UNIVERSIDAD DEL PAIS VASCO

Bot para practicar inglés

por

Eduardo Martínez

Memoria de Trabajo de Fin de Grado para el
Grado en Ingeniería Informática de Gestión y Sistemas de Información

en la
Escuela Universitaria de Ingeniería Técnica Industrial

Junio 2018

UNIVERSIDAD DEL PAIS VASCO

Resumen

Escuela Universitaria de Ingeniería Técnica Industrial

Grado en Ingeniería Informática de Gestión y Sistemas de Información

por Eduardo Martínez

En este proyecto se ha desarrollado un bot para la aplicación Telegram cuyo objetivo es ofrecer al usuario la posibilidad de practicar inglés. Para ello, el bot presentará sus opciones en castellano, para que el usuario las tenga que traducir al inglés. Para conseguirlo, hemos utilizado diferentes tecnologías: Los servicios Cognitive Services de Microsoft y speech to text de Google, qué los utilizamos para procesar el audio, y así el usuario pueda practicar su pronunciación del idioma, y los servicios de google translate API para comparar las respuestas. Asimismo, se utilizan diferentes herramientas: twitterscraper y twitter API con las que obtenemos las frases a traducir y una base de datos MySQL para almacenar las frases recogidas de Twitter (las traducciones posibles) y las interacciones del usuario con el bot.

Contenido

Lista de Figuras

Definiciones

Estas son algunas definiciones importantes que hay que tener en cuenta antes de empezar a leer esta documentacion:

- ***Bot***: Es la palabra robot acortada. Se refiere a un tipo de programa informático autónomo que es capaz de llevar a cabo tareas concretas e imitar el comportamiento humano.
- ***Chatbot***: Programas de ordenador que intentan mantener una conversación con el usuario como si fueran seres humanos. Cada programa tiene una "personalidad específica"
- **Telegram**: Es una nueva aplicación de mensajería móvil desarrollada por los fundadores de VKontakte, la red social más popular en Rusia
- ***Microsoft Cognitive Services***: Es una colección de interfaces de programación de aplicaciones inteligentes (API, por sus siglas en inglés) de transferencia de estado representacional (RESTful, por sus siglas en inglés) que permiten a los sistemas ver, oír, hablar, comprender e interpretar las necesidades de las personas mediante métodos de comunicación naturales. Los desarrolladores pueden usar estas API para hacer que sus aplicaciones sean más inteligentes, atractivas y reconocibles.
- ***speechToText***: Esta tecnología es capaz de recibir un audio y transformar ese audio en texto.
- **PHP**: Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.
- **JSON**: Acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de

objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

- **twitterscrapper**: Es una alternativa a la API de Twitter. Con esto se puede obtener un conjunto de tweets de un usuario. Hace solicitudes de busqueda en Twitter (diferente a la busqueda mediante la API).

Chapter 1

Introducción

Hoy en día, la tecnología está muy avanzada en la sociedad, y esta, nos ayuda entre tantas otras cosas, a poder crear máquinas o programas que sean capaces de interactuar con las personas, de una forma humana e inteligente (Inteligencia artificial).

El desarrollo de la inteligencia artificial cada vez está más presente en nuestras vidas y con el paso del tiempo su presencia irá en aumento gracias al impulso que actualmente están aportando las empresas tecnológicas.

Los Asistentes personales inteligentes son un ejemplo de ello.

1.1 Asistentes personales inteligentes

Un asistente personal inteligente es (en el contexto de la tecnología de la información) un software que puede realizar tareas u ofrecer servicios a un usuario. Está basado en los datos de entrada que le da el usuario, y el software tendrá la habilidad de acceder a información en linea (noticias, horarios, precios...).

La variedad de cosas que pueden realizar los Asistentes personales inteligentes es inmensa. Una variante de estos son los chatbots, que esta orientada a realizar una conversación usuario-maquina, en el que el usuario tenga una experiencia natural interactuando con el bot.

1.2 Origen del proyecto

La idea del proyecto esta propuesta por el mismo tutor del proyecto, Juan Antonio Pereira, profesor de la Universidad del País Vasco (UPV/EHU). Juan Antonio ha utilizado bots en muchas de sus áreas de enseñanza, y es por eso que tenia algunas ideas para poder llevar a cabo. Tiene un grupo de investigación, donde han desarrollado diferentes tipos de bots, y es por eso que este proyecto ganaba fuerza, ya que tener la ayuda de alguien que investiga con estos bots es un apoyo mas. En este caso se construirá un bot para ayudar a personas que estén aprendiendo inglés.

1.3 Motivaciones para la elección del proyecto

Este proyecto esta destinado a abarcar un amplio campo de tecnologías. Estas tecnologías se usan para propósitos diferentes, pero utilizando las herramientas adecuadas y haciendo uso de estas se pueden realizar proyectos interesantes. La principal motivación fue como hacer que las diferentes herramientas interactúen entre si para llevar a cabo el proyecto.

Interactuar con el *bot*

Un bot, se trata de hablar con una máquina, y para eso la maquina debe saber que tiene que contestar. Los bots, están muy asociados a la inteligencia artificial, y es algo que me llama mucho la atención, ya que es un área muy importante en cuanto a tecnologías e informática nos referimos.

