UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND IMPLEMENTATION OF AN EMBODIED COGNITIVE CONVERSATIONAL AGENT FOR ASSISTING IN A WEB SITE BASED ON THE DIALOGFLOW CONVERSATIONAL PLATFORM

ÁNGEL SILVÁN CAMIÑA

2018

TRABAJO FIN DE GRADO

Título:	Diseño e implementación de un agente conversacional cognitivo incorporado para ayudar en un sitio web basado en la plataforma conversacional Dialogflow
Título (inglés):	Design and implementation of an embodied cognitive conversational agent for assisting in a web site based on the Dialogflow conversational platform
Autor:	Ángel Silván Camiña
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:

Vocal:

Secretario:

Suplente:

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DESIGN AND IMPLEMENTATION OF AN EMBODIED COGNITIVE CONVERSATIONAL AGENT FOR ASSISTING IN A WEB SITE BASED ON THE DIALOGFLOW CONVERSATIONAL PLATFORM

Ángel Silván Camiña

Enero de 2018

Resumen

Este proyecto se ha centrado en el diseño y la implementación de un asistente personal integrado en la página web del Grupo de Sistemas Inteligentes (GSI) permitiendo una interacción entre el usuario y el agente conversacional mediante el uso de la computación cognitiva.

Para ello, se ha desarrollado una chat bot para realizar las conversaciones necesarias. Este sistema ha sido desarrollado utilizando la alternativa Dialogflow así como un Sistema de Gestión de Contenidos llamado Joomla!.

Se ha propuesto una arquitectura para nuestro sistema que permitiese realizar las tareas previstas, integrando un sistema pregunta-respuesta, un agente de conversación capaz de identificar y procesar el lenguaje natural en español y un sistema de búsqueda de información en la página web donde está integrado el agente conversacional.

De forma que el chat bot integrado fuese completo se han estudiado dos distintos caminos en función de la petición del usuario. El primero es una conversación simple entre el usuario y el bot que permita realizar preguntar sencillas sin entrar en búsquedas de información. El segundo posible camino es abandonar esta conversación ligera si se detecta en la petición preguntas sobre los datos de miembros, publicaciones o proyectos del GSI. En este último caso, el bot realiza una búsqueda en la base de datos y proporciona al usuario en caso de haber tenido éxito un resumen de los datos entontrados así como la posibilidad de acceder a información más extensa.

Con todo esto, el proyecto permitirá a los usuarios entablar una conversación ligera o de búsqueda de información a través de la página web y hará posible realizar búsquedas exactas.

Palabras clave: Computación Cognitiva, Sistema de Gestión de Contenidos, Agente Conversacional, Grupo de Sistemas Inteligentes, Chat bot.

Abstract

This project has focused on the design and implementation of a personal assistant integrated into the web page of the Intelligent Systems Group (GSI) allowing an interaction between the user and the conversational agent through the use of cognitive computing.

For this, a chat bot has been developed to carry out the possible conversations. This system has been developed using the Dialogflow alternative as well as a Content Management System called Joomla!.

An architecture has been proposed for our system that would enables to perform the planned tasks, integrating a question-answer system, a conversation agent capable of identifying and processing the natural language in Spanish and an information search system on the website where it is integrated the conversational agent.

In order that the integrated bot chat was complete, two different ways have been studied according to the user's request. The first is a light conversation between the user and the bot that enables simple requests without information searches. The second possible way is to abandon this light conversation if questions about member data, publications or GSI projects are detected in the request. In this case, the bot performs a search in the database and provides a summary to the user, if the search has been successful, as well as the possibility of accessing more extensive information.

With all this, the project will allow users to engage in a light conversation or search for information through the website and will make it possible to perform exact searches.

Keywords: Cognitive Computing, Content Management System, Conversational Agent, Intelligent Systems Group, Chat Bot.

Agradecimientos

Quiero aprovechar estas líneas para agradecer a toda la gente que me ha acompañado durante este magnífico *viaje* que tanto he disfrutado.

A mis padres y mi pareja, por su apoyo incondicional y por sus intentos de entender siempre a lo que me dedicaba, a pesar de que raras veces comprendían de que hablaba.

A mis amigos y compañeros de la *Escuela*. A todos aquellos con los que me he cruzado en el camino y me han permitido aprender de ellos.

A todos los del Grupo de Sistemas Inteligentes y en especial a Carlos, que siempre ha confiado en mí y me ha apoyado.

A cualquiera que leyendo estas líneas sabe que ha significado algo para mí.

Contents

Resumen **VII**

Abstract **IX**

Agradecimientos **XI**

Contents **XIII**

List of Figures **XVII**

1 Introduction **1**

1.1 Overview 2

1.2 Project goals 2

1.3 Structure of this document 3

2 Enabling Technologies **5**

2.1 Introduction 6

2.2 Cognitive Computing 6

2.2.1 Cognitive Computing Properties 7

2.2.2 Use Environments 8

2.3 Personal Agents 8

2.4 Conversation Tool - Dialogflow 10

2.5 Joomla Content Management System (CMS) 12

2.5.1 Structure of the platform 13

2.5.1.1 Components 14
2.5.1.2 Modules 14
2.5.1.3 Plugins 15
2.6 Conclusions 15

3 Requirement Analysis **17**
3.1 Overview 18
3.2 System Actors 18
3.3 Use cases 19
3.3.1 Know the users - Use Case 20
3.3.2 Get posts on a topic - Use Case 21
3.3.3 Get information about a project - Use Case 22
3.3.4 Light Conversation - Use Case 23
3.4 Conclusions 23

4 Architecture **25**
4.1 Overview 26
4.2 Content Management System 27
4.2.1 Joomla Module: Bot Dialogflow 28
4.2.1.1 default.php 29
4.2.1.2 helper.php 30
4.2.1.3 getResultApiAi.php 30
4.2.2 Joomla Plugin: JResearch 32
4.3 Conversation System 33
4.4 Conclusions 34

5 Case Studies **35**
5.1 Case Study: Light Conversation with Dialogflow 36

5.1.1 Overview of the system 36
5.1.2 Overall Process 36
5.1.3 Light Conversation Use Case 37
5.2 Case Study: Deep Conversation with Dialogflow 38
5.2.1 Overview of the system 38
5.2.2 Overall Process 38
5.2.2.1 Know the users Use Case 39
5.2.2.2 Get information about a project Use Case 40
5.2.2.3 Get posts on a topic Use Case 41
5.3 Summary 42

6 Conclusions and future lines **43**

6.1 Conclusions 44
6.2 Achieved goals and problems faced 44
6.3 Future work 45

Bibliography **47**

List of Figures

2.1 Dialogflow Tool [1] 10

2.2 Modules of the platform Dialogflow [2] 11

3.1 Use Cases Diagram 20

3.2 Know the users Sequence Diagram 21

3.3 Get Posts on a topic Sequence Diagram 22

3.4 Get information about a project Sequence Diagram 22

3.5 Light Conversation Sequence Diagram 23

4.1 Architecture 27

4.2 Module Dialogflow Light Conversation Sequence Diagram 28

4.3 Module Dialogflow Search Conversation Sequence Diagram 28

4.4 Chat Box 29

4.5 Flowchart 31

4.6 Dialogflow flow [3] 33

4.7 Example of a JSON document from Dialogflow 34

5.1 Overall cycle for the system 36

5.2 Examples of light conversation 37

5.3 Overall cycle for the system 39

5.4 Examples of member searches 40

5.5 Examples of project searches 41

5.6 Examples of posts searches 42

CHAPTER 1

Introduction

This chapter provides an introduction to the problem which will be approached in this project. It provides an overview of the benefits of conversational agents and the artificial intelligence. Furthermore, a deeper description of the project and its environment is also given.

1.1 Overview

Nowadays, the development is constant with all the technologies and new ideas which appear as: digital transformation, big data, machine learning, artificial intelligence ... etc. Therefore, doubts and questions appear usually in order to resolve how to use certain tools or where to find certain information, and this is where the use of artificial intelligence and chat bots comes into play.

Usually lot of questions related to a web page or actions related to users are attended by a small group of people depending on the size of the company. Therefore, if there are more problems or doubts than personal that can solve them, the service is no longer optimal and the solution tends to stop being a deep and personal solution.

For all these reasons, chat bots appear with the aim of helping users and allow us through this tool to gain time and user satisfaction. Personal assistants with artificial intelligence are necessary to engage users by offering customized solutions, wherever and whenever they are requested. Personal agents are present in many fields, from educative platforms to virtual city tours or database querying.

To solve that, this project focuses on the design and implementation of a personal agent integrated into a web site allowing it to interact with users through the use of natural language. We will therefore study the technologies available for understanding natural language, as well as information retrieval and indexing technologies, allowing us to build a system that will interact with users, both answering their questions in natural language and proposing answers to topics related to the web of the Intelligent Systems Group (GSI).

The GSI has a website designed with Joomla in which they store content about their department. They have all kinds of information related to publications, members, events, research, projects and news.

Therefore, our system will allow us to manage the desired tasks in relation to the functionalities of the web page providing an integrated platform to solve doubts, answering the questions of the users in Spanish.

1.2 Project goals

Above all things, this project aims to develop a system that would allow users to interact using natural language with a linked webhook, receiving the information they have asked for, as well as suggestions and information about related topics. With this objective the

project will be formed by a conversation module, a module for Joomla [4] which will call Dialogflow [5] and a plugin for Joomla in order to do the searches.

Among the main goals inside this project, we can find:

- *Design a personal assistant* with Dialogflow [5] who is able to have a light conversation.
- *Integrate external information* through web hooks to be able to solve questions related to the web page
- *Integrate the personal assistant* into the web page using a Joomla plugin.
- *Testing* different cases.
- *Analyze the results* of the testing in order to improve the personal assistant.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document.

