Asistente virtual para servicios de la biblioteca de la UCM - Janet

Trabajo de Fin de Grado
Curso 2018-2019

Autores
Mauricio Abbati Loureiro
Jose Luis Moreno Varillas

Director
Alberto Díaz Esteban
Antonio Fernando García Sevilla

Grado en Ingeniería del Software
Facultad de Informática
Universidad Complutense de Madrid

Asistente virtual para servicios de la biblioteca de la UCM - Janet

Trabajo de Fin de Grado en Ingeniería del Software
Departamento de Inteligencia Artificial

Autor
Mauricio Abbati Loureiro
Jose Luis Moreno Varillas

Dirigido por
Alberto Díaz Esteban
Antonio Fernando García Sevilla

Grado en Ingeniería del Software
Facultad de Informática
Universidad Complutense de Madrid

30 de mayo de 2019

Dedicatoria

Quiero dedicar este proyecto a mis padres, Marco Abbati Pellegrini y Concepción Loureiro Amor, y a mi hermano, Diego Abbati Loureiro, que durante todos estos años han creído en mí, dándome todo su apoyo y confianza.

Mauricio.

La realización de este proyecto ha sido posible gracias a la fe ciega y confianza que mis padres, María Isabel Varillas y Jose Luis Moreno Clemente, han depositado en mi durante toda la carrera, incluso en los momentos de flaqueza, que es cuando más lo necesitaba. Este proyecto es para vosotros.

Jose Luis.

Agradecimientos

En primer lugar queremos agradecer a nuestros directores, Alberto Díaz Esteban y Antonio García Sevilla, por el apoyo y dedicación que han puesto en este proyecto.

También, queremos agradecer a todo el personal de la biblioteca de la Universidad que han hecho posible este proyecto, en concreto:

- A Gumersindo Villar García Moreno, por ponernos en contacto con el personal de la biblioteca central.
- A Pilar Gómez Bachmann, Isabel Gutierrez Sánchez y Maria Luisa Martínez Ordóñez, por ayudarnos a analizar qué funcionalidades serían útiles para este proyecto.
- A Inmaculada Fernández Sáez y Eugenio Rubén Tardón González, por explicarnos y ayudarnos a implementar las funcionalidades del catálogo de la biblioteca en el proyecto.

Por último, agradecemos la ayuda prestada por Alberto Banet para enviar la aplicación de iOS a la App Store.

Resumen

Este proyecto nace con la idea de desarrollar un asistente de voz para facilitar el acceso a los servicios de la biblioteca de la Universidad Complutense de Madrid a través de un dispositivo móvil. Para ello, se han utilizado tecnologías de `procesamiento de lenguaje natural` para conseguir que el sistema reconozca las peticiones que un usuario le realiza y ser capaz de devolver una respuesta apropiada en un contexto acotado. Todo ello, simulando el comportamiento de un ser humano para que el sistema sea amigable. Así pues, con el objetivo de obtener la información de ejemplares solicitados en la aplicación, se hace uso del catálogo online de la biblioteca de la UCM. Además, se utiliza una base de datos propia que almacena información básica de las bibliotecas con el fin de responder a preguntas sobre datos institucionales de estas.

Puede acceder a todos los recursos, así como ejecutables, código y scripts del proyecto en el siguiente enlace

`https://github.com/NILGroup/TFG-1819-Biblioteca`

Palabras clave

Asistente virtual, Chatbot, Buscador, Biblioteca, Accesibilidad, Rasa, spaCy, OCLC, WMS

Abstract

This project was born with the idea of developing a voice assistant to facilitate access to the library services of the Complutense University of Madrid through a mobile device. To this end, technologies have been used `natural language processing` to ensure that the system recognizes the requests made by a user and be able to return an appropriate response in a limited context. All this by simulating the behaviour of a human being so that the system is friendly. Thus, in order to obtain the information of copies requested in the application, use is made of the online catalogue of the library of the UCM, and in addition, an own database is used that stores basic information of the libraries in order to answer questions about institutional data of these. Resources, such as executables, code and sripts of the project can be accessed at the following link

`https://github.com/NILGroup/TFG-1819-Biblioteca`

Keywords

Virtual assistant, Chatbot, Search engine, Library, Accesibility, Rasa, spaCy, OCLC, WMS

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

"Salutaciones terrícola, quiero decir, lector."
— Siri, Apple.

1.1. Motivación

En los últimos años, el avance de los dispositivos móviles y de la nube han cambiado la forma en la que nos comunicamos y accedemos a la información.

Con la premisa de mejorar todavía más el acceso a internet y a su información, hace casi 8 años aparece el primer asistente de voz inteligente comercial para dispositivos móviles, permitiendo al usuario realizar cualquier consulta en internet simplemente formulando una pregunta a su dispositivo con la voz.

Algunas de las tecnologías que se desarrollaron, y se desarrollan actualmente, para el funcionamiento de estos servicios implementan tecnologías como la inteligencia artificial o el aprendizaje automático ("Machine Learning"[1]), permitiendo a los sistemas mejorar, ya sea prediciendo las posibles consultas o ofreciendo una respuesta más concreta a medida que más usuarios utilizan el servicio.

Con el paso de los años, cada vez más instituciones y empresas hacen uso de estas tecnologías para facilitar el acceso a información y llegar a la mayor cantidad de gente posible. En nuestro caso, el número de personas que acceden a la universidad cada vez es mayor, y por ello aumenta la diversidad entre estas.

Actualmente, el servicio de la biblioteca de la Universidad Complutense de Madrid carece de este tipo de tecnología, teniendo un buscador alojado en la web, por lo que para algunas personas puede no resultar fácil, teniendo que dedicar más esfuerzo del necesario a la hora de hacer una consulta o

[1] `https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico`

directamente personarse en la biblioteca de su facultad.

1.2. Objetivo

El objetivo de este proyecto consiste en trasladar estas tecnologías y experiencias de usuario a los sistemas de la biblioteca de la Universidad Complutense de Madrid, añadiendo nuevas herramientas que permitan un acceso más rápido e intuitivo a la información del catálogo y sus servicios. Para ello, hemos desarrollado una aplicación para los dispositivos móviles iOS y Android para poder acceder al catálogo de la biblioteca de la UCM usando la voz.

1.3. Contenido

El documento incluye un capítulo del estado del arte, en el que se explica la situación con la que comienza el proyecto; un capítulo en el que se analizan los requisitos de este proyecto, y dos capítulos técnicos en los que se explican las herramientas y metodologías que hemos utilizado y el diseño, tecnologías y arquitectura que hemos desarrollado. Por último, se incluye un ejemplo de uso de la aplicación.

En los anexos de este documento se incluyen los manuales de usuario tanto para los clientes de iOS y Android, como para el servidor. Además se incluye un histórico de versiones del proyecto.

Chapter 2

Introduction

2.1. Motivation

In recent years, the advancement of mobile devices and the cloud have changed the way we communicate and access information.

With the premise of further improving access to the Internet and its information, almost 8 years ago appeared the first commercial intelligent voice assistant for mobile devices, allowing the user to make any query on the Internet simply by asking a question to his device with the voice.

Some of the technologies that were developed, and are currently being developed, for the operation of these services implement technologies such as artificial intelligence or automatic learning ("Machine Learning" [1]), allowing systems to improve, either by predicting possible queries or by offering a more concrete response as more users use the service.

Over the years, more and more institutions and companies are using these technologies to facilitate access to information and reach as many people as possible. In our case, the number of people who access the university is increasing, and therefore the diversity among them increases.

Currently, the library service of the Complutense University of Madrid lacks this type of technology, having a search engine hosted on the web, so that for some people may not be easy, having to devote more effort than necessary when making a query or go directly to the library of your faculty.

2.2. Objective

The aim of this project is to transfer these technologies and user experiences to the library systems of the Complutense University of Madrid, adding new tools that allow faster and more intuitive access to the information in

[1] `https://en.wikipedia.org/wiki/Machine_learning`

the catalogue and its services. To do this, we have developed an application for mobile devices iOS and Android to access the catalog of the library of the UCM using voice.

2.3. Content

The document includes a chapter on the state of the art, explaining the situation in which the project begins; a chapter analysing the requirements of this project, and two technical chapters explaining the tools and methodologies we have used and the design, technologies and architecture we have developed. Finally, an example of the use of the application is included.

In the annexes of this document the user manuals for the iOS and Android clients are included, as well as for the server. A version history of the project is also included.

Capítulo 3

Estado del arte

Resumen: En este capítulo se explica cuál es el estado de los sistemas bibliotecarios de la Universidad Complutense de Madrid y de las tecnologías de procesamiento de lenguaje natural existentes en el momento de empezar este proyecto (septiembre de 2018). Para explicar toda la información, dividimos este capítulo en dos secciones, una dedicada a los servicios que ofrece la universidad y otra dedicada a las tecnologías de mencionadas anteriormente.

3.1. Servicios de la Universidad Complutense de Madrid

La universidad ofrece diversa información a través de varios medios. Los medios que se analizarán son la aplicación "La Complutense" y el sistema bibliotecario, accesible a través de una página web.

3.1.1. Aplicación La Complutense

La Universidad Complutense de Madrid, dispone de una aplicación desarrollada para iOS y Android en colaboración con CRUE y banco Santander 3.1, en el que se integran ciertos servicios en línea.

Para poder utilizar todas las funcionalidades que ofrece la aplicación es necesario tener una cuenta de la UCM. Si no se dispone de una, se puede consultar información general sobre la universidad, así como vías de contacto para cualquier consulta. Una vez hemos accedido a la aplicación con una cuenta válida, podemos realizar las siguientes acciones:

- Pestaña de Noticias: desde esta pestaña podremos informarnos de las

Figura 3.1: Aplicación "La Complutense"

noticias y/o artículos que publique la Universidad Complutense de Madrid.

- Pestaña de Retos: en esta pestaña nos aparecerán retos propuestos por la Universidad.

- Pestaña de Ventajas: aquí saldrán promociones y/u ofertas para todo aquel que posea una cuenta de la UCM.

Debido a la ausencia de contenido en la pestaña de Retos y la pestaña de Promociones, no podemos describir con detalle cada uno de estos apartados.

Además de esas pestañas, a la derecha tenemos un botón que nos abre un menú donde podremos chatear con profesores y alumnos que estén matriculados en las mismas asignaturas que nosotros y unirnos o crear grupos de chats. El siguiente botón que aparece es acceso al calendario. Este apartado, está separado en tres secciones: calendario de la UCM, en el que aparecen eventos relacionados con las asignaturas matriculadas, calendario personal,

el cual se sincroniza con el calendario de tu dispositivo móvil, y un calendario general, en el cual se muestra los dos calendarios anteriores en uno solo.

A la izquierda tenemos un botón que nos abre un panel lateral, en el cual tenemos varias opciones, todas ellas relacionadas con nuestro perfil de la complutense (Campus Virtual y gestión de datos académicos). La opción del campus virtual abre en un contenedor web el acceso al campus, como si se accediera desde un navegador. Por la parte de gestión de datos académicos, podemos consultar el horario para el curso actual, las asignaturas matriculadas, así como el historial de notas de cada una de estas últimas.

Tras realizar este análisis, nos encontramos que esta aplicación no tiene ningún tipo de integración ni con el catálogo de la biblioteca ni con sus servicios, limitándose exclusivamente a mencionarla en artículos de noticias en los que la biblioteca está involucrada. Esto, por tanto, es un asunto por mejorar, dotando a la biblioteca de su propia App para facilitar su acceso o, incluso, integrar estas funcionalidades en la App institucional de la Universidad Complutense de Madrid como proyecto futuro.

3.1.2. Sistema bibliotecario de la Universidad Complutense de Madrid

En este apartado se explica cómo funcionan los sistemas principales de la biblioteca.

3.1.2.1. Gestión del catálogo

Desde julio de 2018, la Universidad Complutense de Madrid utiliza el software "WorldShare Management Services" (WMS) de la cooperativa OCLC. Con este software se gestiona el catálogo, los préstamos (únicamente para los usuarios de la UCM), las renovaciones y las reservas. A parte de este sistema, la Universidad utiliza software de diversa índole para interconectar WMS con sistemas heredados y préstamo interbibliotecario. Estás tecnologías que no están relacionadas con el propio WMS no se utilizarán para el desarrollo de este proyecto.

3.1.2.2. API's de WMS

En este punto se detalla el funcionamiento de las API's que ofrece OCLC (OCLC, 2011) y sus limitaciones.

Funcionamiento

En la reunión realizada con el departamento de servicio de desarrollo tecnológico y sistemas bibliotecarios, se dan a conocer las distintas `API's`[1]

[1] `https://platform.worldcat.org/api-explorer/apis`

(3.1) que son usadas por el sistema de la biblioteca, y se solicita el acceso a las API's más relevantes para la realización de este proyecto.

Nombre de la API	Descripción
WorldCat Search API	Acceso al catálogo a bajo nivel. Permite obtener una lista de recursos bibliográficos y datos respecto su localización.
WMS Availability API	Dado un identificador bibliográfico único, permite obtener información acerca de la disponibilidad de los ejemplares en una biblioteca concreta.
WorldCat Discovery API	Servicio de consulta para el descubrimiento y la vinculación de recursos electrónicos. Proporciona acceso a los desarrolladores a la información de las bibliotecas en la base de datos de WorldCat. La base de conocimiento de WorldCat combina todos los datos sobre el contenido de las bibliotecas.

Tabla 3.1: API's solicitadas y su funcionalidad.

Además, también nos otorgan acceso a las aplicaciones antiguas de la biblioteca con conexión a WMS con la finalidad de facilitar la integración de este sistema en el proyecto y buscará información sobre una posible API para el acceso de usuarios de cara a implementar una posible característica de renovación y reserva de ejemplares en el proyecto.

Limitaciones

Estas son las siguientes limitaciones de las API's de WMS encontradas hasta la fecha:

- Ninguna de las API's devuelve una imagen de portada del ejemplar, ya sea por URL o por fichero. Para solucionarlo, en las primeras versiones, accedemos a la URL del ejemplar ofrecido por la API WorldCat Search API y obtenemos el enlace de la imagen. Esta operación es extremadamente costosa tanto en tiempo como en recursos, por lo que hemos limitado las listas de libros hasta 3 ejemplares para no exceder un tiempo de consulta de 10 segundos. En las versiones finales (a partir de la 0.9, que puede consultarse en el anexo D.4) modificamos la funciona-

lidad, trasladando la búsqueda de las portadas a los clientes móviles y utilizando como servicio de referencia la API de Open Library.[2]

- La API WorldCat Search no dispone de filtros de búsqueda para seleccionar ejemplares de una biblioteca concreta de la UCM. No hay forma posible de implementar esta funcionalidad de forma alternativa.

3.1.3. Información institucional de las bibliotecas

La información institucional de las bibliotecas, como números de teléfono, direcciones, o consultas sobre el funcionamiento y/o las normas de la biblioteca, se accede a través de la web de la Universidad. Cada biblioteca de cada una de las facultades que componen la Universidad dispone de su propia página con libertad absoluta para estructurar el contenido como deseen. Esto genera un importante problema, ya que esta gestión descentralizada de la información supone que la información puede estar en diferentes páginas dentro de su web, lo que lleva a que el usuario tarde más tiempo en buscar la información que necesite.

Por otro lado, la biblioteca ofrece un servicio de chat para resolver dudas generales. En este chat trabajan diferentes personas de todas las bibliotecas de la Universidad, turnándose los días en los que se debe atender.

Una ventaja de este proyecto es que se puede englobar toda la información de las diferentes webs de las bibliotecas en un único lugar, y se puede eliminar parte de la carga de trabajo al personal encargado de gestionar el chat.

3.2. Estado tecnológico actual

En este apartado analizaremos las tecnologías que puedan ser necesarias para el desarrollo del proyecto, y las analizaremos entre una u otra.

3.2.1. Plataformas para bot's

En primer lugar, enumeramos las plataformas disponibles de cara a desarrollar un bot, destacando las siguientes: Telegram, WhatsApp, Twitter y Facebook Messenger. En caso de desarrollar un bot, elegimos Telegram por las siguientes razones:

- WhatsApp no dispone por el momento de una plataforma de bot's como tal. En la actualidad ofrece una aplicación para pequeñas empresas que exclusivamente envía mensajes de forma automática cuando esta indica que está cerrada en un determinado horario. En un futuro se espera que implementen una funcionalidad más completa.

[2] `https://openlibrary.org/dev/docs/api/covers`

- Twitter, pese a tener una API muy rica para bot's, no puede enviar ni recibir archivos de audio.
- Facebook Messenger acepta el envío y recepción de archivos de audio y dispone de una API para bot's. No obstante, debido a los múltiples problemas de privacidad que ha tenido en los últimos tiempos, a su importante descenso de popularidad y a la exigencia de ser un usuario registrado de Facebook, optamos por utilizar otra alternativa.
- Telegram dispone de envío y recepción de archivos de audio, dispone de una API abierta y muy documentada.

3.2.2. ¿Utilizar un bot o un cliente nativo?

Tras determinar una plataforma en la que poder trabajar para desarrollar un bot nos cuestionamos: ¿Qué ventajas e inconvenientes tiene usar un bot frente a un cliente nativo?

Enumeramos las ventajas y desventajas de utilizar un bot en la tabla (3.2)

Ventajas	Desventajas
· Desarrollo único para todas las plataformas en las que está disponible Telegram.	· No existe API nativa para voz.
· Simplicidad de desarrollo.	· Necesidad de contar con un servicio externo para el reconocimiento de voz (principalmente de pago o gratuito con funcionalidades muy limitadas).
· No es necesario disponer de cuenta de desarrollador.	· Interfaz de usuario más compleja (necesidad de escribir comando /start como mínimo). · Dificultad de uso para personas con discapacidad visual. · Necesario disponer de un servidor en el que procesar la información pedida por el usuario antes de poder enviarla al back-end. · Desconocimiento de programación usando Telegram.

Tabla 3.2: Ventajas y desventajas uso de bot's.

Mientras que las ventajas y desventajas de utilizar un cliente nativo se describen en la siguiente tabla (3.3)

Ventajas	Desventajas
· Interfaz de usuario personalizable completamente, pudiéndose adaptar su desarrollo a cualquier escenario. · API de reconocimiento de voz y transformación de texto a voz nativa gratuita. · Procesamiento casi inmediato de la solicitud del usuario en el propio dispositivo/servidor de la plataforma del dispositivo gratuito. · Posibilidad de acoplar la funcionalidad a la aplicación ya existente de La Complutense con unas pocas adaptaciones. · Conocimientos de programación para dispositivos móviles. · Menor tiempo de conexión con el Back-End (muchas operaciones se hacen desde el dispositivo y no requieren un tratamiento especial en un servidor externo).	· Desarrollo más complejo, teniendo que realizar un cliente para cada plataforma. · Necesidad de cuentas de desarrollo para algunas plataformas.

Tabla 3.3: Ventajas y desventajas uso de un cliente nativo.

