Universidad Politécnica de Madrid

Escuela Técnica Superior de

Ingenieros Informáticos

Grado en Ingeniería Informática

**Trabajo Fin de Grado**

**Desarrollo de un Asistente para Público Infantil para Clasificar Sonidos del Cielo**

Autor: **Jhosef Anderson Cardich Palma**

Tutora: **Raquel Cedazo León**

Madrid, junio 2020

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

*Título:* **Desarrollo de un Asistente para Público Infantil para Clasificar Sonidos del Cielo**

Mes Año: Junio de 2020

*Autor:* Jhosef Anderson Cardich Palma

Tutor: Raque Cedazo León

ETSI Informáticos

Universidad Politécnica de Madrid

**Resumen**

El proyecto tiene como objetivo el diseño de una arquitectura y desarrollo de una aplicación para la clasificación de sonidos del cielo mediante un juego orientado al publico infantil.

Entre las taras del proyecto, se encuentra la definición y desarrollo de un asistente inteligente, el cual este dotado con la habilidad de poder comunicarse con el usuario de la manera mas natural posible, además de la visión de poder asegurar la integración con terceras aplicaciones. Se presenta también la investigación de que tecnologías estándar, framework, patrones de diseño poder usar para poder implementar dicho asistente virtual.

Para el diseño de la aplicación se ha seguido el patrón de diseño Modelo Vista Controlador (MVC), consiguiendo así una estructura clara de todos los componentes del sistema.

Para parte del modulo del asistente inteligente se ha seguido una implementación marcada por el framework *open-source* Rasa, de la cual se han utilizado todas sus características para implementar todos los requisitos de la parte conversacional (asistente inteligente) del proyecto.

También se ha diseñado la lógica del juego la cual se ha implementado usando el lenguaje de programación Python. Esta encarga de los cálculos, procesamientos necesarios y el acceso a los datos (sonidos, clasificaciones) que el usuario requiera.

Otra de las tareas descritas es el desarrollo de una aplicación frontal, para poder dotar de una interfaz con la cual el usuario pueda interactuar de manera amigable con el asistente. En el componente de la interfaz de proyecto, se ha realizado una aplicación web basada en el framework web Django. Esta aplicación tiene la capacidad de capturar los mensajes del usuario para enviárselos al asistente y poder transmitir los mensajes del asistente al usuario.

En cuanto al funcionamiento del juego, se tienen tres opciones principales: Aprendizaje, entrenamiento y clasificación. En el aprendizaje, el usuario podrá ser capaz de aprender los cinco tipos de sonidos de meteoros que existen. En la opción de entrenamiento se presentan los cinco tipos de sonidos y se le pide al usuario que intente clasificarlos. La ultima opción es la de clasificación, se presentan sonidos de cualquier tipo para que el usuario intente clasificarlos, cuando el usuario termine de clasificar un sonido, recibirá información sobre el tipo de clasificación que han ido dándole otros usuarios.

Al final de ese documento se presentan otras características interesantes con una visión de desarrollo a futuro.

**Abstract**

The project aims to design an architecture and develop an application for the classification of sky sounds using a game aimed at children.

Among the flaws of the project, is the definition and development of an intelligent assistant, which is endowed with the ability to communicate with the user in the most natural way possible, in addition to the vision of ensuring integration with third-party applications. Research is also presented on what standard technologies, framework, design patterns can be used to implement said the virtual assistant.

For the design of the application, the Design Model Controller View (MVC) has been followed, thus achieving a clear structure of all the components of the system.

For part of the smart assistant module, an implementation marked by the Rasa open-source framework has been followed, from which all its features have been used to implement all the requirements of the conversational part (smart assistant) of the project.

The game logic has also been designed and implemented using the Python programming language. It is responsible for the calculations, necessary processing, and access to the data (sounds, classifications) that the user requires.

Another of the tasks described is the development of a front-end application, to provide an interface with which the user can interact in a friendly way with the assistant. In the component of the project interface, a web application based on the Django web framework has been made. This application can capture user messages to send to the assistant and to transmit the messages from the assistant to the user.

Regarding the operation of the game, there are three main options: Learning, training, and classification. In learning, the user will be able to learn the five types of meteor sounds that exist. In the training option, the five types of sounds are presented, and the user is asked to try to classify them. The last option is the classification, sounds of any type are presented for the user to try to classify them when the user finishes classifying a sound, he will receive information about the type of classification that other users have been giving him.

At the end of this document, other interesting characteristics are presented with a vision of future development.

Tabla de Contenidos

**I Resumen**

**II Abstract**

**III Tabla de Contenidos**

**IV Índice de Tablas**

**V Índice de Ilustraciones**

[1 Introducción 8](#_Toc44286561)

[1.1 Motivación del proyecto 8](#_Toc44286562)

[1.2 Contexto 8](#_Toc44286563)

[1.3 Objetivos 10](#_Toc44286564)

[2 Estado del Arte 11](#_Toc44286565)

[2.1 Asistentes Virtuales 11](#_Toc44286566)

[2.2 ¿Qué son los Asistentes Virtuales? 12](#_Toc44286567)

[3 Desarrollo 16](#_Toc44286568)

[3.1 Requisitos 16](#_Toc44286569)

[3.2 Planteamiento de la Arquitectura 19](#_Toc44286570)

[3.2.1 MVC – Diseño de las Aplicaciones Lógicas del Controlador del Proyecto 20](#_Toc44286571)

[3.2.2 MVC – Esquema de Diseño para el Frontal-Vista del Proyecto 21](#_Toc44286572)

[3.2.3 MVC – Descripción de la Aplicación Acceso para el acceso de Datos 22](#_Toc44286573)

[3.2.4 Requerimientos y herramientas 24](#_Toc44286574)

[3.3 Implementación de los Módulos del Controlador del Proyecto 25](#_Toc44286575)

[3.3.1 Módulo del Comportamiento del Asistente 25](#_Toc44286576)

[3.3.2 Rasa Framework 25](#_Toc44286577)

[3.3.2.1 Rasa NLU 25](#_Toc44286578)

[3.3.2.2 Rasa Core 26](#_Toc44286579)

[3.3.2.3 Rasa NLG 26](#_Toc44286580)

[3.3.2.4 Procesamiento de un mensaje en Rasa 26](#_Toc44286581)

[3.3.3 Componentes Rasa 27](#_Toc44286582)

[3.3.3.1 Intents 27](#_Toc44286583)

[3.3.3.2 Utterances 28](#_Toc44286584)

[3.3.3.3 Actions 28](#_Toc44286585)

[3.3.3.4 Stories 29](#_Toc44286586)

[3.3.3.5 Slots 30](#_Toc44286587)

[3.3.3.6 Dominio 30](#_Toc44286588)

[3.3.3.6.1 Implementación de Rasa para el Asistente – Stories 30](#_Toc44286589)

[3.3.3.6.1.1 Ejemplo de implementación de Rasa 31](#_Toc44286590)

[3.3.4 Módulo-Aplicación de Acciones del Juego 32](#_Toc44286591)

[3.3.4.1 Implementaciones de funcionalidades 32](#_Toc44286592)

[3.3.5 Integración entre el Módulo de Acciones y el Asistente 35](#_Toc44286593)

[3.3.5.1 Función Aprendizaje y Función de Entrenamiento 35](#_Toc44286594)

[3.3.5.2 Función Clasificación 35](#_Toc44286595)

[3.3.5.3 Pruebas de Integración 36](#_Toc44286596)

[3.3.5.3.1 Pruebas 38](#_Toc44286597)

[ Saludo y reconocimiento: 38](#_Toc44286598)

[ Función de Aprendizaje: 38](#_Toc44286599)

[3.3.6 Módulo-Aplicación Frontal Web del Proyecto 39](#_Toc44286600)

[3.3.6.1 Frontal Web del Proyecto – DJango 39](#_Toc44286601)

[3.3.6.2 Audio 40](#_Toc44286602)

[3.3.6.2.1 Speech to Text 40](#_Toc44286603)

[3.3.6.2.2 Google Speech to Text 40](#_Toc44286604)

[3.3.7 Interacción entre el módulo de Conversacional y el Frontal 40](#_Toc44286605)

[3.3.7.1 Diseño de mensajes entre módulos 40](#_Toc44286606)

[3.3.7.2 Procesamiento de Mensajes en el Frontal 42](#_Toc44286607)

[3.3.7.3 Pruebas de integración 42](#_Toc44286608)

[3.4 Funcionamiento del Sistema 43](#_Toc44286609)

[3.4.1 Comandos y Funcionalidades 43](#_Toc44286610)

[3.4.1.1 Bienvenida 43](#_Toc44286611)

[3.4.1.2 Aprendizaje sobre los Sonidos del Cielo 43](#_Toc44286612)

[3.4.1.3 Entrenamiento 43](#_Toc44286613)

[3.4.1.4 Clasificación 43](#_Toc44286614)

[3.4.1.5 Despedida 43](#_Toc44286615)

[4 Resultados y conclusiones 44](#_Toc44286616)

[5 Bibliografía 45](#_Toc44286617)

[6 Anexos 46](#_Toc44286618)

Índice de Tablas

[Tabla 1. Asociación entre actions y clases asociadas para la funcionalidad de Aprendizaje 21](#_Toc44173658)

Índice de Ilustraciones

[Ilustración 1. . Lluvia de estrellas. Fotografía de Casey Horner. 7](file:////Users/jhos/Desktop/GIT-LOCAL/UPM-ChatBot-SoundsOfMeteors/documentation/003%20TFG%20Entrega%20Memoria%20Final.DOCX#_Toc44189477)

[Ilustración 2. Modelo Vista Controlador para el desarrollo del proyecto. 11](file:////Users/jhos/Desktop/GIT-LOCAL/UPM-ChatBot-SoundsOfMeteors/documentation/003%20TFG%20Entrega%20Memoria%20Final.DOCX#_Toc44189478)

[Ilustración 3. Diagrama general organizativo de los módulos del proyecto según el patrón MVC. 13](#_Toc44189479)

[Ilustración 4. Diagrama de pasos básicos del asistente para responder a un mensaje. 16](#_Toc44189480)

# Introducción

## Motivación del proyecto

Desde el campo de la inteligencia artificial, la interacciones entre las personas y los ordenadores lleva ya un tiempo implantándose de manera asidua, también evolucionando junto con las necesidades de los consumidores tecnológicos, o para proveer automatización en ciertas tareas. Estos consumidores ya no solo son personas con conocimientos técnicos acerca de tecnologías, nos podemos encontrar con muchos tipos de perfiles debido a que la tecnología esta revolucionando la manera en la que los humanos realizamos ciertas tareas que llevan ya implantadas desde los inicios de las civilizaciones. El acceso a la información para la educación, o la manera en la que la enseñanza ha ido cambiando, se ha ido adaptando para hacer frente a los problemas que puede presentar la disponibilidad de la enseñanza dirigida. Solamente hay que pensar como ha sido la enseñanza desde siempre: presencial, mediante un profesor, en un aula, con alumnos escuchándole y resolviendo las dudas que puedan surgir. Pero ahora imaginemos ¿Y si tuviéramos al profesor en cualquier momento y en cualquier lugar? o ¿Y si no pudiéramos ir clases por algún motivo?