Google API y twitter API

En la universidad ya habíamos usado anteriormente otro tipo de APIs como Flickr, pero cuando Juanan me explico, al inicio del proyecto, que íbamos a necesitar la ayuda de dos grandes de la tecnología como son Google y Twitter, sin duda me anime, ya que es importante aprender el funcionamiento de estas APIs para un futuro. La API de Twitter la utilizamos para recoger las traducciones posibles, y las respuestas a estas traducciones. Para poder abarcar mas posibles traducciones disponibles, y que la aplicación sea lo mas real posible con respecto a las similitudes que puedan tener las diferentes frases que puedan ofrecer los usuarios.

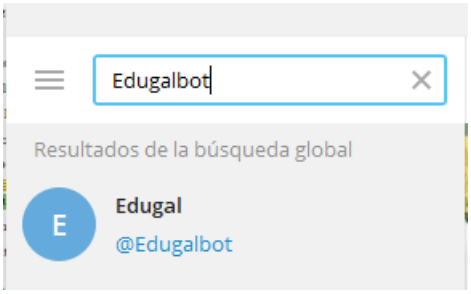

FIGURA 1.1: Esquema de la arquitectura de un bot de Telegram.

1.4 Caso de uso general

El caso de uso mas importante de este proyecto sería el de una persona que quiere aprender inglés, pero no tiene tiempo para ir a clases extra-escolares de ingles, ni mucho menos para ponerse en casa a hacer ejercicios. Un amigo le comenta que hay un *bot* de Telegram que le puede ayudar, y que aunque ya sabe que un *bot* no es como hablar con una persona, en algún rato libre que tenga se puede practicar con el.

Para usarlo por primera vez, buscamos al *bot* por su alias, como podemos ver en la figura 1.1, y lo añadimos a nuestros contactos de Telegram. A continuación pulsamos el botón *inicio* que tendremos en la conversación. Una vez hecho eso ya podemos utilizarlo.

Una vez que ya hemos utilizado nuestro *bot*, podremos hacer diferentes cosas con el. Cada vez que queramos practicar sabemos que con pulsar el comando nos ofrece diferentes posibilidades, en un principio tenemos la opción de elegir entre todas las traducciones posibles, también tenemos la opción de elegir solo las que mas nos cuestan(las que hemos fallado), o una opción de que sea al azar.

Una vez que ya tenemos la traducción que queremos traducir, solo tenemos que escribir, o mandarle una nota de audio al bot, con lo que creemos que es la traducción correcta. Si lo que mas nos cuesta es escribir la traducción podemos probar escribiendo, pero si queremos saber la pronunciación correcta, podemos enviar audio también y el *bot* nos dirá si hemos acertado con la traducción.

Chapter 2

Planteamiento inicial

En este capitulo se procederá a nombrar y explicar los objetivos, las herramientas utilizadas, la arquitectura, el alcance y la planificación del proyecto.

2.1 Objetivos

Debatir y establecer los objetivos es uno de los pasos mas importantes a la hora de planificar un proyecto, ya que los objetivos son los que definen por donde debe ir avanzando el desarrollo de este, y cual es la meta a la que hay que llegar. Aquí se describirán los objetivos que se establecieron al inicio del proyecto.

Objetivos para el desarrollo del *bot* de Telegram

- **Robustez de cara al usuario**: En un bot, lainteracción con el usuario debe de ser natural, es decir, el usuario no debería de tener problemas a la hora de mantener la conversación con el bot, de tal forma que el bot sepa reaccionar adecuadamente en cada estado en el que este se encuentre.

- **Robustez de cara a actualizaciones futuras**: El bot ha de estar preparado para las actualizaciones que puedan darse en un futuro, tales como los cambios en la base de datos, o añadir diferentes tipos de retos que requieran de diferentes herramientas, para así poder ofrecer mas opciones al usuario, como por ejemplo: añadir mas idiomas a traducir o introducir niveles.

- **Manejo sencillo**: El usuario debería de estar cómodo usando el *bot*, y su consulta debería de ser rápida y eficaz. Para esto se requiere un diseño adaptado al usuario, para que tenga que teclear lo menos posible hasta llegar a la traducción que desea.

Objetivos para la documentación y el código

- **Claridad y simpleza**: La documentación y el código del programa deben estar de una forma clara y simple para cuando se lea se entienda perfectamente, y si en un futuro la aplicación se expande, o alguien quiere seguir trabajando en ella, se entienda perfectamente cada paso.

Archivos de configuración para cada parte del proyecto : Algunas herramientas del proyecto requieren de archivos de configuración, o de claves que deben estar fácilmente accesibles en una ruta. Cada área del proyecto tendrá un archivo de configuración.

