UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Computación

MEMORIA CHATBOT MULTIAGENTE

ChatBot Multiagente
Sistemas Multiagentes

Iago García Suárez

enero, 2022

UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

Computación

MEMORIA CHATBOT MULTIAGENTE

ChatBot Multiagente
Sistemas Multiagentes

Autor: Iago García Suárez

Tutor(a): Rubén Rodríguez Cardos

enero, 2022

ChatBot Multiagente

Este texto ha sido preparado con la plantilla LaTeX de TFG para la UCLM publicada por Jesús Salido en GitHub[1] y Overleaf [2] como parte del curso *«LaTeX esencial para preparación de TFG, Tesis y otros documentos académicos»* impartido en la Escuela Superior de Informática de la Universidad de Castilla-La Mancha.

[1] https://github.com/JesusSalido/TFG_ESI_UCLM, DOI: 10.5281/zenodo.4574562
[2] https://www.overleaf.com/latex/templates/plantilla-de-tfg-escuela-superior-de-informatica-uclm/phjgscmfqtsw

Tribunal:

Presidente: _____

Vocal: _____

Secretario(a): _____

Fecha de defensa: _____

Calificación: _____

Presidente	Vocal	Secretario(a)
Fdo.:	Fdo.:	Fdo.:

A mi.

TRABAJO FIN DE GRADO - ESCUELA SUP. DE INFORMÁTICA (UCLM)

ChatBot Multiagente

Iago García Suárez
Ciudad Real, enero 2022

Resumen

El chatbot diseñado constará de 6 funcionalidades distintas que ejecutará un agente receptor o servidor una vez recibido el mensaje del agente emisor o cliente. Este comando deberá ser escrito por el usuario y se ha realizado su tratamiento mediante expresiones regulares para así lograr una mayor transparencia. De esta manera, sabiendo qué funciones tiene el chatbot, no es necesario un mayor conocimiento a priori de cómo funciona y sólo se le deberá escribir mensajes como si de un ser racional se tratara.

Para perfeccionar este tratamiento de mensajes sería recomendable utilizar ontologías pero, dado el gran incremento de la complejidad del sistema que esto provocaría, se ha optado por el uso de expresiones regulares, disminuyendo así su eficacia pero simplificando su implementación.

Se explicará, a continuación, la arquitectura del sistema donde se ha desarrollado, cómo se tratan los mensajes para identificar la funcionalidad que se desea ejecutar, los protocolos utilizados, los problemas encontrados durante su desarrollo y cómo se han abordado, y una breve guía de como desplegar y poner en marcha el proyecto.

Agradecimientos

Gracias a todos :)

Iago García Suárez
Ciudad Real, 2022

Índice general

CAPÍTULO 1

Introducción y Arquitectura del Sistema

Este proyecto ha sido realizado con fines educativos para la asignatura Sistemas Multiagentes de la ESI en el curso 2021-2022. Consiste en un chatbot que necesitará dos agentes, uno emisor y uno receptor, entre los cuales se llevará acabo un proceso de comunicación cada vez que el usuario introduzca un comando en el cliente. Ambos agentes se han implementado con SPADE y por lo tanto necesitan un servidor XMPP para funcionar correctamente, por lo que se ha utilizado el servicio de 404city.

Además, se ha llevado a cabo una dockerización del proyecto para su posible despliegue.

1.1. ARQUITECTURA DEL SISTEMA

Para la implementación y diseño de este chatbot se ha utilizado Visual Studio Code en un sistema Ubuntu 20.04. El funcionamiento básico requerirá dos archivos python, uno para el agente emisor y uno para el agente receptor y un archivo JSON de donde se leerán las credenciales de cada agente para el servidor XMPP.

Durante la implementación se ha requerido de distintos paquetes python que se han indicado en un archivo *requirements.txt* para su instalación cómoda y sencilla. Entre estos paquetes se encuentra el de *qrcode* para la generación de un código QR, que a su vez necesita el paquete *Image* de *PIL*. También se utilizarán los paquetes *bs4* para poder realizar el Web Scrapping en las búsquedas en la Wikipedia, el paquete *requests* para realizar peticiones HTTP y, como es de suponer, el paquete *SPADE* para la implementación de los agentes así como de sus comportamientos.