3.2.3. Tecnologías de reconocimiento de voz en dispositivos móviles

La mayoría de los dispositivos móviles en la actualidad disponen de asistentes de voz, como pueden ser “Google Assistant”, “Amazon Alexa”, “Siri” o “Cortana”. Estos asistentes convierten la voz del usuario en un mensaje de texto y lo envían a un servidor externo para que se procese esta consulta. Ninguno de estos asistentes de voz tiene la funcionalidad de realizar consultas al catálogo de la biblioteca de la Universidad Complutense de Madrid ni a su información institucional. Esto es un problema, dado que estas tecnologías facilitan el acceso del usuario a la información, ahorrando bastante tiempo.

3.2.4. Base de datos

Entre los tipos de bases de datos a utilizar nos encontramos con dos alternativas: Una base de datos relacional (SQL)[3] o una base de datos NoSQL[4].

Las ventajas y desventajas de utilizar una base de datos SQL son las descritas en la tabla (3.4)

Ventajas	Desventajas
· Su amplia documentación y cantidad de plugins dada su antigüedad. · La atomicidad en sus operaciones, pudiendo abortar una operación entera haciendo un rollback. · Su estricto control sobre la integridad de los datos. · Nuestra experiencia utilizando este tipo de bases de datos.	· El mantener la atomicidad de las operaciones hace que el rendimiento de estas se vea muy afectado. · Su poca o nula escalabilidad. · Instalación y puesta en marcha más complicada. · Incompatible con Rasa.

Tabla 3.4: Ventajas y desventajas uso de una base de datos SQL.

Las ventajas y desventajas de utilizar una base de datos NoSQL son las descritas en la tabla (3.5)

Ventajas	Desventajas
· Su alta escalabilidad y descentralización. · La estructura de los datos es más abierta y flexible. · Permite modificar el esquema de los datos sin necesidad de realizar paradas de mantenimiento. · Alto rendimiento. · Están diseñadas para hacer consultas que devuelvan grandes cantidades de datos. · Nuestra experiencia utilizando este tipo de bases de datos. · Compatible con Rasa.	Generalmente no aceptan la atomicidad en las operaciones. Son más recientes y, por tanto, existe menor documentación. · Las utilidades para la administración de estas generalmente se realizan mediante terminal.

Tabla 3.5: Ventajas y desventajas uso de una base de datos NoSQL.

[3] `https://es.wikipedia.org/wiki/SQL`
[4] `https://es.wikipedia.org/wiki/NoSQL`

3.3. Procesadores de lenguaje natural

El Procesamiento Natural del Lenguaje o NLP (del inglés, Natural Language Processing) es un apartado de la Inteligencia Artificial combinado con la lingüística enfocado en la interacción entre máquinas y el lenguaje "natural humano". Dentro de este campo, existe un subcampo conocido como Comprensión del Lenguaje Natural o NLU (del inglés Natural Language Understanding), que abarca en profundidad ciertos subcampos del NLP que requieran un análisis más exhaustivo, como, por ejemplo, análisis de sentimientos o traducción automática de un lenguaje. En la figura 3.2 (MacCartney, 2014) se pueden observar algunas de ellas y su relación con cada campo.

Estos otorgan a un sistema la capacidad de generar una posible respuesta adecuada y mantener un hilo de diálogo, recogiendo lo que introduce el usuario, descomponiéndolo y haciendo un análisis de ello.

Para que un asistente virtual o chatbot, que contienen internamente un NLP y NLU, tengan la capacidad mencionada anteriormente, son necesarios los siguientes elementos: Intents, Entities y flujo de diálogo.

- Intent: la función de este elemento consiste en captar la intención del usuario, pudiendo estar realizada de diferentes formas gramaticales y teniendo el mismo significado.

- Entity: gracias a este elemento, podemos recolectar un objeto o término que aportará más información junto a un intent. En los distintos NLP existentes, es posible añadir una librería con ejemplos de entities del contexto en el que se encuentre nuestro sistema.

- Flujo de diálogo: para poder mantener un hilo conversacional y poder dirigirla por un camino u otro, se hace uso del flujo de diálogo. Este componente, una vez que se ha extraído los intents y entities, determina cuál es la respuesta más adecuada, teniendo en cuenta además las peticiones realizadas anteriormente.

En un NLP, el componente que se encarga de obtener los intents y entities se conoce como extractor de relaciones. Para que un sistema aprenda a extraerlos, debe proporcionársele un modelo que contenga casos de prueba como se muestra en la figura 3.3[5]. En este ejemplo, la categoría de un intent va precedida del símbolo # y los posibles ejemplos se enumeran con - y luego la frase. Por último, un entity se define rodeándole con corchetes y, seguidamente, entre paréntesis, el tipo de entity que es.

[5] `https://github.com/RasaHQ/rasa_core/blob/master/examples/restaurantbot/data/nlu.md`

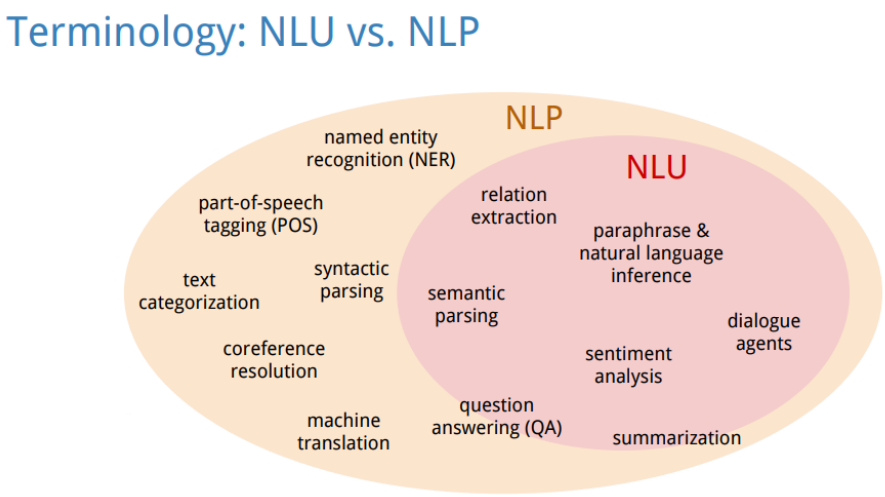


Figura 3.2: Distinción de tareas entre NLP y NLU

3.3.1. Alternativas disponibles en la nube

En el presente apartado explicaremos cuales son los posibles candidatos a ser el motor de NLP en la primera fase del proyecto. Las herramientas analizadas han sido las siguientes:

- IBM Watson Assistant[6]
- Amazon Lex[7]
- Microsoft Azure Cognitive Services - Language Understanding[8]
- Google Cloud DialogFlow[9]

En todas estas, el análisis se ha realizado en igualdad de condiciones, creando cuentas gratuitas en periodo de prueba y escogiendo la misma característica común: la capacidad de comprender una frase en formato textual, entender su contexto y devolver una respuesta adecuada. Las plataformas que no soportan esta característica han quedado directamente descartadas.

En nuestras pruebas se han realizado una serie de configuraciones preparadas para ofrecer respuestas en referencia a información muy básica de una biblioteca ficticia, además de saludos, despedidas o información sobre

[6] `https://www.ibm.com/cloud/watson-assistant/`
[7] `https://aws.amazon.com/es/lex/`
[8] `https://azure.microsoft.com/es-es/services/cognitive-services/directory/lang/`
[9] `https://dialogflow.com/`

```
## intent:deny
    - no danish food
    - no north
    - no
    - uh yes restaurant that serves danish food
    - let's do it
    - yeah
    - yes that sells korean food
.......................
## intent:affirm
    - uh yes im looking for a cheap restaurant
        in the west part of town
    - yeah a cheap restaurant serving international food
    - correct
    - ye
.......................
## intent:inform
    - [afghan](cuisine) food
    - how bout [asian oriental](cuisine)
    - im looking for a [moderately](price:moderate)
        priced restaurant in the [east](location) part of town
    - find [moderately](price:moderate) priced restaurant
        in the [west](location) part of town
    - what about [indian](cuisine) food
```

Figura 3.3: Ejemplos de intents y entities

el propio asistente. Los resultados de las pruebas no son concluyentes, ya que todos los servicios han respondido correctamente a cada petición en un tiempo razonable. Por tanto, procedemos a indagar más en las limitaciones por el tipo de suscripción y en su API.

Funcionalidades por suscripción

A continuación, se detalla las solicitudes disponibles en las suscripciones gratuitas:

- IBM Watson Assistant[10]: 10000 solicitudes/mes.
- Amazon LEX[11]: 10000 solicitudes/mes durante el primer año. A partir del siguiente año el servicio cuesta 0.00065 euros por solicitud más

[10] `https://www.ibm.com/cloud/watson-assistant/pricing/` - Último acceso en octubre de 2018

[11] `https://aws.amazon.com/es/lex/pricing/` - Último acceso en octubre de 2018

gastos por el uso de servicios añadidos de AWS no incluidos en la suscripción.

- Microsoft Azure CS-LUIS[12]: 5000 solicitudes gratuitas al mes. A partir de la 5000 el precio es de 63,01 euros al mes hasta alcanzar las 25000 peticiones.
- Google DialogFlow[13]: Gratis, con la limitación de procesar 180 solicitudes al minuto (a partir de la 180 la solicitud se descarta). Además, es necesario incluir una tarjeta de crédito para registrarse.

Funcionalidad de la API

Con respecto a las API's, IBM Watson Assistant y Amazon Lex ofrecen una API sencilla para ser utilizada con Python. En cambio, Google DialogFlow ofrece una API muy sencilla de configurar con su asistente para una serie de plataformas como Facebook Messenger, Telegram, etc. Sin embargo, su API para Python es más compleja que las alternativas anteriores. Como uno de los pilares básicos de este proyecto es su alta modularidad y la alternativa de Google supone un problema para desarrollar un Back-End propio, esto puede limitar un desarrollo más personalizado. Por otro lado, Amazon Lex también dificulta el desarrollo de un Back-End externo, ya que está pensado para utilizar el servicio AWS Lambda[14]. Sin embargo, el desarrollo de un BackEnd propio es posible y el nivel de dificultad no es tan alto como con la alternativa de Google.

3.3.2. Alternativas disponibles open source

En este apartado explicaremos cuales son los posibles candidatos a ser el motor de NLP en la segunda fase del proyecto. Un requisito ineludible para este procesador de lenguaje natural será que su licencia sea de código abierto. Esta decisión viene dada por los siguientes motivos:

- IBM Watson es una solución cerrada, de manera que no podemos ver ni manipular su código fuente, impidiendo realizar mejoras o adaptaciones al tipo de proyecto en el que nos encontramos.
- Watson ofrece compatibilidad con la lengua castellana, sin embargo, posee algunas limitaciones con ciertos tipos de entidades (como los nombres propios), esto dio problemas en la primera fase y limitó enormemente el potencial que se podría usar en este proyecto.

[12] `https://azure.microsoft.com/es-es/pricing/details/cognitive-services/language-understanding-intelligent-services/` - Último acceso en octubre de 2018

[13] `https://cloud.google.com/dialogflow-enterprise/pricing` - Último acceso en octubre de 2018

[14] `https://aws.amazon.com/es/lambda/features/`

- Pese a disponer de una licencia para estudiantes, en el fondo Watson es un servicio de pago, además, la modalidad gratuita utilizada en la actualidad tiene un límite de procesamiento de "Intents", "Entities" y llamadas a la API.
- Un motor de lenguaje propio permitiría que cualquier persona interesada en este proyecto pudiese continuar con el desarrollo de este.

spaCy	
Ventajas	Desventajas
· Alto rendimiento. · Soporte para el lenguaje español. · Facilidad de uso, con API sencilla y concisa. · Enfocado a orientación a objetos. · Uso de redes neuronales para el aprendizaje de algunos modelos. · Gran cantidad de documentación y soporte de la comunidad. · Soporte nativo de vectores de palabras	· Menor flexibilidad con respecto a otras librerías. · Tokenización de frases más lenta respecto a otras librerías. · Soporte para 7 idiomas diferentes (en la versión 2.0.18).

Tabla 3.6: Ventajas y desventajas de spaCy

Tanto spaCy como NLTK, no proveen de capacidades de los NLU, como el reconocimiento de "Intents", necesarias en este proyecto.

3.3.3. Rasa

Rasa (Bocklisch et al., 2017) es un una librería open-source para Python que facilita el desarrollo de chatbot's. Esta librería no incluye un motor NLP y necesita de uno de terceros. Comprobando su compatibilidad encontramos que este se integra perfectamente con spaCy, por tanto, es la solución ideal para este proyecto.

Rasa se divide en dos módulos clave (hasta la versión 0.14.3[15]):

- Rasa NLU: Como su nombre indica, este módulo provee funcionalidades de NLU. La funcionalidad que provee es muy básica, limitándose al reconocimiento de intents y entities de una oración. Este módulo es fundamental para el proyecto, dado que se utiliza para analizar qué es lo que desea el usuario.

[15] `http://legacy-docs.rasa.com/docs/core/0.14.3/`

NLTK	
Ventajas	Desventajas
· Soporte para extensiones. · Tokenización de frases rápida. · Soporte para gran cantidad de lenguajes (entre ellos, el español).	· Rendimiento lento. · Más difícil de utilizar respecto a otras librerías. · No tiene soporte nativo de vectores de palabras. · No utiliza redes neuronales para el aprendizaje. · El tokenizador únicamente divide el texto por oraciones, no analiza su estructura semántica. · No está pensado para ser utilizado con programación orientada a objetos.

Tabla 3.7: Ventajas y desventajasde NLTK

- Rasa Core: Este módulo añade las características necesarias para soportar diálogos con el usuario manteniendo un hilo conductor coherente.

Como funcionalidad extra, provee compatibilidad total con JSON para la devolución de las consultas (formato de objeto estándar en la actualidad). Además, a diferencia de Watson, Rasa y spaCy contienen más diccionarios de entidades para la lengua castellana, evitando tener que entrenarlo en algunos aspectos como los nombres propios.

Para poder entrenar al sistema usando Rasa, se debe hacer manualmente con un modelo creado desde el principio para obtener las consultas y respuestas deseadas.

Rasa NLU

Como se explica en el punto anterior, este módulo se usa para el reconocimiento de intents y entities. Permite utilizar múltiples "Named Entity Recognition" (o también conocido como "NER")[16] haciendo uso de un motor NLP compatible (entre los que se encuentra spaCy). Además, soporta la creación de modelos de sinónimos para evitar duplicidades en el modelo "CRF".

y, por tanto, utilizaremos el NER de spaCy para la extracción de entidades generales (como nombres o localizaciones) como base y el NER "CRF", construido específicamente para este proyecto, proporcionando un modelo

[16] `https://es.wikipedia.org/wiki/Reconocimiento_de_entidades_nombradas`

preparado para entender consultas relacionadas con contenido bibliográfico. También construimos manualmente un modelo de sinónimos para evitar duplicidades en el modelo "CRF".

Rasa Core

Este módulo guiará por el flujo de la conversación tanto al usuario como al resto de módulos del proyecto. El usuario primero indicará su consulta, Rasa Core enviará la petición al NLU y en función del reconocimiento que este realice, podrá hacer lo siguiente:

- Lanzamiento de un "Action": Los action realizan comprobaciones sobre la consulta realizada, comprobando si la información proporcionada por el usuario es suficiente, dando una respuesta adecuada en este caso, o si la información es insuficiente, solicitando mediante texto al usuario más información para poder continuar con la consulta.
- Lanzamiento de un "FormAction": Los FormAction se encargan de mapear los múltiples entities detectados en la consulta a los slots.
- Lanzamiento de una respuesta de texto: Se encarga de devolver una respuesta de texto al usuario, todos los puntos anteriores finalizan su ciclo invocando esta funcionalidad. Los casos más simples solo lanzan esta función.

Durante el flujo de la conversación, es necesario que Rasa almacene algunos datos significativos de esta, como puedan ser el nombre del usuario (si este se lo indica) u otros como pueden ser nombres de libros, autores, localizaciones de bibliotecas, etc. Para esto, utilizaremos una característica llamada "Slots". Los Slots permiten almacenar datos durante todo el flujo de la conversación, y permiten enviar respuestas personalizadas a los usuarios.

Además, también es interesante mantener algunos de los datos para nuevos flujos de conversación (como el nombre del usuario). Para esto, utilizamos los "Tracker Stores"[17], que permiten almacenar las conversaciones en su totalidad en una base de datos.

[17] `https://rasa.com/docs/core/tracker_stores/`

Capítulo 4

Análisis de funcionalidad

Resumen: En este capítulo se detalla como se ha procedido a la hora de captar los requisitos con los posibles clientes interesados en este proyecto y se ha construido un documento de historias de usuario a partir de estos.

4.1. Captación de requisitos

Para el desarrollo de este proyecto hemos necesitado llevar a cabo reuniones con el personal de la Biblioteca Central de la Universidad Complutense de Madrid y personal de la Biblioteca de la Facultad de Informática de la Universidad Complutense de Madrid.

El objetivo de estas reuniones consistió en conocer las necesidades y los problemas expuestos por el personal de la biblioteca, así como analizar el entorno en el que está implantado el sistema informático de la biblioteca. Además, como punto a destacar, hemos podido reunirnos con una persona invidente, para conocer de primera mano las necesidades que tienen diariamente y como manejan los dispositivos móviles, ya que uno de nuestros intereses de realizar este proyecto es para dar mayor accesibilidad a los servicios de la Biblioteca de la Universidad Complutense de Madrid.

Durante esta primera etapa del proyecto, comprendida entre los meses de septiembre y noviembre de 2018, se procedió a organizar las siguientes reuniones.

En la primera reunión, celebrada el 30 de octubre de 2018 en el edificio Multiusos de la Universidad, nos entrevistamos con dos personas del departamento de Servicios de Información y Apoyo a la Docencia e Investigación Bibliotecaria. Estas personas se encargan principalmente de la atención del

chat de la web de la biblioteca. Aquí se exponen los siguientes problemas o necesidades a los que se busca una solución o mejora desde este servicio:

- Los usuarios desconocen el funcionamiento en profundidad de la herramienta de búsqueda de libros/documentos en el nuevo catálogo de la Universidad.
- Los usuarios suelen preguntar por información institucional de cada biblioteca.
- Existen quejas por el poco alcance de las noticias y entradas del blog de cara al usuario corriente.
- La asistencia a los cursos ofrecidos por las bibliotecas difiere mucho del registro de usuarios inscritos, ya sea por no estar interesados a asistir aún estando inscritos o por olvido de este.

Con todo esto, presentamos la idea de qué queremos desarrollar, explicando algunas posibles alternativas sobre cómo se pueden solucionar algunos de los problemas antes descritos. Dado que una de las personas que asiste a la reunión tiene una discapacidad visual, esta nos demuestra como es su día a día utilizando un dispositivo móvil, dándonos una idea sobre cómo funcionan las características de accesibilidad. Además, nos ofrece una serie de recomendaciones para el desarrollo del cliente, como es la posibilidad de que el cliente no lea en voz alta las respuestas devueltas o incluir un modo de alto contraste para otro tipo de discapacidades.

