UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Computación

Documentación del Chatbot

Pablo Velasco Crespo

enero, 2022

Documentación

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

Índice general

1. Introducción **1**

2. Arquitectura **3**
2.1. Emisor 3
2.2. Receptor 3

3. Decisiones **7**
3.1. Clasificación de mensajes 7
3.2. Servicio de mensajería 7
3.3. Correo electrónico 7

4. Protocolos **9**

5. Manual **11**
5.1. Pasos previos 11
5.2. Cómo se ejecuta el sistema 12
5.3. Cómo comunicarse con el bot 12

Bibliografía **15**

CAPÍTULO 1

Introducción

Esta práctica consiste en desarrollar por completo un **sistema multiagente**, es decir, desde el diseño hasta la implementación.

Para el desarrollo del sistema, teníamos **dos opciones**: simular un proceso **ETL** o desarrollar un **bot conversacional**, también conocido como chatbot. El sistema desarrollado ha sido la segunda opción, la del **chatbot**.

Actualmente en el ámbito de los chatbot existen de varios tipos y con diferentes finalidades. Por ejemplo, Facebook tiene chatbots integrados en su aplicación Messenger, lo cual proporciona ventajas como poder pedirle a la CNN que nos **mande un resumen de las noticias** más importantes o las que más nos interesen según nuestros gustos.

Durante este documento se van a desarrollar los siguientes temas:

1. **Introducción**. Donde se mostrará una idea global de los apartados.

2. **Arquitectura**. Donde se explicará cómo se ha formado el sistema.

3. **Decisiones**. Donde serán explicadas las decisiones más importantes que se han tomado durante el desarrollo y, obviamente, la justificación de las mismas.

4. **Protocolos**. Donde se desarrollarán los protocolos de comunicación para los agentes.

5. **Manual**. Donde se hará una descripción de cómo se ejecuta el sistema.

CAPÍTULO 2

Arquitectura

La arquitectura del chatbot se basa en dos programas, **Emisor.java** y **Receptor.java**, los cuales, mediante los mensajes que el **usuario** (emisor) envíe al **bot** (receptor) se ejecutarán las diferentes funcionalidades del sistema.

2.1. EMISOR

La misión del emisor es **recibir el mensaje del usuario**, **clasificarlo**, **enviárselo al receptor** y por último, **mostrar al usuario** lo que le haya respondido el bot.

Para clasificarlo y enviarlo, se **divide** el mensaje del usuario en 2, la primera parte del mensaje indica qué **tipo de funcionalidad** está pidiendo el usuario y la segunda parte contiene el posible **contenido complementario** para que el bot complete la funcionalidad.

2.1.1. Clasificación

Para la clasificación se han creado varias **frases claves** en lenguaje natural para cada funcionalidad, por ejemplo, para pedir información sobre una persona, se puede decir: *"puedes decirme quien es <persona>"*, *"quien es <persona>"*, ... entonces el mensaje se clasifica en tipo **buscarPersona**, para que luego al enviar el mensaje al bot, este sepa que su deber es buscar información de la persona indicada. Lo que **haya después** de estas frases claves, se enviará como **contenido el mensaje**.

Cuando ya se sabe de qué tipo de mensaje se trata, se cambia el **protocolo** del mensaje a ese tipo, es decir, cuando nos encontramos un mensaje del tipo **buscarPersona**, cambiamos el protocolo del mensaje a enviar por *"buscarPersona"*.

2.1.2. Mostrar respuesta

Mostrar la respuesta del bot es muy sencillo, lo único que hay que tener en cuenta es si el mensaje que hemos recibido del chatbot es de tipo **INFORM**, lo cual significa que todo ha funcionado **sin problema** o de tipo **FAILURE**, lo cual significa que **algo ha ido mal** y se le muestra al usuario el problema que ha habido.

2.2. RECEPTOR

La labor del receptor es **recibir el mensaje** (ya clasificado) enviado por el emisor, hacer la **funcionalidad** correspondiente y **enviar el resultado** al emisor. Para que eso sea posible, el agente receptor tiene a su vez **11 comportamientos**, uno para cada funcionalidad. Cada uno de estos comportamientos está tratando de recibir un mensaje con un **filtro** para que solo reciba mensajes de su tipo, cuando lo reciben, comienza su labor y responde con la solución. Para detectar su tipo se filtra por la **variable protocolo** del mensaje ACL.