La visión tecnológica que tenemos de la sociedad con la implantación de asistentes virtuales presenta un panorama futurista, estos están presentes en muchas de las tareas cotidianas que antes se hacían de manera rudimentaria, en otros casos son capaces de poder realizar tareas de manera independiente, sin ningún tipo de intervención humana. Ahora también estos toman parte en la educación. Estos asistentes son cada día más humanos, algunos pueden dotarse de capacidades, como poder interactuar mediante voz con el usuario haciendo así la experiencia mas cercana y fluida. Estas herramientas tecnológicas hacen que el aprendizaje haya evolucionado a otro nivel y aprovechando este potencial, se desea que este proyecto pueda acercar la ciencia al público en general, para que puedan permitirse interactuar con una fuente de conocimiento sobre los sonidos del cielo, de manera entretenida y bajo demanda.

## Contexto

La experiencia de poder entender los elementos del espacio exterior siempre ha sido algo emocionante para los seres humanos. Los meteoros son elementos que viajan por el espacio y han sido siempre objeto de investigación por la toda la información que pueden brindar a los científicos para desarrollar sus investigaciones y entender así el espacio infinito. La astronomía y su comunidad tienen que entender las características de los meteoros, saber clasificarlos. Uno de los fenómenos astronómicos mas espectaculares en los que podemos disfrutar los meteoros de manera visual son las lluvias de estrellas.

Imagen que contiene naturaleza, exterior, cielo, montaña

Descripción generada automáticamentePero ¿podríamos ser capaces de poder escuchar esa lluvia de meteoros? La tecnología puede

ayudar en esto, aprovechados de las características de este fenómeno. Pero para ello hay que entender un poco mejor que es lo que pasa cuando observamos el rastro de una estrella fugaz en el cielo. Cuando un meteoro, entra en la atmósfera, este lo hace con gran velocidad y teniendo una gran fricción con la atmósfera, el rastro brillante que vemos en el cielo es debido a la ionización del aire alrededor del meteoro [1], así podemos ser capaces de ver el recorrido que hacen hasta que terminan de desintegrarse o incluso llegar a la superficie terrestre (meteoritos). Los meteoros están conformados también por metal y otros elementos, gracias a ello y con ayuda de ondas de radio reflejadas en los meteoros, se puede capturar con un receptor los ecos de esas ondas reflejadas y transformarlas en sonido.

De esa manera ya podemos escuchar los meteoros. Pero ¿con que tipos de sonidos podemos identificarlos? Se puede decir que existen cinco tipos de sonidos meteoros: Meteoro Underdense, Meteoro de tipo M, Overdense largo, Overdense medio y Overdense corto.

Imagen que contiene dibujo, reloj

Descripción generada automáticamente

Es interesante y despierta curiosidad poder clasificar los sonidos del cielo y este es el objetivo de este proyecto de ciencia ciudadana, con el trasfondo de que el publico general forme parte de este experimento, involucrando así no solo a las personas con conocimientos técnicos sobre astrología, sino todo aquel que este interesado, para convertirlos en protagonistas de este experimento de clasificación. Esta aplicación será una  herramienta educativa para los usuarios, los cuales podrán aprender de manera entretenida, ya que el asistente que guiara al publico durante la clasificación aporta una experiencia nueva de aprendizaje y también, los datos recogidos de la aplicación servirán como fuente de datos de posibles clasificaciones para un sonido, lo cuales podrían tener otras aplicaciones de provecho para el estudio de los meteoros.

## Objetivos

El principal objetivo de este proyecto es poder desarrollar la implementación de la inteligencia del asistente, que servirá a la aplicación, siguiendo este unos estándares dirigidos por un framework. En términos de  un modelo, vista, controlador (MVC).

Se desea que las tecnologías  usadas para la lógica del juego y el desarrollo del asistente sean compatibles con implementación de una vista (interfaz web) y además con el modelo (acceso a los datos, sonidos, con los que trabajará la app). Los objetivos se resumen a continuación:

* Diseño de la Arquitectura del Proyecto.
* Búsqueda de Tecnologías que satisfagan los requerimientos del Proyecto.
* Estudio de la compatibilidad entre las diferentes implementaciones.
* Adaptación del comportamiento del asistente a la tecnología a emplear.
* Implementación del Asistente Inteligente.
* Implementación de la Lógica del Juego.
* Implementación de una aplicación Frontal basada en Tecnología Web.

# Estado del Arte

## Asistentes Virtuales

Antes que nada, nos tenemos que detener a conocer una de las partes mas interesantes del proyecto. Los asistentes virtuales están presentes hoy en nuestro día a día, incluso aunque nos los usemos. La mayoría de estos están implementa en muchos dispositivos, los teléfonos móviles por ejemplo cuentan con asistentes, los cuales ejecutan tareas sencillas como hacer llamadas, mandar mensajes, etc., pero la evolución de estos ha hecho que estos se vuelvan cada vez mas personales, mas inteligentes. Pero ¿Qué es un asistente virtual?

Captura de pantalla de un celular

Descripción generada automáticamenteCaptura de pantalla de un celular

Descripción generada automáticamente

## ¿Qué son los Asistentes Virtuales?

Como casi toda la mayoría de las tecnologías desarrolladas existentes hoy en día, los asistentes virtuales nacen, en principio, para poder cubrir necesidades de consulta de información o para de guiar a través de pasos predefinidos para la realización de alguna tarea. Estas tareas en principio eran bastante sencillas, sin embargo los asistentes virtuales aportaban algo mas, la disponibilidad de ejecutar esas tareas en cualquier momento, es decir, ponen a disposición, dan soporte al usuario, de manera ininterrumpida.

Su implementación en las empresas que tienen presencia online a aumentado con mucha rapidez. Por ejemplo, en tiempos actuales, la pandemia del COVID-19 ha obligado a todas las empresas a emplear medidas para hacer frente a la reorganización en el modelo en el que desarrollaban sus tareas, ahora el trabajo a distancia se ha vuelto una moda necesaria.

Captura de pantalla de un celular

Descripción generada automáticamenteAsí entonces el empleo de asistentes virtuales se ha visto impulsado por la necesidad de ofrecer los servicios al cliente, brindando una experiencia distinta, sin importar la ubicación o el momento, ya que pueden ser usados desde cualquier lugar y cuando el usuario lo requiera.

Los asistentes virtuales funcionan mediante la comunicación con el usuario. En la idea mas básica, estos tienen que ser capaces de poder entender o reconocer ciertos comandos o palabras clave para identificar el servicio o tarea que el usuario quiere realizar.

La comunicación entre el usuario y el asistente puede tener diferentes medios y formas. Se tienen asistentes con los que podemos comunicarnos de manera escrita, enviándole nuestras ordenes mediante una entrada de texto, el asistente también responderá de la misma manera. Por ejemplo, el asistente de KLM, es capaz de brindarnos soporte en cuanto a los viajes en avión, tramites con el billete del vuelo y demás gestiones de reserva. Se encuentra disponible en su pagina web pero también esta integrada dentro de la aplicación *Messenger* de Facebook, haciendo la interacción bastante menos robótica, como si Una captura de pantalla de un celular con letras

Descripción generada automáticamente*chateáramos*  con cualquier otra persona usando este servicio de mensajería social.

Captura de pantalla de un celular con texto e imagen

Descripción generada automáticamenteSiri de Apple es otro asistente, con el que podemos tener una conversación mediante voz, el sistema es capaz de reconocer todas las tareas y de mantener conversaciones cortas gracias al tecnóloga de síntesis de voz que usa [2], de esa manera puede comprender que es lo que queremos decir y responder por voz. Con respecto a esto, hay una característica presente en los asistentes mas modernos. Muchos de los asistentes se centran en ser mas humanos. Por ejemplo tenemos los asistentes conversacionales, lo cuales usan sistemas de diálogos bastante amplios, permitiéndoles expresarse de la forma mas humana posible, haciendo uso de alguna interfaz grafica, mostrando gestos y voz.

De todo lo comentado hay que tener en cuenta la inteligencia que está por detrás de estos asistentes, ya que son capaces de reconocer una infinidad de comandos, incluso muchos de ellos que no se habían predefinido, sino que estos también aprenden de los datos de las conversaciones de todos los usuarios que los utilizan, extrayendo patrones de las acciones que solicitan, de la manera en la que se expresan los usuarios, de sus gustos, brindándoles este aprendizaje, a los asistentes, experiencia mas amplia para reconocer lo que el usuario quiere y como lo quiere.

Hoy en día los asistentes son mas personales, esto quiere decir que nos entienden cada día mejor, pues ahora aprenden de los usuarios constantemente. Esto es gracias a la implementación de la inteligencia artificial en este tipo de software, que es capaz de reconocer acciones, predecir comportamientos esperados ante un comando del usuario, basado en la experiencia previa de infinidad de casos.

Existen asistentes que ahora forman parte del hogar, aportando automatización en las tareas en la administración, todo esto se refiere a otro campo comercial de la tecnología llamada domótica [3]. Estas tecnólogas se encargan de la seguridad, control de energía y administración de otros dispositivos presentes en el hogar. La mayoría de estos dispositivos son dedicados a estos y se presentan en formato altavoz, como son los casos de Amazon Echo, el cual tiene integrado el asistente de voz Alexa y Google Nest Mini, el cual usa la tecnología Google Assistant, la cual esta presente en muchos productos.