En la segunda reunión, celebrada el 13 de noviembre de 2018 en la biblioteca de la Facultad de Informática, nos entrevistamos con la jefa de proceso e información especializada. Uno de sus roles de este cargo es la atención del chat de la biblioteca.

Con la experiencia obtenida con la reunión anterior, estructuramos la reunión de una forma algo diferente. En primer lugar, nos presentamos y explicamos desde el primer momento el proyecto, que objetivos queremos alcanzar y su experiencia con el tratamiento del chat.

Con esto, nos informan de los siguientes problemas a los que suele dar respuesta con este servicio:

- Nuevamente, la principal cuestión a la que se enfrenta es el desconocimiento de los usuarios sobre el funcionamiento del nuevo catálogo de la Universidad.
- Atención y explicación sobre el funcionamiento del préstamo interbibliotecario y las restricciones de uso que este tiene con usuarios externos a esta Universidad.

- Problemas con la lectura de documentos electrónicos, principalmente producido por las restricciones de las licencias de uso para cierto tipo usuarios.
- Igual que en el punto anterior, también suelen preguntar información institucional de las bibliotecas.

Con esto quedan cerradas las ideas clave sobre qué debe solucionar este proyecto.

En la última reunión, realizada el 14 de noviembre de 2018 en el edificio Multiusos, nos entrevistamos con una persona del departamento de servicio de desarrollo tecnológico y sistemas bibliotecarios. En esta reunión exponemos la idea del proyecto y explicamos la necesidad que disponemos de acceder al sistema utilizado por la universidad. Durante esta reunión, hacemos un visionado global de las API's disponibles de WMS y finalmente añade a la lista de peticiones el acceso a las API's.

Con todo lo obtenido de estas reuniones, creamos el siguiente documento de "Historias de Usuario".

4.2. Historias de Usuario

Tras los requisitos obtenidos en las reuniones mencionadas en el punto 4.1, priorizamos las necesidades, problemas y características que puedan darle un valor añadido al proyecto, y se redactaron las historias de usuario, accesibles en las tablas 4.1 y 4.2.

La estructura de estas es la siguiente:

- ID: Representa el identificador de la historia.
- ROL: Define el papel que toma una persona para el problema y/o necesidad.
- Característica/Funcionalidad: Define el problema y/o necesidad.
- Razón/Resultado: Define la razón por la que se debe implementar la característica.
- Título: Definición de un posible escenario que podría ocurrir.
- Contexto: Explica qué condiciones se han dado al realizar la llamada a la funcionalidad.
- Evento: Indica cuándo se produce la respuesta.
- Resultado/Comportamiento esperado: Indica la respuesta ofrecida por el sistema.

	ENUNCIADO DE LA HISTORIA			CRITERIOS DE ACEPTACIÓN				
ID	ROL	CARACTERÍSTICA / FUNCIONALIDAD	RAZÓN / RESULTADO	Nº	TÍTULO	CONTEXTO	EVENTO	RESULTADO / COMPORTAMIENTO ESPERADO
01-00-01	Como un usuario	Necesito poder saludar al asistente de voz	Con la finalidad de que este responda	1	Conexión correcta	La aplicación se ha conectado con el servidor y ha generado un saludo	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con un saludo aleatorio.
				2	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.
01-00-02	Como un usuario	Necesito poder despedirme del asistente de voz	Con la finalidad de que este responda	1	Conexión correcta	La aplicación se ha conectado con el servidor y ha generado una despedida	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con una despedida aleatorio.
				2	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.
01-01-01	Como un usuario	Necesito poder buscar una serie de libros en el catálogo de la biblioteca por palabras clave	Con la finalidad de obtener la información que busco	1	Información encontrada	El sistema ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con una lista de hasta tres resultados en pantalla expandible a más en caso de solicitarlo.
				2	Información no encontrada	El sistema no ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que no existen términos relacionados con esa consulta.
				3	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.
01-01-02	Como un usuario	Necesito poder buscar una serie de libros en el catálogo de la biblioteca por título	Con la finalidad de obtener la información que busco	1	Información encontrada	El sistema ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con una lista de hasta tres resultados en pantalla expandible a más en caso de solicitarlo.
				2	Información no encontrada	El sistema no ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que no existen términos relacionados con esa consulta.
				3	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.
01-01-03	Como un usuario	Necesito poder buscar una serie de libros en el catálogo de la biblioteca por autor	Con la finalidad de obtener la información que busco	1	Información encontrada	El sistema ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con una lista de hasta tres resultados en pantalla expandible a más en caso de solicitarlo.
				2	Información no encontrada	El sistema no ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que no existen términos relacionados con esa consulta.
				3	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.
01-02-01	Como un usuario	Necesito poder buscar un único libro en el catálogo de la biblioteca	Con la finalidad de obtener la información detallada sobre dicha búsqueda	1	Información encontrada	El sistema ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con un único resultado y permitirá al usuario acceder a la web del objeto a través de un botón.
				2	Información no encontrada	El sistema no ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que no existen términos relacionados con esa consulta.
				3	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.
01-03-01	Como un usuario	Necesito poder preguntar por la hora de apertura de una biblioteca concreta	Con la finalidad de obtener la información de apertura de dicha biblioteca	1	Información encontrada	El sistema ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con la hora de apertura de la biblioteca.
				2	Información no encontrada	El sistema no ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que no existe la biblioteca dada o pidiendo que sea más concreto, indicando una biblioteca existente.
				3	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.

Tabla 4.1: Historias de Usuario - 1

01-03-02	Como un usuario	Necesito poder preguntar por la hora de cierre de una biblioteca concreta	Con la finalidad de obtener la información de cierre de dicha biblioteca	1	Información encontrada	El sistema ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con la hora de cierre de la biblioteca.
				2	Información no encontrada	El sistema no ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que no existe la biblioteca dada o pidiendo que sea más concreto, indicando una biblioteca existente.
				3	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.
01-03-03	Como un usuario	Necesito poder preguntar por los días que abre una biblioteca concreta	Con la finalidad de obtener la información de cierre de dicha biblioteca	1	Información encontrada	El sistema ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con los días que abre la biblioteca.
				2	Información no encontrada	El sistema no ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que no existe la biblioteca dada o pidiendo que sea más concreto, indicando una biblioteca existente.
				3	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.
01-04-01	Como un usuario	Necesito poder preguntar la dirección de una biblioteca concreta	Con la finalidad de obtener la localización de dicha biblioteca	1	Información encontrada	El sistema ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con la dirección de la biblioteca y un botón que redirigirá a la aplicación de mapas del dispositivo.
				2	Información no encontrada	El sistema no ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que no existe la biblioteca dada o pidiendo que sea más concreto, indicando una biblioteca existente.
				3	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.
01-05-01	Como un usuario	Necesito poder preguntar el teléfono de una biblioteca concreta	Con la finalidad de obtener el teléfono de dicha biblioteca	1	Información encontrada	El sistema ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un globo en la aplicación con el teléfono de la biblioteca y un botón que redirigirá a la aplicación teléfono del dispositivo.
				2	Información no encontrada	El sistema no ha encontrado la información que desea el usuario	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que no existe la biblioteca dada o pidiendo que sea más concreto, indicando una biblioteca existente.
				3	Fallo de conexión	La aplicación no ha sido capaz de conectarse con el servidor	Cuando hacemos la consulta	El sistema mostrará un mensaje de error indicando que existe un problema con la conexión.

Tabla 4.2: Historias de Usuario - 2

Capítulo 5

Herramientas y metodología

Resumen: Aquí se explica brevemente las herramientas utilizadas para el desarrollo del proyecto y la metodología empleada.

5.1. Herramientas para el desarrollo

En esta sección se enumeran las diferentes herramientas utilizadas durante el desarrollo del sistema. Dado que el sistema está dividido en Clientes y Servidor, se desarrolla cada uno en un apartado diferente. Cabe destacar que la única herramienta común para todos los módulos del proyecto fue un repositorio git, en concreto, utilizamos GitHub. Las características utilizadas en GitHub fueron la carga de código en ramas y la sección de "Issues" para la comunicación interna entre los desarrolladores y los directores del proyecto.

5.1.1. Servidor

Para el desarrollo del servidor se utilizaron las siguientes herramientas:

- PyCharm IDE[1]: Es un entorno de desarrollo creado por la compañía "JetBrains" basado en "IntelliJ IDEA". Es un entorno de desarrollo multiplataforma muy completo para el lenguaje Python. Utilizamos la versión profesional, dado que la herramienta GitHub, de la que hemos hablado anteriormente, ofrece licencias gratuitas de esta herramienta para estudiantes.
- Sublime Text[2]: Es un editor de texto muy potente que acepta coloreado sintáctico para cualquier lenguaje. En nuestro caso fue utilizado para

[1] `https://www.jetbrains.com/pycharm/`
[2] `https://www.sublimetext.com/`

modificar algún fragmento de código de Python y para la lectura y escritura de los ficheros JSON del proyecto. Utilizamos una edición de prueba.

5.1.2. Cliente iOS

Para el desarrollo del cliente para iOS se utilizaron las siguientes herramientas:

- Xcode[3]: Es el entorno de desarrollo de Apple. Es el único entorno en el que se pueden desarrollar y enviar a la App Store las aplicaciones para dispositivos iOS en el lenguaje Swift. La licencia de uso es gratuita siempre que se utilice en un ordenador Mac.
- iOS Simulator: Este es el simulador de iOS para ordenadores Mac, permite probar la aplicación en desarrollo sin necesidad de instalarla en un dispositivo físico, viene incluida con el IDE Xcode.
- Dispositivo iOS: Se utilizó un iPhone SE y un iPad (6ª generación) para realizar las pruebas de la aplicación en un entorno real.

5.1.3. Android

Para el desarrollo del cliente para Android se utilizaron las siguientes herramientas:

- Android Studio[4]: Es el entorno de desarrollo de Google para Android. Este entorno incluye un simulador para realizar pruebas en un dispositivo simulado. La licencia de uso es gratuita.
- Dispositivo Android: Se utilizó un Xiaomi Mi A1 para realizar las pruebas de la aplicación en un entorno real.

5.2. Metodología

La metodología utilizada ha sido una "metodología ágil genérica", dado que el trabajo ha sido desarrollado por un equipo muy pequeño y que por disponibilidad tanto del equipo como de los clientes no se podía usar una metodología ágil concreta. El método de comunicación consistió en realizar una reunión semanal física entre los desarrolladores para exponer los puntos de conflicto que surgiesen durante esa semana o puesta en conocimiento de los trabajos realizados. Una reunión cada dos semanas con nuestros directores para explicar los avances en el proyecto y reuniones con los clientes para

[3] `https://itunes.apple.com/es/app/xcode/id497799835?l=en&mt=12`
[4] `https://developer.android.com/studio`

realizar la captación de requisitos o búsqueda de información sobre el funcionamiento de los sistemas bibliotecarios. Además, se realizaron reuniones telemáticas en cualquier momento según las necesidades del proyecto.

Por último, durante el desarrollo se ha seguido una metodología "gitflow" en el repositorio del proyecto, de manera que existía una rama master en la que se almacenan versiones del proyecto listas para subir a producción (el historial de versiones está disponible en el anexo D), y una rama Dev, de la que a su vez se crearon ramas de desarrollo para los distintos módulos del proyecto. En esta rama Dev el código no debe contener ningún problema en cuanto a compilación, y puede ser o no funcional, es decir, que cumpla alguno de los requisitos o no.

5.2.1. Plan de contingencia

Después de celebrarse las reuniones con el personal de la biblioteca, como se explica en el punto 4.1, hemos decidido hacer un plan de contingencia, en el que explicaremos las funcionalidades a implementar en función de los resultados de las peticiones realizadas al personal de la biblioteca. En concreto, solicitamos acceso a distintas API's para poder obtener información sobre recursos de la biblioteca de la universidad. Los dos escenarios que pueden ocurrir son los siguientes:

- Si no es posible acceder a esas API's , obtendríamos la información solicitada realizando peticiones HTTP pasando los datos clave para realizar la búsqueda y a través del código HTML de la página web conseguir la información del recurso en cuestión, utilizando como referencia lo que se conoce como webcrawler[5], que sirve para inspeccionar páginas web metódicamente para recoger información. En este momento, la información básica a obtener sería título, año, autor.
- Si es posible tener acceso a las API's solicitadas, podríamos conseguir información sobre los recursos a través de peticiones a estas herramientas, y poder obtener dicha información a través de un intercambiador de objetos estándar, como por ejemplo JSON o XML.

[5] `https://en.wikipedia.org/wiki/Web_crawler`

Capítulo 6

Diseño, tecnologías y arquitectura del sistema

Resumen: A continuación, se procede a indicar los elementos que componen el sistema. Después, se explica detalladamente el funcionamiento de cada uno de estos y su función en el sistema. Por último, se indica cómo se realiza el despliegue del sistema.

6.1. Arquitectura

En este apartado se enumeran los componentes del proyecto.

La arquitectura se compone de seis módulos principales (como puede observarse en la figura 6.1): cliente, back-end, controlador, procesador de lenguaje natural "Jarvis" (basado en Rasa y spaCy), buscador de biblioteca (conectado a los servicios de OCLC a través de una API) y DAO (conectado a una base de datos MongoDB).

- Cliente: El cliente es el punto de acceso del usuario al sistema. Desde este módulo, el usuario envía su consulta al Back-end y recibe su respuesta. La entrada de datos se realiza a través del micrófono del dispositivo, mientras que la salida se muestra por pantalla y se reproduce a través del altavoz del dispositivo. Este componente se comunica con el servidor a través del Back-end.
- Back-end: El módulo Back-end se encarga de traducir los datos que envía el cliente y enviárselos al controlador del sistema. También se encarga de encapsular la información devuelta por el controlador en un JSON para devolvérsela al cliente.

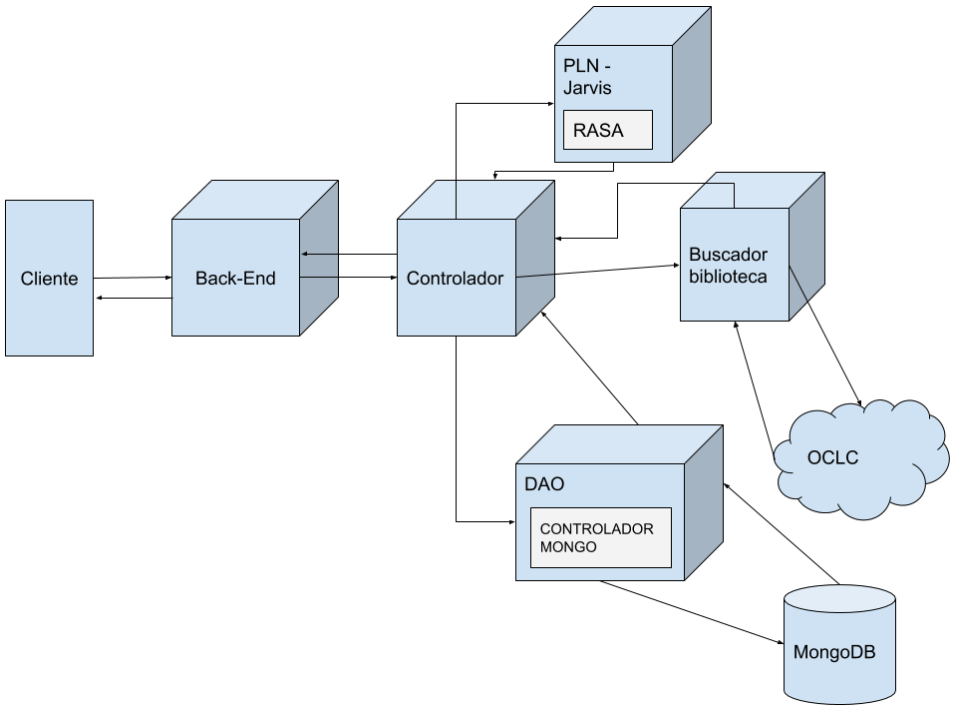


Figura 6.1: Representación de los módulos del proyecto

- Controlador: El controlador de la aplicación se encarga de tratar la consulta realizada por el usuario enviándo la y recibiendo las respuestas a los distintos módulos de la aplicación, finalmente devuelve la respuesta final al Back-end.
- NLP "Jarvis": El módulo de NLP se encarga de procesar los datos recibidos del controlador de manera que el sistema comprenda qué le está pidiendo el usuario y envíe la respuesta adecuada, tanto al usuario como al controlador que redirigirá la consulta al módulo correspondiente. La respuesta devuelta por este módulo será un intent con sus entidades (si las hay) y un mensaje textual que se mostrará en el cliente.
- Buscador biblioteca: El módulo "buscador biblioteca" se encarga de estructurar las entidades recibidas del módulo "Jarvis" en una URL para enviar una consulta a los servidores de OCLC, de manera que el módulo será capaz de hacer búsquedas en el catálogo de la biblioteca mediante filtros de búsqueda proporcionados por la API de OCLC. También podrá consultar, a través de un código "oclc"(que es el identificador único del ejemplar en el sistema bibliotecario de OCLC, similar al ISBN) proporcionado por el controlador, la disponibilidad de un ejemplar concreto.

- DAO: El módulo "DAO" permite a la aplicación conectarse a una base de datos MongoDB para acceder a horarios, teléfonos y localizaciones de las bibliotecas de la Universidad. También permite consultar y modificar los historiales de búsqueda de los usuarios.

El cliente se instalará en el dispositivo móvil del usuario (Smartphone iPhone o Android), mientras que el resto de módulos se alojarán en internet, ya sea en un servidor de la UCM (servidor de Janet), un cloud (módulo "Jarvis") o en un servidor externo (catálogo OCLC).

6.2. Diseño

A continuación, explicaremos en detalle los distintos flujos que seguirá nuestro sistema en lo referente al diálogo dependiendo de la petición realizada por el usuario.

6.2.1. Flujos de conversación

Esta sección está divida en varias subsecciones, cada una de ellas compuesta por un flujo de dialógo concreto. En las figuras, los cuadros sobreados de color verde representan acciones realizadas por el usuario, mientras que los cuadros sobreados de color azul, representan acciones llevadas a cabo por el sistema.

Flujo de diálogo - Saludos

En este flujo de diálogo (figura 6.2), cuando el usuario introduce un saludo o despedida, si el sistema reconoce el intent de saludo o despedida, responderá con una frase aleatoria del apartado correspondiente del modelo.