The structure is the following:

Chapter 1 provides an introduction to the problem which will be approached in this project. It explains the context in which this project is developed. Moreover, it describes the main goals to achieve in this project.

Chapter 2 contains an overview of the existing technologies on which the development of the project will rely.

Chapter 3 describes one of the most important stages in software development: the requirement analysis using different scenarios. For this, a detailed analysis of the possible use cases is made using the Unified Modeling Language (UML). This helps us also to focus on key aspects and take apart other less important functionalities that could be implemented in future works.

Chapter 4 describes the architecture of this project, including the design phase and implementation details.

Chapter 5 describes the selected use cases. It is going to be explained the running of all the tools involved and its purpose. It demonstrates a prototype in order to extract the information about the Intelligent Systems Group (GSI), its members, publications and projects.

Chapter 6 sums up the findings and conclusions found throughout the document and gives a hint about future development to continue the work done for this project.

CHAPTER 2

Enabling Technologies

This chapter describes a brief introduction of the state of the art for conversational agents. Likewise, it talks about how we are going to manage the content referring to the project.

2.1 Introduction

For the development of this project we must first talk about the technologies used in this project. In order to do that, in this section we will explain and analyze the following technologies.

At first, the technology used for all this type of applications: cognitive computing [6]. We are going to give a likely definition of this technology and then we will analyze the characteristics of it. After that, we must study there possible use environments.

Secondly, we have to talk about how personal agents and chat bots work. The ability of this type of software to identify words and phrases in natural language and convert them to a machine-readable format will be inspected in this section. After that, we must talk about a conversation motor used in this project: Dialogflow [5] tool in collaboration with Google Actions [7]. Nowadays, Dialogflow [5] is one of the most powerful conversational platforms with a word error rate less than 4,9%.

Fourthly, the Joomla platform [4], a free open source Content Management System (CMS) for publishing web content will be analyzed. This CMS is multilingual, with easy updates and above all with an integrated help system and a content manager that allows us to categorize and nest the components that are necessary in an easy and effective way.

2.2 Cognitive Computing

Nowadays, increasingly companies are investigating in this field. Important communities like HP, Microsoft or IBM have in the last decade made a great leap towards this sector. If we look for a definition of this term we can find numerous whether on the Internet or in books. However, there is a definition accepted by all experts in this field, and it is the definition proposed by the Cognitive Computing Consortium. This explanation of the term cognitive computing appear in 2014 by a cross disciplinary group of experts. According to the Cognitive Computing Consortium [8], the definition of cognitive computing is:

"Cognitive computing makes a new class of problems computable. It addresses complex situations that are characterized by ambiguity and uncertainty; in other words it handles human kinds of problems. In these dynamic, information-rich, and shifting situations, data tends to change frequently, and it is often conflicting. The goals of users evolve as they learn more and redefine their objectives. To respond to the fluid nature of users' understanding of their problems, the cognitive computing system offers a synthesis not just of information

sources but of influences, contexts, and insights. To do this, systems often need to weigh conflicting evidence and suggest an answer that is "best" rather than "right" [6]."

In other words, cognitive systems differ from current computing applications in that they move beyond calculating based on configured rules and programs. Although they are capable of basic computing, they can also infer and even reason based on broad objectives. For that reason, before giving an answer some parameters have to be taken into account: machine learning, question analysis, current natural language. So, we can say that cognitive computing add a context to the question input, which has as result a specific response to each problem.

Cognitive computing [9] redefines the nature of the bridge between people and the digital environment. Consequently, it is used on virtual assistants and automation tasks systems. However, even if the importance and the paper of this technology is increasing, the boundaries of the processes and domains these systems will affect are still elastic and emergent.

2.2.1 Cognitive Computing Properties

According to the Cognitive Computing Consortium, the main features that cognitive systems have are the following [6]:

- Adaptive: They must be trained to learn as interactions, as information changes, and as goals and requisites evolve. They must resolve ambiguity and tolerate unpredictability. They must be engineered to feed on dynamic data in real time, or near real time. They must handle ambiguous data or unpredictable data with minimal human supervision.
- Interactive: It have to be easy for the users to interact with them so that those users can define their requirements comfortably. They may also interact with other devices as well as with people in order to adapt their response to each user.
- Iterative: They must help the users anticipating the user intent by asking questions or finding additional information if a problem is ambiguous or incomplete. They need to remember the history of past interactions in order to resolver user present questions.
- Contextual: They must be capable of extracting, understanding and identifying information about the context of the intent. Besides, they have to be able to access and use structured and unstructured digital information, as well as sensory inputs in case of having

2.2.2 Use Environments

Nowadays, cognitive computing systems are used in plenty of situations like say IBM [10]. The reason is that they are capable to use plenty of information since personal data to weather data for example.

Some usually case environments are:

- Speech recognition: they develop methodologies in order to enable the understanding and translation of spoken language into text by computers.
- Sentiment analysis: they identify, extract and quantify personal information by the use of the voice of the customer or biometrics. This information gives an analysis about the states and the subjective emotion of the user.
- Face detection: They identify human faces in digital images.

2.3 Personal Agents

"Intelligent bots are computer programs that leverage artificial intelligence to enable natural conversations with people. The recent advancements in machine learning and the convergence of compute power and big data are bringing artificial intelligence into the mainstream [11]."

The most frequent model is the personal agent or chat bot [12], a bot capable of simulating a conversation with a person. They use the speech recognition to translate the spoken language into text and therefore they are increasingly present in messaging applications.

If we select speech recognition from the environments just above, then we must add an important property to those previously described for cognitive computing:

- Natural Language Interaction: They understands queries and interacts with humans in natural language.

Even if a personal agent must be capable to understand a human, it is a limited entity in terms of topics. We must be concise that the ability to reason external issues is not possible. The challenge of designing a chat bot is to take into account the general context and apply it to topics in which they must be experts.

For this reason, a personal agent is a bot with the following properties in order to become expert in certain topics:

- Strong Natural Language Understanding (NLU) capabilities.
- Capability to work with human agents.
- Ability to learn specific properties based on machine learning.
- Scalability during training, deployment and conversations.

Once the properties of bots are studied, their main work and the scope they have in speech recognition will be analyze. From of all those mentioned before, personal agents stand out for their effective work in the field of speech recognition. That is the ability of a software to identify words and phrases in natural language and convert them to a machine-readable format.

Looking the technology perspective, it still has some issues to work through. Actually, the only downside of this type of cognitive computing is the inability to catch some words or expressions due to variations of pronunciation or writing errors. Rudimentary speech recognition software has a limited vocabulary of words and phrases, and it may only identify that if they are spoken or typed very clearly.

However, companies like Microsoft, IBM, Apple, Amazon or Google are investigating successfully in this field taking advantage of the new tools which exist today like: deep learning and big data. For this reason, more sophisticated software has appeared with the ability to accept natural speech. The effectiveness of speech recognition is mainly evaluated in terms of speed and accuracy. Speed is usually measured with a real time factor but the accuracy is always rated with word error rate (WER).

If we look for the word error rate on Internet, we can find many articles that talk about it. Nevertheless, we can find two enterprises that stand out for their research in this sector. The opinion of IBM about the WER is the following:

In the world of speech recognition software, 5.1% is kind of a magic number. Companies that can create software with error rates falling in that threshold are essentially matching the capabilities of humans, who miss roughly 5% of the words in a given conversation. [13]"

By the way, Google [7] says that from June of the last year to today they have decreased the word error rate from 8.5% to 4.9%. Here comes into count the conversation tool named Api.ai now Dialogflow which was bought by Google [7] on September 2016.

2.4 Conversation Tool - Dialogflow

The number of conversation tools has been increased more and more from the last ten years, at the same time as cognitive computing and low cost computers have brought the software to many people. These APIs based on natural language give us a good and none-exploited platform to make chat bots and conversational human computer interactions.

Usually, all these tools have in common parameters or fields to be configured like entities or intents. Nevertheless, despite the idea that these tools are very powerful, the platform has to be implemented for doing more than a simple conversation without any action or context. Once the conversational tool has been chosen we have to create a conversational agent [14]. Secondly we have to design the possible routes of the conversation with the entities to be required as a parameter if necessary in each intent and with the intents to pass across these entities.

All conversations with a bot must be based on a context and be related to previous searches. In this way, conversational agents can guide the conversation by asking for information (if any entity is required) or asking for confirmation from the user. In addition, the agents must be trained in the best possible way so that the interactions with the user are as optimal as possible.

However, this training can be done only by the agent studying the inputs of each user while the conversation is being developed using machine learning tools or can be done by the developer observing the answers provided by the agent and indicating which do not have been correct.

In this project, the conversational tool Dialogflow which was bought by Google one year ago like we said before will be used.

Figure 2.1: Dialogflow Tool [1]

Dialogflow [5] is a conversational tool which allows us to handle conversations from our applications. Basically, it is based on the use of entities and intents to generate some responses. However, this is usually the case with all the conversational tools, and what really makes Dialogflow more powerful is the possibility of using concepts such as actions, events or parameters.

The actions are specific to the intents and can be read by our program to execute or perform certain operations or desired tasks. Events can be triggered by our main application to start a conversation even if the user do not have said anything.

Figure 2.2: Modules of the platform Dialogflow [2]

In the other hand, the idea of context explained above allows the chat bot not to get lost during the conversation, knowing at all times the previous interactions if necessary. This tool allows us to define sequences of interactions and study different answers depending on what happened before.

Besides, another feature that not all conversational agents have is the possibility of modifying an automated response through the use of web hooks. This feature allows the conversational platform to use external data to modify its responses based on various parameters and is known in Dialogflow as fulfillment.

Finally, probably mention one of the greatest properties of this powerful tool that is the possibility of integration [15] with numerous applications such as: Google Actions, Line, Telegram, Skype or Facebook.

The platform of Dialogflow develops the conversational agents only in the web interface.