Imagen que contiene interior, tabla, ventana, grande

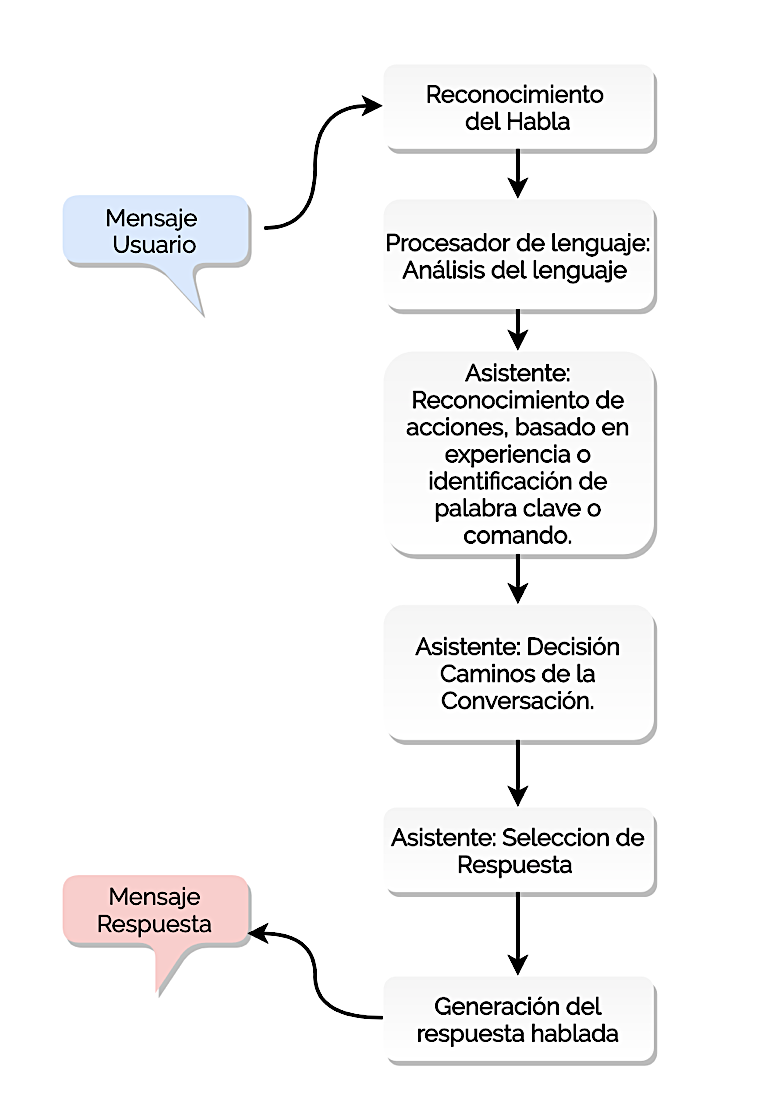
Descripción generada automáticamenteCaptura de pantalla de un celular

Descripción generada automáticamente

Imagen que contiene interior, tabla, hecho de madera, escritorio

Descripción generada automáticamenteImagen que contiene tabla

Descripción generada automáticamente

Los asistentes de voz presentan en su arquitectura una serie de elementos los cuales son importantes conocer, ya que es de interés en este proyecto, por ser la creación de un asistente un requisito fundamental. La arquitectura que presentan se puede extrapolar de una conversación, en este caso por voz, y de manera general los elementos que se identifican se organizan secuencialmente como se muestra en el diagrama:

Podemos describir una conversación en el diagrama presentado: El usuario es el que normalmente inicia el dialogo, entonces el sistema conversacional se activa y espera mas información del usuario. Cuando el usuario empieza a hablar, lo primero que suele suceder, es la captura de la voz del usuario para poder analizar el lenguaje. Cuando la tecnología apropiada para este trabajo analiza el lenguaje, devuelve dicho análisis en información procesable que es analizado por el asistente con la finalidad de reconocer la acción o comando que el usuario desea realizar, en función de esto, el asistente tendrá que decidir que acción tomar a continuación y además modificar el flujo el dialogo para definir un camino de la conversación. La respuesta devuelta al usuario suele ser seleccionada función de las experiencias del usuario o por el reconocimiento de alguna palabra clave, que el asistente entiende como un comando. Dentro del análisis de reconocimiento puede intervenir tecnología de aprendizaje, para poder guardar esta interacción como una experiencia mas. La respuesta se genera, y luego este es pasado a un sintetizador de voz, el cual generar la respuesta hablada.

# Desarrollo

## Requisitos

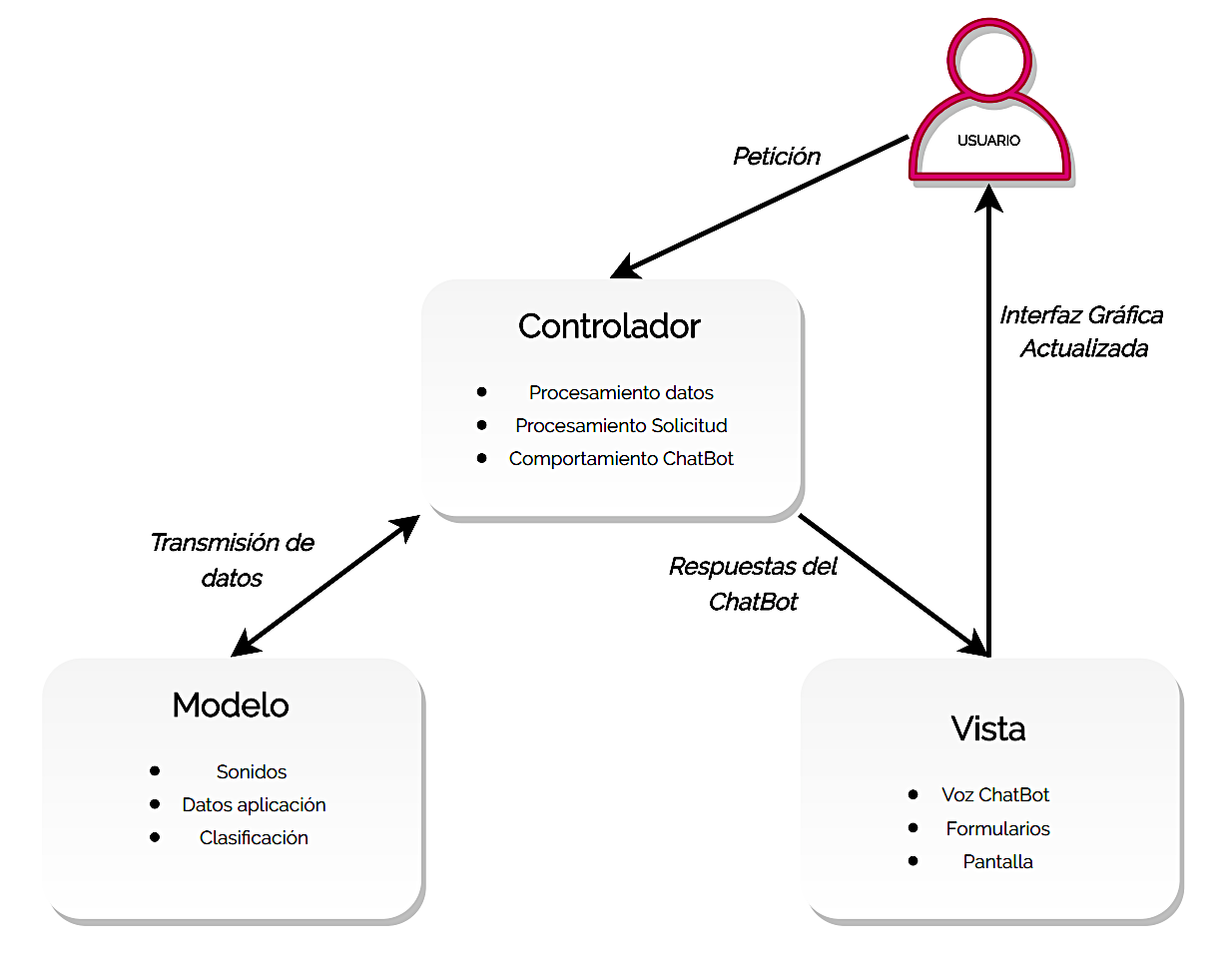
Los requerimientos de la aplicación van ligados directamente con las funcionalidades presentes dentro de la experiencia de uso que ofrecerá el juego, el componente que se encarga de la parte conversacional y el frontal de la aplicación . Por lo tanto mencionamos de manera general, los principales requisitos a nivel de usuario y también a nivel de software que se han identificado, asociando a una tarea para la realización del proyecto.

|  |  |  |
| --- | --- | --- |
| **Requisito** | **Tareas** | **Componente Asociado** |
| El usuario interactúa por el navegador web. | * Implementación de navegación en un frontal web. * Diseño de navegación y apariencia de la experiencia visual de usuario. | * Frontal. |
| El usuario es capaz de introducir texto para los comandos. | * Captura de datos en el frontal web. | * Frontal |
| El usuario interactúa con el asistente en forma de dialogo. | * Diseño de flujos conversacionales. * Diseño de la forma en que los mensajes son presentados. | * Frontal |
| El juego ofrece la opción de aprendizaje | * Desarrollo de la lógica del Juego. * Funciones de procesamiento para el acceso a los sonidos. * Definición del flujo conversacional en asistente. | * Frontal * Lógica de procesamiento del Juego. * Asistente conversacional. |
| El juego ofrece la opción de entrenamiento | * Desarrollo de la lógica del Juego. * Definición del flujo conversacional en el asistente. * Diseño de mensajes a intercambiar entre el frontal y mensajes * Diseño implementación de interacción entre el frontal y el asistente. * Diseño entre el asistente y la lógica del juego, para el acceso a los sonidos. | * Frontal * Lógica del procesamiento del Juego. * Asistente conversacional. |
| El juego ofrece la opción de clasificación. | * Desarrollo de la lógica del Juego. * Funciones de acceso a los sonidos. * Almacenamiento de resultados. * Tratamiento de archivos de clasificación. * Definición de *path* conversacional en asistente. | * Frontal * Lógica de procesamiento del Juego. * Asistente conversacional |
| El sistema ofrece la opción de ayuda | * Definición de mensajes de ayuda en los mensajes del asistente. | * Asistente conversacional. |
| El sistema tendrá que reproducir sonidos para el usuario. | * Implementación de un reproductor de sonidos en el frontal. | * Frontal. |
| El sistema tiene que ser capaz de almacenar la información de las clasificaciones. | * Tratamiento de respuestas en el flujo conversacional. * Almacenamiento de los datos de clasificación por la lógica del juego. * Diseño de los tipos de Archivos donde se guardan las clasificaciones. * Definiciones de funciones de escritura, lectura de archivos de texto. | * Lógica de procesamiento del juego. * Asistente conversacional. |
| El sistema tiene que mostrar información de clasificación de los sonidos. | * Cálculos sobre las clasificaciones existentes para un sonido. | * Lógica de procesamiento del juego. |
| El sistema tiene ser capaz de poder desarrollar un dialogo lo mas natural posible. | * Implementación de un lenguaje natural. * Definir las posibles variaciones para una conversación. | * Asistente conversacional |
| El aspecto del sistema tiene que ser amigable. | * Diseño grafico orientado al publico infantil. | * Frontal |
| El frontal tiene que ser capaz de solicitar sonidos al asistente. | * Diseño de mensajes para la comunicación de sonidos en JSON. * Acceso mediante funciones usando JQuery. | * Frontal * Lógica de procesamiento del Juego. |

## Planteamiento de la Arquitectura

Una vez se han identificado los requisitos de la aplicación, surge la necesidad de tener una organización que permita una visualización global del proyecto, en donde se ubiquen los diferentes componentes a implementarse, la interacción entre estos, con la finalidad de guiarnos en los diferentes contextos del desarrollo.

