Explicación del desarrollo en curso del chatbot de Francisco José Checa Medina para la FCT

Se trata de un chatbot aún en desarrollo que simula un asistente para el caso ficticio de una página web que está creada por un local o empresa enfocada en la venta y consumo de sus productos, en este caso cervezas. El resultado final será la mejora y actualización del chatbot para que pueda dar respuestas más precisas y que logre simular el rol para lo que está pensado, junto con su posterior implementación en una página web que se subirá a AWS.

Desarrollo del Chatbot:

Realizado con Python y empleando RASA para brindar un conjunto de herramientas de código abierto para el desarrollo de chatbots y asistentes virtuales. Proporcionando así una especie de librería que sirve de plataforma para construirlos y entrenarlos, la idea me vino durante el desarrollo de un proyecto de PHP que hicimos antes de las navidades, vi que en un entorno de producción podía ser caótico si un usuario quería ver ciertas características de un producto, ya que en un caso práctico podría haber una base de datos bastante grande y el usuario tendría que estar bajando y abriendo pestañas buscando algún producto con unas características concretas, para este problema supuse que un chatbot podría ayudar al usuario a realizar sus búsquedas aún más rápido y ahorrarle tiempo. De este modo el chatbot realiza consultas a una base de datos que tiene implementada, en estos momentos se trata de una base de datos aún bastante simple porqué primero quería entrenarlo en su dia antes de seguir subiendo un escalón más, cuando el usuario pregunta sobre el país de una cerveza, su graduación alcohólica, entre otras opciones, el chatbot le devuelve respuestas coherentes, sin embargo, lamentablemente los tres archivos de python de “actions”, sql_aux, y las validaciones dieron algunos errores, aún están pendientes de ser corregidas, como solución temporal comenté el código para que al menos el chatbot pudiese realizar algunas respuestas al usuario.

ACTIONS:

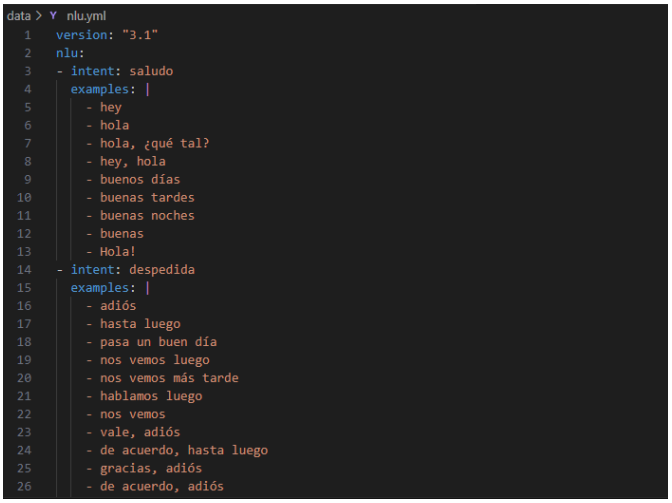
Para acciones y validaciones importé librerías de RASA, el planteamiento que tuve fue que el chatbot realizase las consultas en el “sql_aux”.py. Por otra parte, “actions.py” sirve para crear acciones personalizadas en el chatbot en respuesta a ciertas intenciones detectadas por el usuario. El código además define varias clases, cada una de las cuales representa una acción personalizada. Cada clase tiene dos métodos principales: name y run.

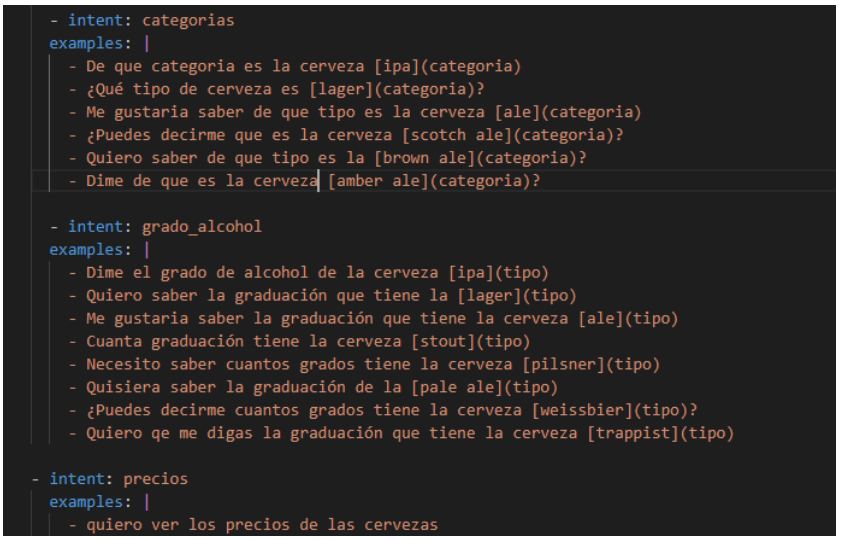
Por último “validaciones.py” lo hice para crear validaciones personalizadas en el chatbot. Las validaciones personalizadas permiten verificar la información proporcionada por el usuario antes de que se procese, también se encuentran en proceso de mejora.