- **Archivo de configuración de Telegram**: El fichero hook.php, dentro del proyecto, sirve para conectar con un webhook, pero también como archivo de configuración, donde se pueden guardar las credenciales de las APIs, y luego usarlas en los comandos que creamos.
- **Archivo de configuración de Google**: La API de Google de speechToText, que sirve para obtener texto desde un audio, requiere de un archivo de configuración aparte, que nos descargamos de Google, ya que Google tiene diferentes niveles de APIs, y este es uno de los servicios mas costosos.
- **Archivo de configuración de Twitter**: Para hacer el Twitter-script y obtener las preguntas y respuestas de Twitter, hay que usar la API de Twitter, y esta se introduce en un fichero aparte.

2.2 Herramientas utilizadas

En este apartado se definirán las herramientas que se han utilizado a lo largo del proyecto.

PHP

Es el lenguaje principal en el que se ha desarrollado la mayor parte del proyecto. Con este lenguaje se han programado todas las funcionalidades del *bot* de Telegram, incluyendo las comunicaciones con los servicios de *Microsoft Cognitive Services* y de *speechToText* y las llamadas a Twitter con *Twitter-script*.

PHP longman Telegram-bot

Esta biblioteca es un bot de Telegram que se puede extender y añadirle funcionalidades. Contiene todas las funcionalidades básicas y los comandos básicos para empezar a programar un bot. Es una de las principales bibliotecas que se usan para programar bot de Telegram.

Composer

Este paquete es un administrador de dependencias para PHP que nos permite descargar paquetes desde un repositorio para agregarlo a nuestro proyecto. Composer se puede encargar de actualizar las dependencias de las librerías que hayamos descargado por una nueva versión.

API de *Microsoft Cognitive Services*

Microsoft ofrece un potente paquete de servicios en relación a la interacción usuario-máquina. Este proyecto utiliza el servicio *Microsoft Cognitive Services*, para convertir archivos de audio a texto.

API de *Google* (GoogleTranslate y SpeechToText)

Google ofrece una potente API con infinidad de servicios. En este proyecto se usan dos servicios de la API de Google: GoogleTranslate, y SpeechToText. El primero para traducir frases de un idioma a otro, y el segundo para convertir audio en texto escrito.

FFmpeg

FFmpeg es una herramienta que puede grabar, editar, convertir, decodificar, codificar, etc, todos los formatos de audio y vídeo. Está desarrollado en GNU/Linux, pero puede ser compilado en la mayoría de los sistemas operativos, incluyendo Windows

CURL CURL es un proyecto de software consistente en una biblioteca (libcurl) y un intérprete de comandos (CURL) orientado a la transferencia de archivos. Se podría decir que es una herramienta que permite la comunicación con servidores externos, y así conectándonos a estos servidores obtenemos los datos, o subimos los datos que queremos. Soporta varios tipos de protocolos

MySQL workbench

MySQL Workbench es una herramienta visual de diseño de bases de datos que integra desarrollo de software, Administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

MySQL

El sistema de gestión de bases de datos que se usa. Este sistema es el que nos permite crear tablas y gestionar los datos guardados. El paquete php longman Telegram bot ofrece unas tablas por defecto, pero para la creación del proyecto se necesitó definir una serie de nuevas tablas para añadir una capa de persistencia a la lógica de negocio del bot.

GIT

Es un software de control de versiones, que esta pensado en la eficiencia y la fiabilidad del mantenimiento de las diferentes versiones del proyecto. Su propósito es registrar todos los cambios que se hacen en los archivos del proyecto, y en coordinar el trabajo entre mas de una persona.

Bitbucket

Bitbucket es un servicio de alojamiento para los proyectos que utilizan el sistema de control de versiones GIT. Ofrece servicio en la nube para gestionar el control de versiones de nuestros proyectos, tanto públicos como privados

Sharelatex

Sharelatex es una herramienta en linea que se utiliza para redactar la documentación de proyectos en linea. Pagina web: https://www.sharelatex.com

Cacoo

Cacoo es una herramienta que utiliza un sistema de guardado en la nube, y que se utiliza para hacer figuras, diagramas, esquemas de todo tipo de formas que luego se introducen en la documentación del proyecto.

2.3 Arquitectura

Aqui se procederá a mostrar en lo que a la arquitectura del bot se refiere, y como funciona.

Bot de Telegram

Para dar de alta un *bot* de Telegram, la plataforma de mensajería nos ofrece un *botFather*, algo asi como el padre de los bots, y este nos da la oportunidad de además de otras cosas, registrar un nuevo bot y administrarlo.

Una vez que ya tenemos el *bot* dado de alta, tenemos que ponerlo en funcionamiento, es decir, obtener y procesar los datos que el usuario envía a Telegram través del *bot*, y para ello lo básico que necesitaremos sera un servidor Apache2, y una de las bibliotecas disponibles que existen para el desarrollo de bots. En este proyecto sutilizará'a un servidor con el paquete de desarrollo de *bots* PHP *longman Telegram-bot* que estará programado en PHP.