2.2.1. Buscar Persona

Este comportamiento, cuando recibe un mensaje, primero comprueba que el usuario haya puesto algún nombre, en caso de que no lo haya puesto, responde con un mensaje para que el usuario lo corrija. En caso de que si haya un nombre, **transforma** el nombre para que se pueda buscar en la página de *wikipedia*, para ello, le quita las mayúsculas y sustituye los espacios en blanco por barras bajas. Por ejemplo: Bill Gates ->bill_gates. A continuación, el nombre parseado se concatena a *"https://es.wikipedia.org/wiki/"*, es decir, que la forma final de la URL sería:

"https://es.wikipedia.org/wiki/bill_gates"

Tras todo el preproceso de la URL, se accede a dicha página gracias a JSoup, en caso de que la página no exista se le responderá al usuario que no se encuentra la página.

Si se encuentra la página, se accede al **primer párrafo** y se envía al usuario.

2.2.2. Mostrar Hora

El comportamiento mostrarHora, obtiene la hora y fecha actual, le da un **formato** y le envía al usuario el resultado, el formato es el siguiente: "Hoy es <*día*> de <*mes*> de <*año*>. Son las <*hora*> y <*minutos*>."

2.2.3. Crear Fichero

La labor de este comportamiento es, como su nombre indica, **crear un fichero** en la ruta indicada. El fichero que se crea es **vacío** pero se le puede dar la extensión indicada. Otra funcionalidad que tiene este comportamiento es que si al crear el fichero en la ruta indicada hay algún **directorio** que **no existe**, lo crea.

2.2.4. Borrar Fichero

Esta es una de las funcionalidades extra que se le ha añadido a este chatbot, la cual es exactamente igual que la anterior pero **elimina el fichero indicado**.

2.2.5. Teminar Ejecución

Este comportamiento **finaliza todos los agentes**.

2.2.6. Recomendar Juego

Este comportamiento incorpora una **funcionalidad adicional** al bot, la cual consiste en **recomendar un videojuego aleatorio** y dar información sobre el mismo. Para llevar a cabo su tarea, accede a la carpeta datos donde se encuentra el directorio Juegos, el cual contiene bases de datos (elaboradas por mi mismo con JSoup). Cada una de estas bases de datos contiene un listado de **todos los juegos** que tienen página de wikipedia de diferentes **consolas** y **PC**.

Este comportamiento elige una base de datos al azar y después selecciona uno de los juegos dentro de esta y lo busca. A continuación se selecciona la información más importante y se la envía al usuario.

2.2.7. Enviar Correo

Este es uno de los **comportamientos adicionales** que se ha añadido al chatbot. Su funcionalidad es enviar un correo. Para ello el bot le pedirá al usuario la siguiente información:

1. **Receptor del correo**. Un correo electrónico, que no es necesario que sea un *gmail*, pero si es necesario que exista, en caso de que no exista, se mostrará un mensaje de error al usuario.
2. **Asunto**. Asunto del correo.
3. **Cuerpo**. Mensaje del correo.

Cada uno de estos campos irá separado por "///":

- *<receptor>*///*<asunto>*///*<cuerpo>*
- prueba@gmail.com///reunion///Hola buenas, necesito reunirme contigo, gracias.

Para este comportamiento es muy importante la ruta en la que estén los credenciales. Esto se desarrolla más abajo.

2.2.8. Meme Random

Este comportamiento es una **funcionalidad extra** para el trabajo, la cual te muestra un **meme aleatorio**. Para ello, accedemos a este enlace, el cual muestra memes aleatorios que se cambian cada vez que se recarga la página, lo cual nos viene perfecto, ya que como se cambia cada vez que se accede simplemente **descargamos** la primera imagen y se **abre** en el escritorio.

El meme se descarga en el directorio de jade, pero se puede cambiar modificando rutaMeme en Receptor.java. Cada vez que se accede a este comportamiento el meme se **sobrescribe**, para no llenar el ordenador del usuario de memes de dudosa calidad.

2.2.9. Conversación

Este apartado engloba 3 comportamientos: saludos, preguntas y respuestas. Son funcionalidades **extras añadidas** para que el usuario y el bot puedan "conversar", son muy simples. Sirven para que el bot responda cuando le saludan, responda cuando le preguntan cómo está...

