

ChatBot para Clasificar Sonidos del Cielo

Sky Sounds Chatbot Classifier

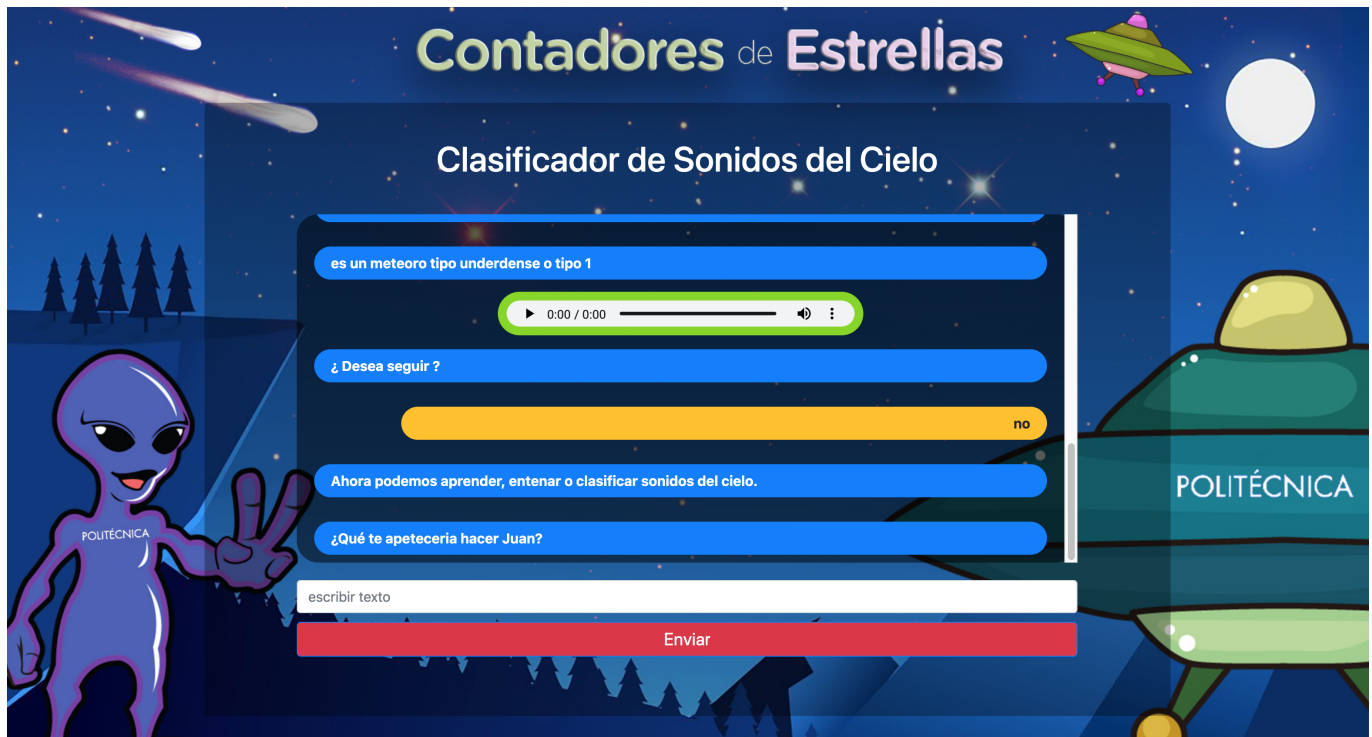



Tabla de contenidos

- [Introducción](#)
- [About](#)
- [Arquitectura](#)
- [Instalación](#)
- [Características](#)
- [Team](#)


Introducción

- El proyecto tiene como objetivo el diseño y desarrollo de una arquitectura para la implementación de una aplicación, que tiene la principal funcionalidad de la clasificación de sonidos del cielo mediante un juego orientado al público infantil.