And it is here when we are creating agents when we really see the power of this tool. Firstly, we have to define what our bot will be used for, that is, specify the field in which our bot will become an expert to give it a suitable name. Once we have the conversational agent prepared, we have to think about the entities that we will need to respond to the user. Subsequently we must define the intents in order to recognize the possible entities created.

In case you need the agent to start the conversation without the human saying anything, it is the moment to declare these events. Also, it is time to add the actions, fulfillments or integrations that we want with the aim of making our conversational agent more powerful.

All these platform skills can be tested in real time from the web interface. It is enough to write or speak through the microphone of the platform so that the conversational agent can take out the answer that he considers appropriate.

Finally, after all these tools are defined, the conversational platform matches all the inputs asked by the user to the most suitable intent, considering the information named before like entities, events, actions, fulfillments and parameters required.

To achieve a good accuracy in the answers, it is important that the agent has as much entities and intents as possible. To make the learning process easier to use, Dialogflow has developed in the application a training tool that allows it to analyze previous conversation records and adding annotated examples to relevant intents or even informing the agent that the answer has been erroneous in some case. In other cases, it is possible to assign unmatched inputs which are marked by an exclamation mark, add inputs to one of the existing intents or construct a new intent with this input.

In conclusion, the agent will interact with the user only by the definition of a series of parameters and actions. This response will be better the more different ways the agent can take.

The platform is going to answer the input sending a JSON object as a query response which will contain all information about the intent including the answer speech. Currently, Dialogflow has the following SDK's to make text events or entities queries for all most popular programming languages like: Android, Botkit, C++, Cordova, HTML+Javascript, iOS, Java, Javascript, Node, Python, Ruby, Unity, Xamarin and more to come.

2.5 Joomla Content Management System (CMS)

Joomla [4] is a very powerful tool that requires little prior knowledge and allows us to create web pages.

This platform is a content management system for web developments: it is a kind of program to manage web pages. It is based on own standards of web developments such as HTML, CSS, JavaScript and PHP. Joomla is a web application, that is, it does not run on our local computer but on a server (although we can make our local computer work as a server). Joomla is free software: it is developed by a community of users who improve it, update it and make it available for its use. It is distributed with some basic contents and we call this base version "native Joomla".

The operation of Joomla is carried out thanks to its two main elements:

- The mySQL database: this is where all the information and most of the configuration of the system is stored, in an orderly manner and in different tables, each of which stores specific and determined information.
- PHP scripts: are those that execute the query actions and make changes to the database by converting the data into simple web pages interpretable by Internet browsers (Browsers) and perfectly intelligible for users and administrators.

This Content Management System works independently on the contents of its visual aspect [16]. This is a great advantage when redesigning or renewing a website in operation since it is not required to make changes to its appearance on the site. The design is worked independently and then it is loaded from its administration section on the web in a few minutes without altering its functionality with the users.

2.5.1 Structure of the platform

The structure of its code and operation is designed to allow easy understanding by search engines and web robots therefore it is a very interesting platform in this project.

In Joomla there are three very different elements that we must know and distinguish before going into detail. It is very important to understand what each of these elements does.

That helps us when we are developing these accessories ourselves as at the time of knowing how to distinguish which of them we need to look for to achieve the desired functionality [17].

Joomla uses a very specific nomenclature to refer to each of these elements:

- The components are always named as follows: com_component name. The prefix

"com_" is set to the name of the component.

- For the modules, the same system is used, only the prefix "mod_" is used.
- Finally the plugins, use the prefix "plg_"

2.5.1.1 Components

Perhaps the most complex elements of the three. A component of Joomla, is a program that we insert into our CMS. They are independent applications that manage information within Joomla. Components add different features to Joomla and make it much more than a web of articles or news.

In general, components have two distinct parts. On the one hand, if we access the backend of our Joomla, there is an item called "Components". Here we can see all the components installed in our Joomla portal and its configuration.

The other part of the components is the one seen in the public part of the portal. Usually, it is the central part of the page. In order to see a component in the public part of our website, it is necessary to create a menu item and associate it with the component.

The components are the software parts of our portal, which give the basic functionality to it.

2.5.1.2 Modules

Modules are much simpler elements to understand and create than components. They are small elements that are inside the structure of Joomla and are used to show the information received from the database, information of the elements, or characteristics that allow interacting with the site, for example the surveys, a visit counter, search form, etc ...

When we create a template, we define some positions or gaps, in which we can load our modules.

Examples of default modules that are preinstalled with Joomla are the search engine, the login module, or the menu module. When in the backend of our website, we create a new module, we must indicate in which position of the template will be positioned. Each template can define its own positions.

The modules are usually used to help the component they are associated with. For example, the search engine module, when we perform a search, does not redirect the search

engine component. The module would be the text field and the search button that we put in the position we want, and the component would be in charge of providing us with the results of the search.

Unlike components, we can have as many modules as we want on a page of our Joomla.

2.5.1.3 Plugins

Plugins are pieces of code that will be executed when a specific event happens.

Plugins are essentially small portions of code or mini-programs that can be called by several parameters or actions, for example, after activating a program, script or carrying out some function from a component or from the database . These actions are invariably taken before any type of output is sent to be displayed in the appropriate module or component, either from the site or the administration.

In essence, there are two types of plugins: those that are critical for the operation of the Joomla core programs and others that serve to optionally provide greater extensibility and functionality to the core programs or third-party extensions.

In addition to the types of plugins, within each type, there are different events to which they can respond. But to give an example, we can perform a certain action, each time a user logs in our site.

2.6 Conclusions

In this chapter we have introduce some of the technologies which are part of the project and conform the base for this embodied cognitive conversational agent.

It is necessary to understand first what a personal agent is for and how the Dialogflow tool works. We have also introduced the Joomla platform to show how it works and the different tools we can choose in this powerful content management system.

CHAPTER 3

Requirement Analysis

This chapter describes one of the most important stages in software development: the requirement analysis using different scenarios. For this, a detailed investigation of the possible use cases is made using the Unified Modeling Language (UML) [18]. This language allows us to specify, build and document a system using a graphic language.

3.1 Overview

We are going to describe the requirement analysis using different scenarios and cases.

"Requirements engineering is the process of conforming engineering designs to a set of core software requirements. This is critically important for creating accurate results in software engineering. Requirements engineering is also known as requirements analysis [19]."

These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications.

In order to explain correctly all the requirements, we will draw diagrams with UML [20] (Unified Modeling Language) in order to use a standard way to visualize the design of the system throw the use of graphic language.

To achieve that, we are going to identify the actors and their interactions, distinguishing the importance of their functions in the development of each case of use.

Secondly we will resume a complete specification of all the requirements based on the actors and the different environments.

3.2 System Actors

Identifying the actors of the system is an important step on the requirement analysis. Each actor specifies a role played by a user or even by other systems that interacts with the subject. They have a unique name, description and a goal that must be satisfied by the system.

The actors involved in the scenario are:

- Joomla (CMS): This actor will be the bridge between the end user and the module created. The Joomla platform is going to be the front end of the application. It has to interact with the module created and the plugin for the searches displayed on Joomla.
- User: End user of the system, and the main actor. Access the intelligent environment and interact with it by using questions through the chat application. He can make queries to the system by using the conversation application implemented.
- Dialogflow: This is another principal actor. It receives the user queries through the Joomla module and generates a response.

- Module Bot Dialogflow (Joomla): This actor will call the Dialogflow platform in order to answer the user query if it is an easy conversation or to call the Joomla JResearch plugin to get all the relevant information for the user query if it is a search conversation.
- Plugin JResearch (Joomla): This actor is imported from the copy of the GSI website. It is the plugin used on the web to perform searches with filters, so in our project it will be a main actor for conversations with staff searches, publications or projects which are three areas that the plugin knows how to differentiate.

3.3 Use cases

This section identifies the use cases of the system. This helps us to obtain a complete specification of the uses of the system, and therefore to define the complete list of requisites to match.

First, we will use the actors involved in the system listed before and a UML diagram representing all the actors participating in the different use cases. This representation enables, apart from specifying the actors that interact in the system, the relationships between them.

These use cases will be described in the next section and we will show each type of data used from the Intelligent Systems Group (GSI) web page. Of all the information mentioned in the introduction that stores the website designed in Joomla of the GSI, only three of them will be focused for this project: the information about the members of the group, their publications and the projects the group has taken part of. Each type comes from a different part of the website, and therefore will be considered independently. Using these tables, we will be able to define the requirements to be established. In the following picture 3.1, it is shown the principal application uses cases and the connections between the system actors.

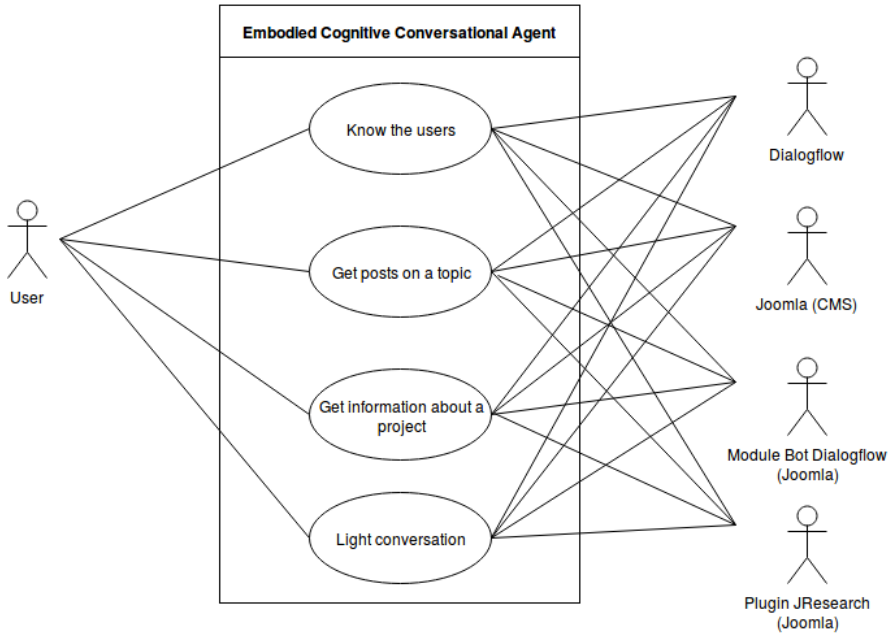


Figure 3.1: Use Cases Diagram

3.3.1 Know the users - Use Case

The main actor starts this use case when he asks to the chat bot information about the staff of the GSI. The user enters to the embodied conversational agent and starts throw Joomla a conversation with the other actor: Dialogflow.

