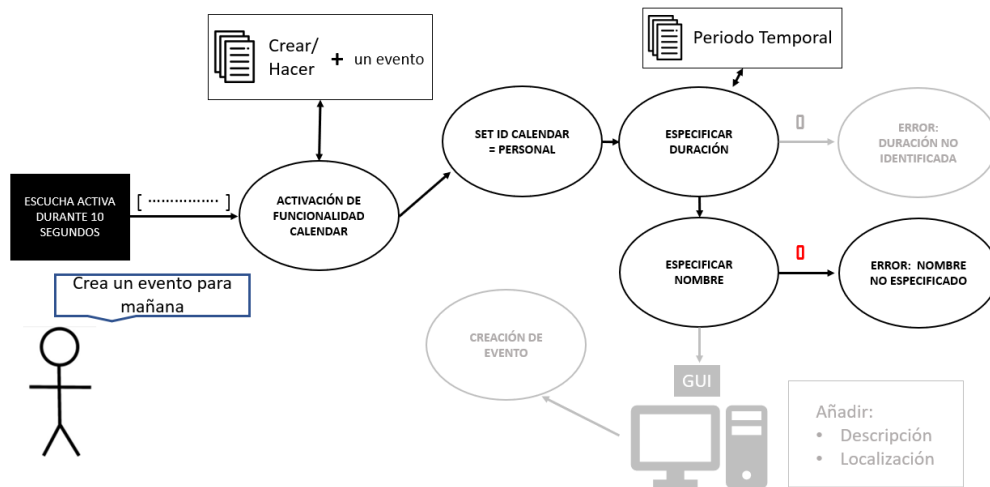
- Usuario especifica el evento a crear correctamente:



- Usuario NO especifica la duración del evento a crear:



- Usuario NO especifica el nombre del evento a crear:



3. Modificación de evento.

Descripción: Teodoro conecta con Google Calendar para poder modificar un evento. Solo se podrá modificar eventos previamente creados en el calendario personal de usuario. **(TbD)**

Requisitos funcionales

- **CAL-1.** Teodoro debe poder acceder a la API de Google Calendar, siempre y cuando el usuario haya realizado los pasos de configuración para el acceso a la API y a su calendario personal.
- **CAL-2.** Teodoro debe poder acceder a las tareas de Trello previamente vinculadas al calendario del usuario a través de los pasos especificados en los documentos correspondientes.
- **CAL-3.** Teodoro debe mostrar eventos dentro de Google Calendar hasta una fecha concreta. Esta fecha irá desde el día actual a los próximos “X” meses.
- **CAL-4.** Teodoro debe mostrar estos eventos mediante la GUI *GetCalendar*. Para mostrar estos eventos se utilizará esta GUI que mostrará una lista con cada uno de estos eventos hasta una fecha concreta.
- **CAL-5.** Teodoro debe poder crear eventos dentro de Google Calendar con una duración y nombre concretos. Si el usuario solicita crear un evento, Teodoro accederá a Google Calendar y tras introducir una serie de datos como el nombre, la duración, la localización y una descripción se creará un evento que será incluido en el Google Calendar del usuario.
- **CAL-6.** Teodoro deberá mostrar una GUI donde el usuario podrá introducir una descripción y la localización del evento creado. Al crear un evento se activará la GUI *SetCalendar* para que el usuario realice lo previamente comentado.
- **CAL-7.** Teodoro deberá ser capaz de modificar un evento existente en el Google Calendar personal del usuario.

Características de Teodoro

Descripción y Prioridad

La clase Teodoro es el sistema base, la clase que hereda todos los atributos y métodos del resto de sistemas, por lo que es la encargada de definir el flujo del programa. Por esta razón, aparte de contener aquellas funcionalidades más relacionadas con la interacción social con el usuario, también es la encargada de ejecutar el método especificado por la petición realizada por el usuario al sistema. Debido a su importancia, su prioridad es máxima.