Una vez que tenemos el paquete instalado, podremos empezar a pensar en como obtener los datos que el usuario envía al *bot*. Para esto tenemos dos opciones: *polling* y *webhook*, el primero consiste en consultar periódicamente Telegram en busca de nuevos mensajes y no requiere disponer de un servidor externo, es decir, hay que hacer consultas constantemente para ver si el usuario ha enviado alguna petición. El segundo, consiste en que Telegram nos avise cuando haya nuevos mensajes del usuario y por tanto requiere una URL pública en un servidor, en este caso, sera Telegram quien nos avise cuando hay mensajes nuevos. Dependiendo del tipo de *bot* que se quiera, habría que tener una base de datos MySQL y/o un certificado SSL, este certificado es obligatorio para la opción de desarrollo de bots con webhook.

En nuestro *bot* el manejo de las peticiones al servidor shará'a mediante un *webhook*También shará'a uso de una base de datos MySQL para guardar la información relativa a los chats (en lo que al *bot* se refiere). Esta base de datos, la obtendremos del paquete que instalemos para desarrollar el bot.

En la figura 2.1 se detalla un esquema del funcionamiento del *bot* de Telegram

Cuando un usuario le manda un mensaje al *bot*, este se transfiere a los servidores de Telegram, y cuando estos detectan que el *bot* esta asociado a una *webhook URL*, manda el mensaje recibido al servidor, y este procesa ese mensaje para devolver la respuesta deseada.

Los mensajes son enviados mediante el protocolo HTTPS que asegura su encriptación, y aporta un canal apropiado para este tipo de información. Asimismo, Telegram utiliza su propio protocolo para la encriptación interna de los mensajes, el protocolo MTProto.

Una vez que el servidor ha recibido la petición de Telegram, el paquete *longman-Telegram-bot* será el que se encargue de la gestión de dicha petición. Por lo tanto, dependiendo del comando que se haya utilizado para envíar la respuesta, y dependiendo de que tipo de respuesta se haya enviado, el paquete *longman-Telegram-bot* tratara esa respuesta de una forma u otra.

Los servicios *Google cloud services (speechToText)* y *Microsoft Cognitive Services* solo se utilizan en caso de que el bot reciba una nota de audio, y su propósito es el de transformar ese audio en texto. Cuando el *bot* recibe el audio, lo recibe en formato *.ogg*, y para que los servicios que hemos mencionado antes puedan procesar el audio, debe de estar en formato *.wav*, por lo que para eso, tendremos que hacer una llamada FFmpeg y cambiarle el formato, y así sea compatible con estos servicios.

Finalmente, cuando la biblioteca termina de procesar el mensaje, genera una respuesta que devuelve al servidor de Telegram y éste a su vez, envía al usuario que dio comienzo a todo el proceso.