When the main actor enters a query related to the information about GSI members, the conversational agent identify the intent and starts looking for the information in the database. If there is a match between the searched member and the database, the information is presented to the user on the Joomla module. If not, the agent will ask the user for more information if he has made a mistake writing or wants to reformulate the question.

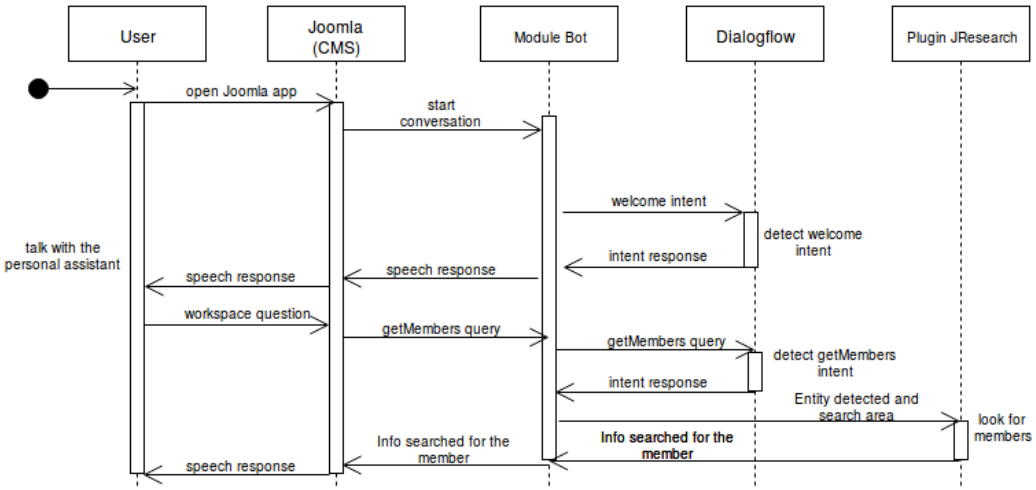


Figure 3.2: Know the users Sequence Diagram

3.3.2 Get posts on a topic - Use Case

The main actor starts this use case when he asks to the chat bot information about the publications of the GSI. The user enters to the embodied conversational agent and starts through Joomla a conversation with the other actor: Dialogflow.

Like before, when the main actor enters a query related to the information about GSI publications, the platform identifies the intent and starts looking for the information. By the same steps, the agent extracts the information and answers the user with the data stored or indicates to the user that there has been no match in the search.

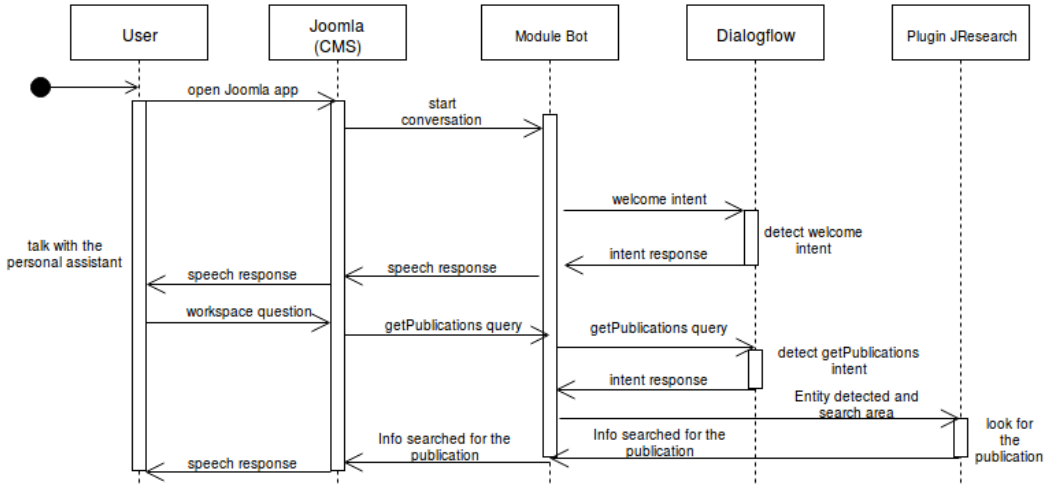


Figure 3.3: Get Posts on a topic Sequence Diagram

3.3.3 Get information about a project - Use Case

Another time, the main actor begins the conversation with the other actor. Using the Joomla module he makes a new query to the conversational agent but this time about the GSI projects. In the same way as in the other two use cases, the agent consults the external data and presents the requested information to the user in case it has found it.

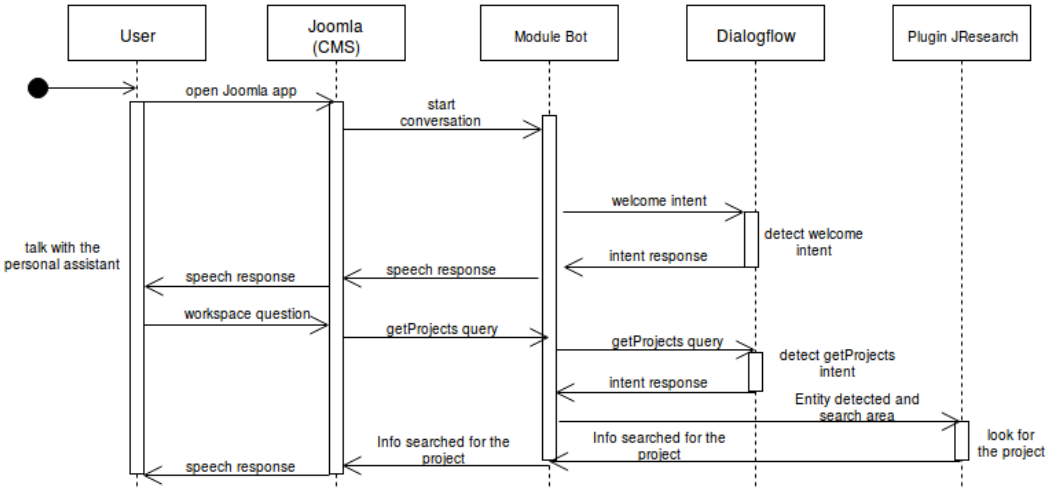


Figure 3.4: Get information about a project Sequence Diagram

3.3.4 Light Conversation - Use Case

The main actor starts this use case talking with the chat bot. In this use case, the conversation will not perform any information search and it will simply be a light conversation. The user can ask questions and talk with the chat bot so that the conversation can be as human as possible.

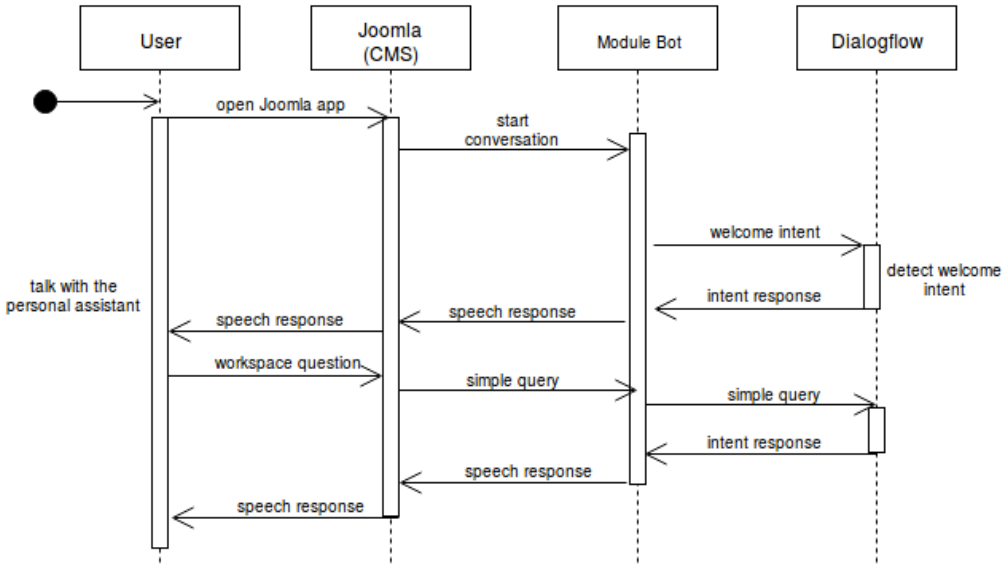


Figure 3.5: Light Conversation Sequence Diagram

3.4 Conclusions

After that description of the use cases, we have presented the basic functionalities that we have implemented in the project. By the diagrams above, we can understand the way in which the actors interact with each other as well as the flow of the application since the user needs a type of information until he obtains it. However we can see that the use cases are very similar, and for a future we could start thinking about different use cases that provide new functionalities or even new information to the user in order to complete the field covered by the conversational agent.

With all that, we can conclude that this is a large field of development, therefore it can be part of future lines of work to be able to generate a more complete assistant and with all the information of an organization such as the GSI.

CHAPTER 4

Architecture

The main purpose of this chapter is to explain the architecture of this project. Firstly, we will focus on the global vision of the architecture in the overview, looking at the modules which form the system and their connections. Later, each module will be presented explaining its function in this project.