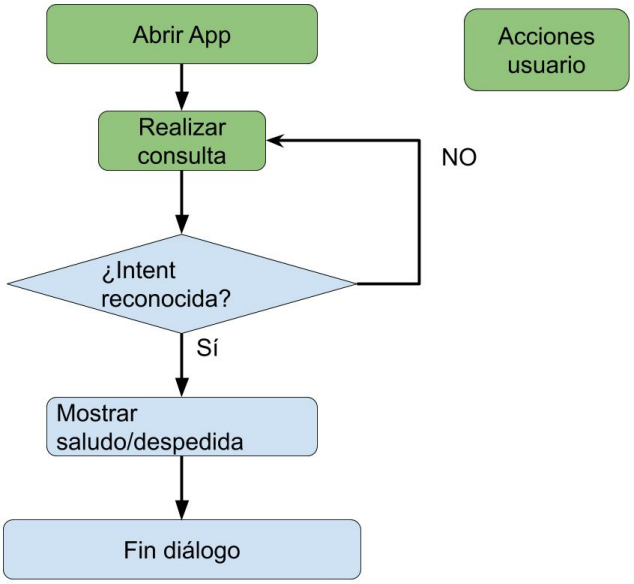


Figura 6.2: Diagrama de flujo - Saludos

Flujo de diálogo - Libros

En este flujo de diálogo (figura 6.3) el usuario realizará una petición relacionada con libros. Una vez que el sistema reconozca el intent, comprobará que tiene los entities necesarios. Para ello, como podemos ver en la figura 6.4, el sistema preguntará por la información necesaria para seguir con la petición. Una vez que lo consiga irá por una rama u otra dependiendo de si el usuario ha solicitado un único libro o varios. En el caso de solicitar un unico libro, el sistema sacará la información sobre este, pudiendo solicitar otro libro que cumpla las entites introducidas. En cambio, si el usuario ha solicitado varios ejemplares, el sistema mostrará por pantalla tres libros distintos. En este punto, el usuario puede solicitar más libros y el sistema devolverá otros tres libros. Si por el contrario decide que quiere más información sobre uno de los tres últimos libros, puede solicitar información sobre uno de ellos. En el caso de que el último mensaje proporcionado por el sistema sea un único libro, si se pulsa sobre la carátula de este, el sistema abrirá el navegador predeterminado del sistema con el enlace al libro solicitado. Cabe destacar que en el caso de querer conocer la disponibilidad de un libro, ha de solicitarse

un único ejemplar, además de abrir una página con el libro en la web de la biblioteca de la universidad al pulsar la portada.

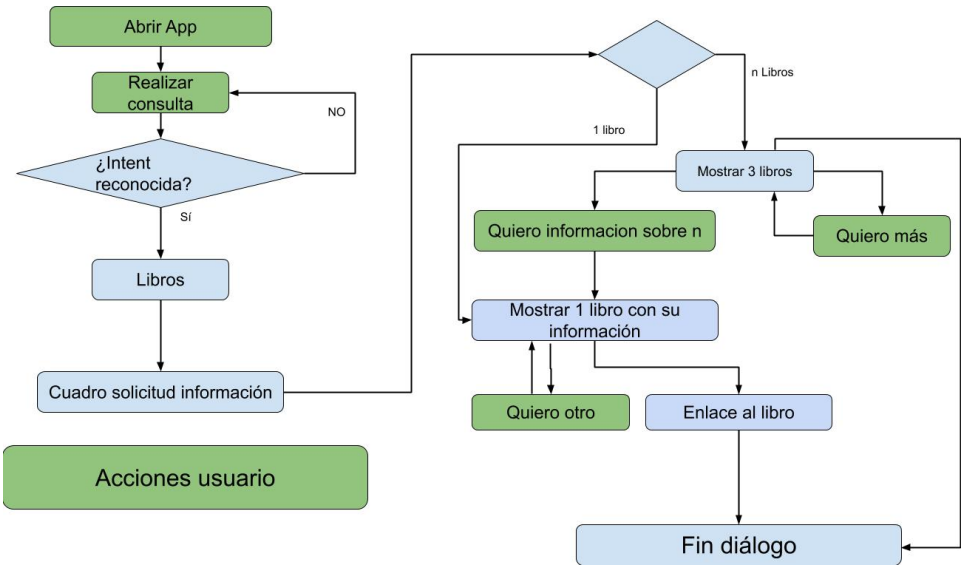


Figura 6.3: Diagrama de flujo - Libros

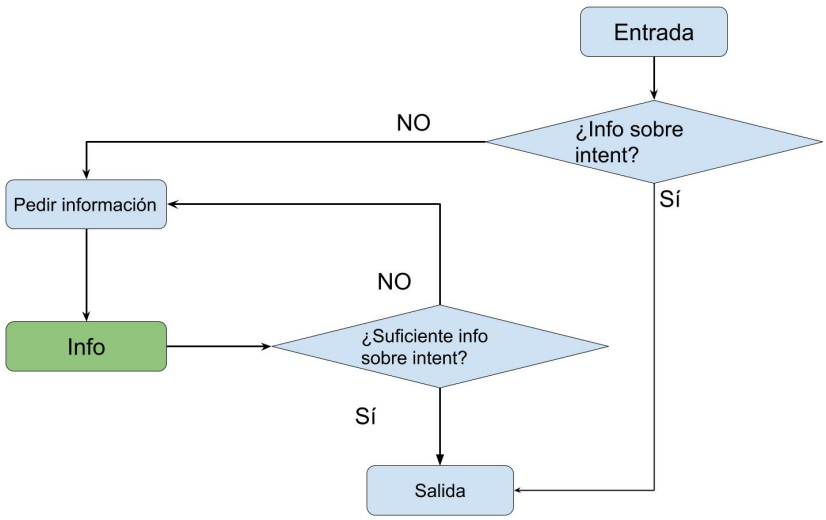


Figura 6.4: Diagrama de flujo - Cuadro solicitud más información

Flujo de diálogo - Teléfonos

En este flujo de diálogo (figura 6.5), cuando el usuario pide un teléfono, si el sistema reconoce el intent de teléfono, comprobará si en la petición se incluye la información necesaria para poder devolver una respuesta o si esta ya ha sido proporcionada anteriormente. La información debe ser una biblioteca o facultad. Si el sistema no encuentra esta información se la pedirá al usuario, en caso contrario, responderá el teléfono solicitado extraido de la base de datos.

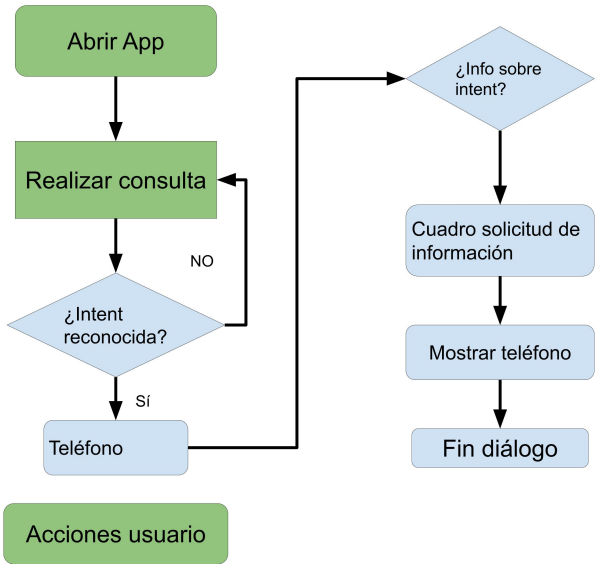


Figura 6.5: Diagrama de flujo - Teléfonos

Flujo de diálogo - Localización

En este flujo de diálogo (figura 6.6), cuando el usuario pide una localización, si el sistema reconoce el intent de localización, comprobará si en la petición se incluye la información necesaria para poder devolver una respuesta o si esta ya ha sido proporcionada anteriormente. La información debe ser una biblioteca o facultad. Si el sistema no encuentra esta información se la pedirá al usuario, en caso contrario, responderá la dirección solicitada, extrayéndola de la base de datos.

Flujo de diálogo - Horarios

En este flujo de diálogo (figura 6.7), cuando el usuario pide un horario, si el sistema reconoce el intent de horario, comprobará si en la petición se incluye la información necesaria para poder devolver una respuesta o si

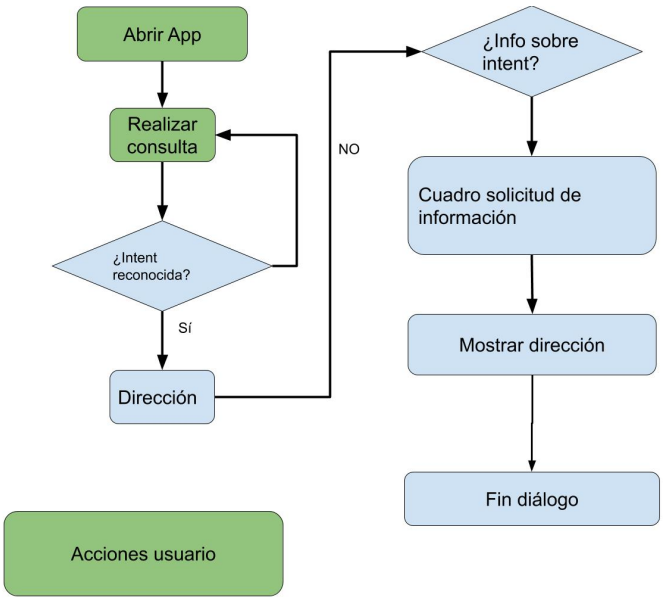


Figura 6.6: Diagrama de flujo - Localización

esta ya ha sido proporcionada anteriormente. La información debe ser una biblioteca o facultad. Si el sistema no encuentra esta información se la pedirá al usuario, en caso contrario, responderá el horario solicitado, buscándolo en la base de datos.

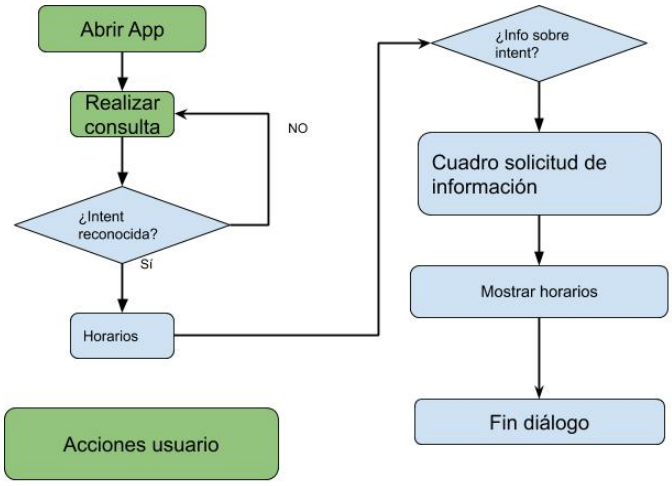


Figura 6.7: Diagrama de flujo - Horarios

6.3. Módulos del sistema

En este apartado se explican los diferentes módulos en los que se divide, así como las tecnologías empleadas en el proyecto con sus debidas justificaciones de uso.

6.3.1. Servidor propio

El servidor propio, en el que se alojan los componentes principales del proyecto, se compone de diferentes tecnologías para su funcionamiento. Los componentes del sistema son los siguientes:

- Servidor HTTP.
- Servicio de aplicación.
- Framework Web.
- Base de datos.

Además, el servidor proporcionado por la Facultad de Informática ejecuta como sistema operativo la distribución de GNU/Linux - Debian 9.5 "Stretch"

Servidor HTTP

La facultad de informática alojó este proyecto en un servidor suyo dentro de una máquina virtual accesible mediante redirecciones en un servidor NGINX, por tanto, esta capa de abstracción proporcionada hizo que no necesitásemos instalar un servidor HTTP propio.

Lenguaje para el servicio de aplicación

Los lenguajes candidatos a ser utilizados para desarrollar la aplicación fueron PHP y Python.

- Python: Las principales ventajas de utilizar Python son nuestra mayor experiencia de uso en este, el ser un lenguaje desarrollado con la idea de estar orientado a objetos, su sintaxis clara, la extensión con módulos de terceros para utilizar módulos NLP externos y su mayor facilidad a la hora de utilizar la herencia. Como desventaja principal se encuentra una mayor dificultad de despliegue, siendo necesaria la instalación de diferentes módulos extra de forma semimanual.
- PHP: Las principales ventajas de utilizar PHP son su amplia compatibilidad y fácil despliegue con en servidores HTTP estándar , una mayor cantidad de instrucciones nativas del lenguaje y una gran cantidad de documentación. Como desventajas está la poca experiencia de uso de

este lenguaje, su sintaxis y su mayor complejidad a la hora de escalar recursos.

Finalmente escogemos Python 3.6.8. (Browning y Alchin, 2019)

Framework Web

Los framework web Python analizados han sido Bottle y Flask, sin embargo, las diferencias entre ambos son mínimas. Por tanto, por experiencia de uso y por recomendación de los directores del proyecto, optamos por utilizar el Framework Bottle 0.12.16

Base de datos

El propósito de esta base de datos consiste en almacenar la información institucional de las bibliotecas y, además, un historial de conversaciones y "Slots" de Rasa. El historial de conversaciones permite el soporte de múltiples usuarios en Rasa y la capacidad de ofrecer una conversación manteniendo un hilo conductor. Por último, también se almacenan los identificadores únicos de los ejemplares consultados y los intents, de esta manera se pueden agilizar algunas de las consultas.

Para esto, como primera idea, pensamos en utilizar un crawler[1] para extraer la información directamente desde la web. Para la implementación de estas características, procedemos al análisis de dichas páginas web de las bibliotecas. Durante este análisis, nos encontramos con un importante problema de descoordinación referente a la estructura de cada página, ya que cada biblioteca introduce la información de manera diferente y sin etiquetar, por tanto, esto dificulta enormemente la extracción de los datos de estas y prácticamente nos impide utilizar estas tecnologías.

Como alternativa, estudiamos el recopilar los datos de las bibliotecas de forma manual e insertarlos en una base de datos propia, siendo una alternativa viable y con un tiempo de implementación razonable.

Tras analizar las alternativas en el punto 3.2.4. Decidimos utilizar una base de datos NoSQL por los siguientes motivos:

- En principio, no se prevé realizar cambios constantemente en los datos, por tanto, preferimos ganar en rendimiento a disponer de las características de atomicidad de consultas.
- Durante el desarrollo de este proyecto, la estructura de los datos puede variar con el tiempo, por tanto, las facilidades que brindan las bases de datos NoSQL al realizar estas operaciones son un importante punto para considerar su uso.

[1] `https://en.wikipedia.org/wiki/Web_crawler`

- En caso de necesitar más rendimiento, es más fácil su ampliación que en las bases de datos SQL.
- Rasa ofrece compatibilidad nativa con MongoDB para la serialización de conversaciones.

Con esto, optamos por utilizar MongoDB Community Edition, dado que ambos hemos utilizado esta base de datos, conocemos bien su funcionamiento y da muchas facilidades para ser utilizado con Rasa Core.

6.3.2. Procesador de lenguaje

Esta sección está divida en dos fases. La primera, haciendo uso de un proveedor cloud que nos han ayudado al desarrollo con un gran nivel de abstracción para poder comprender el funcionamiento de un chatbot. La segunda, habiendo recogido toda la información posible de la primera fase, optamos por el desarrollo de un chatbot utilizado librerías NLP y NLU.

Primera fase

Con los datos analizados en el punto 3.3.1 descartamos los siguientes proveedores:

- Microsoft Azure CS-LUIS, por ser la que menos características ofrece en su modalidad gratuita.
- Google DialogFlow, en primer lugar, por requerir el uso de una tarjeta de crédito y por disponer de una API más compleja que las demás alternativas.
- Amazon Lex, por su dificultad añadida en comparación con IBM Watson Assistant.

Por consiguiente, escogemos IBM Watson Assistant (Azraq et al., 2017) dado que es la herramienta que más facilidades pone a la hora de desarrollar y sus funcionalidades incluidas en la suscripción cumplen con los requisitos esperados.

Segunda fase

Con los datos analizados en el punto 3.3.2, decidimos utilizar spaCy. Con motivo del cambio de procesador de lenguaje natural a spaCy, nos encontramos con la problemática de que este no incluye características fundamentales para este proyecto como el reconocimiento de intents o la capacidad de mantener un flujo de diálogo con hilo conductor con el usuario.

Dado que la versión 0.2 (que puede consultarse en el anexo D.2) del proyecto utilizaba IBM Watson Assistant, buscamos alternativas que tuviesen una funcionalidad muy parecida, igual o superior a Watson. Añadimos pues como nuevo requisito que esta fuése una solución open source capaz de ejecutarse en el servidor proporcionado por la universidad.

Para solucionar esta carencia y cumpliendo con los requisitos anteriores utilizamos Rasa, introducido en la sección 3.3.3

6.3.3. Tipo de aplicación para el dispositivo móvil

Es la parte de la aplicación que recogerá las peticiones del usuario, la implementación se lleva a cabo en dispositivos móviles con sistema operativo Android y iOS. Debido a este objetivo, existen varias tecnologías para llevar a cabo la implementación.

- Cliente nativo: La gran ventaja de realizar los clientes de ambas aplicaciones en nativo, es decir, en el lenguaje perteneciente a cada sistema operativo, es su gran rendimiento frente a las demás alternativa. La única desventaja que nos encontramos es la realización de un mismo cliente para cada sistema operativo.

- Cordova: Su función es la creación de una aplicación sin depender del sistema operativo, así como su versión. Utilizar esta tecnología conlleva una ventaja sobresaliente, y es la realización de una única implementación para cualquier dispositivo móvil, evitando múltiples implementaciones de la misma aplicación. Además, tiene gran compatibilidad entre las distintas versiones de cada sistema operativo. Como desventaja cabe destacar el desconocimiento de su uso e implementación.

Con estos puntos analizados, optamos por usar un cliente nativo tanto para Android como iOS, dado que tenemos experiencia desenvolviéndonos con la implementación de aplicaciones para estos dos sistemas operativos.

6.3.4. Cliente para iOS

Introducción

El cliente para iOS está desarrollado con el lenguaje Swift 4.2 (Apple, 2018), dado que es la última versión de Swift pública en el momento en el que se empezó a desarrollar el cliente para iOS (noviembre de 2018) proporcionada por Apple y que el antiguo lenguaje Objective-C, lanzado por primera vez en 1980, está en desuso. La aplicación será de tipo universal, es decir, en un único binario ejecutable se incluye una versión para iPhone y para iPad, aumentando las opciones de uso al usuario. El cliente funciona sobre cualquier dispositivo iOS que sea compatible con iOS 10.3.3 o superior, permitiendo su uso en dispositivos de hasta 6 años de antigüedad como

pueden ser el iPhone 5 o el iPad (4ª generación). El motivo por el cual no se soportan versiones anteriores son los siguientes:

- Las versiones anteriores a iOS 10 se decide excluirlas porque iOS 10.3.3 es la versión más estable y segura lanzada por Apple de esta iteración del sistema operativo. Además, todos los dispositivos compatibles con iOS 10 y sus subversiones son compatibles con iOS 10.3.3, pudiendo actualizar sin ninguna dificultad e inconveniente. Permitir el uso en versiones más antiguas supondría tener que hacer pruebas en estas para garantizar su correcto funcionamiento.

- iOS 9.3.5 y anteriores no son compatibles con las API's de reconocimiento de voz proporcionadas por Apple, siendo imposible adaptar la interfaz de entrada de datos a estas versiones. Por otro lado, el rango de dispositivos en funcionamiento en la actualidad que utiliza versiones anteriores a iOS 11 no supera el 8 %[2].