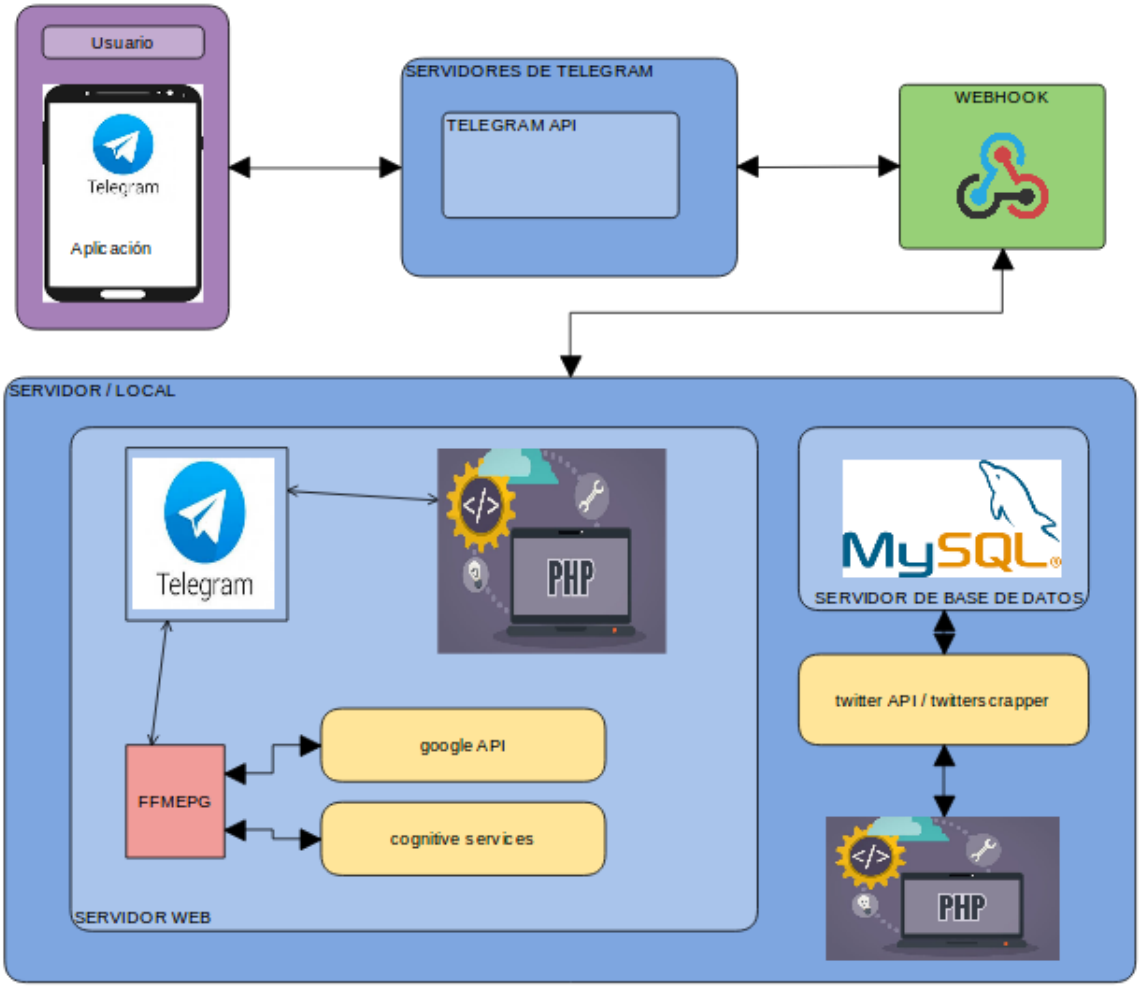

FIGURA 2.1: Esquema de la arquitectura de un bot de Telegram y twitter-script.

Twitterscrapper / Twitter API

El funcionamiento del comando */retos* requiere de una base de datos con información actualizada de las frases a traducir. Esto quiere decir que necesitamos idear un sistema que permitiera la extracción de frases en un idioma (castellano) y sus posibles traducciones a otro (inglés). Analizando posibles soluciones se optó por explotar el API oficial de Twitter y recabar dichas frases a partir de los retos públicos que propone periódicamente un profesor de inglés (@imkylemillar) en Twitter

En un principio, se planteo hacer uso de *la API de Twitter* para poder empezar a obtener las preguntas, pero durante el desarrollo, surgieron algunas complicaciones, y nos dimos cuenta de que solo nos devolvía los tweets de los últimos 7 días, en caso de querer tweets mas antiguos habría que pagar una cuota. Debido a las complicaciones y tras analizar

varias alternativas que evitaran dicho pago, se optó por el uso del paquete Twitterscrapper. Este paquete, es un simple script que permite obtener unos determinados tweets mediante técnicas de web scrapping, y de esta forma, pudimos obtener mas frases y sus respectivas respuestas correctas, además de las respuestas que los usuarios de Twitter enviaban a dicho tweet.

Finalmente, tras completar la extracción de todos los tweets requeridos de la cuenta Twitter indicada usaremos la *Twitter API* para descargarnos los tweets recientes, y programaremos el script *get-tweets.php* para que se ejecute 1 vez a la semana, y obtenga los tweets y las respuestas de toda la semana, así el usuario tendrá semanalmente nuevas traducciones disponibles.

En la figura 2.1 podemos observar a la derecha como mediante PHP obtenemos los tweets y los introducimos en una base de datos.

2.4 Alcance

En este apartado expondremos todo el trabajo necesario para realizar el proyecto, y para eso se han fijado los distintos requerimientos para llegar al objetivo.

Las tareas se han ido realizando a lo largo de un periodo de tiempo, y no siguen un orden en concreto, por lo que hemos ido planteándolas dependiendo de las necesidades que teníamos que cubrir.

2.4.1 Aprendizaje

A lo largo del desarrollo del proyecto se han utilizado diferentes herramientas, de las cuales algunas no habíamos utilizado, por lo que se requiere de su aprendizaje y tener una base de conocimiento antes de empezar a realizar la tarea que queramos desarrollar.

A continuación se muestra un diagrama EDT con los puntos que componen la tarea del aprendizaje en este proyecto.

Junto al diagrama de la figura 2.2, estas tablas desglosan cada apartado del aprendizaje.