About

-  This chatbot is being developed by [Jhosef A. Cardich Palma](#) as part of his Final Degree Project (TFG) for the [Polytechnic University of Madrid](#) as a part of its Computer Science Degree on the [Higher Technical School of Computer Engineers](#). The **Sky Sounds Chatbot Classifier** is an application that offers an entertaining experience to classify sounds of the sky, bringing the general public closer to

science. This application is part of [Star counter](#), a large project in collaboration with the **Instituto Astrofísico de Canarias**.

-  Este proyecto esta siendo desarrollado por [Jhosef A. Cardich Palma](#) como el trabajo de fin de carrera en la [Universidad Politécnica de Madrid](#) para el grado de Ingeniería Informática en la [Escuela Técnica Superior de Ingenieros Informáticos](#). La aplicación **Chatbot Clasificador de Sonidos del Cielo** tiene como objetivo la realizacion de un bot para el publico infanti, el cual podra clasificar los sonidos del cielo interactuando con el asistente virtual. Esta aplici3n forma parte de un proyecto mas grande. [Contadores de Estrellas](#) es un proyecto realizado en colaboraci3n con el **Instituto Astrofísico de Canarias**.



Arquitectura

La implementaci3n general se ha seguido un patr3n **Modelo Vista Controlador (MVC)** , para definir los componentes y sus interacciones.

Instalaci3n

- En este repositorio se encuentra el proyecto, en el que se incluye el entorno virtual con el que se ha trabajado en local (carpeta venv) sin embargo, en esta carpeta, se encuentra otra llamada "lib" (/venv/lib), la cual no se esta sincronizando con el repositorio aqui (pero si usa en local), debido a que esta carpeta tiene dos modulos que pesan demasiado, por ello es recomendable, si deseas hacer funcionar esto:

Configuraci3n Entorno

- Antes que nada, tenemos que instalar [RASA](#)

Instalaci3n RASA

```
$ pip3 install rasa
```

- Una vez clonado el proyecto, hay que hacer una serie de comprobaciones. Se ha usado el entorno Pycharm para el desarrollo del asistente, el siguiente paso hace referencia las opciones de Pycharm. Hay

que asegurarnos que estamos usando la version de Phyton 3.7 y que el entorno venv esta configurado correctamente, también lo haremos cuando hayamos instalado RASA. Podemos encontrar las configuraciones en :

Abrir el proyecto con Pycharm:

```
> Pycharm > preferencias ..
```

Configuración del módulo del juego y del ChatBot

Modulo Back - Lógica del Juego

Servicios RASA

Otra parte esencial del proyecto es la lógica del juego, para ello se ha desarrollado una aplicación en Phyton, integrada en framework de RASA, la cuál define las funcionalidad de acceso a los sonidos, la clasificación, almacenamiento de los datos de clasificación, y otros procesamientos independientes de la parte conversacional que es el bot. Estos servicios serán usados por el bot cuando este reconozca un comando por parte del usuario.

- Para iniciar los servicios solo tenemos que situarnos en el directorio anterior y ejecutar el siguiente comando:

```
$ rasa run actions...
```

Módulo Back - ChatBot

- Para entrenar nuestro modelo nos situamos en el siguiente directorio:

```
"/UPM-ChatBot-SoundsOfMeteors/mvc_control_bot_juego"
```

- Estando ya en el directorio, ejecutamos el siguiente comando para entrenar el modelo:

```
$ rasa train
```

- Una vez que nuestro rasa ha terminado de formar el universo de nuestro bot este se encuentra ya listo para poder usarlo. Entonces podemos conversar con el bot conectando un frontal web o la consola de comandos. La segunda opción es la mas inmediata. Para poder comunicarnos con nuestro bot via consola, ejecutamos el siguiente comando en una terminal. Este inicia el servidor RASA y ademas nos da una terminal de entrada para poder comunicarnos via texto con nuestro bot:

Para iniciar el servidor rasa donde corra nuestro bot:

```
$ rasa shell
```

- Si el comando se ha realizado con éxito, se mostrara un mensaje como este:

```
jhos@MacBook-Pro ~/Desktop/GIT-LOCAL/UPM-ChatBot-SoundsOfMeteors/mvc_control_bot _juego (master) $ rasa shell
2020-06-25 19:36:31 INFO     root - Connecting to channel 'cmdline' which was specified by the '--connector' argumen
t. Any other channels will be ignored. To connect to all given channels, omit the '--connector' argument.
2020-06-25 19:36:31 INFO     root - Starting Rasa server on http://localhost:5005
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input ->  hola
Buenas, como va todo ?
Your input ->  
```

Integración con APPS externas

En el caso de la integración con aplicaciones externas lo que tenemos que hacer para exponer los servicios de nuestro tenemos que seguir el siguiente comando . Se pueden realizar pruebas sobre mensajes con el software postman.

Exponer los servicios de nuestro bot

- Nos situamos siempre en el directorio donde hemos entrenado nuestro bot y ejecutamos:

```
$ rasa run
```

Consumir mediante POSTMAN

- En el caso de que el servidor se haya iniciado sin nignun problema, el aspecto de la terminar es la siguiente:

```
jhos@MacBook-Pro ~/Desktop/GIT-LOCAL/UPM-ChatBot-SoundsOfMeteors/mvc_control_bot _juego (master) $ rasa run
2020-06-25 19:48:30 INFO     root - Starting Rasa server on http://localhost:5005
```

Consumir Mediante un Navegador - Cliente Web

Exponer los servicios de nuestro bot con Extra Google Chrome

- Si queremos conectar nuestro asistente desde un navegador, es decir, conectar el frontal con el controlador/modelo del proyecto, hay que usar el siguiente comando que desactivara ciertas características de seguridad de RASA las cuales entran en conflicto con los navegadores. Esto se tiene que hacer al usar el frontal creado para el proyecto. Ejecutar el siguiente comando:

```
rasa run --enable-api --cors "*"
```

Consumir servicios

- Podremos consumir los servicios en la siguiente URI (**POST**):

```
http://localhost:5005/webhooks/rest/webhook
```

Una prueba de operación **POST** en postman, donde se ve el mensaje enviado y la contestación del asistente. Hay que notar un detalle, y es que el la ultima parte del mensaje enviado por el bot ha sido generado por la aplicación del juego definida en las acciones de RASA, las cuales están activas porque las hemos activado previamente (Modulo Back - Lógica del Juego) .

The screenshot shows a REST client interface with a top bar containing three tabs, each with a 'POST' method and a URL: 'http://localhost:5005/webhoo...', 'http://localhost:5005/webhoo...', and 'http://localhost:5005/webhoc...'. Below this is a section titled 'Untitled Request'. The main area is divided into two parts. The top part shows the request configuration: a dropdown menu set to 'POST', a text field with the URL 'http://localhost:5005/webhooks/rest/webhook', and a series of tabs: 'Params', 'Authorization', 'Headers (9)', 'Body' (selected), 'Pre-request Script', 'Tests', and 'Settings'. Under the 'Body' tab, there are radio buttons for 'none', 'form-data', 'x-www-form-urlencoded', 'raw' (selected), 'binary', and 'GraphQL', followed by a 'JSON' label and a dropdown arrow. The body content is a JSON object:

```
1 {
2   "sender": "Rasa",
3   "message": "clasificar"
4 }
```

 The bottom part of the interface shows the response configuration: tabs for 'Body' (selected), 'Cookies', 'Headers (6)', and 'Test Results'. Below these are buttons for 'Pretty', 'Raw', 'Preview', and 'Visualize', followed by a 'JSON' dropdown and a refresh icon. The response body is a JSON array:

```
1 [
2   {
3     "recipient_id": "Rasa",
4     "text": "Muy bien! vamos a clasificar"
5   },
6   {
7     "recipient_id": "Rasa",
8     "text": "Estoy escogiendo un sonido para ti..."
9   },
10  {
11    "recipient_id": "Rasa",
12    "text": "Dime, ¿De que tipo crees que es el siguiente sonido ?"
13  },
14  {
15    "recipient_id": "Rasa",
16    "custom": {
17      "soundUri": "sounds/classify_sounds/sound9.wav"
18    }
19  }
20 ]
```

En ese momento los servicios estarán disponibles para que nuestro asistente pueda llamarlos si reconoce alguno en la conversación con el usuario.