El desarrollo de un cliente para este sistema conlleva una serie de retos y de obstáculos, todos ellos relacionados con la filosofía cerrada del ecosistema de Apple. El principal problema de esta plataforma es que iOS solo puede ejecutar aplicaciones creadas en Objective-C o en Swift. Si bien es cierto que existen proyectos externos que permiten desarrollar las aplicaciones en otros lenguajes como en C# con el entorno de desarrollo Microsoft Visual Studio (antiguamente, Xamarin), no es viable su utilización por sus limitaciones a la hora de utilizar API's más concretas como puede ser la API del reconocimiento de voz.

Para desarrollar una aplicación en Swift es un requisito obligatorio disponer de un Mac con el entorno de desarrollo gratuito Xcode. Este entorno incluye todas las herramientas necesarias para crear la aplicación, diseñar su interfaz gráfica, compilar y depurar la aplicación, ya sea a través del simulador de iOS o a través de un dispositivo físico conectado a través de un puerto USB.

Por último, Apple ofrece una modalidad gratuita de sus cuentas de desarrollo, con la limitación de probar la aplicación en hasta 3 dispositivos y en un máximo de 7 días, a partir del cual la aplicación deja de funcionar y se requiere conectar el dispositivo a Xcode para obtener nuevamente la licencia de prueba. Este tipo de cuentas solo permiten instalar la aplicación en un dispositivo del desarrollador y no permite su instalación en ningún dispositivo para el usuario final. Para probar la aplicación en múltiples dispositivos sin limitaciones de tiempo y publicar la aplicación en la tienda de aplicaciones de Apple (App Store) o distribuir un paquete de instalación (.ipa) en un entorno cerrado, se exige pagar por una cuenta de desarrollador de Apple. El coste de este tipo de cuentas asciende a 99$ al año. Como paso final para

[2] `https://developer.apple.com/support/app-store/` - A 24 de febrero de 2019

publicar el cliente en la App Store, Apple realiza una comprobación de la aplicación para comprobar que el desarrollador cumple con las normas de publicación de la App Store[3]. Los ejecutables distribuidos a través de ficheros .ipa no requieren realizar este paso.

Arquitectura de la aplicación

La arquitectura de la aplicación se basa en un modelo cliente-servidor, en el cual la aplicación cliente realiza operaciones muy sencillas y el servidor procesa las tareas complejas del proyecto.

La aplicación se compone de un View Controller principal, sobre el que se depositan los diferentes controladores de vistas de la aplicación, y un DAO, que se encarga de enviar y recibir información al servidor.

El View Controller contiene diferentes elementos, entre los que se encuentran:

- Dos UIView: En estos se depositan la imagen de fondo que cubre toda la aplicación y una vista que ofrece una capa grisácea para facilitar la lectura de la información en la pantalla.
- UIButton: Este es el botón del micrófono que inicia y detiene la captación de voz a texto.
- UIActivityIndicatorView: Contiene un "spinner" de carga oculto. Este se activa cuando se realiza una petición, ocultando el botón del micrófono anterior y con la finalidad de indicar al usuario que se está cargando su petición.
- UITableView: En esta vista de tabla se depositan los diferentes globos de mensajes de la aplicación.

El UITableView es una parte fundamental de la aplicación, por tanto, ha sido construido pensando en cada uno de los mensajes que la aplicación es capaz de recibir. Dentro del UITableView se almacenan UITableViewCell, estos almacenan los propios globos de mensajes. Para garantizar la compatibilidad con los distintos tipos de mensaje y con el objetivo de facilitar su desarrollo, utilizamos herencia y polimorfismo para esta clase. De manera que queda de la siguiente manera:

- TableViewCell: Será la clase padre, que contendrá una variable UILabel que almacenará la respuesta de texto y un UIImage que almacena la imagen de la burbuja.
- MessageViewCell: Hereda directamente de la clase TableViewCell y determina la posición del globo en función del remitente de dicho mensaje.

[3] `https://developer.apple.com/app-store/review/guidelines/`

- SingleBookViewCell: Hereda de TableViewCell y almacena una UIView que contiene la portada del libro y varios UILabel con la información de un único libro.
- ListBooksViewCell: Hereda de TableViewCell y almacena tres UIView que contiene la portada del libro y varios UILabel con la información de los tres libros.
- MapViewCell: Hereda de TableViewCell y almacena un MKMapView que contiene el mapa en su ubicación correspondiente y varios UILabel con la información de localización de la biblioteca solicitada.
- PhoneViewCell: Hereda de TableViewCell y almacena un UIView que contiene teléfono de la biblioteca correspondiente y varios UILabel con la información de la biblioteca solicitada.

Para el paso de información entre los distintos módulos se utiliza la clase Globos.swift como contenedor y transfer de información.

6.3.5. Cliente para Android

El cliente nativo para dispositivos móviles Android está desarrollado con el lenguaje Java 1.8.0_191[4]. El cliente funcionará como mínimo sobre cualquier dispositivo móvil Android cuya versión sea igual o superior a Android 5.1 (nombre en clave: Lollipop), aunque la aplicación se haya desarrollado teniendo como objetivo Android 9.0 (nombre en clave: Pie).

- Hemos decidido optar como versión mínima Lollipop debido a que, según la figura 5.2, la versión 5.1 Lollipop es la versión mínima más utilizada, llegando así a la inmensa mayoría de dispositivos móviles Android (como se puede observar en la imagen 6.8)[5].
- La versión objetivo ha sido 9.0 Android Pie, debido a que ha sido la última versión al mercado en el transcurso del desarrollo de este proyecto. La versión objetivo indica los requisitos recomendables para ejecutar la aplicación, para que la experiencia de uso no se vea afectada por bajo rendimiento.

El desarrollo de una aplicación de Android es muy amigable, ya que el IDE de desarrollo (Android Studio) se puede instalar en Windows, Linux y Mac, dando así amplia libertad. Además, cuenta con una gran comunidad de usuario que ayudan con cualquier problema que surja.

[4] `https://www.oracle.com/technetwork/java/javase/8u191-relnotes-5032181.html`

[5] `http://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide` - A enero de 2019

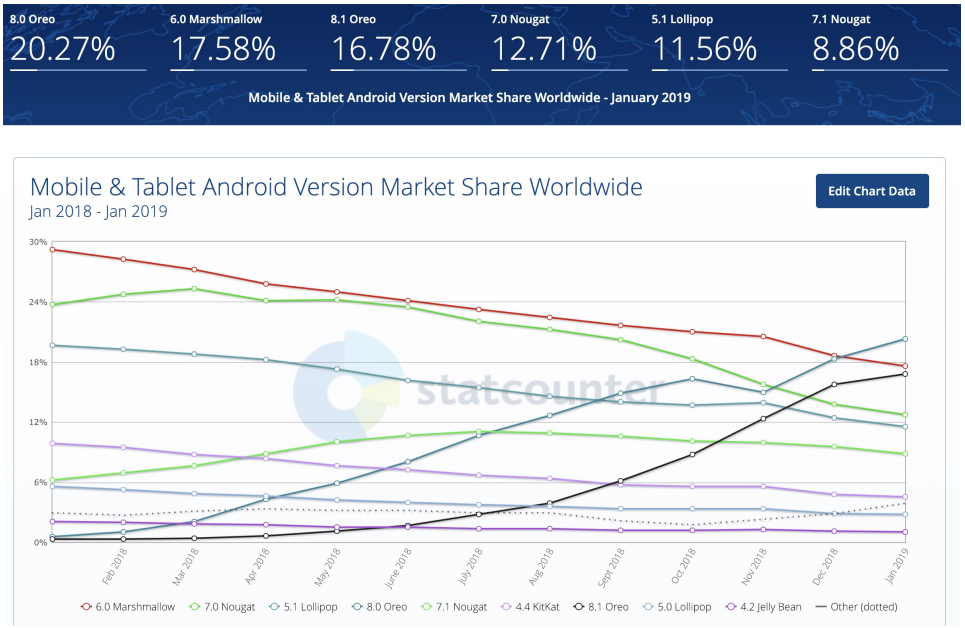


Figura 6.8: Porcentaje de distribución de Android

Para desplegar la aplicación con el objetivo de realizar pruebas, se puede ejecutar tanto en un dispositivo móvil físico conectado al ordenador, como ejecutar una máquina virtual que simula un móvil a través de Android Studio.

Para poder desarrollar en Android no se requiere ningun tipo de cuenta de desarrollo. Sin embargo, si se desea publicar la aplicación en la Play Store para distribuirla, es necesaria una cuenta, con una única cuota de 25$ [6].

6.4. Despliegue del sistema

En esta sección se detallan los requisitos y la forma de actuar para desplegar el sistema a producción.

6.4.1. Servidor

El conjunto de aplicaciones desarrollados para el servidor del sistema son multiplataforma, sin embargo, a continuación, indicamos los requisitos mínimos 6.1 y recomendados 6.2

Cabe destacar que el proyecto ha sido desarrollado y probado con Python 3.6.8, sin embargo, es posible que en un futuro versiones modernas de Rasa acepten compatibilidad con versiones más modernas como la 3.7.3

[6] `https://support.google.com/googleplay/android-developer/answer/6112435?hl=es` - 26 de Mayo de 2019

Requisitos mínimos	
Sistema Operativo	Windows 7 SP1 Ubuntu 14.04/Debian 8 "Jessie" OS X Mavericks
Procesador	Intel Core i5 doble núcleo con conjunto de instrucciones AVX
Memoria RAM	2GB
Python	3.4
MongoDB	3.6

Tabla 6.1: Requisitos mínimos del servidor

Requisitos recomendados	
Sistema Operativo	Windows 10 Ubuntu 19.04/Debian 9 "Stretch" macOS Mojave
Procesador	Intel Core i7 cuatro núcleos con conjunto de instrucciones AVX
Memoria RAM	8GB
Python	3.6.8
MongoDB	4.0

Tabla 6.2: Requisitos recomendados del servidor

Las dependencias del proyecto están recopiladas en el archivo "requirements.txt" del repositorio para automatizar su instalación.

El proyecto incluye scripts de instalación diseñados para distribuciones GNU/Linux "Debian" y "Ubuntu". (Parker, 2011)

Problemas durante el despliegue

En primer lugar, es muy importante destacar que Rasa Core 0.13.7 tiene un fallo a la hora de cargar de disco los modelos de entrenamiento, esto implicaría problemas al instalar el sistema en función del sistema operativo de la máquina. Por ello, en el repositorio, hemos cargado una versión del módulo de Rasa afectado ("`regex_featuryzer.py`") con correcciones propias. El script de instalación desarrollado para el proyecto tiene en cuenta este defecto y sustituye el módulo incorrecto por el nuestro.

En segundo lugar, el módulo "tensorflow" utilizado por spaCy requiere desde la versión 1.6.0 de una máquina con procesador compatible con el conjunto de instrucciones AVX[7] (Advanced Vector Extensions). El servidor proporcionado por la facultad de informática no posee un procesador que cumpla este requisito, por tanto, Rasa no es capaz de ejecutarse.

Para este problema estudiamos tres posibles soluciones:

- Instalar la versión 1.5.0 de "tensorflow"
- Compilar "tensorflow" de manera que no precise el uso de este conjunto de instrucciones.
- Realizar el despliegue en una máquina compatible apoyándonos de una plataforma cloud.

La primera solución resultó imposible, dado que es una versión muy antigua y tanto Rasa como spaCy utilizan funcionalidades no disponibles en esta versión. Rasa no logró ejecutarse en ningún momento.

La segunda solución requirió varios días de formación, dado que los desarrolladores de "tensorflow" (Google), facilitan únicamente el código fuente de la solución. Para compilar "tensorflow" se necesitó instalar la herramienta "bazel"[8] y ajustar en los parámetros de compilación las características exactas de la máquina donde el sistema se va a ejecutar. La compilación resultó con éxito y Rasa pudo ejecutarse correctamente. Sin embargo, al realizar las pruebas de funcionamiento, se detectó que el sistema ofrecía resultados incorrectos respecto a máquinas locales, siendo la única diferencia entre estas el componente.

Finalmente se opta por desplegar la solución en un entorno cloud. Dadas las capacidades de la máquina y sus costes, optamos por utilizar Microsoft

[7] `https://en.wikipedia.org/wiki/Advanced_Vector_Extensions`
[8] `https://bazel.build/`

Azure (Microsoft, 2018), dado que es el único que da una prueba mensual con crédito de 170€, suficientes para lanzar una máquina cumpliendo los requisitos recomendados de este punto.

Para evitar problemas de latencia, decidimos desplegar Rasa en este cloud y el resto de los módulos del sistema en el servidor proporcionado por la facultad de informática. Los resultados de hacer este despliegue fueron los esperados.

6.4.2. Clientes

Las aplicaciones para los dispositivos móviles, como se indica en el capítulo 6, han sido desarrolladas exclusivamente para iOS y Android.

Los requisitos de estas aplicaciones son los siguientes:

- iOS: iPhone, iPad o iPod Touch con iOS 10.3.3 o superior.
- Android: Dispositivo Android con Lollipop (5.1) o superior.

El proceso de instalación es muy sencillo para ambas plataformas. En el caso de iOS el usuario simplemente entrará en la App Store de su dispositivo, escribirá "Janet ucm" en el buscador y pulsará en botón instalar. En el caso de Android, el usuario accederá a la Play Store, escribirá "Janet ucm" en el buscador y pulsará en el botón de instalar. Opcionalmente, podrá descargar un instalador en formato APK y activará el ajuste de seguridad "Permitir origines desconocidos".

Capítulo 7

Ejemplo de uso

Resumen: En este capítulo explicaremos el funcionamiento de todo el sistema mediante un ejemplo de uso.

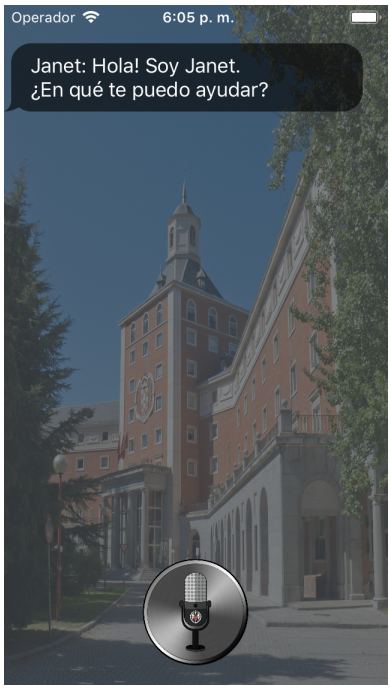


Figura 7.1: Ejemplo 1 - Petición del usuario

Supongamos que queremos obtener libros que tengan relacion con una determinada saga, en este caso, queremos libros relacionados con la saga de El Señor de los Anillos. Para ello, una vez que tengamos la aplicación abierta

(figura 7.1), pulsaremos sobre el botón del micrófono y le indicaremos la consulta deseada.

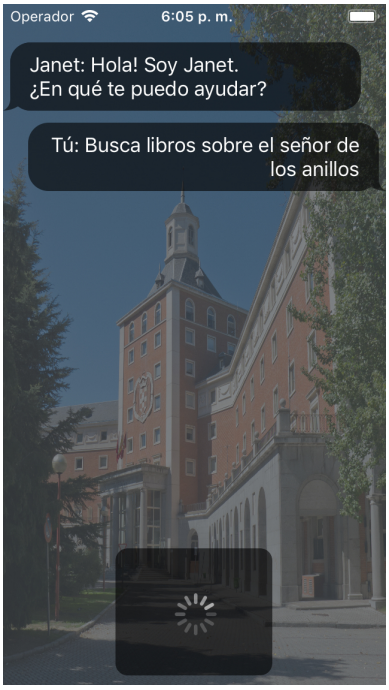


Figura 7.2: Ejemplo 2 - Petición del usuario

Una vez que el cliente ha reconocido la petición realizada mediante voz (figura 7.2), esta petición será enviada al back-end. Este se encargará de transmitirle esta información al módulo controlador y este se la enviará al NLP para su procesamiento. En nuestro caso, el sistema ha reconocido que necesitamos varios libros sobre un tema como intent, y "El Señor de los Anillos" como entity y la respuesta de texto que el cliente ha de mostrar en pantalla. Así pues, el NLP enviará esta información estructurada de vuelta al controlador, siendo este último el que le enviará la información al módulo "buscador biblioteca". Este módulo se escargará de solicitar la información a OCLC mediante su API. El resultado obtenido será devuelto al controlador, después al back-end para que finalmente, llegue al cliente. Al mismo tiempo el controlador enviará los identificadores únicos (oclc) de los ejemplares buscados y el identificado único del usuario al módulo DAO para guardar la consulta en nuestra base de datos con el fin de agilizar futuras consultas relacionadas con estos libros.

En el momento que el cliente ha recibido toda la información necesaria sobre la petición, en este caso una lista de libros, procederá a estructurar dicha información para mostrarla por pantalla (figura 7.3).

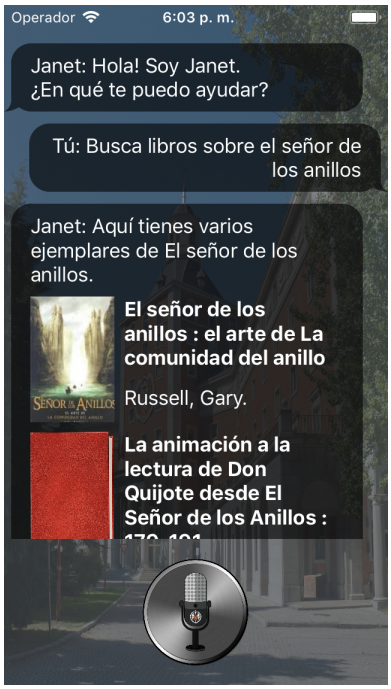


Figura 7.3: Ejemplo 3 - Resultado de la consulta

Capítulo 8

Conclusiones y Trabajo Futuro

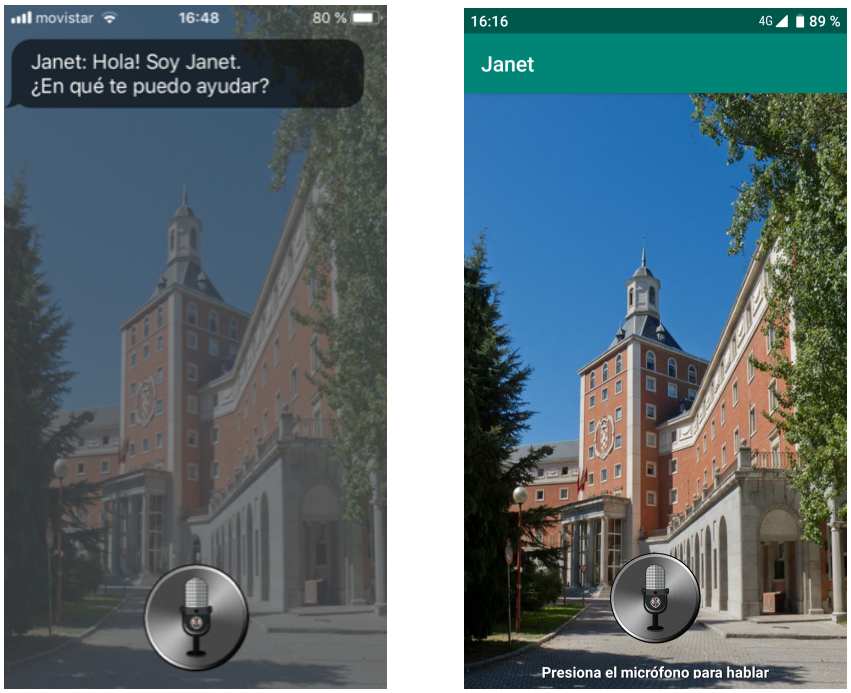
Resumen: En este capítulo se incluyen las conclusiones del trabajo realizado en este proyecto y cómo se puede mejorar y/o ampliar en el futuro.