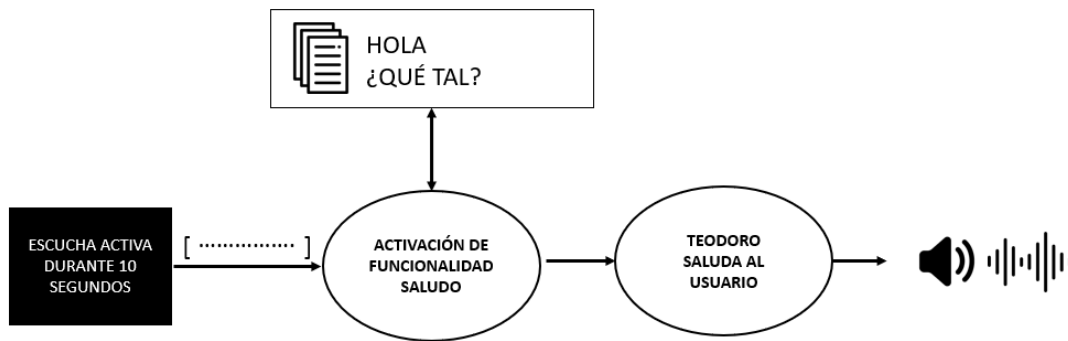
Secuencias Estímulo / Respuesta

1. Saludo.

Descripción: Teodoro saluda al usuario. Es el método que se ejecuta por defecto al inicio del programa.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario saluda a Teodoro con frases como “Hola” o “¿qué tal?” y similares.
3. Teodoro responde con un saludo.



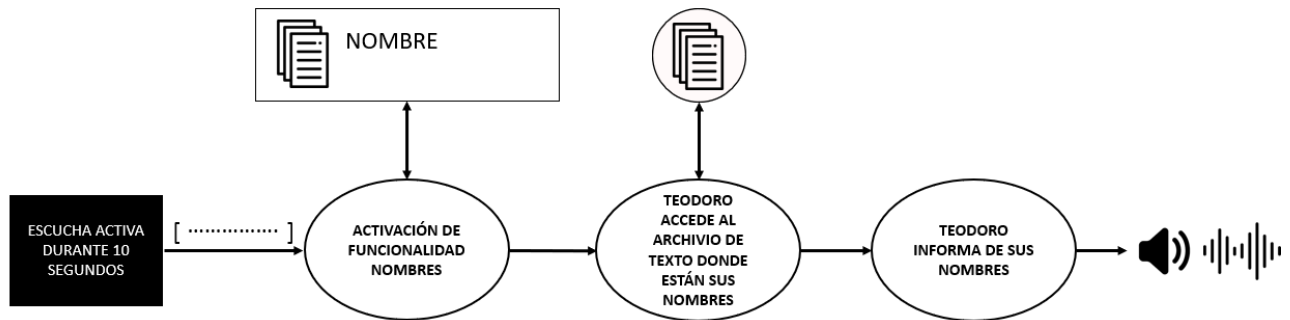
2. Informar sobre los posibles nombres del asistente.

Descripción: Teodoro lista los posibles nombres con los que se le puede llamar.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario consulta los nombres del asistente con frases como: “¿Cómo te llamas?”, “¿dime tus nombres?” y similares.
3. Teodoro responde por voz la lista de sus nombres establecidos en el archivo de texto correspondiente de la BdC.