Configuración del Frontal

Por la arquitectura propuesta, se ha desarrollado una aplicación frontal basada en el framework Web de Django.

- Para poder hacer funcionar el frontal de la aplicación tenemos que situarnos en el directorio:

```
/UPM-ChatBot-SoundsOfMeteors/mvc_vista_fronal/mysite
```

- Hay que arrancar el servidor de la siguiente manera:

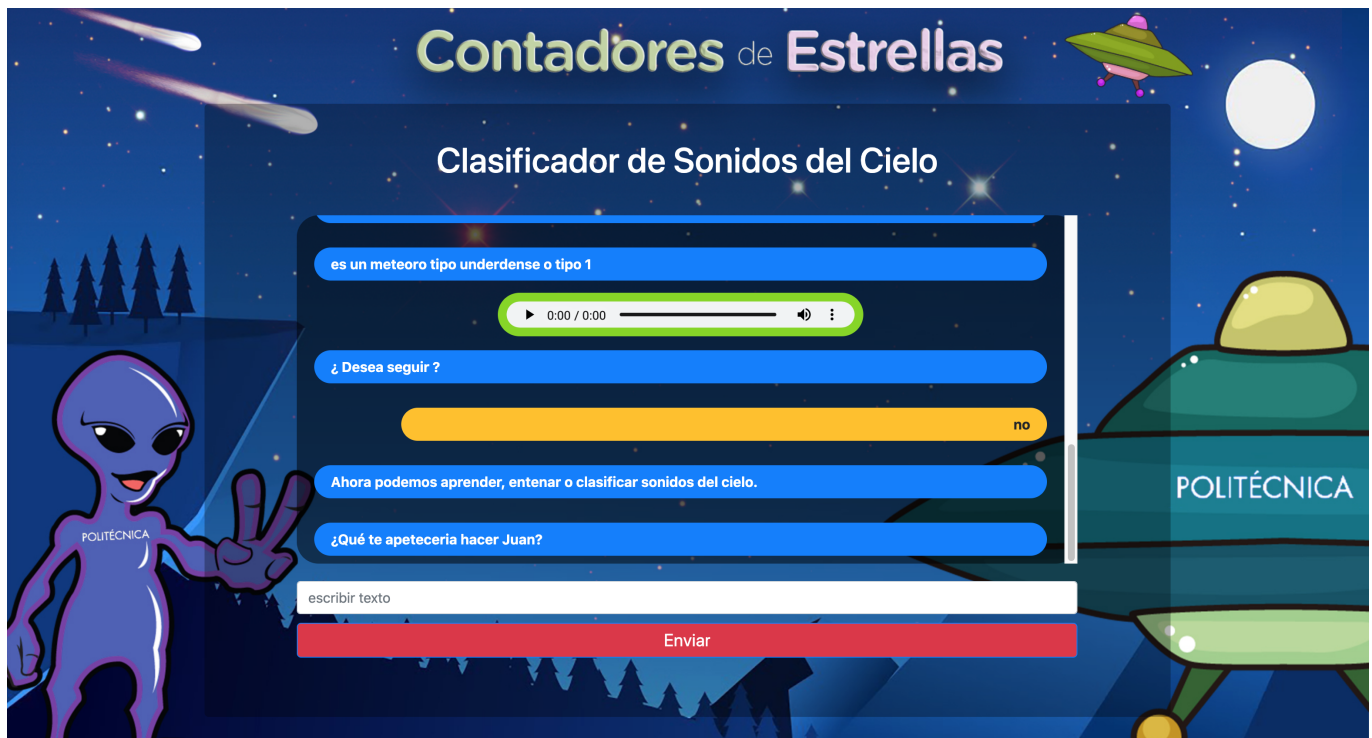
```
$ python3 manage.py runserver 0.0.0.0:8000
```