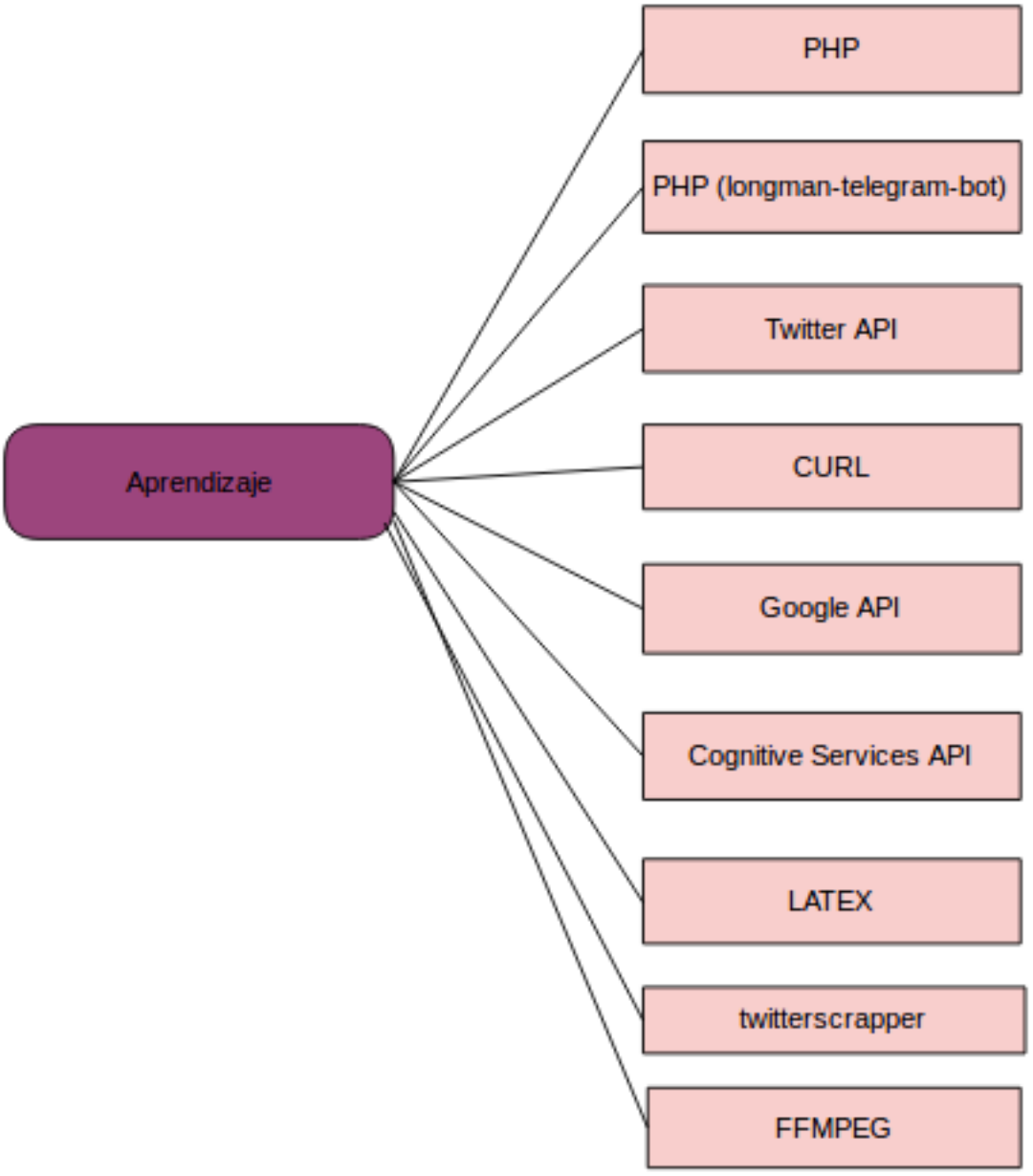

FIGURA 2.2: Diagrama EDT de la tarea del aprendizaje.

Aprender el lenguaje de programación *PHP*	
Descripción	Aprender a realizar tareas simples en PHP, para tener un conocimiento de la base del lenguaje
Recursos necesarios	PHP ,Navegador Web.
Duración estimada	30 horas

Aprendizaje del paquete PHP longman Telegram-bot	
Descripción	Aprender a utilizar los métodos que este nos ofrece, para poder desarrollar los comandos
Recursos necesarios	PHP, biblioteca *longman Telegram-bot*, Navegador Web.
Duración estimada	20 horas

Aprender a hacer consultas usando cURL	
Descripción	Aprender a realizar peticiones HTTP utilizando la biblioteca cURL integrada con PHP.
Recursos necesarios	Navegador Web, PHP.
Duración estimada	10 horas

Aprendizaje de la *API de Twitter*	
Descripción	Adquirir conocimientos de el funcionamiento de la API de Twitter.
Recursos necesarios	Navegador Web, php, MySQL
Duración estimada	10 horas

Aprendizaje del uso de Twitterscrapper	
Descripción	Aprender a hacer llamadas a Twitterscrapper.
Recursos necesarios	Navegador Web, php
Duración estimada	2 horas

Aprendizaje del uso de la biblioteca FFmpeg	
Descripción	Aprender a utilizar las funciones del paquete FFmpeg, con el que convertiremos los ficheros a otro formato y los haremos compatibles a la hora de ejecutar el comando */retos*, cuando la respuesta que ofrecemos es mediante un audio.
Recursos necesarios	Navegador Web, PHP, FFmpeg
Duración estimada	2 horas

Aprendizaje del lenguaje LaTeX	
Descripción	Aprender a crear documentos bien estructurados usando el lenguaje LaTeX.
Recursos necesarios	Navegador Web y cuenta en Sharelatex para hacer pruebas.
Duración estimada	15 horas

Aprendizaje de el servicio de Google API	
Descripción	Estudiar como se utiliza el servicio de Google que ofrece con su API, tanto el de Translate como el de speechToText
Recursos necesarios	Navegador Web, cuenta de Google
Duración estimada	5 horas