8.1. Conclusiones del trabajo

El objetivo de este proyecto ha consistido en trasladar tecnologías de procesamiento de lenguaje natural para construir un chatbot capaz de resolver consultas relacionadas con el catálogo y la información institucional de las bibliotecas de la Universidad Complutense de Madrid a través de un dispositivo móvil (como se puede observar en la figura 8.1).

Para ello, hemos aprovechado las tecnologías de NLP disponibles en el mercado y los dispositivos móviles para ofrecer un sistema con la mayor calidad posible, partiendo de cero con los modelos de entrenamiento para el ámbito de este proyecto. En un principio utilizando la solución ofrecida por IBM con su producto Watson Assistant, con el objetivo de tener un campo de visión mayor en el campo de los chatbot y los NLP. Posteriormente, sustituimos Watson por una solución que fuera de código abierto, para no depender de las limitaciones de utilizar una solución cerrada y con funcionalidades limitadas por el uso de una suscripción gratuita. Esta solución fue spaCy más Rasa. Este conjunto de librerías en Python otorga las herramientas necesarias para la construcción de un NLP y un NLU. Con esto, podemos configurar el sistema de tal forma que tenga la capacidad de dar una respuesta acorde al servicio que se desea ofrecer.

Este proyecto ha logrado también, en cierta medida, unificar información, tanto institucional como del catálogo, de las bibliotecas, de manera que el



(a) (b)

Figura 8.1: Aplicaciones disponibles en App Store (a) y Play Store (b)

usuario no necesita bucear por las diversas webs de las bibliotecas para obtener información como los números de teléfono, horarios o direcciones.

Como conclusión, podemos determinar que el proyecto ha sido desarrollado con éxito, lanzando al público dos aplicaciones para dispositivos iOS y Android, disponibles en sus respectivas tiendas de aplicaciones, así como la implementación del servidor donde se procesan todos los datos enviados por los clientes.

8.2. Líneas de trabajo futuro

Pese a lo anterior, las dimensiones de este tipo de proyectos son muy grandes, y es complicado tratar todos los puntos que se quisieran realizar. Por ello, este trabajo puede ser ampliado desde muchos puntos. Aquí enumeramos qué puntos podrían ampliarse y/o mejorarse en este proyecto:

- Mejorar los modelos de entrenamiento: De manera que el sistema pueda reconocer intents y entities de forma más eficaz.
- Soporte multilenguaje: El proyecto actualmente solo soporta la lengua castellana para las consultas (excluyendo nombres de libros y autores, que si pueden estar en otro idioma), para facilitar el acceso a personas que hablen diferentes idiomas este punto es necesario desarrollarlo.
- Ampliar la integración con más servicios de la biblioteca de la UCM: Dados los problemas iniciales como consecuencia del cambio del sistema de gestión de la biblioteca en verano de 2018, varias funcionalidades que quisimos incorporar en este proyecto no pudieron ser desarrolladas. Algunas de estas mejoras podrían ser la capacidad de reservar y renovar ejemplares desde la propia aplicación e integrar funcionalidades de préstamo interbibliotecario, o introducir la capacidad de hacer búsquedas en el catálogo de una biblioteca concreta.
- Ampliar el dominio del proyecto con más servicios de la UCM: Sería interesante ampliar la capacidad de este sistema para poder realizar consultas a otros sistemas como el campus virtual de la UCM (moodle), o el sistema de gestión académica (GEA).
- Implementar mejoras de accesibilidad, como puedan ser que la aplicación sea capaz de controlar la velocidad a la habla mediante comandos de voz, un modo de alto contraste en el cliente de Android y mejoras también de este modo en el cliente de iOS.

Chapter 9

Conclusions and Future Work

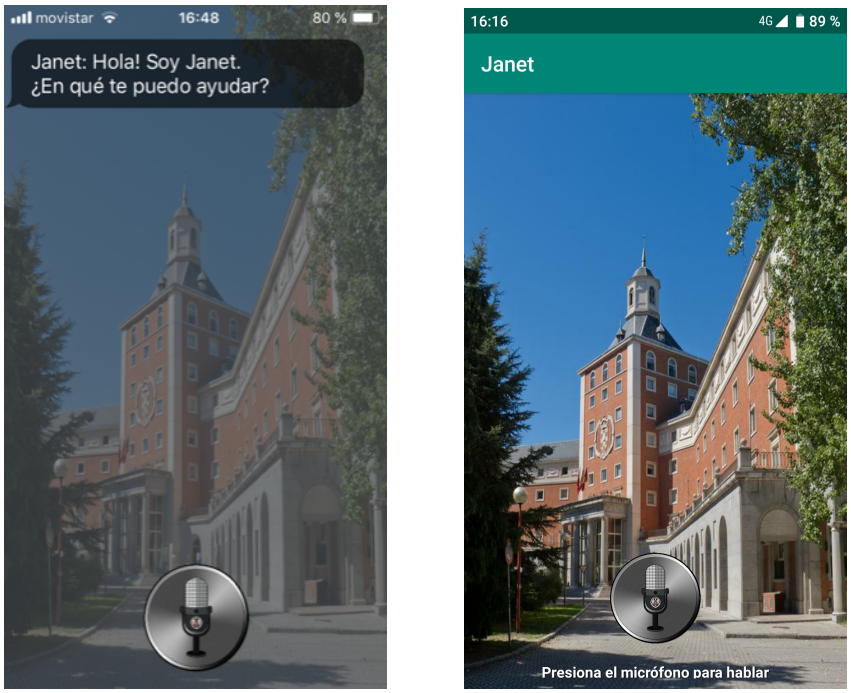
Resumen: This chapter includes the conclusions of the work carried out in this project and how it can be improved and/or expanded in the future.

9.1. Conclusions about work

The aim of this project has been to transfer NLP technologies to build a chatbot capable of resolving queries related to the catalogue and institutional information of the libraries of the Complutense University of Madrid through a mobile device (as can be seen in the figure 9.1 below).

To this end, we have taken advantage of the NLP technologies available on the market and mobile devices to offer a system with the highest possible quality, starting from scratch with the training models for the scope of this project. Initially using the solution offered by IBM with its product Watson Assistant, with the aim of having a greater field of vision in the field of chatbot and NLP. Subsequently, we replaced Watson with a solution that was open source, so as not to depend on the limitations of using a closed solution and with limited functionality by the use of a free subscription. This solution was spaCy plus Rasa. This set of libraries in Python provide the necessary tools for the construction of an NLP and an NLU. With this, we can configure the system in such a way that it has the ability to give a response according to the service you want to offer.

This project has also managed, to a certain extent, to unify information, both institutional and catalog, libraries, so that the user does not need to dive through the various web's of the libraries to obtain information such as telephone numbers, schedules or addresses.



(a) (b)

Figure 9.1: Aplicaciones disponibles en App Store (a) y Play Store (b)

On the other hand, in the beginning we wanted to make an application thought about accessibility and it has not been possible to implement it given the great ambition of the fields we wanted to deal with in this project and due to time and human resource limitations.

As a conclusion, we can determine that the project has been developed successfully, launching to the public two applications for devices iOS and Android, available in their respective application stores, as well as the implementation of the server where all the data sent by customers are processed.

9.2. Future lines of work

In spite of the above, the dimensions of this type of project are very large, and it is complicated to deal with all the points that you would like to carry out. Therefore, this work can be expanded from many points. Here we list which points could be expanded and/or improved in this project:

- Improve training models: So that the system can recognize intents and entities more effectively.
- Multilanguage support: The project currently only supports the Spanish language for queries (excluding names of books and authors, if they can be in another language), to facilitate access to people who speak different languages this point needs to be developed.
- Extend the integration with more services of the library of the UCM: Given the initial problems as a consequence of the change of the management system of the library in summer 2018, several functionalities that we wanted to incorporate in this project could not be developed. Some of these improvements could be the ability to reserve and renew copies from the application itself and integrate interlibrary loan functionalities, or introduce the ability to search the catalogue of a particular library.
- Extend the domain of the project with more services of the UCM: It would be interesting to expand the capacity of this system to make queries to other systems such as the virtual campus of the UCM (moodle), or the academic management system (GEA).
- Implement accessibility improvements, such as the application being able to control speech speed through voice commands, a high-contrast mode in the Android client and improvements in this way in the iOS client.

Capítulo 10

Aportaciones individuales al proyecto

Resumen: En este capítulo exponemos el trabajo realizado por los integrantes del equipo.

10.1. Mauricio

Para la creación de este proyecto ha sido necesaria una estrecha colaboración entre nosotros y una clara división de tareas.

En los inicios del proyecto destacamos tres bloques fundamentales:

- Aplicaciones del servidor.
- Cliente iOS.
- Cliente Android.

En esta primera fase acordamos que me encargaría del cliente iOS en su totalidad y colaboraría en el desarrollo de los diferentes módulos que se ejecutasen en el servidor.

Durante la fase de captación de requisitos (hasta noviembre de 2018), las tareas que realicé fueron las siguientes:

- Asistir a las reuniones concertadas con el personal de la biblioteca.
- Investigar las diferentes tecnologías a utilizar en el proyecto.
- Programar las distintas reuniones con el personal de las bibliotecas.

- Iniciar el desarrollo de un primer prototipo de las tareas asignadas como describo anteriormente.

Tras finalizar la captación de requisitos y determinar que el proyecto se dividirá en dos fases, comienzo a trazar las líneas de diseño del futuro backend, dejando claro cuales serán los elementos accesibles por los clientes por medio de una API propia, implemento una primera conexión con el NLP de ese momento (IBM Watson Assistant) y con la API WMS. Los elementos de la API pueden consultarse en el manual de usuario del servidor, disponible en el anexo C. Además, experimento con el funcionamiento de las API's de reconocimiento de voz de iOS y desarrollo un primer prototipo del cliente para iOS, diseñando ciertos elementos de la interfaz de usuario desde cero, como pueden ser la iconografía o los sonidos de interacción de la aplicación.

Una vez acabada la primera fase del proyecto, investigo un posible sucesor de IBM Watson, encontrando spaCy para la parte NLP y Rasa para el NLU. Al descubrir estas librerías, colaboro con su implementación en el proyecto, ayudando en el diseño de los modelos de entrenamiento y desarrollando modificaciones propias para añadirle compatibilidad con los módulos ya desarrollados en el proyecto. Además, ayudo con la búsqueda de errores en el modelo realizando pruebas utilizando el prototipo del cliente de iOS desarrollado para la primera fase del proyecto.

Tras implementar una primera versión del nuevo PLN, continúo con el desarrollo del cliente de iOS y hago las modificaciones necesarias en los módulos del servidor para garantizar la compatibilidad con Rasa. Durante estas versiones integro los módulos restantes en el cliente, como son el mapa para mostrar las localizaciones, como una interfaz interactiva para mostrar números de teléfono y ofrecer la posibilidad de llamarlo con un toque.

Por otro lado, contacto nuevamente con el personal de la biblioteca para obtener cierta información institucional (números de teléfono, correos electrónicos y direcciones). Con esta información realizo el fichero de configuración para la base de datos.

Al finalizar el desarrollo y detectar los problemas de compatibilidad entre este proyecto y el servidor proporcionado por la facultad, me encargo de probar las posibles soluciones para desplegar con éxito el proyecto en un entorno real como se indica el apartado 6.4.1 del capítulo 6.

Como último desarrollo de código para las aplicaciones del servidor, me encargo del desarrollo del script de instalación.

Una vez finalizados los desarrollos principales, me puse en contacto con el personal de servicios informáticos de la Universidad para subir el cliente de iOS a la App Store. Desde servicios informáticos me proporcionaron una cuenta de desarrollador de Apple con permisos para la compilación y distribución de aplicaciones. Por tanto, mi última contribución con el proyecto a nivel de desarrollo consistió en subir la aplicación a la App Store.

De la memoria he colaborado en la redacción de los capítulos Introduc-

ción (en castellano), Estado del arte (puntos 2.2, 2.3 y 2.4), Tareas de investigación (puntos 4.1, 4.2, 4.4), Diseño tecnologías y Arquitectura (puntos 5.1 y 5.3), Despliegue del sistema, Historial de Versiones y diferentes anexos (Historias de Usuario y Manual de usuario de iOS y Servidor).

10.2. Jose Luis

Debido a que hemos realizado este proyecto dos personas, ha sido necesario una estrecha colaboración con el fin de complementarnos mutuamente y conseguir con éxito finalizarlo.

En una primera reunión entre nosotros, establecemos los módulos en los que se divide la arquitectura de nuestro proyecto, así como los hitos a conseguir durante la realización del proyecto, estableciendo unas fechas de guía, para poder lograr una organización mucho más efectiva.

Me encargo del desarrollo del cliente Android y colaboro en la investigación y desarrollo de las diferentes partes de la implementación del servidor.

En lo referente a la primera fase, la captación de requisitos, me encargué de:

- Asistir a las reuniones concertadas con el personal de la biblioteca
- Investigar las diferentes tecnologías a utilizar en el proyecto
- Creación de un primer prototipo.

En el transcurso de la captación de requisitos, se fue analizando todas las necesidades que el personal de la biblioteca nos comentaba, haciendo así un borrador de aquellas historias de usuario, desarrollándolas y asignándoles prioridad.

Al finalizar la captación de requisitos, determinamos dividir el proyecto en dos fases. La primera consistió en implementar un NLP (IBM Watson Assistant) para ayudarnos a la comprensión de los conceptos usados en el NLP. En esta fase, colaboro en la investigación sobre estos conceptos y en el funcionamiento de la API de reconocimiento de voz de Android, y además realizo el desarrollo de una primerísima versión del cliente Android.

Una vez acabada la primera fase de nuestro proyecto, investigo a la par que mi compañero sobre una alternativa a IBM Watson Assistant (a poder ser open source). Para la implementación del NLP, nuestros tutores nos recomiendan algunas librerías en Python para la implementación de este. Entonces, realizo una lista de ventajas y desventajas sobre las distintas librerías, quedándonos finalmente con spaCy. Además, realizo pruebas de desarrollo y uso de esta librería para el reconocimiento de intents y entities.

Pero con esta librería no sería suficiente para sustituir al IBM Watson Assistant, ya que faltaría el flujo de diálogo y las respuestas que daría nuestro sistema. Así que investigo junto con mi compañero en una posible solución a este problema. Es entonces cuando, después de una lista de sustitutos con pros y contras, decidimos usar Rasa. En colaboración con mi compañero, desarrollamos modelos de entrenamiento para que el sistema "aprenda", el uso de funcionalidades ofrecidas por esta librería y la corrección de posibles errores encontrados en nuestra implementación.

A la par que finalizamos con una primera versión de nuestro NLP, soluciono posibles fallos encontrados en el cliente de Android debido al cambio por completo de una parte importante del backend.

En lo referente a la elaboración de esta memoria, he colaborado en la redacción y corrección de errores de los capítulos Introducción (en inglés), Estado del arte, Análisis de funcionalidad, Diseño, tecnologías y arquitectura del sistema, Contribuciones y Anexos.

Bibliografía

Apple. *The Swift Programming Language (Swift 4.2)*. Apple, 2018. Disponible en https://books.apple.com/es/book/the-swift-programming-language-swift-4-2/id881256329?l=en (último acceso, Marzo, 2019).

Azraq, A., Aziz, H., Nappe, N., Rodriguez, C. y Sri, B. *Building Cognitive Applications with IBM Watson Services: Volume 2 Conversation*. IBM Skills Academy Program, 2017. Disponible en http://www.redbooks.ibm.com/redbooks/pdfs/sg248394.pdf (último acceso, Diciembre, 2018).

Bocklisch, T., Faulkner, J., Pawlowski, N. y Nichol, A. *Rasa: Open Source Language Understanding and Dialogue Management*. Versión electrónica, 2017. Disponible en https://arxiv.org/pdf/1712.05181.pdf (último acceso, Mayo, 2019).

Browning, J. B. y Alchin, M. *Pro Python 3 : features and tools for professional development*. Apress, 2019. Disponible en https://link.springer.com/book/10.1007%2F978-1-4842-4385-5 (último acceso, Marzo, 2019).

MacCartney, B. Understanding natural language understanding. 2014. Disponible en https://nlp.stanford.edu/~wcmac/papers/20140716-UNLU.pdf (último acceso, Abril, 2019).

Microsoft. *La guía de Azure para desarrolladores*. Microsoft, 2018. Disponible en http://download.microsoft.com/download/3/F/2/3F2FFA90-1B49-478A-9199-94106D5FB89A/Azure_Developer_Guide_eBook_es-ES.pdf (último acceso, Mayo, 2019).

OCLC. *OCLC Developer Network handbook*. OCLC, 2011. Disponible en https://www.oclc.org/content/dam/developer-network/Handouts/usb_web_services_booklet.pdf (último acceso, Mayo, 2019).

Parker, S. *Shell scripting : expert recipes for Linux, Bash, and more*. John Wiley & Sons, Incorporated, 2011. Disponible en `https://ebookcentral.proquest.com/lib/universidadcomplutense-ebooks/detail.action?docID=818941` (último acceso, Mayo, 2019).

Apéndice A

Manual de usuario Android

A.1. Historial de Versiones

Versión	Descripción
0.1	Funcionamiento para la versión 0.2.2 de la aplicación.
0.2	Funcionamiento para la versión 0.9.0 de la aplicación.
1.0	Versión final.

A.2. Requisitos de uso

Para utilizar la aplicación se requiere disponer de un Smartphone Android con la versión 5.1 (Lollipop) o superior. Además, se puede usar en el ADV Manager de Android Studio.

A.3. Pantalla principal

Figura A.1: Pantalla principal de la App.