CAPÍTULO 2

Protocolos de comunicación

Durante la comunicación de los agentes implicados en el funcionamiento del chatbot, se emplean dos protocolos de comunicación: el protocolo *simple-request* para peticiones estándar, resultado de la simplificación del protocolo *FIPA-request-protocol*, y un protocolo diseñado con el único fin de satisfacer el funcionamiento correcto de la funcionalidad de generación de códigos QR nombrado como *qr-gen*.

2.1. SIMPLE-REQUEST

Este protocolo ha sido diseñado basándose en el protocolo del estándar FIPA *FIPA-request-protocol*, y es utilizado para comunicaciones entre un agente que haga peticiones con la performativa *request* y un agente que las reciba, ejecute la orden asignada y devuelva el resultado o un mensaje de error en el que se incluye como cuerpo el motivo del error y con performativa *failure* si ha ocurrido un error durante la ejecución de la función, o con performativa *refuse* si ha recibido la petición pero no se dispone de los recursos necesarios para llevar a cabo la acción, como puede ser el caso de la petición de las condiciones meteorológicas de una ciudad sin haber especificado la ciudad.

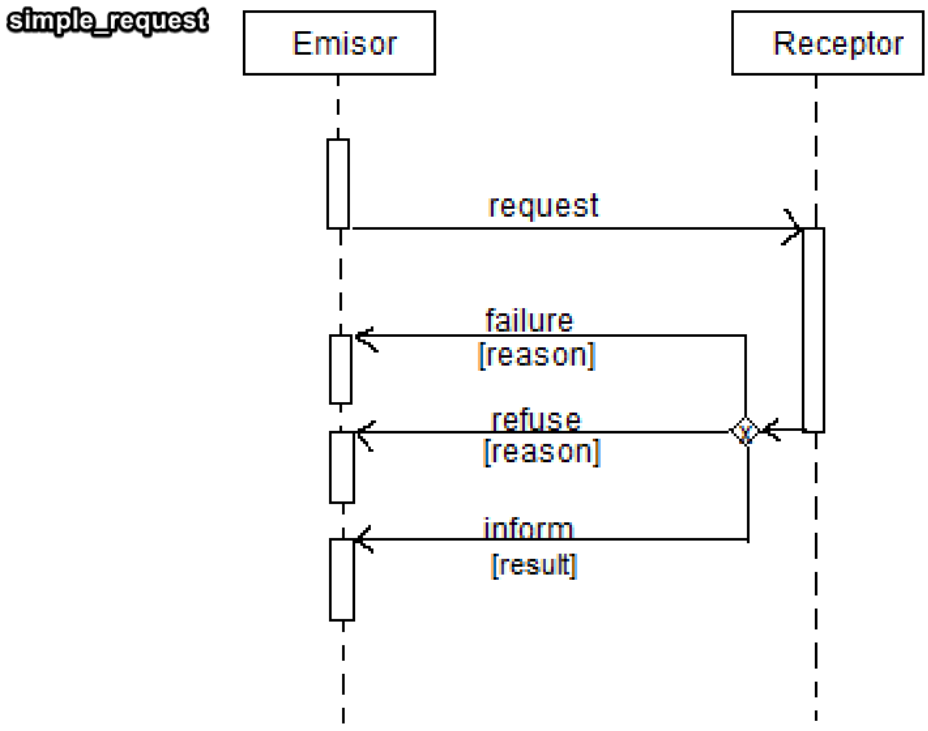

Figura 2.1: Diagrama del protocolo de comunicación simple-request

2.2. QR-GEN

El protocolo *qr-gen* tiene el fin de cumplir con el procedimiento de generación de un código QR y hacerlo disponible en el directorio del agente emisor. Es muy similar al protocolo descrito en la sección anterior pero con un paso añadido al emisor que debe ejecutar un comando de copia del archivo indicado en el cuerpo del mensaje, en caso de respuesta satisfactoria, al directorio desde donde se está ejecutando.