4.1 Overview

In this section the global architecture of the project will be presented, defining the different subsystems that participates in the entire system.

In the previous chapter, the main actors of our work have been described. Now, the connection between those actors and how they are formed will be illustrated.

When the user types a query in the chat bot, the first thing our created module will do is capture the information and send it to Dialogflow. This principal actor will be responsible for collecting the query, identifying and classifying it according to the rules that has been defined as explained in chapter 2 and will return a JSON with the answer that it considers most appropriate. The answer will be received by the module and at this moment it is when the type of conversation is decided. If the name of the intent sent by Dialogflow does not match "getMembers", "getPublications" or "getProjects" which are the intents that are triggered to look for the staff information, publications or projects respectively, then the answer which is going to be displayed will be extracted from the JSON received.

On the other hand, if the name of the intent matches the above mentioned intents then the simple conversation will be abandoned and a call will be made to the JResearch plugin. This last actor will receive for its correct search an area (based on the name of the intent) and the entity which is extracted from the initial query. With this data the plugin will look in the received area for an exact match with the entity and will return to the module a response with the requested information or an error message.

The following figure 4.1 illustrates this described architecture, and in the following sections they will be detailed in a deeper way.

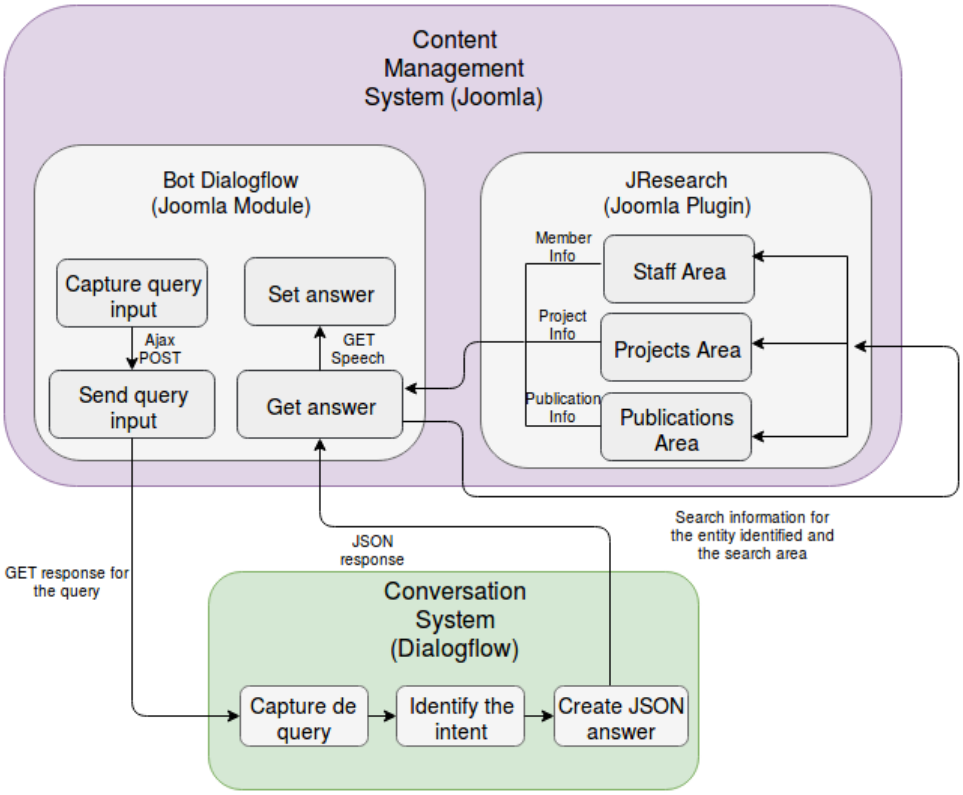


Figure 4.1: Architecture

4.2 Content Management System

As it was said in the chapter 2, Joomla is a computer application that supports the creation and modification of digital content.

Also, as it was said in the introduction, the GSI website is made with Joomla. This web has a series of Joomla components and modules that will help us when we will look for information, so the basis of the project will be based on a copy of the web and the database called gsibbdd in order to have all the information about the Intelligent Group System. After that, the web page of the GSI has been linked to the database in order to have a complete web site.

4.2.1 Joomla Module: Bot Dialogflow

In this scenario that replicates the website, a Joomla module [21] has been developed named **mod_botdialogflow** which contains all the logic to communicate with Dialogflow and to call the plugin if the user looks for some information of the web site.

The module behaves a little differently depending on the conversation that is taking place. The following figures 4.2 and 4.3 show the two possible behaviors in a general way. In the following subsections the three main scripts of the module will be explained in detail:

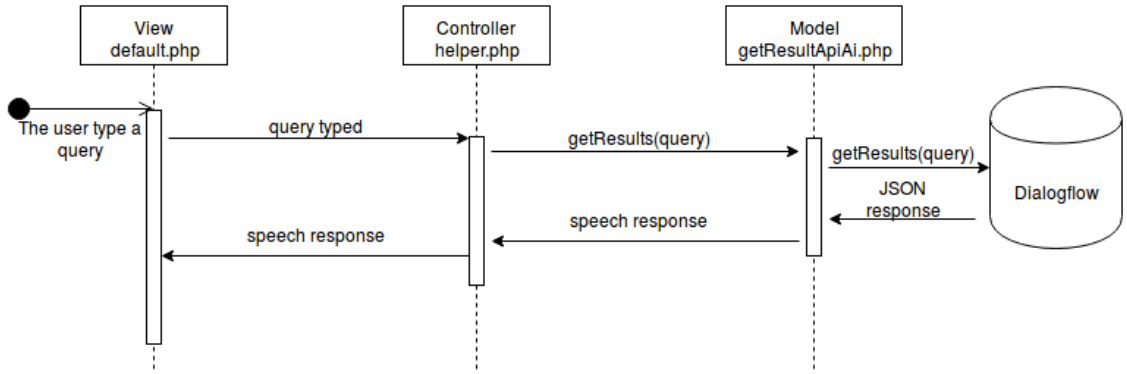


Figure 4.2: Module Dialogflow Light Conversation Sequence Diagram

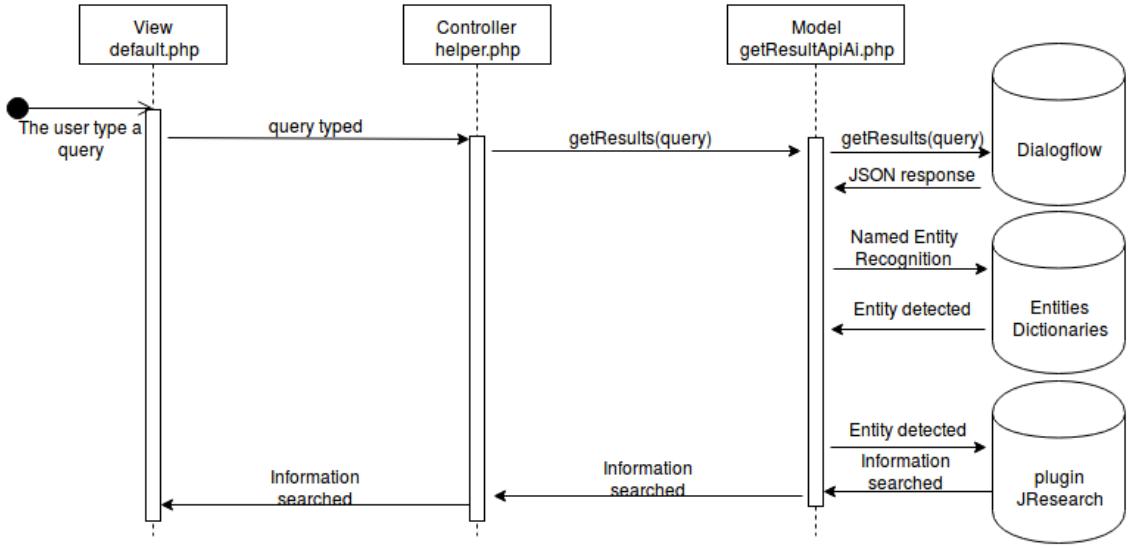


Figure 4.3: Module Dialogflow Search Conversation Sequence Diagram

4.2.1.1 default.php

Firstly, this script is the graphic part of this module for Joomla. This file generates the HTML code that is displayed on the page with the data which are set. It shows a box to interact with the user which will store the entire conversation as well as the input that will allow the user to ask their questions. The generated HTML consists of a box that simulates a chat window. This box is separated in three parts: the header that contains a title to show that it is a chat and an image of the bot that will be in charge of saying the answer aloud, a central part with the whole conversation with messages and at the bottom a space where the user can type his query and send it.

The style of the conversation box follows the basis of an example of bot [22] to which some personal touches and an image which display a little gif that will read the answer given to the user out loud have been added. Also, bootstrap has been used to include some icons and default styles in order to accompany the original colors of the web. To ensure that the image of the upper header is able to read the response aloud, an animated image is reproduced synchronously with the Speech Synthesis Web API [23] to retrieve information about the synthesis voices available. For this, the talkr app repository code [24] has been used and the system gifs have been modified. Several gifs have been created so that depending on which is deployed in Joomla it can be changed. In addition, the chosen gifs should be face to face and moving their mouths so that when they are talking they coincide with the reproduction of the response and for the user more real.

The following picture 4.4 shows the result of the HTML code with all the styles applied and with the same icon: one talking and the other don't.

Figure 4.4: Chat Box

This script is also in charge of getting the input of the user and retrieve it to the controller named helper.php by a POST request. The process of this action is basically that the user writes the query on the input and submit it. At this moment, the view executes an Ajax

request [25] using the component of Joomla named **com_ajax**.

The Joomla ajax interface [26] is a lightweight and extensible component designed to act as an entry point to HTTP requests for independent plugins and modules stored in Joomla, allowing them to use the potential of Ajax functionality.