CAPÍTULO 3
Decisiones

A continuación se expondrán las decisiones que se han tomado durante el desarrollo del Chatbot y su correspondiente justificación.

3.1. CLASIFICACIÓN DE MENSAJES

La primera decisión está relacionada con cómo puede identificar el bot lo que desea el usuario, es decir, que si el usuario escribe: ´´quien es Elvis Presley" que el bot sepa identificar que lo que el usuario quiere decir que busque en wikipedia Elvis.

Por lo que la decisión tomada fue **encapsular los mensajes** de la siguiente forma: *<tipo mensaje><posible contenido>* y para identificar de qué tipo es el mensaje, se han recogido diferentes frases para decir, por ejemplo "quien es", como: "que hizo", "a que se dedica", ...

3.1.1. Justificación

Se ha utilizado esta metodología para la clasificación de los mensajes porque es la **más sencilla**, debido a que cuando encuentra alguna de estas frases se puede clasificar el tipo de mensaje.

3.2. SERVICIO DE MENSAJERÍA

Otra decisión importante fue elegir qué **servicio de mensajería** a utilizar, a pesar de esta importancia fue muy sencillo. Se utilizó el servicio de correos electrónicos de Google, Gmail.

3.2.1. Justificación

Con diferencia este servicio de correos electrónicos es el que más documentación, librerías y tutoriales tiene.

3.3. CORREO ELECTRÓNICO

También relacionado con el correo electrónico, había dos opciones, utilizar el **correo electrónico del usuario** o utilizar una **cuenta nueva** de correo electrónico. La decisión final fue crear una nueva cuenta de Gmail y el chatbot será la cuenta que utilizará para enviarlos.

Por otra parte, se puede **modificar** el archivo *redenciales.txt* con una cuenta de correo diferente y su contraseña si se desea. Aunque para que funcione se debe activar la opción de **dar acceso a apps menos seguras**. Para ello, vamos al gmail, después hacemos clic arriba a la derecha a nuestra foto de perfil, después a "Administra tu Cuenta Google". A continuación, en la parte izquierda accedemos a la sección de "Seguridad". Bajamos a "Acceso de apps menos segura" y lo activamos.

3.3.1. Justificación

El motivo de esta decisión es muy simple, es muy tedioso poner el correo electrónico y activar la opción mencionada, de esta forma es más **sencillo** y **cómodo** para el usuario introducir los datos.

3.3.2. Recomendación de videojuegos

Para esta funcionalidad se tomó la decisión de hacer una **base de datos local** con todos los **videojuegos** de 10 consolas (incluido PC). Para ello se hizo *web scrapping* en los siguientes enlaces de este enlace de wikipedia:

1. Juegos de PC
2. Juegos de Nintendo 64
3. Juegos de Wii
4. Juegos de Game Cube
5. Juegos de Play Station 2
6. Juegos de Play Station 3
7. Juegos de Play Station 4
8. Juegos de XBOX
9. Juegos de XBOX 360
10. Juegos de XBOX ONE