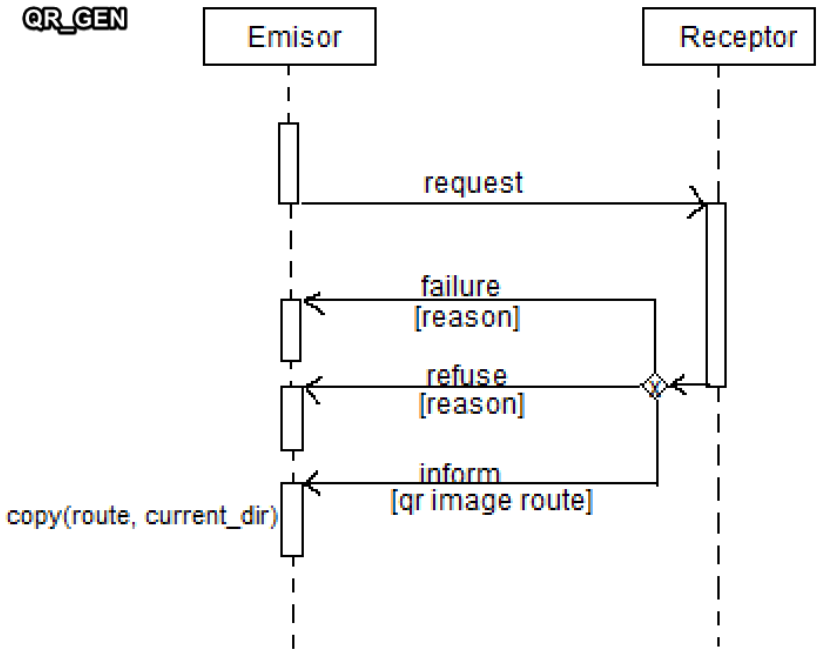

Figura 2.2: Diagrama del protocolo de comunicación qr-gen

2.3. TERMINATION

El protocolo *termination* se utiliza para expresar la intención de terminar la ejecución. El emisor enviará un mensaje con performativa *inform* y cuerpo *exit* al receptor y a continuación terminará su ejecución. Cuando el receptor lo recibe, también terminará su ejecución.

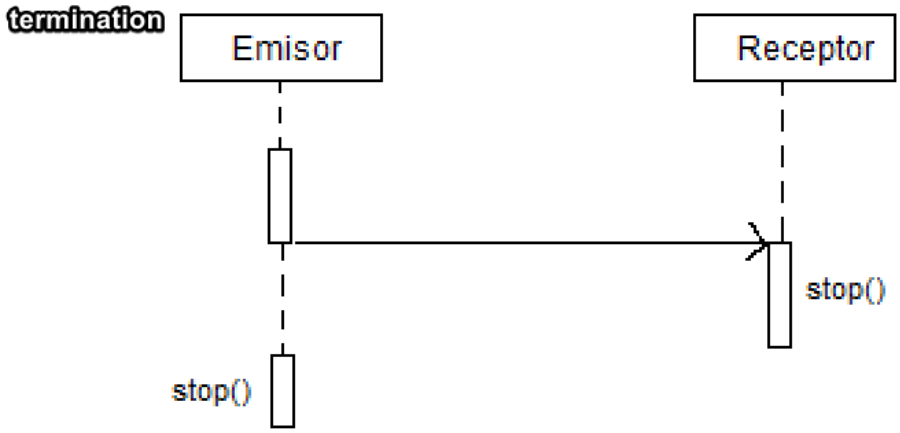

Figura 2.3: Diagrama del protocolo de comunicación simple-request

CAPÍTULO 3

Interpretación de comandos

La interfaz de la que dispone el usuario para la interacción con el chatbot consiste en una simple CLI en la que puede escribir comandos para que el agente receptor responda con el resultado correspondiente.

Existen 5 funcionalidades que interpretarán el comando recibido para averiguar qué función deberá ejecutar y 2 funcionalidades que requerirán una palabra específica para su ejecución. Para la interpretación del comando se han utilizado expresiones regulares, de modo que se pueda hablar con normalidad al chatbot y que éste entienda qué desea ejecutar el usuario y actuar en consecuencia.

El objetivo de esta decisión de diseño es el de disminuir en la medida de lo posible el conocimiento requerido del bot previo a su uso, limitándolo de esta manera a saber qué puede hacer y los comandos de ayuda y de terminación.

A continuación se explican con detalle las expresiones regulares asociadas a cada comportamiento, ordenados de manera que el último sólo tendrá lugar cuando ninguno de los anteriores ha sido detectado.

3.1. GENERACIÓN DE CÓDIGO QR

3.1.1. Funcionalidad

La funcionalidad de este comportamiento consiste en la generación de un código QR para una URL recibida, el almacenamiento en memoria del resultado en formato PNG y el envío al emisor de la ruta del archivo para que pueda copiarlo a su directorio de ejecución.