In order to get a success response, the following parameters in the request have to be set:

- *data*: this variable contains the information that will be sent to the controller script. Therefore, the query collected by the input of the box has to be sent here.
- *url*: it must contain the com_ajax option, the module to which we want to send the request, in our case it is the mod_botdialogflow module and the method that will be executed in the destination, in our case is named setQuery.
- *success function*: the callback action that is going to be run if the POST request has been successful. In our case the answer given by the controller is set in order to show it in the conversation box.

4.2.1.2 helper.php

This file contains the helper class that is used to do the real work in retrieving the information that is displayed in the module.

In our case, the principal function of the controller is to receive the POST request of the previous script and to get the answer from the model.

For that, once the data sent by the POST request is detected and it is verified that it is not empty, the script that allows us to consult the Dialogflow platform is called.

Finally when the request to the conversational tool has been successful and the model has the correct information, the answer is returned to the view in order to show it to the end user.

In the following subsection the model will be explained in detail and how it works.

4.2.1.3 getResultApiAi.php

Dialogflow has many possible integrations. Some simpler ones that they call "one click integrations" and others that they call "Dialogflow SDK's". As the basis of the chosen

website is into Joomla and our code is in PHP, the integration has been in PHP in order to have a uniform code with the same technology.

In order to do that, the conversational tool provides the Dialogflow tool SDK that helps us to configure the integration [27].

Using the low level query and importing the files of the repository a client with the access token of our Dialogflow platform is created. Sending the query as a parameter a response of the conversational tool is obtained.

However, to get the message which has to been displayed in the front-end, the JSON response has to be analyzed. If we see the figure 4.7 and evaluate the JSON response we see that at first the attribute "result" has to be obtain to then get the "fulfillment" and finally access the value of "speech". The attribute named speech is the answer which is set in the conversation box.

This process is the one followed in the case of a light conversation. However, if it is a conversation where there is a search for members, publications or projects on the web, the model will keep the name of the intent instead of the speech. For that a simple rule to categorize the inputs is defined: if the intent name of the query identified by Dialogflow is getMembers, getProjects or getPublications it is a search query. Otherwise, it is a simple query.

In the following figure 4.5, the operations of the system depending on the type of the query that is recognized can be seen:

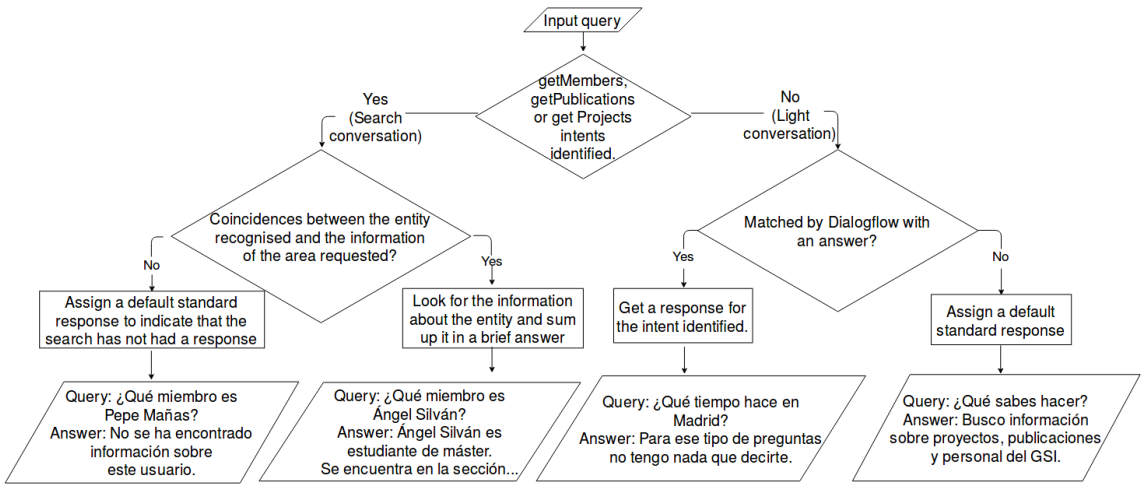


Figure 4.5: Flowchart

After that, the model will send the obtained data to the controller. If it is a normal conversation, the speech obtained from Dialogflow will be sent directly.

However if the conversation has been categorized as a deep search and the basic flow has been abandoned, the search is performed. To get a match on what the user is looking for and the data on the web, three dictionaries have been created with the entities of the three areas. Through three calls to the database, three arrays are obtained with the following information: all the full names of the members of the GSI, all the titles of the projects of the GSI and all the titles of the publications of the GSI. With these three dictionaries, a named-entity recognition (NER) is made to extract the entity that searches from the user's query.

If there is a match and the entity is extracted, then a call is made to the JResearch search plugin. However, if in the user's query there is no entity that matches any of the three dictionaries then no search is performed and the user is notified that there is no information about their request.

4.2.2 Joomla Plugin: JResearch

This plugin is responsible for obtaining the information if it has left the normal process of a simple conversation. In the copy of the initial web that was taken, this plugin was already imported so it can be reused for our searches. This plugin is responsible for making search filters on the GSI website by areas and words so it is perfect for us to search our three types of information.

To obtain optimal data in the model and not having false information, the first thing that should be done is to extract the keyword from the search as it has been said before with the Named Entity Recognition.

The plugin will receive as parameter the area of the information searched: staff, projects or publications and the entity detected by the NER. With all that information, the JResearch plugin will be in charge to collect all the data which will be displayed on the view of the module.

If there is no match and the plugin does not run, it will be shown a warning message to the user in the view of the module that there has been no match and that he or she must enter the name or entire title to redo the search.

4.3 Conversation System

The main objective of conversational agents is to transmit information to people but through the use of a human conversation in which only one person interacts, who is ultimately the one who asks for and receives the information.

Like we have previously told, an implementation with Dialogflow has been made using PHP.

Dialogflow application as already said in chapter 2, allows us to make a conversation with the agent, with plenty of functions, integrations and parameters to characterize the conversation. For this reason we have choose it.

To make our conversational agent the Dialogflow web interface has been used [28]. With that interface, firstly the entities have been designed, which are combinations of words with the same meaning that can appear during the conversation.

Secondly, the intents have been developed, which are based on user queries. To achieve optimal performance and in order to get correct answers, the queries of an intent have to be similar and with resembling meanings. Also, as already explained above, these intents contain a context field to organize the conversation and so the agent does not get suddenly lost. The answers can be enriched by the use of external data called web-hooks.

The process of assigning a response through Dialogflow is quite simple. With entities and intents created, when a query arrives, the first thing that the conversational agent does is identify with which intent it has more similarity. Once the intent has been obtained, the conversational agent makes a call to the web hook if necessary to obtain additional information.

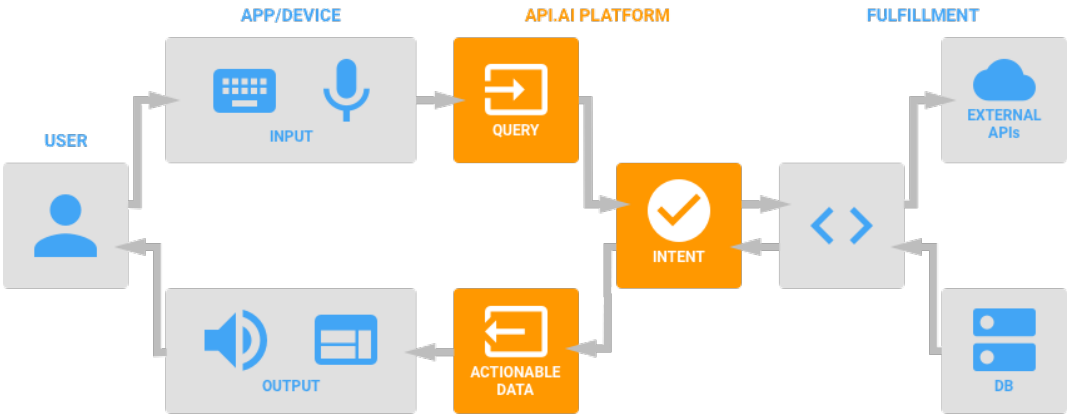


Figure 4.6: Dialogflow flow [3]

With all this information, the agent creates a response JSON 4.7 with all the data and sends it.

JSON


```
{
  "id": "771267f7-af75-4c38-8f8e-496508770908",
  "timestamp": "2017-12-12T12:24:14.763Z",
  "lang": "es",
  "result": {
    "source": "agent",
    "resolvedQuery": "Busco información sobre los proyectos",
    "action": "showProjects",
    "actionIncomplete": false,
    "parameters": {
      "Proyectos": "Proyectos"
    },
    "contexts": [],
    "metadata": {
      "intentId": "077b4448-f4b0-47f5-8ba1-44a5c5caa967",
      "webhookUsed": "true",
      "webhookForSlotFillingUsed": "false",
      "webhookResponseTime": 1500,
      "intentName": "getProjects"
    },
    "fulfillment": {
      "speech": "Hello from projects",
      "source": "agent",
      "displayText": "Proyectos",
      "messages": [
        {
          "type": 0,
          "speech": "Hello from projects"
        }
      ]
    },
    "score": 0.5199999809265137
  },
  "status": {
    "code": 200,
```


Figure 4.7: Example of a JSON document from Dialogflow

4.4 Conclusions

In this chapter, an architecture fully modular has been proposed at the beginning in which each component can be developed, maintained and deployed separately as well as a short description for each module. Then they have been explained in detail, analyzing its functionality, as well as how each module interacts with each other.

Finally, in order show how it works, two possible case studies will be considered for the system. The first one with a simple conversation between the end user and the conversational tool and then a second case study which will be a deep conversation asking for some information of the web site.