4. Mostrado de los nombres a través de la opción *Show* de la GUI.

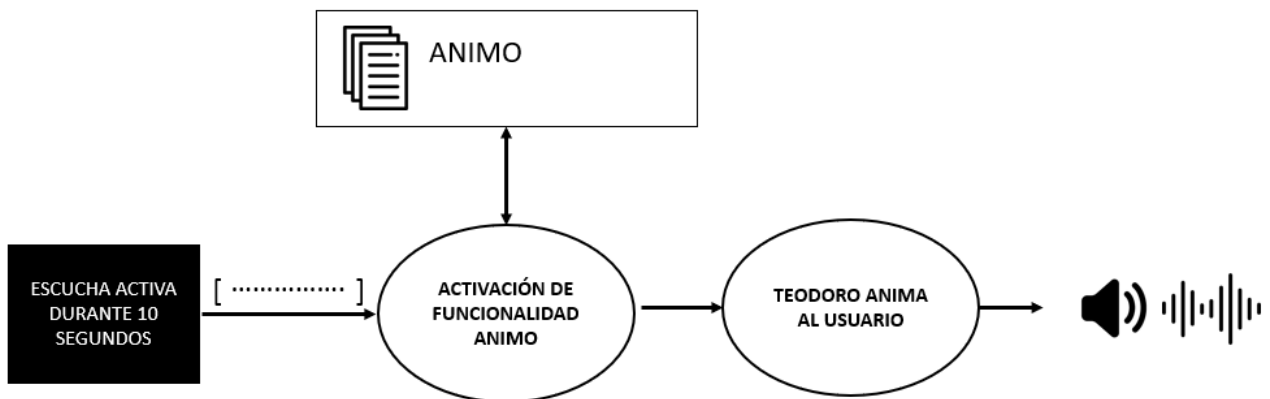


3. Animar al usuario.

Descripción: Teodoro anima al usuario.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario pide ánimos a Teodoro.
3. Teodoro anima al usuario por voz.

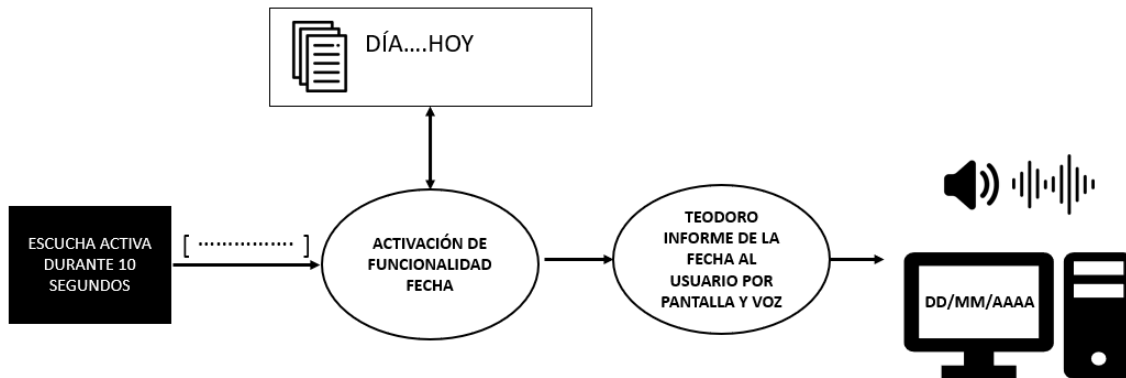


4. Informar sobre el día actual.

Descripción: Teodoro informa sobre el día actual al usuario.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario pregunta por el día actual con frases como: "¿Qué día es hoy?" y similares.
3. Teodoro informa sobre el día actual, con el día de la semana y la fecha exacta (día, mes y año) por voz.
4. Teodoro informa sobre el día actual, con el día de la semana y la fecha exacta (día, mes y año) a través de la opción *Show* de la GUI.