Debido a la complejidad del proyecto, se necesita de una base solida de diseño, en la cual basarse para concretar las ideas y requerimientos. Para conseguir esto, se ha recurrido al patrón de diseño de *software* Modelo-Vista-Controlador(**MVC**), el cual va a permitir definir los componentes necesarios del sistema. Este modelo va a organizar cada parte de la aplicación, haciendo que sea mas claro, en que lugar dentro de la arquitectura se tienen que situar las diferentes funcionalidades a desarrollarse. A continuación se resume la correspondencia entre las funcionalidades de la aplicación y los tres componentes que conforman un proyecto que sigue este patrón.

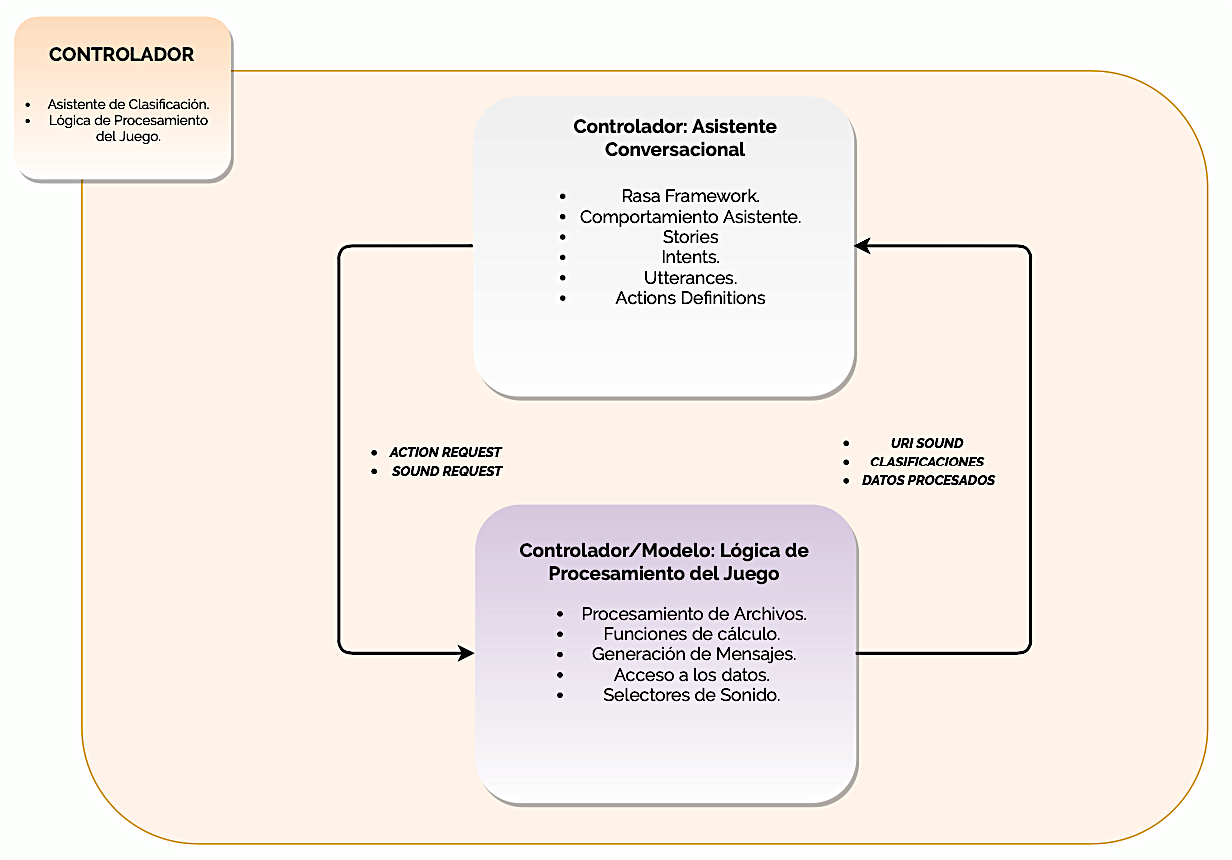


El Modelo-Vista-Controladores un patrón de arquitectura que se caracteriza por la facilidad con la que se puede modularizar el desarrollo de los componentes de un proyecto, además de ofrecer escalabilidad para la expansión de un sistema o modificación de alguno de sus componentes sin afectar al resto.

### MVC – Diseño de las Aplicaciones Lógicas del Controlador del Proyecto

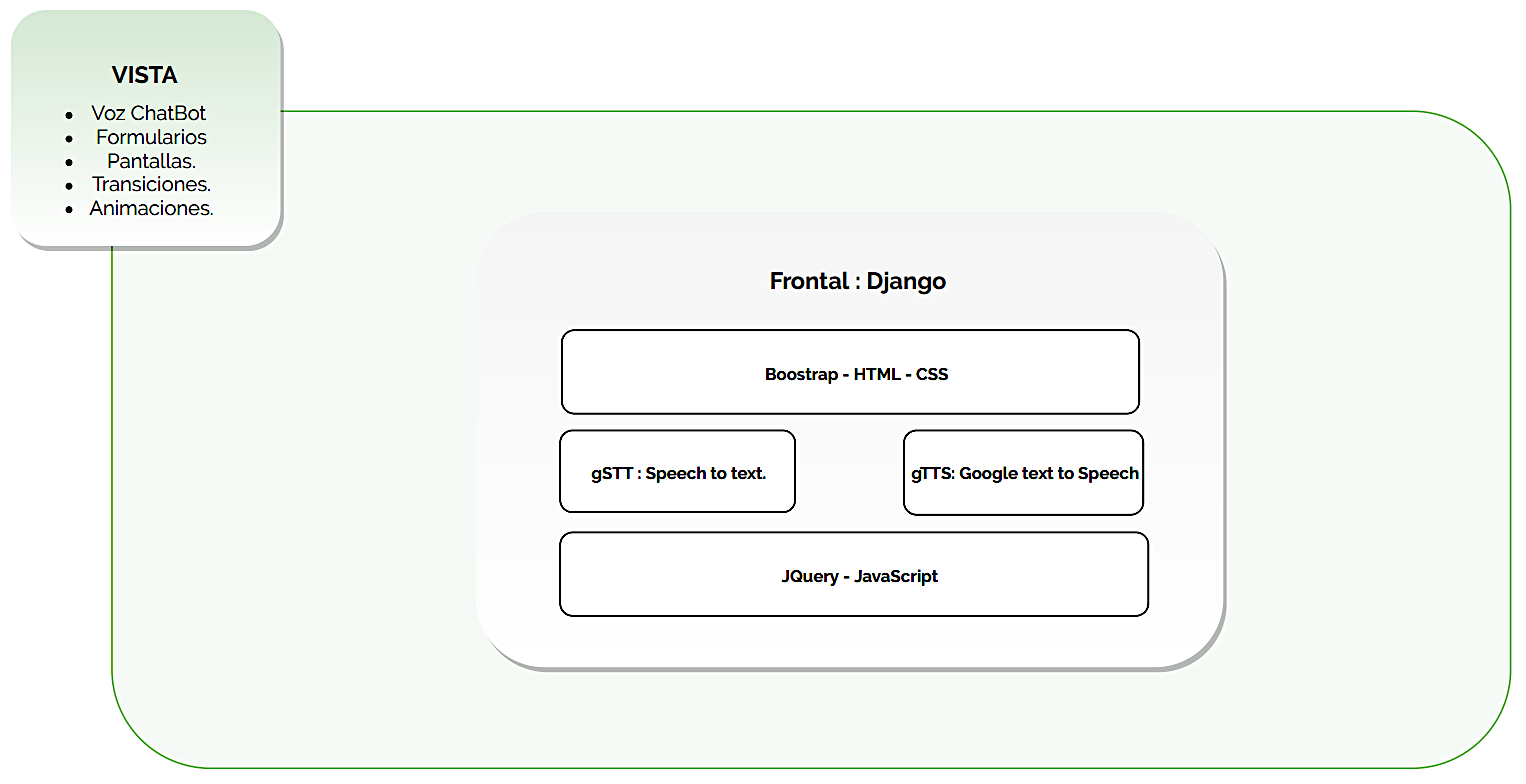
El controlador de la aplicación se encarga de comunicar la vista y el modelo. En este componente del patrón se definen las funcionalidades que realizan los cálculos sobre los datos obtenidos del modelo para enviárselos a la vista, para que así puedan ser presentados al usuario. En palabras mas sencillas, aquí se encuentra la lógica de la clasificación de los sonidos, el cálculo de los resultados del usuario después de una sesión de juego, se definen las acciones que están disponibles en nuestra aplicación a las cuales el usuario podrá llamar mediante la vista.

También se encuentra la inteligencia del asistente, los manejadores que podrán contextualizar las solicitudes, los cuales definen el comportamiento del asistente ante los comandos del usuario, se encuentran también definidas todas las funcionalidades que cubrirán todos los posibles diálogos que pueden ocurrir en una interacción entre el usuario y el asistente. Esta componente será responsable de enviar los sonidos (del cielo) a la vista, para que el usuario los escuche y los clasifique. Lo administración de los sonidos y otros datos procesados como las clasificaciones y el almacenamiento de estos, estarán a cargo de la lógica de procesamiento del juego. La interfaz grafica de la vista y el controlador son los objetivos de desarrollo, por ser los mas componentes mas elementales del proyecto.