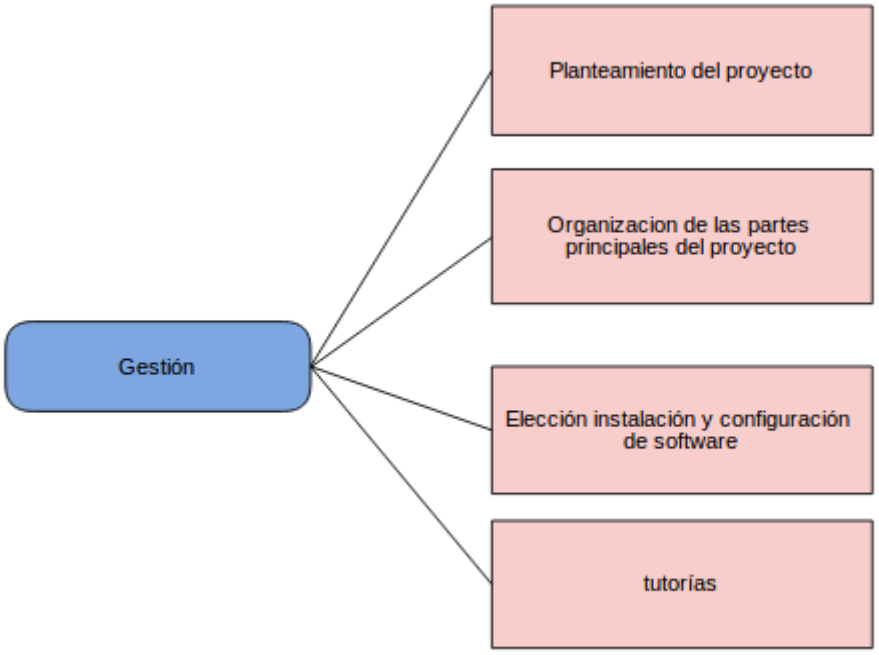

FIGURA 2.3: Diagrama EDT de la tarea de la organización.

2.4.2 gestión

La gestión del proyecto tiene un rol muy importante a la hora de organizarlo. Se trata de la secuenciación de las diferentes partes del proyecto, para que estas realicen sus tareas de forma eficaz.

El diagrama EDT de la figura 2.3 muestra los puntos que componen la tarea de la organización.

A continuación mostramos las tablas que desglosan los apartados de la gestión.

Planteamiento del proyecto	
Descripción	Planear como se podría llevar a cabo el proyecto, basándose en los conocimientos necesarios, y las diferentes tecnologías
Recursos necesarios	Navegador Web, papel y lápiz.
Duración estimada	7 horas

organización de las partes principales y tareas del proyecto	
Descripción	Organizar las diferentes partes que tenemos que tener en cuenta a la hora de desarrollar el proyecto, ya que cada parte o componente del proyecto es importante.
Recursos necesarios	Papel y lápiz.
Duración estimada	10 horas

Elección, instalación y configuración del *software*	
Descripción	Elegir el software apropiado para el desarrollo del proyecto. Instalar y configurar dicho software.
Duración estimada	10 horas

Tutorías	
Descripción	Reuniones con el tutor para esclarecer diferentes problemas que puedan surgir, y para fijar las bases del proyecto.
Recursos necesarios	Papel y lápiz.
Duración estimada	20 horas

2.4.3 Captura de requisitos

Este apartado esta dirigido a los requisitos que hay que cumplir durante el desarrollo del proyecto, utilizando las herramientas necesarias.

El diagrama EDT muestra los puntos que componen la tarea de la captura de requisitos.

A continuación, las tablas que desglosan los apartados de la captura de requisitos.

Casos de uso	
Descripción	Buscar los casos de uso para los diferentes procesos de la aplicación
Recursos necesarios	Papel y lápiz, powerpoint
Duración estimada	3 horas

diseño de la conversación	
Descripción	Hacer un boceto de como fluirá la conversación con el usuario en el comando /retos.
Recursos necesarios	Papel y lápiz, PowerPoint
Duración estimada	10 horas