Vista Previa Frontal

- V1.5

The screenshot shows a web application titled "Clasificador de Sonidos del Cielo". At the top, there is a yellow button labeled "quiero aprender". Below it, a series of blue buttons contain the following text: "Vamos a aprender los sonidos!", "Te voy a ir presentando varios sonidos y me tendras que decir de que tipos son", "El sonido que escucharas a continuacion", and "es un meteoro underdense o tipo 1". Underneath these buttons is an audio player interface with a green border, showing a play button, the time "0:00 / 0:00", a progress bar, a volume icon, and a menu icon. Below the audio player is another blue button that says "¿Quieres continuar?". At the bottom of the interface, there is a white input field containing the text "quiero aprender", and a large red button labeled "Enviar".

- V2.0



Gracias a esta interfaz el usuario podrá ser capaz de interactuar con el asistente de manera más amigable para clasificar sonidos del cielo.

Características

La aplicación permite la clasificación de sonidos del cielo y cuenta con las siguientes funcionalidades:

Aprendizaje

- La aplicación brinda la posibilidad de poder aprender los sonidos del cielo a través del asistente conversacional.

Para aprender el sistema responde a los siguientes comandos:

- quiero aprender
- que tipos de sonidos existen
- dime todos los sonidos
- deseo aprender

Entrenamiento

- Existe un nivel de entrenamiento para que el usuario pueda aprender los sonidos. En este se presentan sonidos del aprendizaje pero sin información sobre su tipo. El sistema se encargara de evaluar la respuesta del usuario para devolver feedback sobre que tan acertada ha sido su respuesta.

- quiero entrenar
- entrenar
- deseo practicar
- entrenamiento
- deseo entrenarme
- quiero practicar

Clasificación

- Es el nivel mas interesante, aqui el sistema presenta diversos tipos de sonidos que el usuario tendra que clasificar. Cuando el usuario clasifique un sonido, su clasificación se guarda junto con la de otros usuarios. Así cuando un sonido es clasificado, al usuario se le muestra el valor promedio de clasificación que otros usuarios le han dado a ese sonido
- Cuando un usuario escucha un sonido y no puede intuir de que se trata, entonces el usuario podra pedir ayuda. En las opciones de ayuda el usuario puede elegir reproducir cualquiera de los 5 sonidos existentes. Si el usuario aun se ve con dudas, puede volver a entrenar o a aprender.

Para clasificar el sistema se activa con cualquiera de los siguientes comandos

- clasificar
- deseo clasificar
- quiero clasificar
- vamos a clasificar

Ayuda para Clasificar

Si el usuario no se siente seguro o necesita ayuda para reconocer un sonido durante una clasificación, entonces el sistema puede ayudarlo mostrandole algun ejemplo o repetir la sesión de entrenamiento.

Para mostrar las opciones de ayuda durante la clasificación el sistema responde a los siguientes comandos:

- como se clasifica ?
- como clasificar ?
- dime como se clasifica
- ayuda para clasificar
- me dices como clasificar
- no se clasificar
- no puedo clasificar
- no se clasificar

Tipos de Sonidos

El usuario podra reproducir un tipo de sonido en cualquier momento para recordar. Para ello se usa cualquier de los siguientes comandos. (Ejemplo para un overdense corto):

Reproducir un Sonido

- recuerdame como suena un overdense corto
- recuerdame como suena un meteoro tipo overdense corto
- como suena un meteoro overdense corto
- como suena un meteoro tipo overdense corto
- Como suena un meteoro tipo overdense corto ?
- quiero escuchar un overdense corto
- reproduce un meteoro overdense corto
- reproduce un tipo overdense corto

Dependencias

Los siguientes paquetes de software son necesarios en el sistema para poder hacer funcional la aplicación:

- [Python 3.7](#)
- [Rasa](#)
- [Django 3.0.7](#)
- Django Rest framework 3.11.0

Team

Contributors/People

| [Jhosef Anderson Cardich Palma](#) |