### MVC – Esquema de Diseño para el Frontal-Vista del Proyecto

El proyecto necesita de una interfaz con la cual el usuario tendrá que interactuar para comunicarse con el asistente. En este componente se agrupan todas las funcionalidades relacionadas con el desarrollo del *frontend,*  tales como la programación de las pantallas que se mostrarán al usuario, el diseño gráfico, la ubicación de los botones, formularios y los resultados o respuestas que el asistente pueda generar por la solicitud de un usuario. Una parte muy importante de nuestra aplicación es la captura de los mensajes por parte del sistema. En el diseño de este componente del proyecto tendrían que ir ubicados la interfaz de entrada de texto o la implementación de la voz de nuestro asistente. Todo el desarrollo de las funcionalidades relacionadas con la captura de los comandos de voz del usuario y las respuestas por voz del asistente, estarán en esta categoría dentro del patrón que seguirá el proyecto. En cuanto al desarrollo de este componente, se realizará una aplicación web basada en el framework de Django, el cual nos va a permitir crear una aplicación en la que podremos realizar llamadas a REST para la solicitud de datos al asistente virtual. En esta versión del proyecto, contará con una entrada de texto , de manera que el usuario podrá tener un diálogo con el asistente usando el teclado. Para el diseño grafico del frontal, se usará algunos componentes de Bootstrap, para dotar de calidad la maquetación del diseño. Aparte se requieren ciertas funcionalidades para dotar de movimiento la actualización cuando el usuario escriba un mensaje nuevo o reciba uno del asistente. Para conseguir esa funcionalidad usaremos JavaScript. También se emplearán los lenguajes HTML y CSS para modificar los elementos presentes en el frontal.



### MVC – Descripción de la Aplicación Acceso para el acceso de Datos

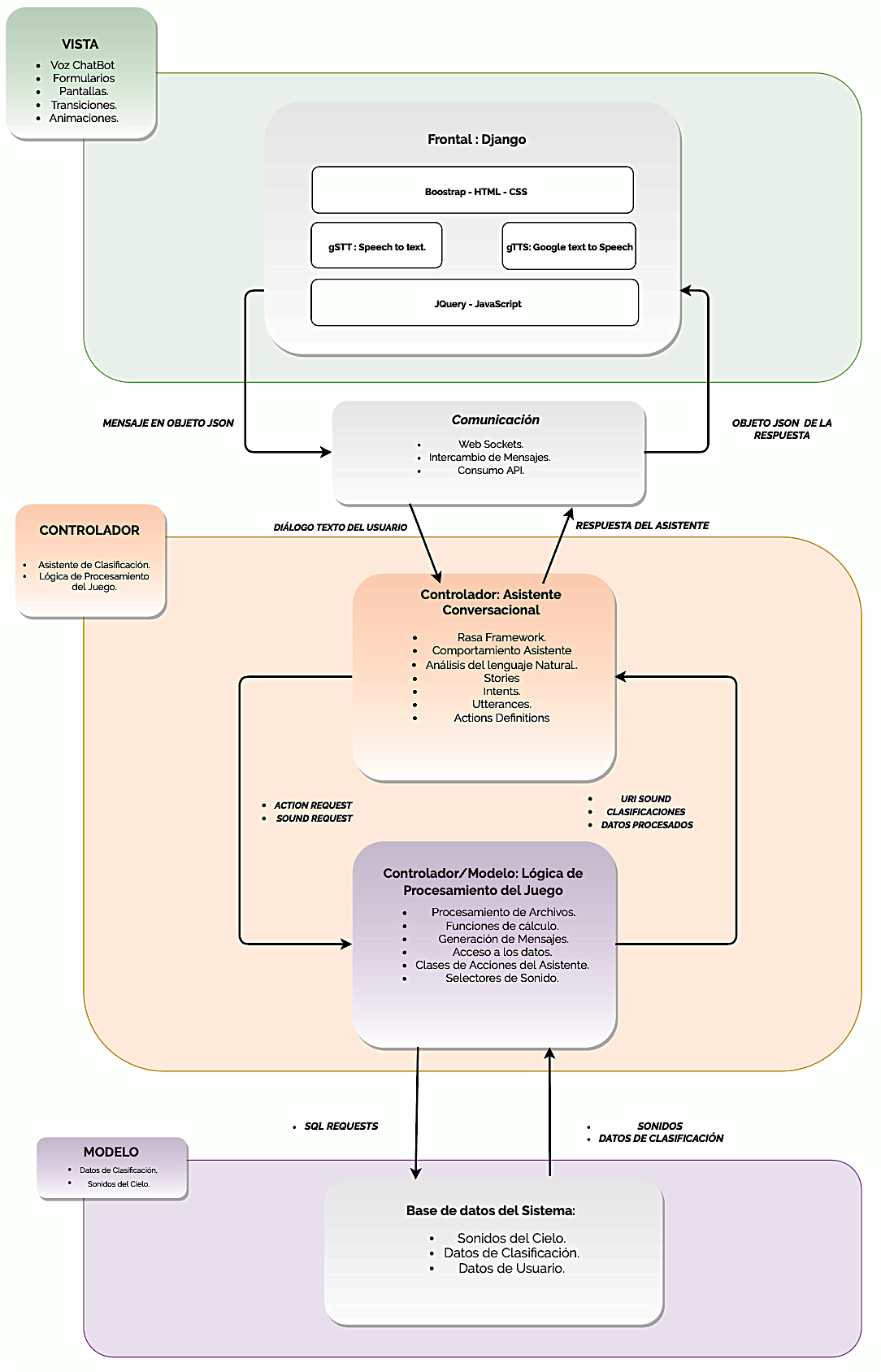
En el modelo del proyecto se situarán aquellas funcionalidades que interactúen con los datos con los que trabajara el asistente. En este módulo, se manipulan y gestionan los accesos a los sonidos del cielo, que serán clasificados dentro del juego, los accesos a la base de datos donde residen los sonidos o cualquier otra información, también la actualización, búsqueda, ordenación, y mantenimiento de los sonidos, los registros de clasificación obtenidos después de una sesión de clasificación, etc. Para una aproximación, según los requerimientos del proyecto con vistas a futuro.

Captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente

En la siguiente imagen se puede ver una vista en conjunto de los todos los componentes que forman parte del sistema.

Ilustración 4. Diagrama general organizativo de los módulos del proyecto según el patrón MVC.



### Requerimientos y herramientas

Analizados todos los requisitos se van a pasar a mencionar todas las herramientas usadas en el desarrollo de este proyecto.

* **Lenguaje de programación Python**:

Para el desarrollo del asistente se ha empleado dicho lenguaje ya que la parte de la lógica del juego se ha desarrollado en Python, además así se mantiene el uso de este lenguaje en otros componentes ya que en el frontal se usará Python también.

* **Rasa:**  
  Este framework es una de las principales que usaremos para la implementación del asistente conversacional.
* **Django:**  
  Es el framework para desarrollo de aplicaciones Web, este se utilizará para marcar el patrón de diseño del frontal del proyecto.
* **PyCharm**:  
  Es uno de los mejores entornos de desarrollo avanzado para Python.
* **WebStorm**:   
  Entorno de desarrollo Web. Este se empleará para el desarrollo de la capa frontal de proyecto.
* **JavaScript:**  
  Ciertos elementos del frontal requieren acciones un poco mas complejas para poder actualizar las vistas. Se emplea **JQuery** para poder hacer solicitudes cuando el usuario desea enviar un mensaje al asistente.
* **JSON:**   
  Se usará en el formato de los mensajes que se envíen entre el frontal y el y la parte del controlador.
* **Bootstrap**:  
  Se empleará para la maquetación de la aplicación ya que ofrece gran adaptabilidad.
* **Postman**:  
  Para las pruebas de envio y recepción de mensajes via REST.
* **Suite Adobe** :  
  Se usará para crear una capa visual amigable para el publico infantil que será usada en el frontal.

## Implementación de los Módulos del Controlador del Proyecto

### Módulo del Comportamiento del Asistente

Para el desarrollo de los requerimientos para la implementación del asistente se ha realizado investigación para conseguir herramientas, *frameworks* que pudieran definir el desarrollo del asistente. Ente las características que se buscan, se necesita que el framework tenga componentes bastante bien definidos, los cuales puedan ser usados para plasmar las necesidades del asistente conversacional del proyecto. También, otra característica es que fuese de código abierto y mas importante, que este tuviera las posibilidades o facilidades para la integración con aplicaciones frontales y aplicaciones o API’s para el consumo de datos. En principio se opto por ChatterBot, pero al presentar Rasa una buena amplitud de herramientas bastante potentes e interesantes se decidió usar este ultimo.

### Rasa Framework

Dentro de la arquitectura planteada, se observa uno de los módulos mas importantes, en el cual se define el comportamiento, las características y funcionalidades del asistente. En el módulo controlador (MVC) se implementan-definen, todas las aplicaciones o funcionalidades que se encargarán de gestionar los mensajes que se reciben: peticiones en una conversación con el asistente, captura de datos del usuario, captura del contexto de los mensajes para que el asistente decida cual es la siguiente acción para realizar, procesamiento para atender las solicitudes del cliente, entregando datos requeridos (en nuestro caso sonidos), etc.

Para el desarrollo de este parte de la aplicación se ha necesitado contar con un conjunto de pautas de diseño que permitan construir el módulo conversacional, es decir la definición de este. Para la realización de esta parte del proyecto se ha decidido emplear Rasa debido a las interesantes características de su arquitectura. Es necesario entender como funciona este framework pues en este se fundamenta la interacción de nuestra aplicación con el usuario, es el corazón del asistente.

Rasa es un framework de código abierto para desarrollar asistentes que emplean inteligencia artificial. En su framework se cuenta con herramientas para la definición datos de entrenamiento de la inteligencia del asistente y también ciertas características ventajosas para su integración, que pasaremos a describir a continuación:

#### Rasa NLU

El componente de compresión natural de lenguaje, en ingles *“Natural Language Understanding*” (NLU), sirve para el reconocimiento de mensajes del usuario, con la finalidad de encontrar una correspondencia con algún *intent.* Los *intents* son todas las posibles frases que el usuario puede llegar a decir/escribir, agrupadas según un tipo de finalidad. Por ejemplo, si alguien quiere saludar podría decir  *“Hola!”, “¿Cómo te va?”, “¿Qué tal todo?”,* todas estas frases se pueden juntar en el concepto *Saludo*, y así cuando un usuario diga alguna de estas frases, entonces el asistente entenderá que el usuario le esta saludando. La NLU de Rasa también tiene la responsabilidad de extraer información del input del usuario para hacerle corresponder con alguna entidad (información clave) con la que rellenara algunos *slots* con información que puede ser necesaria para algún procesamiento. Mas adelante se especifican a detalle todos los elementos de Rasa en el contexto de nuestro proyecto. Este se define como una tubería por donde pasan los mensajes, a la cual se pueden añadir mas “piezas” de procesamiento, para hacer la tubería compleja. Una de estas piezas es *Spacy,*  un componente de lenguajes, que sirve como datos de entrenamiento para el aprendizaje, disponible en varios idiomas. [4]

#### Rasa Core

Es un motor de aprendizaje que usa *machine learning,* el cual define cual será la siguiente accione a tomar en función del mensaje del usuario. Los datos de entrada para este motor y entrenar el asistente, se definen en las *stories,* las cuales se detallan mas adelante.