DATA:

- En data encontramos tres archivos con extensión .yml (YAML Ain't Markup Language"). Es un formato de serialización de datos legible por humanos, estos archivos son:

- **nlu.yml:** Siendo “nlu” las siglas de "Natural Language Understanding" es aquí donde se recogen las preguntas y acciones del usuario, y se guardan en un intent que cumple la función de una “intención” que tiene el usuario, tenemos algunas básicas de saludos y despedidas, y algunas que están relacionadas con la base de datos.





- **rules.yml:** Aquí vienen definidas las reglas para especificar el comportamiento del chatbot en respuesta a ciertos mensajes del usuario.

```yaml
version: "3.1"

rules:

- rule: Saludar cuando el usuario salude.
  steps:
  - intent: saludo
  - action: utter_saludo
  - action: utter_ask_edad

- rule: Decir adiós cuando el usuario diga adiós.
  steps:
  - intent: despedida
  - action: utter_despedida

- rule: Decir que no entiende la frase
  steps:
  - intent: nlu_fallback
  - action: utter_no_entiendo

- rule: Decir que no entiende la frase cuando sea out of scope
  steps:
  - intent: out_of_scope
  - action: utter_no_entiendo
```

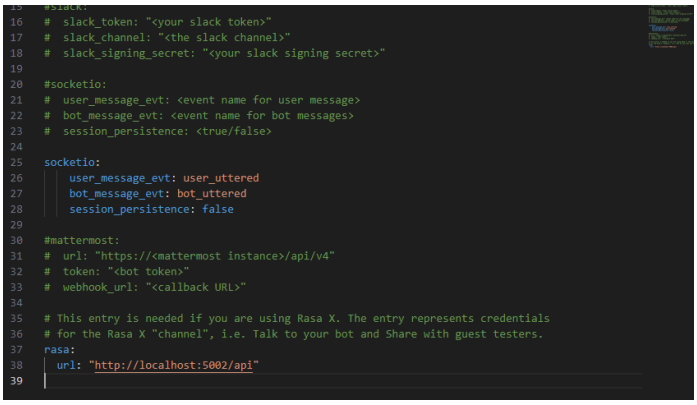
- **stories.yml:** A grandes rasgos son distintos escenarios que usará el chatbot en ciertas situaciones para usar como guia cuando interactúe con el usuario.

```yaml
- story: Consulta cervezas
  steps:
  - intent: saludo
  - action: utter_saludo
  - action: utter_ask_edad
  - intent: respuesta_edad
    entities:
    - edad: '56'
  - slot_was_set:
    - edad:
      - '56'
  - action: action_edad
  - intent: Cervezas
  - action: action_cervezas
```

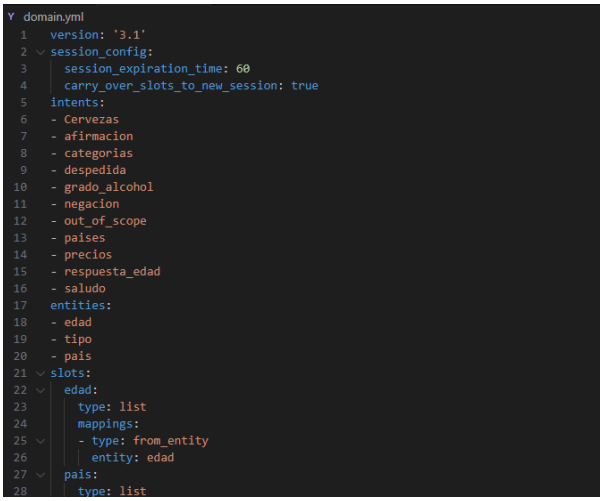
Fuera de DATA:

config.yml: Podremos configurar este apartado para cambiar algunos parámetros

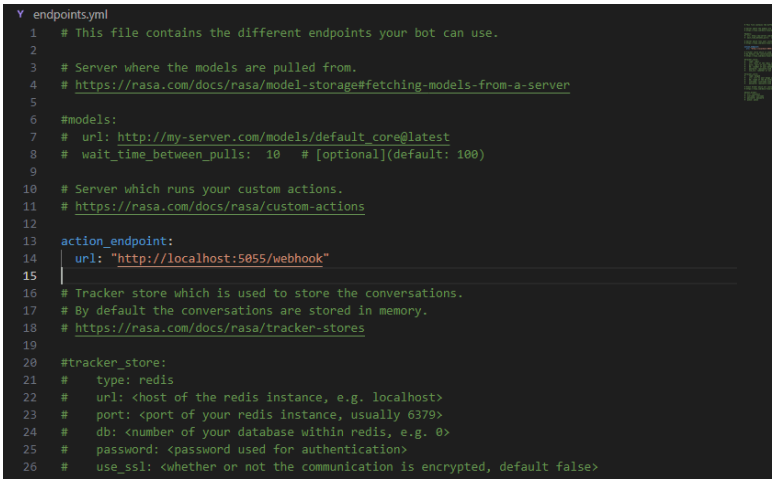
credentials.yml: almacena las credenciales necesarias para que tu bot interactúe con diferentes plataformas de chat y voz.



domain.yml: Es un marco de trabajo para el desarrollo de chatbots. Define las opciones en las que opera el bot, incluyendo las intenciones que puede entender, las entidades que puede extraer, las respuestas que puede dar y las acciones que puede realizar.

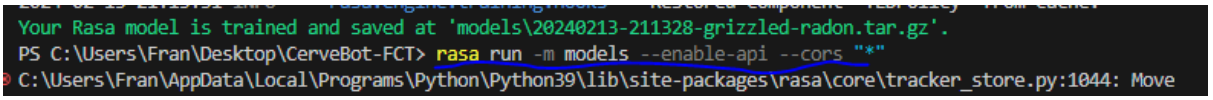


endpoints.yml: Define los diferentes endpoints que el chatbot podrá usar

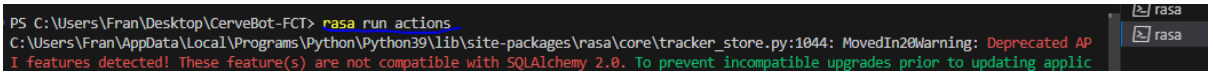


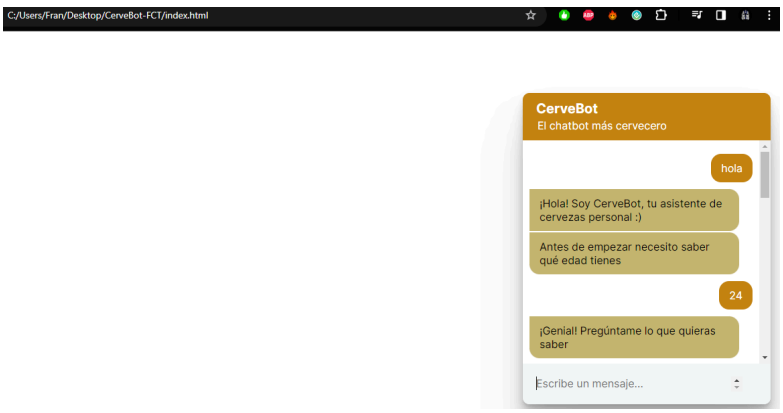
Lanzar el chatbot:

Para lanzarlo nos iremos al terminal y pondremos el comando: rasa run -m models –enable-api –cors “*”. Este comando nos habilita una API HTTP que acepta solicitudes para interactuar con el modelo de Rasa, y permite el acceso desde cualquier origen a través de CORS.



- Para que ejecute acciones tendremos que abrir un segundo terminal y usar el comando: rasa run actions





- De esta forma si el chatbot detecta que somos menores de edad no nos dará permiso para continuar

- Sin embargo como vimos anteriormente la integración con la base de datos da fallos, pero algunos intents como el out_of_scope o el de despedida entre otros si funcionan