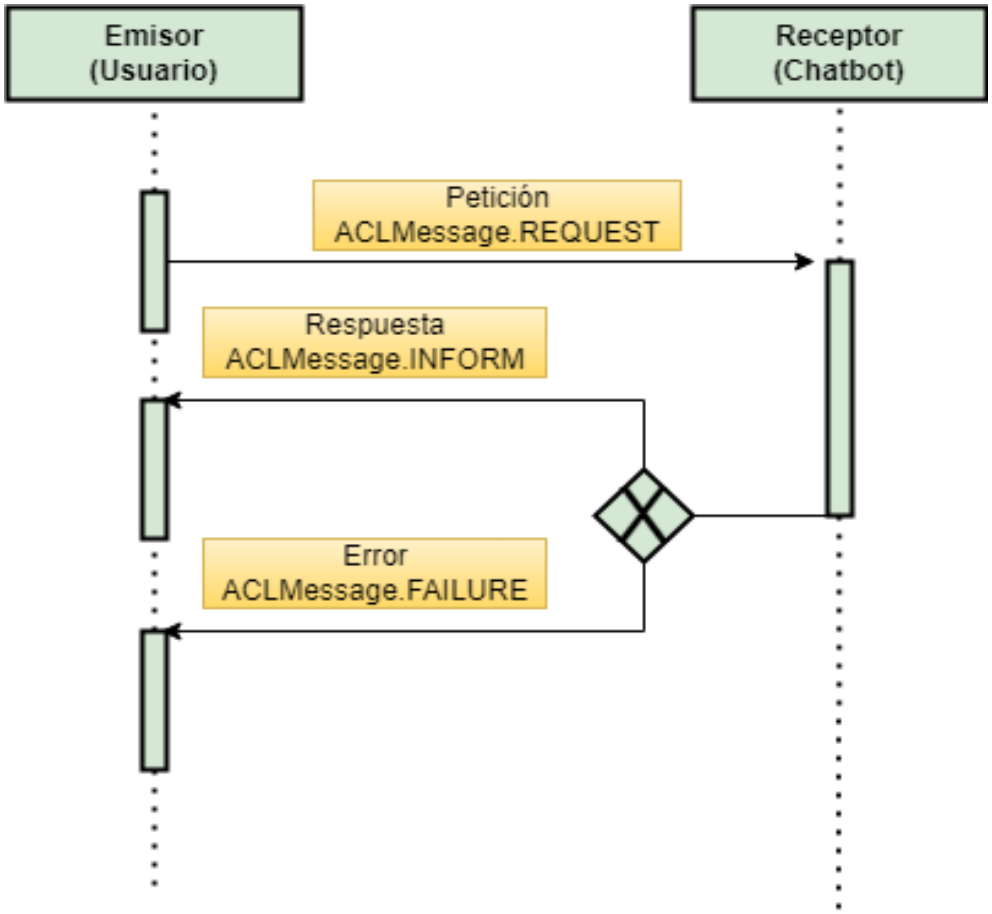
CAPÍTULO 4

Protocolos

Para la comunicación se han utilizado 3 tipos de mensajes ACL diferentes:

1. **REQUEST**. Este tipo de mensajes son enviados desde el **emisor** (usuario), son mensajes de **petición** y son recibidos por el receptor (chatbot) el cuál tratará de resolver la petición realizada.
2. **INFORM**. Serán enviados solo por el **chatbot**, este tipo de mensajes se caracterizan porque **informan** al usuario, es decir, le muestra lo que ha pedido con el mensaje del tipo anterior, lo que significa que todo ha salido **correctamente**.
3. **FAILURE**. Que se enviará desde el **chatbot** también, en las mismas condiciones que el INFORM, pero **cuando algo ha fallado**, por lo que este tipo de mensaje también informa pero sobre el error que se ha capturado.

4.0.1. Diagrama de Protocolos



CAPÍTULO 5

Manual

A continuación se explicará cómo se puede ejecutar el sistema multiagente, para ello serán necesarios varios pasos:

1. **Pasos previos**. Programas que hay que instalar, librerías (jar) que hay que descargar y cómo hay que tener el classpath.
2. **Cómo se ejecuta el sistema**. Qué comandos son necesarios para poner en funcionamiento al chatbot.
3. **Cómo comunicarse con el bot**. Donde se explicará cómo acceder a cada funcionalidad que el bot posee.

5.1. PASOS PREVIOS

5.1.1. Cómo colocar los archivos

jade/

- src/
  - chatbot/
    - Emisor.java
    - Receptor.java
    - datos/
- lib/
  - jade.jar
  - javax.activation-1.2.0.jar
  - jsoup-1.14.3.jar
  - mail.jar

IMPORTANTE: para la funcionalidad de recomendar juegos y la de enviar correo es posible que sea necesario cambiar rutaDatos del archivo Receptor.java a la ruta donde se coloque la carpeta datos si no se colocan de la forma en la que se muestra anteriormente. Y si quiere que los memes se descarguen en una carpeta en específico modifique rutaMeme a un directorio existente.

5.1.2. JADE

Lo primero, es tener JADE operativo en el equipo. En caso de no tenerlo puede seguir el siguiente tutorial.