La pantalla principal de la aplicación consta de una serie de elementos. El primero de ellos es un botón con la imagen de un micrófono. Cuando se pulsa el botón, se inicia la captura de voz. El siguiente elemento que destacar es el conjunto de mensajes. En esta parte, se pondrán por pantalla los mensajes que introduzca el usuario y nuestro sistema. Los mensajes introducidos por el usuario se encuentran en la parte derecha de la interfaz gráfica, y se representan con un bocadillo con el fondo de color blanco. Los mensajes emitidos por nuestro sistema se encuentran en la parte derecha de la interfaz

gráfica. Estos, en cambio, se representan con un bocadillo con el fondo de color verde. Además, la aplicación leerá en voz alta los mensajes emitidos por el sistema.

A.4. Hacer una consulta

Como se ha explicado anteriormente, el usuario deberá formular una consulta pulsando en el botón del micrófono. La aplicación enviará la consulta al servidor y este ofrecerá o bien un mensaje de texto o bien una lista de tres libros (en función de la consulta realizada).Cuando una consulta consta de una serie de libros, el usuario puede realizar la petición para obtener mas información sobre el libro en cuestión. Por ejemplo, si recibimos la lista que se aprecia en la figura A.2 podemos pedir al sistema más información sobre el tercer libro de la lista (fig A.3). Cuando una consulta muestra un mapa, puedes interactuar con el mapa libremente(fig A.4). Si pulsas sobre el indicador rojo, se abrirá la aplicación de mapas por defecto de Android. Cuando una consulta muestra un teléfono (fig A.5), puedes tocar encima del número de teléfono y se marcará automáticamente en la aplicación para llamar por defecto.

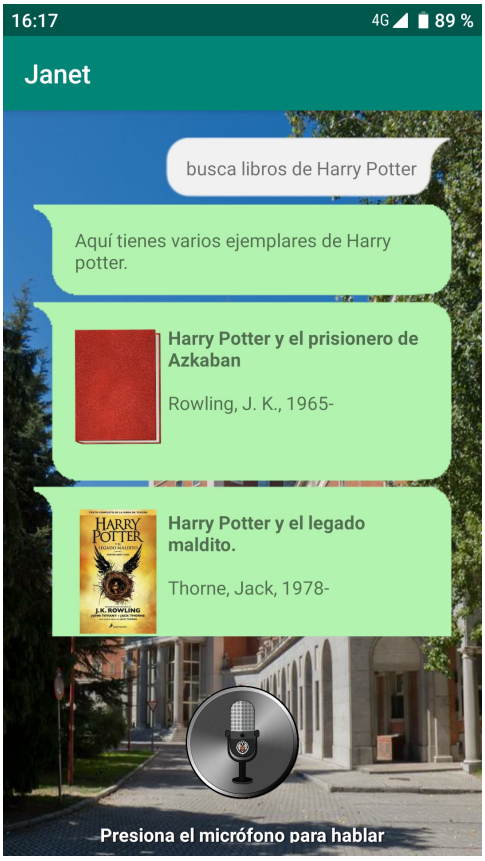


Figura A.2: Consulta de libros

Figura A.3: Solicitud de más información dada una lista de libros

Figura A.4: Mapa para llegar a una localización

Figura A.5: Teléfono de una biblioteca

Apéndice B

Manual de usuario iOS

B.1. Historial de Versiones

Versión	Descripción
0.1	Funcionamiento para la versión 0.2.2 de la aplicación.
0.2	Funcionamiento para la versión 0.9.0 de la aplicación.
1.0.1	Versión final.

B.2. Requisitos de uso

Para utilizar la aplicación se requiere disponer de un dispositivo iOS (ya sea iPhone o iPad) con la versión 10.3.3 o superior. Como alternativa, se puede utilizar el simulador de iOS incluido en el entorno de desarrollo Xcode seleccionando una versión de iOS igual o superior a la anteriormente mencionada.

B.3. Primera ejecución

Al ejecutar por primera vez la aplicación tras su instalación, la aplicación solicitará permisos de utilización del micrófono y reconocimiento de voz. El denegar estos permisos supondrá que la aplicación no funcionará, indicándolo en la pantalla principal de la aplicación (estos ajustes siempre se pueden modificar mediante la App "Ajustes" del dispositivo.

Figura B.1: Mensaje de error cuando se deniegan los permisos en la App.

B.4. Pantalla principal

La pantalla principal de la aplicación consta de una serie de elementos. El primero de ellos es un botón micrófono, que permite al usuario interactuar mediante la voz con la aplicación. Este botón permite iniciar la captura de texto y detenerla. Cuando la aplicación ha reconocido la frase, esta detendrá automáticamente el reconocimiento de texto, igualmente el usuario puede detenerla pulsando nuevamente en el botón del micrófono. El segundo elemento relevante es la interfaz de globos de mensaje. Aquí se interactúa con las respuestas proporcionadas por el servidor, mostrando texto, libros, etc. Los mensajes dictados por el usuario siempre van precedidos por la palabra "Tú:", y van alineados en la parte derecha de la pantalla. Los mensajes enviados por la aplicación siempre van precedidos por la palabra "Janet:" y van alineados en la parte izquierda de la pantalla, además, por defecto la aplicación leerá en voz alta los mensajes enviados por el servidor la primera vez que aparezcan. Si el usuario desea que los mensajes se lean posteriormente a su recepción, deberá habilitar VoiceOver en su dispositivo. La aplicación garantiza la compatibilidad total con los ajustes de accesibilidad.

B.5. Hacer una consulta

Como se ha explicado anteriormente, el usuario deberá formular una consulta pulsando en el botón del micrófono. La aplicación enviará la consulta al servidor y este ofrecerá o bien un mensaje de texto o bien una lista de tres libros (en función de la consulta realizada).

Cuando una consulta consta de una serie de libros, el usuario puede pulsar encima de uno de los libros para obtener más información sobre ese ejemplar (como, por ejemplo, su disponibilidad). Si el usuario necesita todavía más información, puede pulsar nuevamente sobre el globo del ejemplar y automáticamente la aplicación le redirigirá a la web de la biblioteca del ejemplar a través del navegador del sistema.

Cuando una consulta muestra un mapa, puedes interactuar con el mapa libremente. Si pulsas sobre el indicador rojo, se abrirá la aplicación de mapas por defecto de iOS.

Cuando una consulta muestra un teléfono, puedes tocar encima del número de teléfono para llamar a la biblioteca inmediatamente.

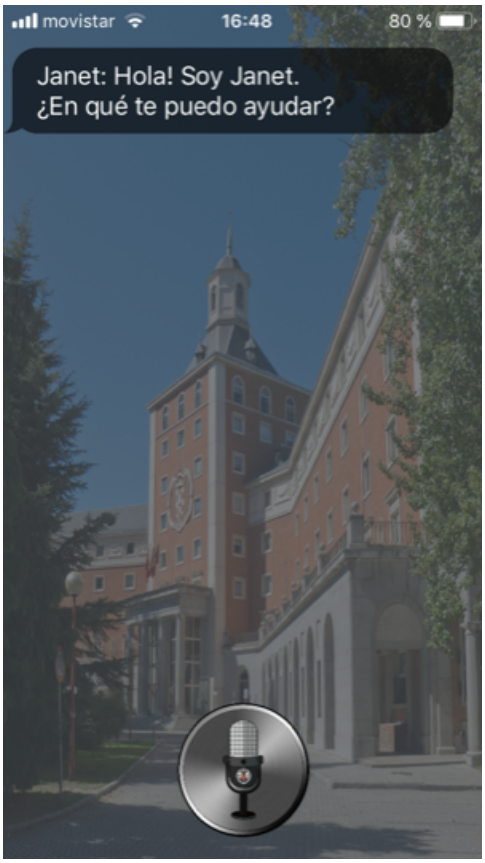


Figura B.2: Pantalla principal.

Figura B.3: Mensaje de una consulta de solo texto.

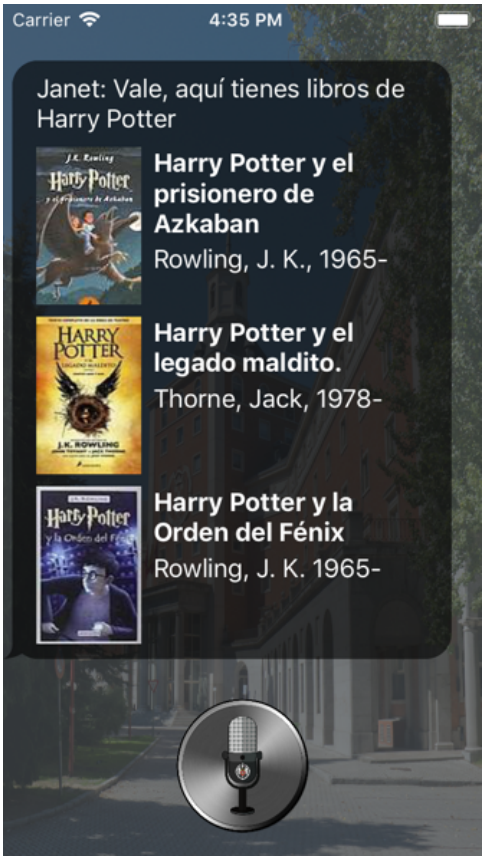


Figura B.4: Lista de libros de una consulta.

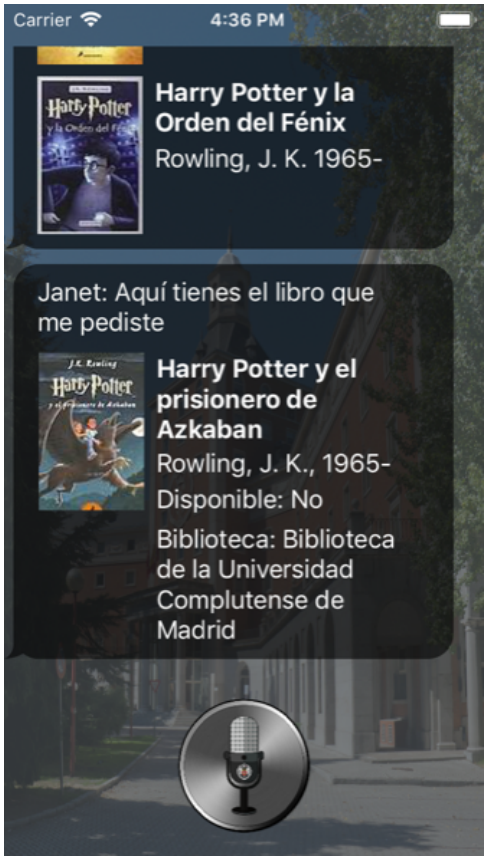


Figura B.5: Más información de un ejemplar seleccionado.

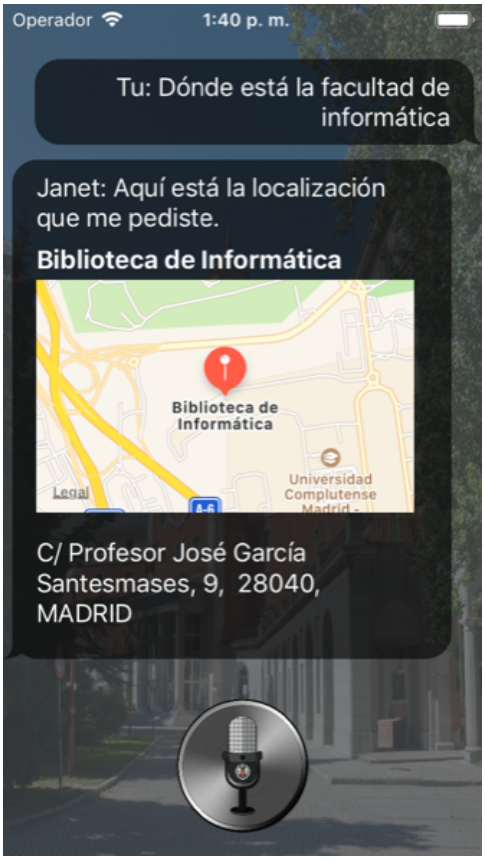


Figura B.6: Mapa para llegar a una biblioteca.

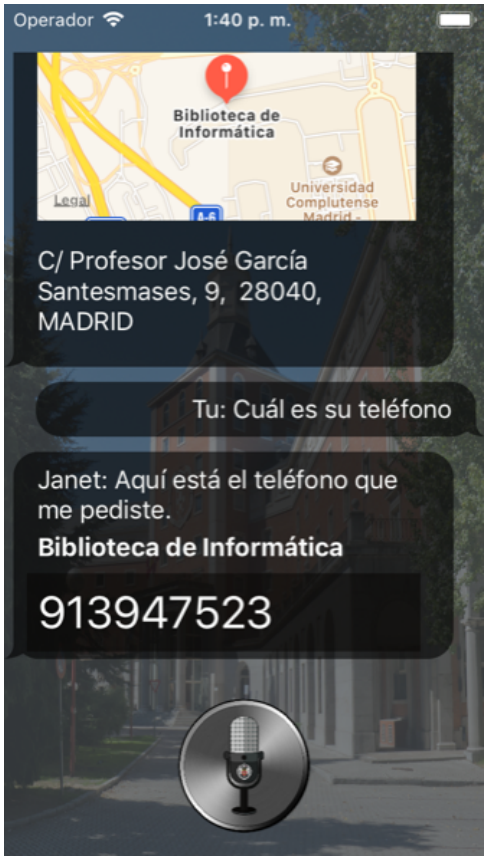


Figura B.7: Teléfono de una biblioteca.

Apéndice C

Manual de usuario Servidor

C.1. Historial de Versiones

Versión	Descripción
0.1	Funcionamiento para la versión 0.1 de la aplicación.
0.2	Actualizado para la versión 0.9 de la aplicación.
1.0	Versión final.

C.2. Requisitos de uso

- Cliente oficial[1]
- Módulo Jarvis (versión 0.5 o superior)
- MongoDB en servidor (versión 4.0.5 o superior)

C.3. Ejecución de una consulta

La aplicación escucha peticiones enviadas mediante POST a la dirección `http://url/api`, utilizando esta dirección desde un cliente oficial o un cliente REST es necesario rellenar la siguiente información en el body de la consulta:

- Type: Tipo de consulta que realiza el cliente.
- Content: Contenido de la consulta.
- user_id: Proporciona el identificador de usuario único.

Type acepta tres parámetros:

- Parámetro "query": Indica a la aplicación que el usuario desea hacer una consulta en lenguaje natural y se requiere la intervención del PLN.
- Parámetro "oclc": Indica a la aplicación que el usuario desea consultar información de un ejemplar concreto, se excluye el uso del PLN.
- Parámetro "restart": Indica a la aplicación que se deben reiniciar los slots del usuario indicado en el "user_id".

La información de "content" varía en función del parámetro establecido en "type", distinguiéndose en los siguientes casos:

- "query": "content" contendrá una consulta en lenguaje natural. Por ejemplo, "Quiero libros de Harry Potter"
- "oclc": "content" contendrá un código oclc del libro a consultar.

C.4. Resultado de una consulta

Una vez enviada esta consulta, la aplicación devolverá una respuesta encapsulada en formato JSON. El contenido del JSON depende de varios factores indicados a continuación:

[1] Además, se permite su uso utilizando un cliente REST. En este caso, recomendamos el complemento RESTClient 3.0.7 para Mozilla Firefox.

1. **Caso de fallo**
   La aplicación devolverá los siguientes parámetros:
   - errorno ->Número distinto de 0 que indica el código de error.
   - errorMessage ->Mensaje informando sobre el error para ser mostrado en el cliente.

2. **Caso de éxito. Respuesta solo texto**
   La aplicación devolverá los siguientes parámetros (como puede observarse en la imagen C.1):
   - errorno ->0
   - content-type ->text
   - response ->Respuesta a mostrar en el cliente.

3. **Caso de éxito. Respuesta lista de libros**
   La aplicación devolverá los siguientes parámetros (como puede observarse en la imagen C.2):
   - errorno ->0
   - content-type ->list-books
   - response ->Respuesta a mostrar en el cliente.
   - Lista "books" ->Lista de libros que contiene los siguientes parámetros:
     - title ->Título del ejemplar.
     - author ->Autor del ejemplar.
     - oclc ->Código OCLC del ejemplar.
     - isbn ->Lista de ISBN's del libro.

4. **Caso de éxito. Respuesta un libro**
   La aplicación devolverá los siguientes parámetros (como puede observarse en la imagen C.3):
   - errorno ->0
   - content-type ->single-books
   - response ->Respuesta a mostrar en el cliente.
   - title ->Título del ejemplar.
   - author ->Autor del ejemplar.
   - url ->URL para WorldCat de la UCM del ejemplar.
   - oclc ->Código OCLC del ejemplar.
   - isbn ->Lista de ISBN's del libro.

- available ->Lista de bibliotecas en las que el ejemplar está disponible actualmente.

5. **Caso de éxito. Respuesta de localización**
   La aplicación devolverá los siguientes parámetros (como puede observarse en la imagen C.4):
   - errorno ->0
   - content-type ->location
   - response ->Respuesta a mostrar en el cliente.
   - library ->Nombre de la biblioteca.
   - location ->Dirección de la biblioteca.
   - lat ->Latitud de la biblioteca.
   - long ->Longitud de la biblioteca.
6. **Caso de éxito. Respuesta de teléfono**
   La aplicación devolverá los siguientes parámetros:
   - errorno ->0
   - content-type ->phone
   - response ->Respuesta a mostrar en el cliente.
   - library ->Nombre de la biblioteca.
   - phone ->Teléfono de la biblioteca.

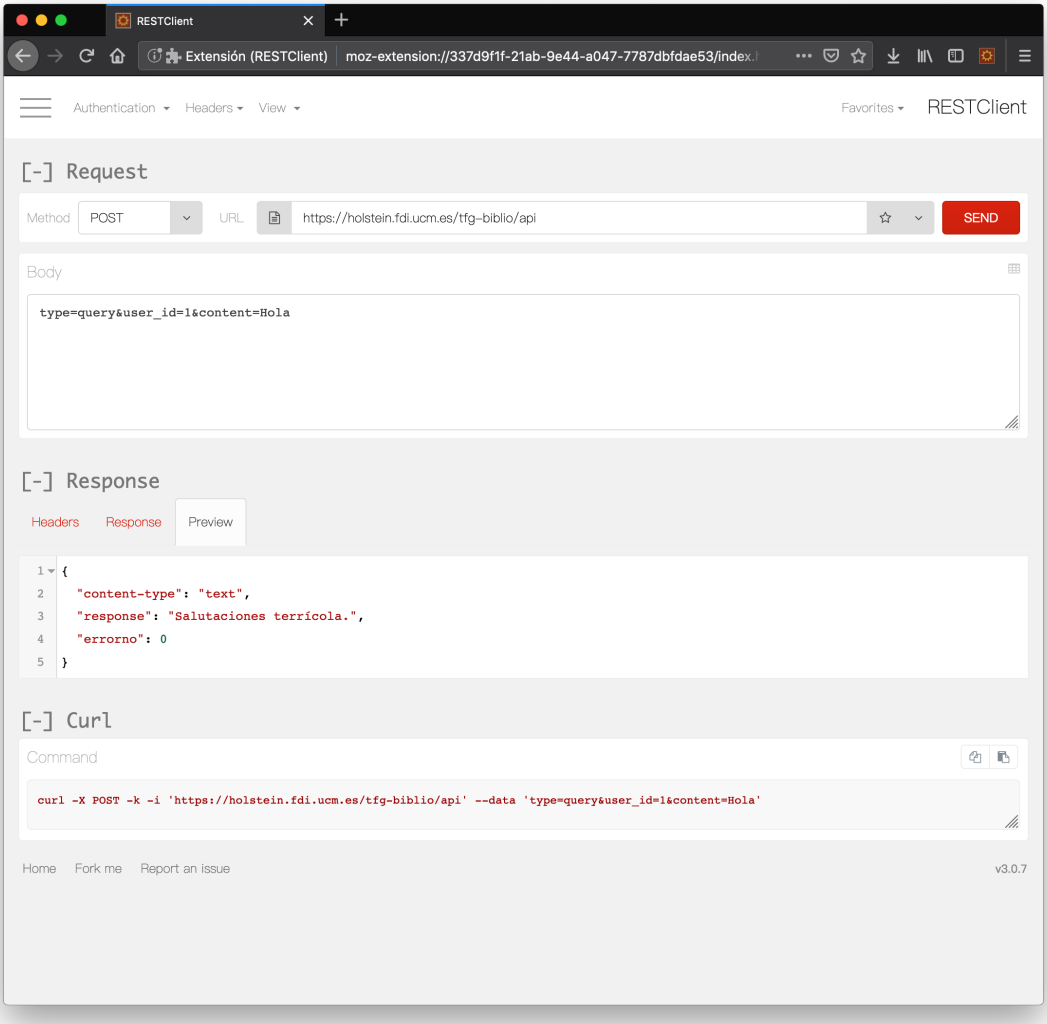


Figura C.1: Ejemplo de consulta de solo texto.