3.1.2. Expresiones regulares

Para diferenciar este comportamiento de los demás se han establecido dos reglas:

- **Identificación**: El comportamiento asignado a esta funcionalidad se añadirá cuando se detecte la palabra *QR* en el mensaje recibido, siendo no sensible a mayúsculas.
- **Parámetros**: Para generar un código QR es necesaria una URL, y para identificarla en el cuerpo del mensaje se aplica una búsqueda con la siguiente expresión regular:
  (?i)\b((?:https?://|www\{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>

3.2. INFORME METEOROLÓGICO

3.2.1. Funcionalidad

Con este comportamiento, el agente receptor utiliza la API de https://openweathermap.org para pedir un informe de las condiciones meteorológicas dada una ciudad y devuelve el resultado formateado para simplificarlo y hacerlo más legible.

3.2.2. Expresiones regulares

- **Identificación**: El comportamiento asignado a esta funcionalidad se añadirá cuando se detecte la palabra *weather* en el mensaje recibido, siendo no sensible a mayúsculas.
- **Parámetros**: Para hacer la petición a la API es necesaria una localización, y para identificarla en el cuerpo del mensaje se aplica una búsqueda con la siguiente expresión regular:
  ([A-Z][a-z]+)
  de modo que todas las ocurrencias de palabras con la inicial en mayúscula se unirán separadas por un espacio para conformar así la localización que se buscará.

3.3. CREACIÓN DE ARCHIVOS

3.3.1. Funcionalidad

Cuando el agente emisor quiera crear un archivo en el directorio del agente receptor, se ejecuta este comportamiento.

3.3.2. Expresiones regulares

- **Identificación**: El comportamiento asignado a esta funcionalidad se añadirá cuando se detecten las palabras *create* y *file* en el mensaje recibido, siendo no sensible a mayúsculas ni al orden de ocurrencia.
- **Parámetros**: Para realizar la creación del archivo es necesario indicar un nombre de archivo, y para identificarlo en el cuerpo del mensaje se aplica una búsqueda con la siguiente expresión regular:
  (\w+[.]\w+)
  de modo que la primera ocurrencia que formada por una palabra, seguida de un punto, seguido de otra palabra, se interpretará como el nombre del archivo, entendiendo que la primera palabra le dará nombre y la segunda palabra indica su extensión.

3.4. BÚSQUEDA EN WIKIPEDIA

3.4.1. Funcionalidad

El comportamiento consiste en realizar una búsqueda de una persona en la wikipedia y, mediante el uso de Web Scrapping, extraer una breve información relativa a esa persona. Generalmente esta información se describe en el primer párrafo, así que se ha elegido éste como resultado.

3.4.2. Expresiones regulares

- **Identificación**: El comportamiento asignado a esta funcionalidad se añadirá cuando se detecte la palabra *wikipedia* en el mensaje recibido, siendo no sensible a mayúsculas.
- **Parámetros**: Para realizar la creación del archivo es necesario indicar un nombre de una persona, y para identificarlo en el cuerpo del mensaje se aplica una búsqueda con la siguiente expresión regular:
  ([A-Z][a-z]+)
  la cual es similar a la del informe climatológico, ya que tendrá exactamente el mismo comportamiento, añadiendo el resultado como parámetro de búsqueda.

3.5. MOSTRAR LA HORA

3.5.1. Funcionalidad

Es el comportamiento más simple, ya que simplemente obtendrá la hora del sistema del agente receptor y la enviará formateada para hacerla más legible.

3.5.2. Expresiones regulares

- **Identificación**: El comportamiento asignado a esta funcionalidad se añadirá cuando se detecte la palabra *time* o la palabra *clock* en el mensaje recibido, siendo no sensible a mayúsculas.
- **Parámetros**: Para esta funcionalidad no es necesario ningún parámetro, así que no se ha empleado ninguna expresión regular más allá que la de encontrar las palabras de identificación del comando.

3.6. AYUDA

3.6.1. Funcionalidad

Esta funcionalidad no es un comportamiento ya que no se llega a enviar al agente receptor, sino que consiste en la identificación de la palabra *help* como comando para así mostrar la información de ayuda correspondiente.