5.1.3. JARS

Después, a parte de tener JADE serán necesarias varias librerías del tipo JAR, las cuales tendrán que estar en el classpath del equipo, para mayor comodidad, tanto a la hora de poner los classpath más adelante como a la hora de ejecutar, vamos a colocar todos los JAR que mencionemos dentro del directorio donde tengamos JADE, dentro de la carpeta lib. Si caben estarán en la entrega, pero por si acaso se pueden descargar aquí:

1. **jade.jar**. Si está instalado JADE este jar debería estar ya en la carpeta mencionada anteriormente.
2. **jsoup-1.14.3.jar**. Esta librería es utilizada para hacer web scraping, se ha utilizado su versión 1.14.3. Para descargar accede a este enlace, hay que hacer clic en jsoup-1.14.3.jar y se descarga el jar.
3. **mail.jar**. La cual será utilizada para enviar correos. Se ha utilizado la última versión de javamail 1.4.7. Para descargar accede a este enlace y descargas el zip de la versión 1.4.7 (en teoría funciona con las demás versiones, pero esa es laque se ha utilizado). Hay que descomprimir el zip y ahí está el mail.jar.
4. **javax.activation-1.2.0.jar**. La cuál se complementa con la anterior, también es para enviar correos. Se utiliza la versión 1.2.0. Para descargar accede a este enlace y en la parte superior derecha haz clic en Downloads y elige la opción de jar.

5.1.4. CLASSPATH

Este paso es muy sencillo, hay que poner en el **classpath** los JARS mencionados anteriormente en el classpath.

5.2. CÓMO SE EJECUTA EL SISTEMA

Primero hay que colocar la carpeta chatbot, dentro de la carpeta src dentro del directorio de JADE, después, si se han colocado todos los JARS mencionados anteriormente en la carpeta de lib dentro del directorio de JADE y se está usando el sistema operativo de Windows, se puede ejecutar de una forma sencilla utilizando el batch proporcionado, para ello este batch tiene que estar situado dentro del directorio de JADE.

Si no es el caso, hay que colocarse en el directorio de JADE y utilizar los siguientes comandos:

- *javac -classpath "<jade.jar>;<jsoup-1.14.3.jar>;<mail.jar>" -d classes src/chatbot/*.java*
- *java jade.Boot -agents "emisor:chatbot.Emisor;receptor:chatbot.Receptor"*

Donde *<jade.jar>*, *<jsoup-1.14.3.jar>* y *<mail.jar>* son los directorios donde están cada uno de los **JARS**.

5.3. CÓMO COMUNICARSE CON EL BOT

La intención que se ha tenido para la comunicación con el bot es utilizar **lenguaje natural** para pedirle cosas. Por lo que, por ejemplo, para decirle que nos recomiende un videojuego, simplemente tendremos que poner: *recomiendame un videojuego* o *me recomiendas un juego*... Hay diferentes frases para cada funcionalidad. Se ponen sin tildes para que no haya fallos con java.

Por ejemplo la funcionalidad de enviar un correo le va a pedir datos, como: correo del receptor, asunto y cuerpo del mensaje.

Frases clave de ejemplo para cada funcionalidad (hay bastantes más por cada funcionalidad, pero he puesto estas para que sirvan de guía):

1. **Buscar Persona**: quien es, que hizo, a que se dedica.
2. **Mostrar Hora**: que hora es, me puedes decir la hora.

3. **Crear Fichero**: crea un fichero, crea un archivo.
4. **Borrar Fichero**: elimina el fichero, elimina el archivo.
5. **Terminar Ejecución**: adios, hasta luego, nos vemos, apagar.
6. **Meme Random**: eseñame un meme, estoy triste, animame.
7. **Recomendar Juego**: recomiendame un juego, me recomiendas un videojuego.
8. **Enviar Correo**: puedes enviar un correo, necesito mandar un correo.
9. **Saludos**: hola, buenas tardes.
10. **Preguntas**: que tal, como estas, estas bien.
11. **Respuestas**: me alegro, guay.

Bibliografía

[1] Code Tech Gyan. Send email using java code tutorial. https://www.youtube.com/watch?v=WKOoAQTioNU&list=WL&index=5&t=195s. Utilizado como guía para que el bot pueda enviar correos electrónicos.

[2] InboundCycle. Qué es un chatbot, cómo funciona y para qué sirve. https://www.inboundcycle.com/diccionario-marketing-online/chatbot. Utilizado para el estado del arte en la introducción.