#### Rasa NLG

El componente de generación de lenguaje natural, en inglés *“Natural Language Generation (NLG)*”, es el componente de Rasa en el que se definen todos los mensajes y acciones que el asistente es capaz de usar para responder ante la solicitud del usuario, estas se definen en las *utterances*.

#### Procesamiento de un mensaje en Rasa

Se presenta de manera general, el procesamiento de un mensaje que llega al sistema y que elementos lo procesan para devolver una respuesta al usuario. [5]

Los pasos son:

* El mensaje se recibe y se pasa a un intérprete, que lo convierte en un diccionario que incluye el texto original, el *intent* y las *entities*  que se pueden encontrar en el mensaje del usuario. Esta parte es manejada por NLU.
* El *tracker* es el objeto que realiza un seguimiento del estado de la conversación. Recibe la información de que ha entrado un nuevo mensaje. Este elemento es importante ya que cualquier aplicación back que se quisiera integrar, usara este objeto para poder acceder al estado del asistente, como pueden ser los valores de *slots* que contienen información de las *entities.*
* La *policy* recibe el estado actual del rastreador y elige qué acción tomar a continuación.
* El *tracker* registra la acción elegida y se envía una respuesta al usuario.

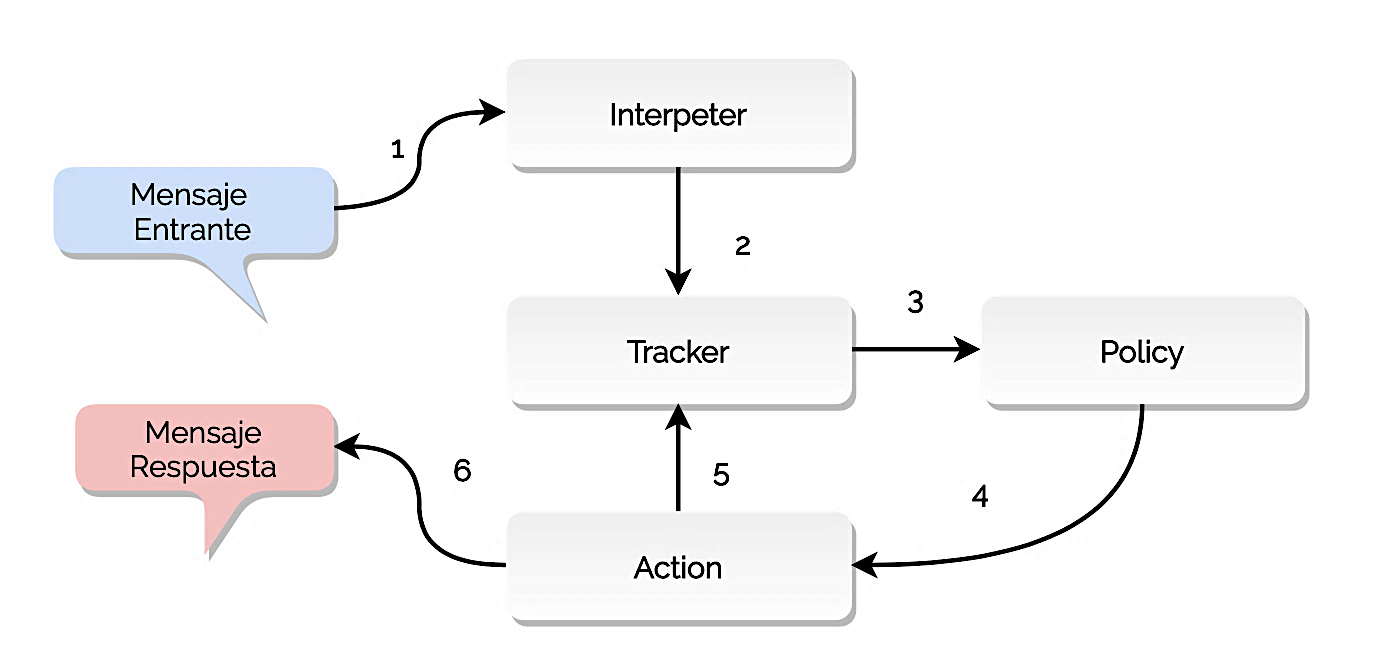


Ilustración 5. Diagrama de pasos básicos del asistente para responder a un mensaje.

La configuración del las *policies* permite, por ejemplo, cuanta cantidad de la conversación tiene que tomar en cuenta el asistente para poder tomar una decisión.

### Componentes Rasa

Se describen con mas detalle los componentes principales del framework Rasa para entender como funcionan y cual es su utilidad, con la finalidad de justificar su uso para la implementación de los requisitos del asistente

#### Intents

Son los mensajes posibles o las conversaciones mas comunes, definidos en el universo (contexto de nuestro asistente) que el asistente puede esperar que ocurran con el usuario. Los *intents* forman parte del componente *Rasa NLU,*  que según la documentación Rasa, la define como una herramienta lenguaje *open-source* para la clasificación de *intents,* para la recuperación de respuestas del usuario, la extracción de entidades que se puede necesitar para definir una conversación o pedir datos del usuario [6]. En la siguiente imagen se muestra la definición del *intent “greet”,*  en la que se reúnen algunas de las posibles frases o palabras que el usuario podría decir cuando quieres expresar un saludo.

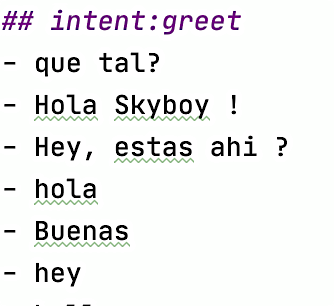
**

Figura 7 . Definición del intent “greet” dentro del archivo nlu.md.

#### Utterances

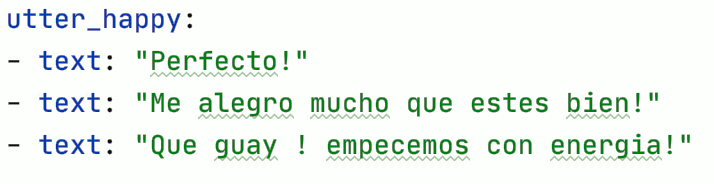
Las *utterances*  son mensajes predefinidos que usa el asistente para poder responder al usuario. Estos mensajes son definidos en el  *Domain*  del asistente; tienen que incluir todas las posibles respuestas que pueden darse cualquier conversación (definida en el dominio del asistente). Las *utterances* forman parte del dominio, que a su vez, forman parte de la componente NLU, que sirve como datos para el entrenamiento de la inteligencia del asistente virtual.

Figura 8 . Definición de la utterance “utter\_happy” y todas sus posibles respuestas, dentro del archivo domain.yml.

#### Actions

Las *actions*  son los nombres de las acciones que serán usadas por el asistente para poder responder al usurario. En detalle, a diferencia de las *utterances,*  las *actions*  es código que se puede ejecutar, para realizar alguna llamada a una base de datos, procesar cálculos, etc. El código asociado a una  *action,* no se encuentra definido en el asistente, sino que son los nombres de los métodos con los cuales el asistente llamará algún modulo Back o aplicación externa para poder ejecutar el código. Todas las *actions*  tienen que estar definidos en el *Domain*  del asistente, ya que forman parte del universo que tiene que conocer y las cosas que puede realizar en cuanto a integración con otras aplicaciones. Al igual que las *utterances* las acciones son mensajes que se envían al usuario, sino que estos contienen información que ha sido procesada por otra o aplicación. Para el desarrollo de las *actions*  de este asistente se ha usado el lenguaje de desarrollo Python y se usado la utilidad de integración que tiene Rasa dentro de su framework, más adelante se detalla la definición de estos. Se muestra a continuación el empleo de una *action*  en concreto, en el dominio y *Imagen que contiene cuchillo

Descripción generada automáticamente*como parte de una *storie.*

Figura 9 . Definición del action “action\_classifying” dentro del dominio.

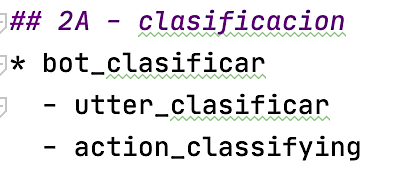


Figura 10 . Uso del action “action\_classifying” dentro del la Storie “clasificación”.

#### Stories

En Rasa, las *stories* definen el comportamiento del asistente ante ciertos comandos enviados a los usuarios. Con ellos podemos definir los diferentes caminos que puede tomar una conversación. En una *storie*  se definen los *intents* y *utterances*  que pueden formar parte de una conversación, con estos elementos podemos recrear las diferentes conversaciones para las cuales el Asistentetiene que estar preparado, siendo así capaz de poder reconocer los comandos del usuario (que se correspondan con las *intents* definidas en la capa NLU) y respondiendo a ellas mediante las acciones las cuales pueden ser simples respuestas (frases) o el desencadenamiento de llamadas al servidor-modulo de acciones en el que están definidos las funciones de clasificación. Dentro de la arquitectura de Rasa, las *stories* forman parte del *CORE***,**  que es el componente de Rasa que reúne todas las características para manejar los contextos y administrar el aprendizaje automático mediante *machine learning.*  Las *stories,*  son la información que usara el *CORE*  para aprender. Gracias a este enfoque de Rasa, el asistente podrá entender lo que dice el usuario y además aprender de esa entrada, lo que hará que el asistente pueda manejarse en situaciones que nosotros no hallamos definido manualmente. Es decir, mediante la experiencia, el asistente podrá definir reglas nuevas, así entonces entendemos que es una gran ventaja ya que no tendremos que pensar que solo los creadores del asistente podrán crear todas las reglas posibles, sino que un asistente pensado en Rasa también lo hará de manera automática. A continuación se describen todas las *stories*  definidas para el funcionamiento para el manejo de las conversaciones entre el Asistentey el usuario.