3.7. TERMINACIÓN

3.7.1. Funcionalidad

El comportamiento de terminación se lleva a cabo en ambos agentes, ya que, al igual que la funcionalidad de ayuda, consiste en identificar el comando como la palabra *exit* y así añadir el comportamiento en el emisor. Este comportamiento enviará un mensaje con performativa *inform* y protocolo *termination*, a diferencia de los demás, que se envían con performativa *request* siguiendo el protocolo *simple-request* explicado en el capítulo correspondiente a los protocolos.

CAPÍTULO 4

Decisiones de Diseño

A lo largo del diseño del chatbot se han tomado diversas decisiones que varían su implementación. A continuación se explicará cada una así como el motivo.

4.1. AGENTES INDIVIDUALES

Se ha separado el agente emisor del agente receptor, haciéndolos independientes de manera que se puedan ejecutar en dos directorios diferentes e incluso en dos sistemas diferentes.

De esta forma, se consigue crear un servicio al que poder recurrir desde varios agentes emisores distintos sin necesidad de crear una relación 1:1 entre ambos.

A pesar de esta decisión, ciertas funcionalidades requieren que ambos se ejecuten, como mínimo, en el mismo sistema, ya que la funcionalidad de la generación de un código QR resultará en una ruta de un directorio donde poder copiar la imagen creada. Se ha intentado realizar mediante transferencia de los bytes, pero al estar restringido el cuerpo del mensaje a *String* no se ha logrado la transformación correcta.

4.2. PROTOCOLO SIMPLIFICADO DE PETICIONES

Para simplificar la comunicación, se ha diseñado un protocolo basado en el *FIPA-request-protocol* eliminando el paso intermedio de información sobre la petición, sino que se enviará la petición y se responderá siempre al emisor, sea con un mensaje de error o con un resultado a la operación.

Se ha diseñado de esta manera para así simplificar la comunicación ya que el transcurso de los eventos siempre es el mismo, y sea un error o no, se mostrará el cuerpo del mensaje por pantalla.

4.3. UTILIZACIÓN DE EXPRESIONES REGULARES

Existen varios métodos para identificar funcionalidades en una cadena de texto de lenguaje natural, ya sea mediante su procesamiento o mediante ontologías. Ambas opciones requerían un aumento considerable de la complejidad del sistema, y dado que éste sólo cuenta con un total de 6 funcionalidades, se ha optado por el uso de expresiones regulares. Con ellas se pueden identificar las intenciones del usuario, disminuyendo la eficacia, ya que es más propenso a equivocaciones en la interpretación, pero también disminuyendo la dificultad de su implementación.

CAPÍTULO 5

Despliegue

Para la ejecución correcta del chatbot será necesario seguir los siguientes pasos:

1. Entrar en el directorio *chatbot_src* donde se encuentra el código fuente del programa.
2. Localizar aquí el script llamado *run_agent* y comprobar si tiene permisos de ejecución y, en caso contrario, añadirlos.
3. Instalar todos los paquetes necesarios con *pip* indicados en el archivo *requirements.txt*. Se recomienda el uso de un entorno virtual.
4. Ejecutar el script mencionado en el paso anterior y esperar a que se muestre el prompt del sistema (» ").
5. Todo el sistema está en inglés, por lo que los comandos deberán también ser en inglés. Con *help* se mostrarán las distintas funcionalidades del bot. Para las demás, excepto la de terminación, se podrá pedir con naturalidad (si no se consiguen llevar a cabo las funcionalidades deseadas se recomienda mirar el capítulo correspondiente a la interpretación de los comandos).
6. Cuando se desee terminar la ejecución, utilizar el comando *exit* para evitar errores, ya que el proceso del agente receptor puede mantenerse activo en segundo plano si se emplea una interrupción.

Cabe mencionar que si se desea utilizar en Docker, la funcionalidad de generación de códigos QR o de creación de archivos no será válida, ya que se crea el archivo en el sistema de ejecución, por lo que sólo será accesible desde el mismo contenedor.

CAPÍTULO 6

Conclusiones

Con la realización de esta práctica se ha logrado la adquisición de conocimientos sobre el uso de *SPADE* en python, así como de *bs4* para realizar Web Scrapping. También se ha conseguido una mayor familiaridad con el funcionamiento de la comunicación entre dos agentes y la necesidad de unos protocolos que definan el comportamiento que deberá llevar a cabo cada uno.

Además, para crear el contenedor Docker, se ha aprendido cómo funciona esta tecnología y cómo desplegar o eliminar contendores con programas y servicios que se necesiten.