CHAPTER 5

Case Studies

In this chapter two selected case studies will be explained. The running of all the tools involved and its purpose will be shown. First a prototype of a bot developed for the GSI web page following the architecture described previously will be explained. It is based on how through a module created in Joomla you get the possibility of having a simple conversation with Dialogflow.

After, the second case study will be shown. For this second example, a more advanced conversation will be displayed. It is based on how through the same module created in Joomla you get the possibility of having a deep conversation with Dialogflow asking for information of the members, projects and publications of the GSI.

5.1 Case Study: Light Conversation with Dialogflow

5.1.1 Overview of the system

For this system, we have developed a prototype utilizing the architecture described in chapter 4.

First, we will identify the main actors in this case study:

- The end user starts a light conversation through Joomla platform.
- An agent on Dialogflow, that handles simple questions from the user. The conversational agent has the objective of answering questions of a light conversation. In this way, the chat bot can provide an approach to the user showing that the conversational agent can interact with the end user and follows a simple conversation.

5.1.2 Overall Process

In this section it is presented a brief explanation on how each module of the system interact with each other and with the main actors. In the following figure 5.1 we can see a simple diagram of the process of the system:

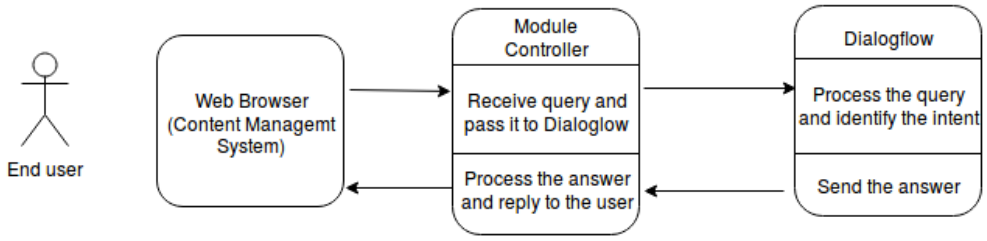


Figure 5.1: Overall cycle for the system

During the simple conversation with the chat bot the agent will always ask first what the user needs and then there are many topics that the user can deal with. The user can say hello, say goodbye or find out what the chat bot is like. The end user can also ask for information about the chat bot and he can even investigate which areas the chat bot covers if he want to go deeper into the conversation.

5.1.3 Light Conversation Use Case

This case study begins when the user enters a query in the bot window. At this moment the controller opens a request to Dialogflow to obtain the response from the platform and to take it out on the screen. Additionally, the bot will answer aloud reading the response provided. If the user needs to listen to it again, simply clicks on the chat icon to repeat the answer orally.

In this use case, the queries are classified by the agent and it assigns an immediate response without having to go through the search plugin.

The following pictures 5.2 show a few examples of the type of light conversation that the agent can make for the light conversation use case:

Figure 5.2: Examples of light conversation

5.2 Case Study: Deep Conversation with Dialogflow

5.2.1 Overview of the system

For this system, we have developed a prototype utilizing the architecture described in chapter 4.

First, we will identify the main actors in this case study:

- The end user is going to start a deep conversation through Joomla platform.
- An agent on Dialogflow, that will handle questions from the user. The conversational agent has the objective of answering questions of a deep conversation. In this way, the chat bot has the abilities of the previous case study but now the chat bot will understand when the user asks for some information and will leave the simple conversation to perform a search.
- A plugin on Joomla which can search in the database of the GSI for information like members, projects o even publications.

5.2.2 Overall Process

Now we will explain how each modules work in this case study and how they interact with the main actors. The process will be the same as before. However, the agent will send a different action if it identifies that the query is about a research of members, projects or publications. With that information the module controller will leave the conversation and call the search plugin to see if there is a match. If there is a coincidence, it will provide the user with the data obtained and include a link to view the information in more detail.

To get an idea of how the system will work now, we have this diagram:

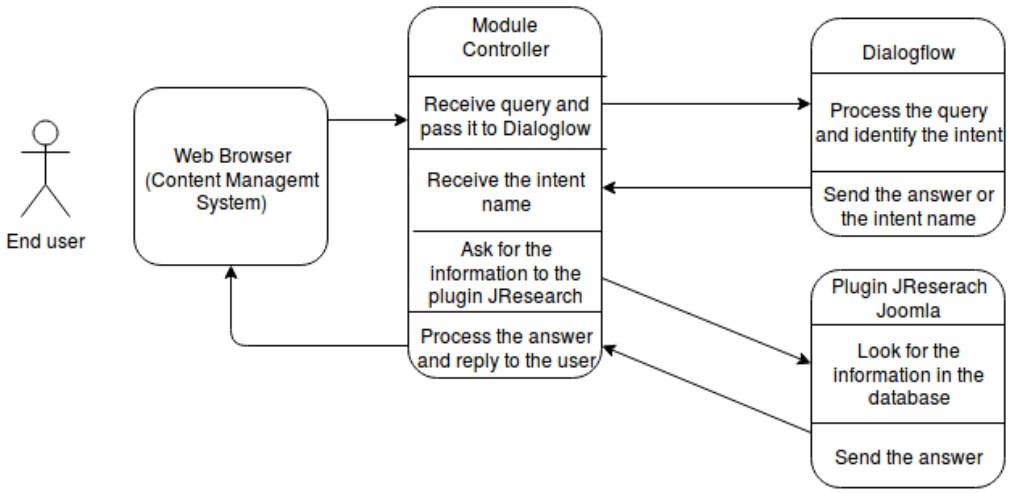


Figure 5.3: Overall cycle for the system

During the conversation the agent will always ask first what the user needs as in the previous case study. So, the user can start by asking information about the members, the projects o the publications of the GSI.

In the three cases of use described in chapter 3, the same systems appear. The change will be the information requested by the user on the web. In the same way, in the three cases of use the bot will show a response message but it will also read it out loud so that the user can hear it.

5.2.2.1 Know the users Use Case

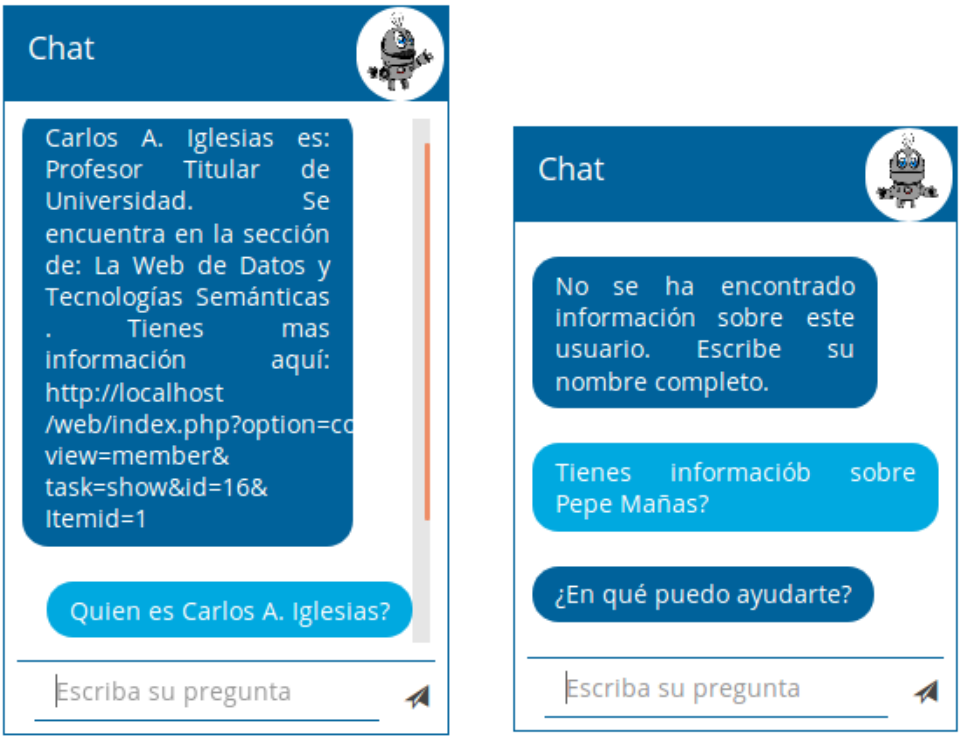
In this use case, the user will ask to the chat bot for information about the GSI staff. The Joomla module will call the JResearch plugin and search for the information stored on the name of the requested user.

If it finds a match, the end user will be informed with a short summary about the position of the user, the section where he works and a link to that user's profile.

On the other hand, if there is no match, it is because the requested name is not found in the database. That it may be due to a typing error, so the chat bot will request more detail to the user in order to indicate either an additional surname or to correct the error that has been committed.

The following pictures 5.4 show an example of conversation for the know the users use

case. Looking for two members, one of them exist but the other don't:



(a) Found (b) Not found

Figure 5.4: Examples of member searches

5.2.2.2 Get information about a project Use Case

In this use case, the user will ask to the chat bot for information about the GSI projects. The Joomla module will call the JResearch plugin and search for the identified project on the query to get the right information.

If it finds a match with the title of the project extracted from the query, a brief summary will be compiled as in the previous case but this time it will only show the section where the project is store, the date of creation of the project and a link to access to that project.

However, if it does not find any information it is because the extracted title of the query is not in the database and therefore it will send a response message warning the user that there have been no coincidences and that in case of writing it wrong or incomplete that the users asks again.