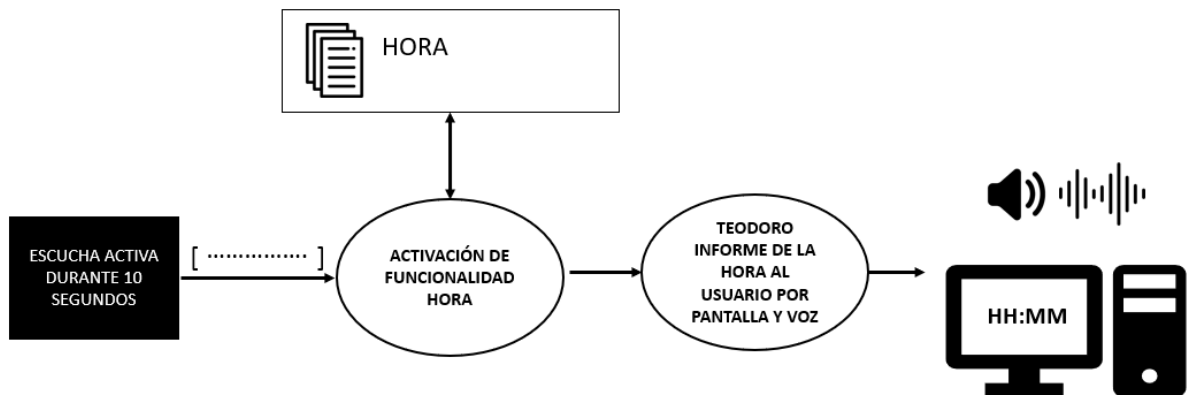


5. Informar sobre la hora actual.

Descripción: Teodoro informa sobre la hora actual al usuario.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario pregunta por la hora actual con frases como: “¿Qué hora es?” y similares.
3. Teodoro informa sobre la hora actual (horas y minutos) por voz.
4. Teodoro informa sobre la hora actual (horas y minutos) a través de la opción *Show* de la GUI.



6. Realizar funcionalidad.

Descripción: Teodoro decide qué funcionalidad ejecutar según la petición realizada por el usuario a través del método “*getAction*”.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario realiza cualquier petición.
3. Si la petición no es una cadena vacía (el usuario no ha dicho nada), se procede a analizar la petición con las palabras y frases clave de activación de funcionalidad (a través del archivo correspondiente a todos los posibles comandos de la BdC). Si es cadena vacía, Teodoro vuelve a su estado inicial.
4. Si la petición no es vacía, Teodoro busca la funcionalidad que corresponde con las posibles palabras o frases clave que la activan. Si no encuentra ninguna, Teodoro pedirá al usuario que repita su petición, a través del método *Repeat* de la clase *Engine*.

5. Si la búsqueda de palabras clave es exitosa, se realiza la lógica de la funcionalidad determinada.

7. Apagado.

Descripción: Teodoro se desconecta y se cierra el programa.

Secuencia inicial paso a paso:

1. El usuario llama por voz a Teodoro.
2. El usuario solicita cerrar el programa con frases como: “Apágate”, “adiós” y similares.
3. Teodoro se despide del usuario.
4. Se termina con la ejecución del programa.

Requisitos funcionales


- **TEO-1.** Teodoro debe informar al usuario de los nombres que se debe utilizar para comunicarse con él.
- **TEO-2.** Teodoro debe realizar funciones básicas predeterminadas como realizar saludos al usuario, animar al usuario, informar sobre la fecha y la hora actual.
- **TEO-3.** Teodoro debe poder realizar las peticiones del usuario. Una vez se haya reconocido por voz la petición del usuario, Teodoro activará la funcionalidad correspondiente a las palabras clave dentro de la petición del usuario.
- **TEO-4.** Teodoro debe pedir al usuario una repetición de la petición en caso de no contener ninguna palabra clave o de no entenderse.
- **TEO-5.** Teodoro debe poder despedirse y acabar su proceso.

Requisitos No Funcionales

Requisitos de Desempeño-Rendimiento



En relación al desempeño, al ser una aplicación para utilizar en tiempo real, se debe asegurar:

- Una **correcta identificación y reconocimiento de voz**, estableciendo una tasa o medida de falsos positivos mínima o inexistente, es decir, que el asistente se active sin haber dicho los nombres para su activación. Esto evitaría que el programa molestara al usuario y/o no respondiera correctamente, lo que iría en contra del objetivo del proyecto. También habría que maximizar la tasa de verdaderos positivos para que el asistente respondiera siempre (o casi siempre) que el usuario requiere de su ayuda. Esto se conseguirá realizando numerosas pruebas con el reconocimiento de audio de la clase *Engine*, con y sin música o ruido de fondo, con diferentes configuraciones de micrófono, etc. 
- Una **respuesta rápida y eficiente**, es decir, que el asistente no tarde demasiado en reconocer la tarea y ejecutarla. Esto va relacionado con medidas de cuál debe ser el tiempo máximo de

cada petición del usuario (de momento se ha establecido en 10 ya que existen peticiones que pueden ser extensas en el tiempo de habla, como mostrar eventos o crear un evento).

- La **interacción con el asistente debe ser natural**, ya que se pretende competir con asistentes personales ya existentes, por lo que las respuestas dadas por Teodoro deben ser naturales y consistentes con el idioma español. Por otro lado, se ha de dotar al usuario de la mayor cantidad de variantes posibles para activar una determinada tarea, para no limitar los comandos a sentencias robóticas y sin sentido en una conversación real.