#### Slots

Captura de pantalla de un celular con letras

Descripción generada automáticamenteLos *slots*  son un componente importante de Rasa, en estos se almacena la información clave, que se ha extraído del dialogo del usuario, por ejemplo, puede ser su nombre o una respuesta a alguna pregunta que el asistente le hace al usuario. En otras palabras, sirven de memoria de almacenamiento de datos a los cuales puede acceder una tercera aplicación para saber el estado del asistente. Estos están ligados a las *entities,*  las cuales están ligadas a piezas de información que puede proveer el usuario, las cuales pueden ser almacenadas en los *slots.*

Figura 11. Definición de algunas entidades y slots dentro de dominio.

#### Dominio

El dominio es en donde se encuentran todos los componentes que definen el comportamiento del asistente, en este se encuentran las *entities, utterance, intents, slots y actions.* Cada vez que se realice algún cambio, hay que asegurarse de que este se vea reflejado en este componente. Su implementación se encuentra en el archivo “domain.yml”

##### Implementación de Rasa para el Asistente – Stories

Se muestran las definiciones de los principales componentes de Rasa, que se han definido para la implementación del comportamiento del asistente ante las posibles conversaciones que pueden ocurrir. En este ejemplo se muestra la implementación para iniciar el diálogo con el usuario, el cual implicar un saludo por parte del asistente, la captura del nombre del usuario y la presentación de las opciones posibles que puede realizar el usuario en el sistema.

###### Ejemplo de implementación de Rasa

Captura de pantalla de un celular

Descripción generada automáticamente

Figura 12. Definición de la Storie para empezar el dialogo con el asistente, en el archivo stories.md

Una captura de pantalla de un celular con letras

Descripción generada automáticamente

Figura 13. Intents posible, asociados a la storie anterior para empezar el dialogo con el asistente, en el archivo nlu.md

### Módulo-Aplicación de Acciones del Juego

Como se ha mostrado ya en el planteamiento de la arquitectura, el asistente tiene que ser capaz de poder ofrecer ciertas funcionalidades que requieren de procesamiento, cálculos y el acceso a datos con los que trabajará el sistema. En nuestro caso, tenemos que ofrecer todas las posibles acciones con las que cuenta la aplicación para la clasificación de los sonidos. Hay que recordar que esta parte es excluyente de la implementación del comportamiento del asistente, ya que , es otra “aplicación independiente” con servicios que están en otro servidor, el cual funcionará para que el asistente pueda hacer uso de estos cuando en la conversación con el usuario, este identifique alguna solicitud de acción (p. ej. “*dame un sonido),* desencadenando llamadas estos servicios cuando sea necesario.

En este apartado se explican estos servicios. Cada uno de estos, o la conjunción de algunos, cumplen con las funcionalidades requeridas para definir los caminos que puede tomar la clasificación o el entrenamiento, y poder brindar procesamiento, etc. Rasa ofrece un endpoint en el que podremos definir todas nuestras acciones. Este endpoint de acciones se encuentra dentro del archivo “actions.py”, en donde están todas las clases las cuales se corresponden con las acciones disponibles para el asistente.

#### Implementaciones de funcionalidades

A continuación se describen la funcionalidad de las clases implementadas:

Las siguientes clases tienen el objetivo de seleccionar un sonido en concreto para el entrenamiento y para el aprendizaje de los tipos de sonidos. Como sabemos, podemos encontrar 5 tipos de sonidos. En el funcionamiento de estas clases, se selecciona un sonido de los disponibles, se analiza su tipo; con esa información lo que se hace es enviarla al asistente para que este se la devuelva al frontal en forma de URI, empaquetado dentro de un mensaje de tipo JSON, el cual va identificado con la clave “soundUri”.



Figura 14. Generación de un mensaje JSON que contiene la URI del sonido.

También una funcionalidad importante es la modificación del estado del asistente, ya que estas clases son capaces de modificar *slots* de información. Retornará mediante el objeto *tracker*, el valor de uno de los Slots llamado 'sonido\_actual', de esa manera, el asistente sabrá sobre de que tipo de sonido se esta conversando con el usuario. Las clases que cumplen esta funcionalidad son:

* **ActionUnderdenseSelector** **:** En esta acción se seleccionará los sonidos de *tipo 1,*  los cuales corresponden con los meteoritos que se clasifican como *underdense.*
* **ActionMSelector**: En esta acción se seleccionará los sonidos de *tipo 2*, los cuales corresponden con los meteoritos que se clasifican como tipo M.
* **ActionLongOverdenseSelector:** En esta acción se seleccionará los sonidos del tipo 3 los cuales corresponden con los meteoritos que se clasifican como *largos overdenses.*
* **ActionMediumOverdenseSelector:** En esta acción se seleccionará los sonidos del tipo 4 los cuales corresponden con los meteoritos que se clasifican como *overdense medio.*
* **ActionShortOverdenseSelector:** En esta acción se seleccionará los sonidos del tipo 5 los cuales corresponden con los meteoritos que se clasifican como *overdense corto.*
* **ActionSoundCheck:** Cuando el usuario responde ante la pregunta sobre la clasificación de sonidos en el entrenamiento, esta acción se encarga de verificar si la respuesta es correcta o no, en ambos casos envía un mensaje al usuario, con una breve descripción del usuario y con el resultado de la clasificación.
* **ActionSoundCheck:** Cuando el usuario responde ante la pregunta sobre la clasificación de sonidos en el entrenamiento, esta acción se encarga de verificar si la respuesta es correcta o no, en ambos casos envía un mensaje al usuario, con una breve descripción del usuario y con el resultado de la clasificación.
* Captura de pantalla de un celular

  Descripción generada automáticamente**SoundListing:** Esta clase sirve para poder listar los sonidos de un directorio.  
  Devuelve una lista con todos los sonidos (y sus localizaciones) y crea un archivo de texto en el *path* especificado, con la lista de todos los sonidos existentes y su clasificación media. Además, a cada archivo de audio, se le asocia un archivo de texto. En la primera línea de dicho archivo se encuentra el numero de veces que ha sido clasificado ese sonido y en las siguientes líneas las clasificaciones con el autor de la clasificación en cada línea.

Figura 14. Archivo de clasificación asociado al archivo de audio “sound1.wav”

Si este archivo de clasificación no existe, entonces lo crea e inicializa a 0 el numero de veces que se ha clasificado ese sonido, poniendo dicha información en la primera línea en dicho archivo de clasificación. En caso de que el archivo ya existiera, lo único que se hace es realizar la lectura de la primera línea de dicho archivo para obtener el numero de veces que ha sido clasificado el sonido asociado. Además crea un objeto de tipo lista, en el que se devuelve las

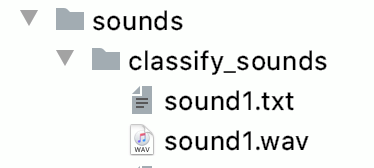


Figura 14. Archivo de clasificación asociado al archivo de audio “sound1.wav”

localizaciones de los sonidos, el número de clasificaciones de cada sonido extraído de cada archivo de clasificación y la ubicación del archivo de texto donde esta la lista de sonidos disponibles para clasificar. El tipo de archivos de texto se crean para la clasificación se ubicará siempre en el mismo directorio donde se encuentra el sonido.

* **ActionSaveClasification:** Esta acción se ejecuta para guardar la respuesta de clasificación, que una vez que se el usuario haya dado una respuesta,la cual se encuentra en el slot\_respuesta. Para ello accedera mediante el *tracker* al estado de asistente y extraerá la respuesta de dicho *slot*. Dicha respuesta se almacenara, en el archivo de clasificaciones asociado a ese sonido, se aumentará en uno el numero de clasificaciones del sonido (la primera línea) y junto con la clasificación se pondrá el nombre del usuario que clasificó el sonido.
* **ActionHelloWorld:** Esta clase se ejecuta cuando la acción “action\_hola\_mundo” es reconocida por el asistente. Este se encarga de almacenar el nombre del usuario y de dar la bienvenida al usuario al sistema.
* **ActionClassifying:** Esta clase se ejecuta cuando el asistente reconoce en la conversación con el usuario la *action*  “action\_classifying”. La funcionalidad de esta es escoger un sonido para ser clasificado y enviárselo al asistente para que este pueda enviárselo al frontal de la aplicación y el usuario lo pueda escuchar. Para enviarle el sonido al usuario, este se encapsula en un objeto JSON para que el frontal pueda presentarlo en un reproductor. Para hacer la selección se sigue la política, en la que siempre se devuelve el sonido que menos veces se ha clasificado, de esta manera nos aseguramos de que ningún sonido se quede sin clasificar, de tener en cuenta de los nuevos sonidos que lleguen; en caso de que todos los sonidos tengan el mismo numero de clasificaciones, entonces se escoge cualquiera de manera aleatoria. Para el funcionamiento de esta clase, se hace un llamado a la clase soundListing, la cual devolverá todos los sonidos y sus datos de clasificación como se ha especificado antes para ser usados en esta clase.

### Integración entre el Módulo de Acciones y el Asistente

Es necesario entender como interactúan el modulo de acciones y el modulo conversacional que define al asistente, ya que la experiencia que tendrá el usuario al realizar un comando depende de llamadas del asistente hacia las funcionalidades implementadas en las acciones. En este apartado se describe la relación entre las funcionalidades implementadas en el módulo de acciones y el módulo del asistente.

#### Función Aprendizaje y Función de Entrenamiento

Esta funcionalidad del controlador del MVC (módulo asistente y modulo de acciones en conjunto), es la que se encarga de poner a disposición del asistente, los tipos de sonidos con sus descripciones, que le servirán al usuario para poder aprender que tipos de sonidos existen o entrenar.

Las implementaciones de las acciones asociadas se encuentran dentro del archivo “actions.py”. Estos hacen la selección de sonidos entre otros procesamientos. Para esta función se usan las siguientes acciones y sus correspondientes elementos definidos en el módulo del asistente definido por Rasa, dentro de los archivos “domain.yml” y en el archivo “stories.md”.

Tabla 1. Asociación entre actions y clases asociadas para la funcionalidad de Aprendizaje

|  |  |
| --- | --- |
| ***Action* definida en el asistente** | **Acciones en el Modulo** |
| action\_underdense\_sample | ActionUnderdenseSelector |
| action\_m\_sample | ActionMSelector |
| action\_long\_over\_sample | ActionLongOverdenseSelector |
| action\_med\_over\_sample | ActionMediumOverdenseSelector |
| action\_short\_over\_sample | ActionShortOverdenseSelector |

Para la función de entrenamiento, se usan todas las anteriores, además de la clase ActionSoundCheck asociado al *action “*action\_check\_sound”.