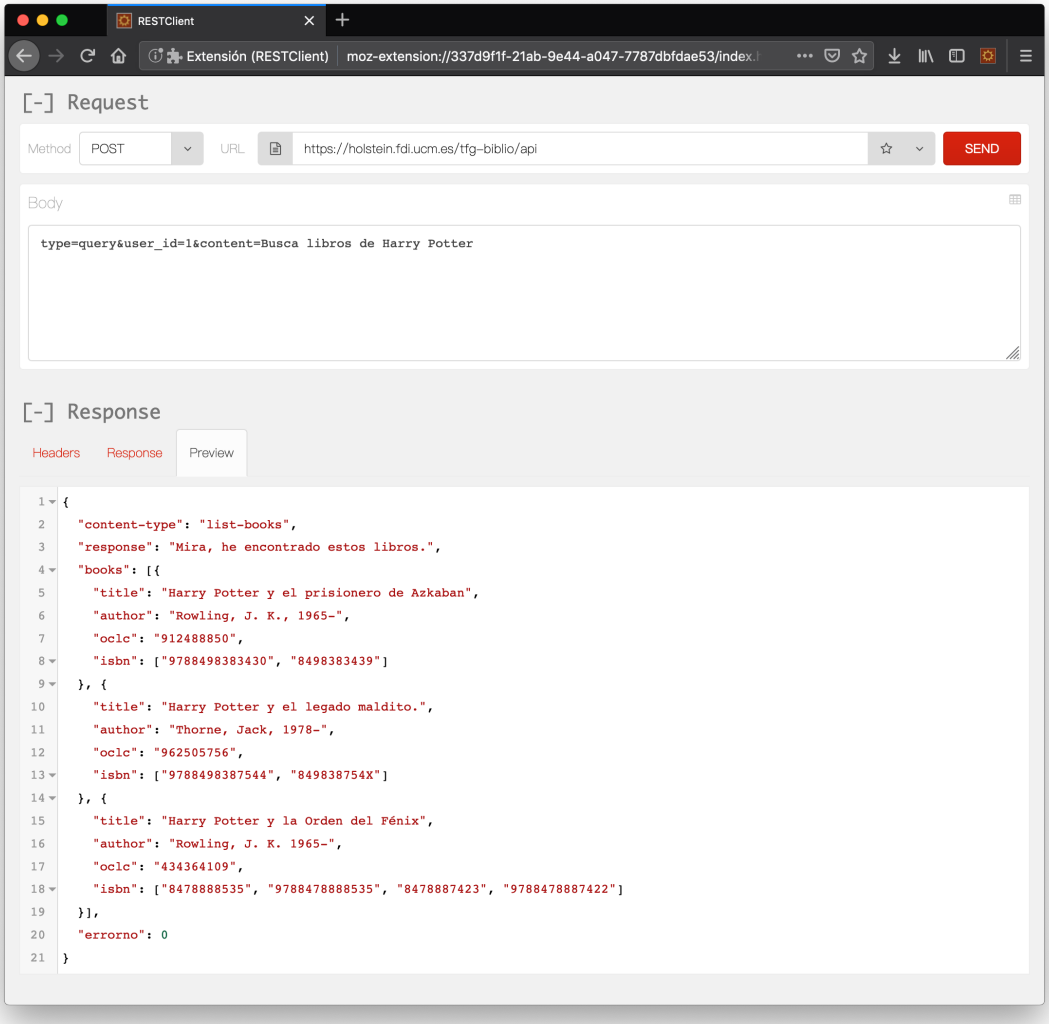


Figura C.2: Ejemplo de consulta para varios ejemplares.

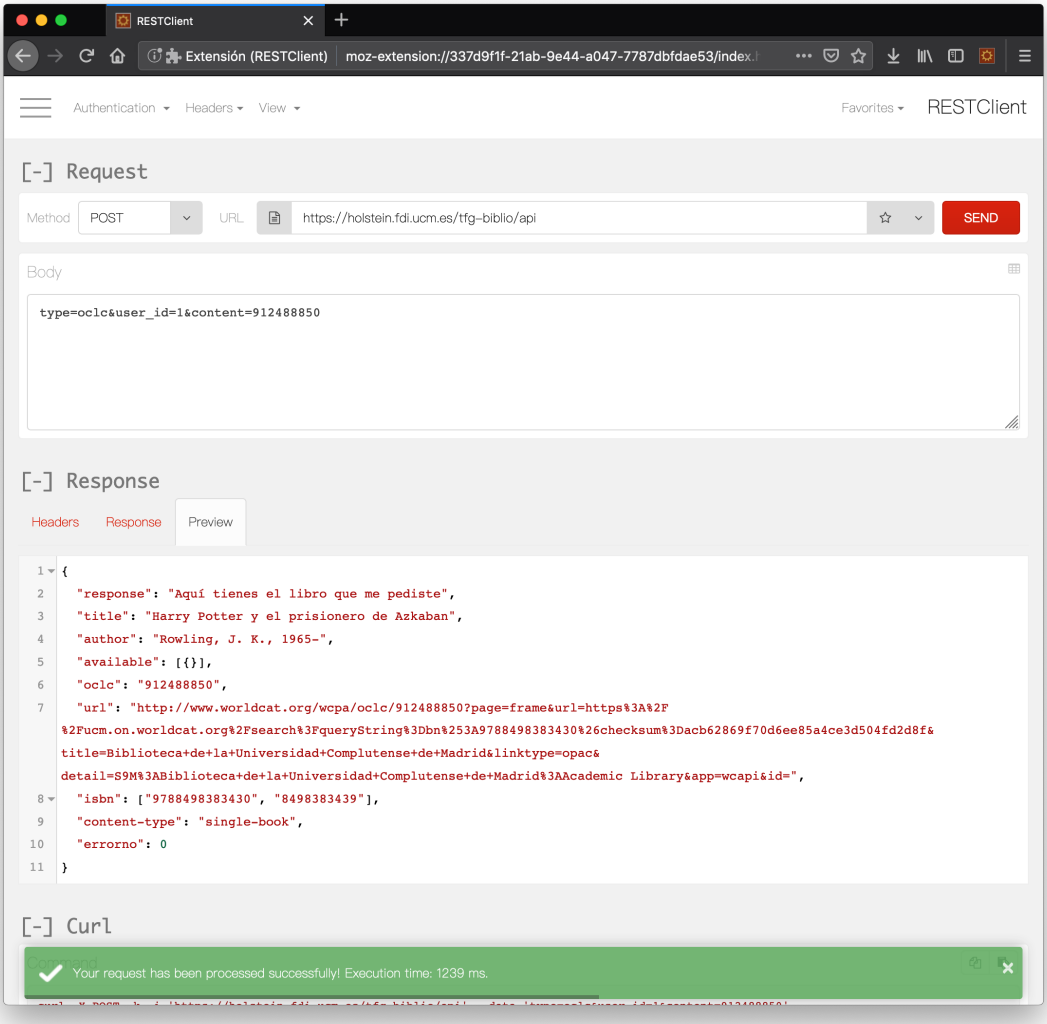


Figura C.3: Ejemplo de consulta de un único libro.

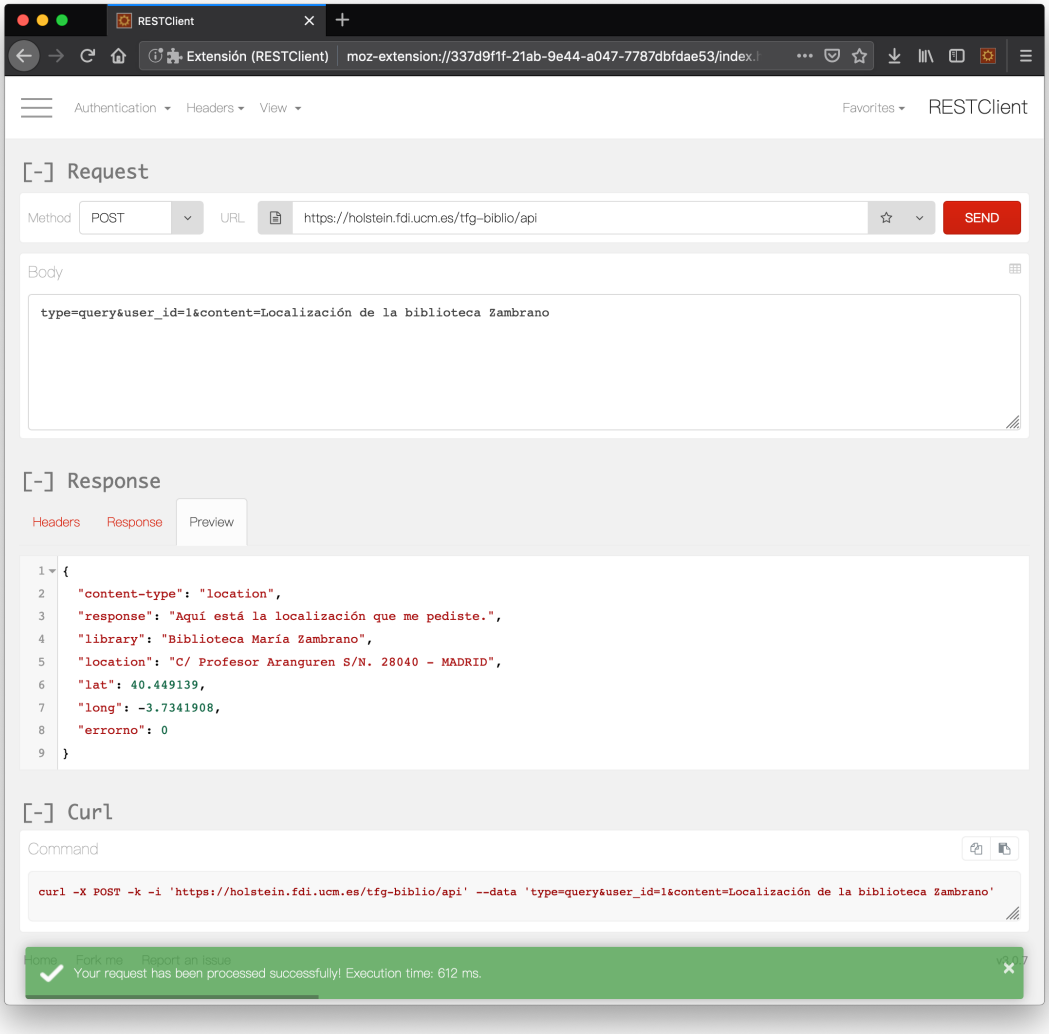


Figura C.4: Ejemplo de consulta para mostrar una localización.

Apéndice D

Historial de versiones

D.1. Versión 0.1 - (16 noviembre 2018)

En esta primera iteración se desarrollaron los diferentes prototipos del proyecto. El desarrollo se hizo en paralelo al análisis y captación de requisitos, de manera que en esta fase se hicieron pruebas de tecnologías a utilizar más adelante.

Durante este sprint se decidieron los lenguajes de programación a utilizar en los clientes y en el servidor, el formato que utilizarían las consultas (JSON), nos permitió tener una primera toma de contacto con un procesador de lenguaje natural (IBM Watson) y se dotó de una primera visión de flujo de uso de la aplicación.

En esta versión existió una primera versión muy inestable de un cliente para la plataforma iOS, capaz de enviar preguntas al servidor y recibir respuestas de solo texto de este. El servidor implementaba exclusivamente de una pasarela de consultas al PLN.

D.2. Versión 0.2 - (3 febrero 2019)

En esta iteración se han implementado numerosa cantidad de funcionalidades. Tras recopilar los análisis y la captación de requisitos realizados durante los meses de noviembre y diciembre de 2018, se estudia a desarrollar historias de usuario (disponibles en el apartado 4.2)

De estas historias de usuario, implementamos las siguientes:

- 01-00-01
- 01-00-02
- 01-01-01

- 01-01-02
- 01-01-03
- 01-02-01

En esta versión se desarrolla una primera versión estable del cliente para iOS (versión 0.2.2 - build 26), capaz de funcionar con las historias de usuario anteriormente descritas. También se crea una primera versión estable del cliente para Android (versión 0.2), igual que el cliente para iOS, este también implementa las funcionalidades para representar las historias de usuario desarrolladas.

La aplicación servidor sufre un cambio de numeración al abandonar el estado "prototipo", asignándole la versión 0.1 de desarrollo. El servidor implementa una arquitectura más rudimentaria que la descrita en el capítulo 5 de este documento . En concreto, la aplicación servidor implementa un back-end que gestiona las llamadas del cliente y envía las peticiones a un controlador muy básico. En el back-end se añade una API de acceso a los clientes oficiales y a cualquier cliente REST con el fin de poder gestionar el flujo de envío y recepción de consultas y/o respuestas. Este controlador, envía las consultas a un módulo de la aplicación que contacta con el PLN IBM Watson . La respuesta obtenida por este modulo, o bien se envía nuevamente al cliente o se envía al módulo encargado de realizar la consulta al catálogo de la biblioteca (OCLC). Dado que esta versión será la última versión en utilizar Watson Assistant como motor de PLN, se descarta desarrollar funcionalidades de gestión del diálogo.

El módulo de conexión al catálogo es implementado por primera vez, con funcionalidad para buscar libros por palabras clave, título o autor en el catálogo general de la biblioteca de la Universidad Complutense. También se añade la posibilidad de consultar los ejemplares disponibles en ese momento.

Durante el desarrollo de este módulo nos encontramos dos limitaciones técnicas:

- La API de OCLC no es capaz de devolver una URL con la portada de los ejemplares buscados.
- La API proporcionada por la universidad es una versión generalizada y no permite filtrar búsqueda por biblioteca de una facultad concreta.

Para la primera limitación la solución aplicada consiste en utilizar la librería lxml de Python, obtener la página web del ejemplar entera y extraer la URL de la imagen desde dicha web. Esta solución es provisional, dado que provoca un importante impacto negativo en el rendimiento del servicio.

Para la segunda limitación contactamos con el personal responsable del catálogo OCLC en la universidad para buscar una posible solución. En esta versión no se soporta dicha funcionalidad y se analizará la viabilidad de incorporarla en versiones posteriores.

D.3. Versión 0.5 - (8 abril 2019)

Este sprint del proyecto inicia la segunda fase del proyecto. En esta versión se culmina con éxito la migración del procesador de lenguaje natural proporcionado por IBM Watson Assistant a uno nuevo de código abierto, Rasa Core + NLU, apoyándose en el PLN spaCy.

Para el desarrollo de esta versión, decidimos mantener por separado los módulos del PLN del resto del sistema para seguir manteniendo la independencia característica de la versión anterior, garantizando la facilidad para modificar los módulos del sistema realizando un mínimo esfuerzo en la adaptación de los demás.

Además del cambio de procesador de lenguaje, también generamos unos nuevos modelos de entrenamiento para este, basándonos en los utilizados en IBM Watson Assistant, y ampliamos las funcionalidades que habían quedado sin implementar en la versión anterior, como el soporte para la serialización de conversaciones para distinguir a un usuario de otro.

Aprovechando la construcción de los nuevos modelos, implementamos todas las historias de usuario del punto anterior, tanto las ya realizadas como las restantes.

En esta versión, no se realizan modificaciones en otros módulos de la aplicación.

D.4. Versión 0.9 - (4 mayo 2019)

Esta versión será la previa a la versión Gold. Los principales cambios realizados han consistido en adaptar todos los módulos al nuevo procesador de lenguaje, incorporando aquellas características nuevas que estos no implementaban en la versión 0.2.

Esta versión implementa un nuevo sistema para la obtención de portadas de libros más eficaz que el utilizado en la versión 0.2. En esta versión trasladamos la carga de trabajo al cliente, que con el uso de funciones asíncronas permite la carga de los ejemplares hasta 2 veces más rápida que en la versión anterior. Dada la imposibilidad de utilizar la solución descrita en la versión 0.2, migramos de servicio para la obtención de portadas a Open Library[1], de la iniciativa Internet Archive[2].

Además, se añade soporte para la consulta de direcciones, números de teléfono y horarios de las diferentes bibliotecas. Al no disponer de un lugar único del que poder acceder a estos datos, se centralizarán en una base de datos propia.

Durante el desarrollo de esta versión nos encontramos dos limitaciones

[1] `https://openlibrary.org/dev/docs/api/covers`
[2] `https://archive.org/`

técnicas:

- No encontramos una alternativa para filtrar la búsqueda de ejemplares por biblioteca de una facultad concreta. Tras contactar con el responsable del catálogo en la UCM, no se dio ninguna solución o alternativa viable. Por tanto, esto queda como un posible trabajo futuro.
- El servidor de la facultad de informática presenta incompatibilidades con Rasa debido a la edad de este, las soluciones adoptadas se explican en el `capítulo 6, sección 6.4` de este documento.

D.5. Versión 1.0 - (12 mayo 2019)

En esta versión aplicamos correcciones de errores menores encontrados en la versión anterior y se establece la licencia de los distintos módulos del proyecto (MIT).

D.6. Versión 1.0.1 - (26 mayo 2019)

En esta versión solucionamos errores menores en la interfaz de usuario del cliente de iOS y Android y se soluciona un error en los módulos del servidor que podía provocar que la aplicación no funcionase correctamente en versiones anteriores a Python 3.6.
Esta es la versión definitiva del proyecto, y la que se adjunta como release en el repositorio.