Requisitos de Seguridad

Para el uso del calendario de Google del usuario, este deberá seguir una serie de pasos que se indicarán y detallarán en documentación a parte (“Instrucciones para la conexión con *Google Calendar*”). Tras estos pasos, el programa, en su primera ejecución no encontrará el *token* para el acceso a los datos del calendario, por lo cual, se pedirá al usuario que certifique el acceso a sus datos a través dirección web que se abrirá automáticamente. Tras esto, se creará el *token* de acceso y el programa evolucionará con normalidad siempre que esos permisos no sean revocados, en el cual se volverá a requerir certificar el acceso. Además, el usuario deberá indicar en el documento correspondiente de la BdC los identificadores (ID) de sus calendarios (cuenta personal de Gmail y Trello, si lo hubiera).




Todos estos pasos están dirigidos a garantizar la seguridad de los datos personales del usuario, pues Google es conocido por ser muy protector con los datos de los usuarios, por ello es el requerimiento de certificar el acceso y todos los pasos que el usuario ha de seguir hasta conseguir el acceso. Además, los archivos que se generen para el acceso (por ejemplo, el *token* personal) serán archivos ocultos y los atributos de la clase *Calendar* serán atributos privados de dicha clase, para asegurar que solo se utilice en dicha clase.

Atributos de Calidad

En consonancia con lo dicho en los últimos puntos de los requisitos de Desempeño-Rendimiento, el asistente debe conseguir dar una respuesta rápida y natural, para simular una interacción lo más realista posible con el usuario. Por otro lado, el sistema debe estar diseñado para optimizar su adaptabilidad y flexibilidad con el usuario, por ello la existencia de una BdC ampliable por el mismo, para que pueda personalizar su asistente personal a su medida y gusto. Además, el sistema debe ser accesible, por ello las primeras versiones se han pensado como proyectos de GitHub para la descarga gratuita para cualquier usuario que quiera probar el sistema.

Otros aspectos, más dirigidos hacia los desarrolladores, son los relacionados con la portabilidad, testeabilidad y de correcto desempeño, pues el asistente debe ser robusto a cambios y a errores de uso por parte del usuario sin que ello suponga un “cuelgue” del mismo o la no posibilidad de su utilización. Además, como se ha mencionado al principio de este documento, el objetivo final del proyecto es el de mejorar su portabilidad con otros S.O., e, incluso, con otros idiomas.

Otros requisitos

- **OTR-1.** La definición de las clases debe ser la establecida, con las herencias e inicializaciones necesarias. 
- **OTR-2.** La lectura de archivos determinados de la BdC debe realizar en aquellas clases que necesiten dicha información, o bien por la clase que las inicializa y también los utilice y que se los pasará en esta inicialización.
- **OTR-3.** Los atributos con información sensible del usuario deberán estar protegidos, así como los archivos externos, que deberán generarse como ocultos. 
- **OTR-4.** Las librerías que utiliza el programa deben ser de público acceso y debe indicarse el comando o recurso web con el que se puedan instalar.
- **OTR-5.** El programa no debe poder quedarse colgado por errores en su implementación o en un uso normal pero incorrecto por parte del usuario.
- **OTR-6.** El programa debe ser respetuoso en todo momento con el usuario, no se debe permitir alguna opción en la que se insulte o se falte al respeto al mismo.
- **OTR-7.** Las respuestas por voz de Teodoro deben ser audibles y de fácil entendimiento.
- **OTR-8.** Los mensajes que aparecen a través de GUI deben ser legibles, estar bien escritos y con un tamaño de letra aceptable. 

Apéndice A: Glosario

- **BdC:** Base de Conocimiento.
- **GUI:** Interfaz Gráfica de Usuario.
- **TbD:** *To Be Determined*

Apéndice B: Lista “To Be Determined”

Aquí se encuentran todos aquellos métodos que no están determinados todavía, bien por su complejidad, como por la no certeza de si se podrán implementar tal y como se han diseñado.

- **Alarma y GUI de alarma no bloqueante - *Applications*.**
- **Desambiguación en Wikipedia - *Applications*.**
- **Modificar evento - *Calendar*.**