#### Función Clasificación

En la funcionalidad de clasificación, lo que se desea es poner a disposición del usuario, sonidos para que este los pueda, clasificar. Cuando un usuario realiza una solicitud de clasificación, el asistente intentará contactar con el modulo de acciones para ejecutar las clases destinadas para este fin. Cuando el asistente haya ejecutado con éxito la llamada a dichas funcionalidades, entonces estas le responderán al usuario mediante el objeto *tracker*  del framework de Rasa, enviándole el sonido al frontal en formato de recurso URI empaquetado en un mensaje JSON. Cuando el usuario haya respondido con su clasificación para un sonido dado, este se guardará en un *slot* en el modulo del asistente llamado “respuesta”, entonces el asistente podrá ejecutar otra acción que se encargará de extraer, mediante el *tracker* la clasificación del usuario del *slot*  “respuesta”. Dicho *slot*  esta definido en el archivo “domain.yml”. Cuando la acción haya extraído la información de dicho *slot*  entonces este guardará esta información en el archivo de clasificación asociado al sonido actual que se esta clasificando, insertando dicha información en una linea, al final. También se guardará en la misma línea junto con la clasificación, el nombre de la persona que realizo dicha clasificación. Las clases de las acciones y los *actions* usados para esta funcionalidad son los siguientes:

Tabla 2Asociación entre actions y clases asociadas para la funcionalidad de Clasificación.

|  |  |
| --- | --- |
| ***Action* definida en el asistente** | **Acciones en el Modulo** |
| **action\_classifying** | **ActionClassifying** |
| **action\_save\_clasificacion** | **ActionSaveClasificacion** |
| **action\_classifying** | **SoundListing** |

#### Pruebas de Integración

En este apartado se muestran las pruebas de integración entre el modulo del asistente y el modulo de acciones, para conformar entre los dos el modulo Controlador, el cual se puede enmarcar dentro del Modelo Vista Controlador.

Antes que nada, tendremos que situarnos dentro de la carpeta “mvc\_control\_bot\_juego” y ejecutar el siguiente comando: “rasa train”, el cual servirá para poder entrenar el modelo de nuestro asistente con el dominio y todos los datos definidos anteriormente (*intents, actions, utterances,* etc). La terminal tendrá el siguiente aspecto:

Imagen que contiene guitarra

Descripción generada automáticamente

Figura 14. Progreso de entrenamiento de nuestro asistente.

Imagen que contiene sostener, tabla, gente, mujer

Descripción generada automáticamenteDespués de haber entrenado nuestro modelo, nos tiene que aparecer el siguiente mensaje de confirmación, mostrándonos la ubicación del archivo donde se encuentra el modelo de nuestro asistente que ya se puede usar:

Mensaje de confirmación de la generación del modelo .

Una vez entrenado ahora solo queda ejecutar el Shell de Rasa para poder interactuar con el asistente. Para eso ejecutamos el comando “rasa Shell” y si no ha habido problemas la terminal presentara el siguiente estado:

Una captura de pantalla de un celular con letras

Descripción generada automáticamente

Figura 1. Rasa Shell donde se puede interactuar con el asistente.

Captura de pantalla de un celular con letras

Descripción generada automáticamenteCuando el asistente este esperando a los comandos del usuario, el siguiente paso es activar el servidor de acciones que Rasa provee, en el que se han desarrollado todas las clases necesarias para implementar las acciones de procesamiento. Para ello se escribirá en otra terminal: “rasa run actions” y cuando el servidor este activo se mostrará el siguiente aspecto en la terminal, listándose todos los servicios disponibles el modulo de acciones:

Figura 1. Rasa Shell donde se puede interactuar con el asistente.

##### Pruebas

* Saludo y reconocimiento: del usuario por parte del modulo de acciones: El usuario saluda al asistente, este captura su nombre y lo usa para poder seguir el diálogo

Imagen que contiene pájaro

Descripción generada automáticamente

* Función de Aprendizaje: Para esta prueba se ha arrancado el servidor y se ha conectado Postman para realizar una operación POST con el comando de aprender.  
  Como se puede ver, el mensaje esta estructurado en formato JSON.

Una captura de pantalla de una red social

Descripción generada automáticamente

La conectividad queda asegurada con la prueba anterior, sin embargo, podemos realizar la propia terminal que Rasa prevee para hacer la misma prueba. Esto es útil, en caso de que no se pueda instanciar la aplicación externa que quiera consumir los servicios de nuestro asistente.

Captura de pantalla de un celular con letras

Descripción generada automáticamente

* Pruebas de Entrenamiento 1: Se le pide entrenar, el asistente hace una llamada al modelo de datos para pedirle un sonido. El asistente espera una respuesta del usuario. El usuario contesta mal.

Captura de pantalla de computadora

Descripción generada automáticamente

En este caso, el sistema le ofrece la respuesta para poder corregir al usuario, con información del tipo de sonido.

* Prueba de Entrenamiento 2: El usuario acierta en la clasificación del tipo de sonido.

Captura de pantalla de un celular

Descripción generada automáticamente

* Prueba de Clasificación: El usuario acierta en la clasificación del tipo de sonido.
* Prueba Información de Ayuda en Clasificación: El usuario pide ayuda cuando no sabe clasificar un sonido.
* Prueba Información de Ayuda General del Sistema El usuario pide ayuda cuando no sabe clasificar un sonido.

### Módulo-Aplicación Frontal Web del Proyecto

La interfaz de la aplicación es un modulo importante, ya que es con lo que el usuario final va a interactuar para clasificar los sonidos del cielo.

Para que nuestro Asistentepueda comunicarse con el usuario, necesitamos dotarle de voz. Como se ha dejado señalado en la arquitectura de la aplicación, las funcionalidades como la que capta la voz del usuario, la que reproduce la voz del Asistentese encontrara dentro de una aplicación web.

rasa run para poder conectar al servicio del chatbot

#### Frontal Web del Proyecto – DJango

En este apartado se describe un diseño de interacción, un diseño gráfico que incluya todos los elementos de una aplicación, tales como: botones, entradas de texto, colores.

#### Audio

Como se sabe, estas acciones (clases) se encurtan dentro de la implementación, en el archivo “actiones.py”, perteneciente al modulo back; si tomamos como referencia el diseño, estas acciones no deberían reproducir los sonidos. Es el cliente front quien se debería encargar de poder capturar o transmitir, correspondientemente, la voz del usuario y la voz del asistente.

A pesar de que no compete el desarrollo de la interacción de la parte front con la parte back, se sugiere altamente tomar en cuenta la siguiente idea para su desarrollo:

Cuando le asistente entienda que tiene que el usuario desea un sonido para poder practicar o para clasificarlo, el asistente deberá ejecutar alguna acción para obtener dicho sonido. Pero la cuestión aquí es ¿ Cómo le paso el sonido al cliente ?, pues bien, no devolveremos el sonido como tal, sino una referencia de él.

Para las personas que desean implementar esto, deberán tener conocimientos de servicios WEB o REST. Al tener la referencia del sonido, es decir, una URI del recurso, la cual es una cadena de texto que el asistente puede devolver sin ningún problema.

Con la URI del recurso, tendremos el acceso al sonido asociado a esa URI. Sobre esta URI podremos ejecutar el método GET y traer el sonido hacia el cliente para poder reproducirlo. Claro esta, que el método GET habrá que ejecutarlo dentro de la implementación WEB usando tecnologías JavaScript o Node. Ya teniendo el sonido en local, podremos reproducirlo usando un reproductor HTML. Esto ha sido una aproximación para dar una idea y no estar perdido del todo.

##### Speech to Text

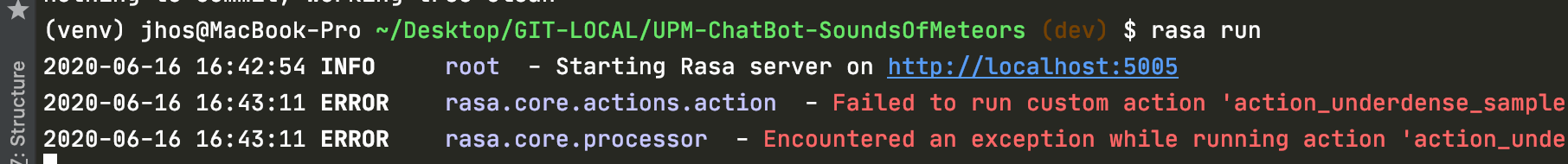
##### Google Speech to Text

### Interacción entre el módulo de Conversacional y el Frontal

#### Diseño de mensajes entre módulos

Imagen que contiene captura de pantalla, tabla, teléfono

Descripción generada automáticamente



Una captura de pantalla de una red social

Descripción generada automáticamente

Una captura de pantalla de una red social

Descripción generada automáticamente

#### Procesamiento de Mensajes en el Frontal

Las siguientes dependencias se necesitan instaladas en el ordenador donde se desee ejecutar nuestro Asistentey así pueda funcionar correctamente:

* *from* pygame *import* mixer # Load the popular external library

Este nos sirve para reproducir los sonidos del cielo en la parte Back

<https://stackoverflow.com/questions/20021457/playing-mp3-song-on-python>

#### Pruebas de integración

## Funcionamiento del Sistema

### Comandos y Funcionalidades

#### Bienvenida

#### Aprendizaje sobre los Sonidos del Cielo

#### Entrenamiento

#### Clasificación

#### Despedida

# Resultados y conclusiones

Resumen de resultados obtenidos en el TFG. Y conclusiones personales del estudiante sobre el trab

Como usar

Mejoras

Vision de funcionalidades para el futuro

# Bibliografía

|  |  |
| --- | --- |
| [1] | AstroCantabria. [En línea]. Available: https://astrocantabria.org/?q=estrellas-fugaces. [Último acceso: 1 6 2020]. |
| [2] | Apple, «Siri de Apple,» [En línea]. Available: https://www.apple.com/siri/. [Último acceso: 5 6 2020]. |
| [3] | Lifegate, «Lifegate,» [En línea]. Available: https://www.lifegate.com/domotics-home-automation. [Último acceso: 5 6 2020 ]. |
| [4] | spaCy, «spaCy,» spaCy, [En línea]. Available: https://spacy.io/. [Último acceso: 7 Jun 2020]. |
| [5] | Rasa, «Rasa Docs,» [En línea]. Available: https://rasa.com/docs/rasa/user-guide/architecture/. [Último acceso: 6 5 2020]. |
| [6] | Rasa, «Rasa,» [En línea]. Available: https://rasa.com/docs/rasa/nlu/about/. [Último acceso: 10 Jun 2020]. |
| [7] | Jhosef, Juan, Lima : Planeta, 2009. |

[6]

# Anexos

Este capítulo es opcional, y se escribirá de acuerdo con las indicaciones del Tutor.