The following pictures 5.5 show an example of conversation for the get information about a project use case. Looking for two projects, one of them exist but the other don't:

(a) Found (b) Not found

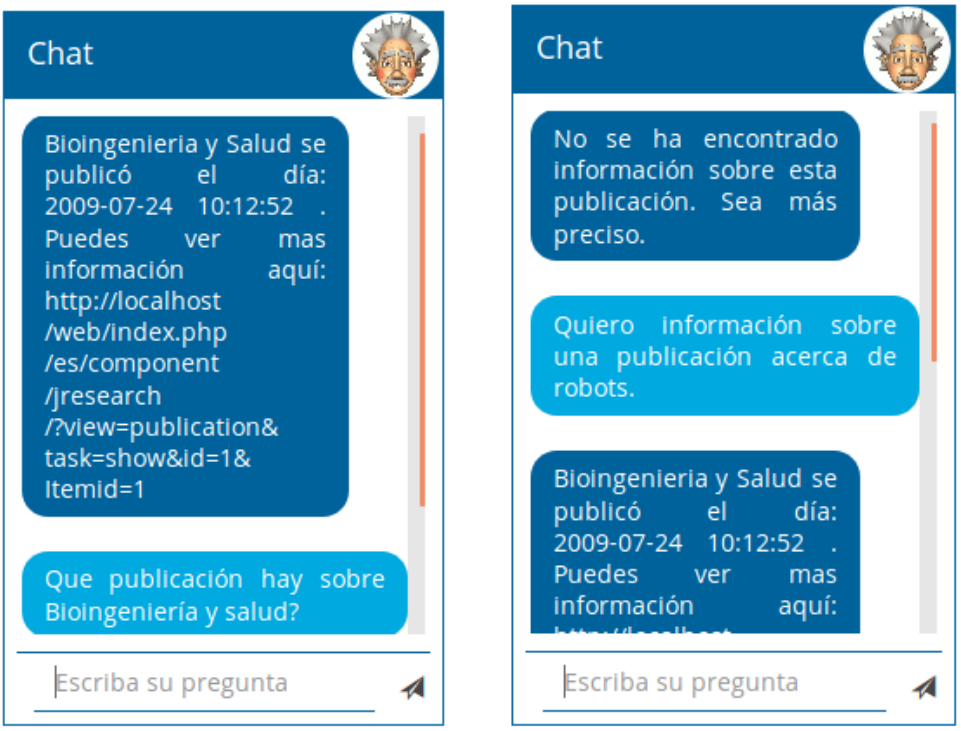
Figure 5.5: Examples of project searches

5.2.2.3 Get posts on a topic Use Case

In this last case of use studied, the user will search information about publications on the web. For this, the same operation will be followed as before.

If the plugin obtains a match then in the response message will appear the section of the publication, the date of creation and a link to access the detail. If the plugin does not get any match then it will warn the user that it has not found the requested information.

The following pictures 5.6 show an example of conversation for the get posts on a topic use case. Looking for two publications, one of them exist but the other don't:



(a) Found (b) Not found

Figure 5.6: Examples of posts searches

5.3 Summary

In this chapter a prototype that allows the agent to answer easy questions from a user has been shown. To sum up, this case study allows the user to cover all the basics questions which can be asked by a user.

In the second case study a deep conversation has been tested obtaining internal data from the web. Indeed, through the JResearch plugin installed in Joomla we managed to extract data from the GSI database and thereby respond to the end user in the most optimal and concise way.

With this second example we can observe the powerful tool that is Dialogflow which complemented with a database allows us to give a series of answers to satisfy the needs of the user.

CHAPTER 6

Conclusions and future lines

In this chapter we will describe the conclusions extracted from this work, take a look at the goals we started in this project and its completion to finally propose several possible improvements for future work.

6.1 Conclusions

During this project we have developed a conversational agent that allows users to both have a simple conversation and perform a search for information on the web. Now we are going to present the conclusions deduced.

We started this work establishing several requirements that we wanted to achieve and the prototypes developed for it. With currently available technologies it is possible to build bots using conversational agents and linked data to help users find the desired information. The construction of these personal assistants comes from the knowledge of the technologies used and how to integrate them among them.

The processing of natural language as well as its understanding has allowed us to obtain an adequate interface for the user. However, it is important to note that with the Dialogflow tool the fact of having to declare all the intents for each type of response makes it easier to perform an expert agent in a sector instead of an agent who knows a lot of information.

Developing two case studies, we have verified that our system is capable of adapting to different areas of conversation although as we have mentioned previously they are limited to the intents declared in the conversational agent.

Finally, the offer of our conversational agent as a web application has proved to be a fundamental tool to spread the use of our system, since it allows access from any device with an Internet connection.

6.2 Achieved goals and problems faced

In the first chapter we discussed the goals we wanted to achieve in this project. In this section I am going to explain the achieved goals.

- Design a personal assistant who is able to have a light conversation: This was one of the main points of our development and to achieve it we have proposed an architecture that can attain that objective.

- Integrate external information to solve questions related to the web page: This was the other main point of our work. Through the classification of the queries with the use of intents of Dialogflow we have managed to separate the cases of light conversation and search conversation shown in the case studies of chapters 5 and 6 respectively.

- Integrate the personal assistant: Thanks to the content manager Joomla we have been able to unify all our work separating it into modules and plugins. The CMS and its tools have allowed us to create a copy of the gsi website by importing the database and thus focus on the construction of a conversational agent.
- Testing different cases and analyzing the results: Finally we have used two case studies with different ways to verify that our conversational agent was able to respond and show the requested information.

Despite having satisfactorily fulfilled the objectives set at the beginning of the work, we can say that the main problem of this work has been the first contact with the world of conversational agents and their functioning.

On the other hand, manage user requests and classify them based on the content has been another main problem. We had to choose between performing a very specific search or otherwise settle for a general search. As we know that the user who is looking for a member or project ig going to know the title or his name and want to see more information we have decided to follow the first type of search. In this way, if the user searches for a very general project or a very common name, we will avoid showing too many answers in the chat and avoid losing the user.

6.3 Future work

There are several lines than can be followed to continue and extend features of this work.

In the following points some fields of study or improvement are presented to continue the development:

- *Increase in the field of work:* The first point to try could be to increase the work area of the conversational agent. As we know that it is necessary to create intents and answers for each type of information we could think about increasing the field covering the news or events of the Intelligent Systems Group.
- *Adding audio interface for the user:* Currently our bot is able to interpret the queries written by the user and answer both written and oral. Therefore, in the future it could be useful to implement a key to send a recording and so the conversational agent can be totally by voice in order that users with low technical capabilities can use it.
- *Keep the chat bot in the navigations in the web:* The chat bot view has been implemented so that when the user navigate on the GSI website, the history is cleaned to

avoid accumulating many messages. Another future line could be to think of a larger location for the chat so that more messages can be stored without putting a lot of scrolling and thus saving searches from one navigation to another.

In conclusion, we have developed a robust and functional system, but it can still be greatly improved with new modules, enhancing the user experience. New developers could take this system as a base to develop better and more powerful personal assistants with more functionalities and a bigger search field.

Bibliography

[1] Dialogflow, "Figure of the icon of dialogflow," 2017.

[2] Dialogflow, "Figure of the modules of the platform dialogflow," 2017.

[3] Dialogflow, "Figure of the dialogflow flow," 2017.

[4] Joomla!, "Joomla free open-source content management system (cms) - https://www.joomla.org/," 2016.

[5] Dialogflow, "Dialogflow getting started basics - https://dialogflow.com/docs/getting-started/basics," 2017.

[6] C. C. Consortium, "Cognitive computing definition - https://cognitivecomputingconsortium.com /resources/cognitive-computing-defined," June 2014.

[7] Google, "Actions on google for build apps for the google assistant - https://developers.google.com/actions/dialogflow/," 2017.

[8] C. C. Consortium, "What is cognitive computing consortium? - https://cognitivecomputingconsortium.com," 2014.

[9] J. E. Kelly III and S. Hamm, *Smart Machines IBM's Watson and the Era of Cognitive Computing*. Columbia Business School, 2013.

[10] I. B. Machines, "Ai and cognitive computing solutions with ibm power systems - https://www.ibm.com/power/solutions/cognitive-computing," 2017.

[11] Oracle, "Intelligent bots definition - https://www.oracle.com/es/solutions/mobile/bots.html," 2017.

[12] R. Kahn and A. Das, *Build Better Chatbots: A complete guide to getting started with Chatbots*. Apress, 2017.

[13] C. Weller, "Ibm speech recognition is on the verge of super-human accuracy - http://www.businessinsider.com/ibm-speech-recognition-almost-super-human-2017-3," March 2017.

[14] J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, *Embodied Conversational Agents*. Massachusetts Institute of Technology, 2000.

[15] Dialogflow, "Dialogflow integrations - https://dialogflow.com/docs/integrations/," November 2017.

[16] D. Mazier, *Joomla! 3.3 Cree y administre sus sitios Web.* Eni, 2015.

[17] D. Rahmel, *Beginning Joomla!: From novice to professional.* Apress, 2008.

[18] U. M. L. Organization, "What is unified modeling language? - http://www.uml.org/what-is-uml.htm," July 2005.

[19] Techopedia, "What does requirements engineering means? - https://www.techopedia.com/definition/21697/requirements-engineering," 2017.

[20] S. W. Ambler, *The Elements of UML 2.0 Style.* Cambridge University Press, 2005.

[21] J. LeBlanc, *Learning Joomla! 1.5 Extension Development: Creating Modules, Components and Plug-Ins with PHP.* Packt, 2007.

[22] Moinisim, "Css styles chat bot - https://botui.org/," October 2017.

[23] M. D. Network, "Speech synthesis web speech api - https://developer.mozilla.org/en-us/docs/web/api/speechsynthesis," December 2016.

[24] Buzfeed, "Respository with an example of animate gif file in sync with the speech syntehsis api - https://github.com/talkr-app/gif-talkr," November 2017.

[25] D. Rahmel, *Advanced Joomla.* Apress, 2013.

[26] D. Rahmel, *Professional Joomla!* Wrox, 2007.

[27] Iboldurev, "Php sdk for dialogflow - https://github.com/iboldurev/dialogflow," November 2017.

[28] M. McTear, Z. Callejas, and D. Griol, *The Conversational Interface: Talking to Smart Devices.* Springer, 2016.