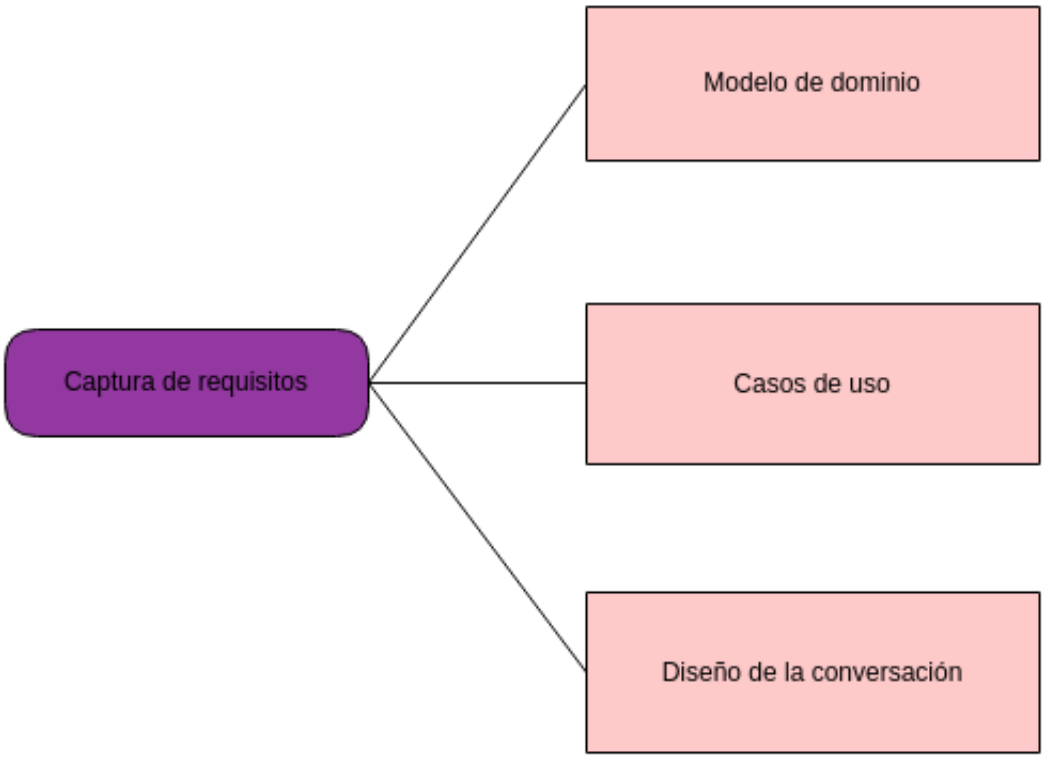

FIGURA 2.4: Diagrama EDT de captura de requisitos.

Modelo de dominio	
Descripción	Diseñar el modelo de dominio del proyecto.
Recursos necesarios	Navegador Web, papel y lápiz.
Duración estimada	5 horas

2.4.4 análisis y diseño

A continuación se muestran los distintos diagramas que se requieren a la hora de analizar y diseñar un proyecto

El diagrama EDT muestra los puntos que componen la tarea del análisis y diseño.

A continuación, las tablas que desglosan el apartado.

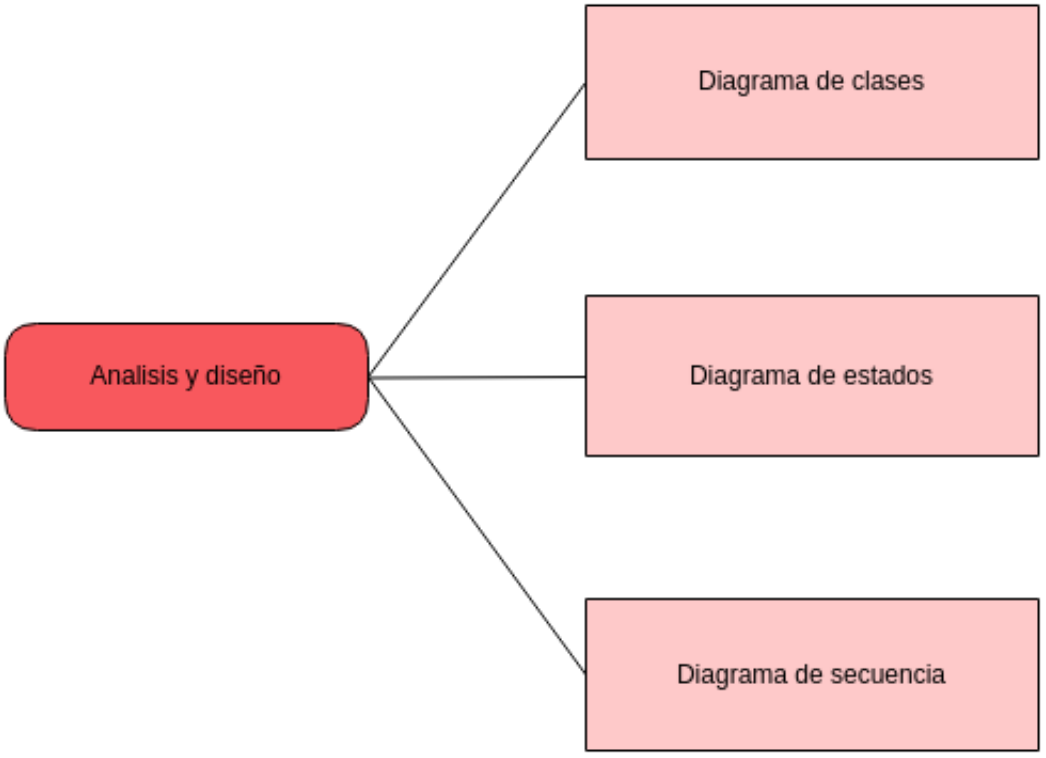

FIGURA 2.5: Diagrama EDT de la tarea del análisis y diseño.

Diagrama de estados	
Descripción	Diseñar y construir el diagrama de estados del proyecto.
Recursos necesarios	Papel y lápiz, Cacoo.
Duración estimada	5 horas

Diagrama de clases	
Descripción	Diseñar y construir el diagrama de clases del proyecto..
Recursos necesarios	Papel y lápiz, Cacoo.
Duración estimada	5 horas

Diagramas de secuencia	
Descripción	Diseñar y definir la secuencia de ejecución del bot.
Recursos necesarios	Papel y lápiz, Cacoo.
Duración estimada	8 horas

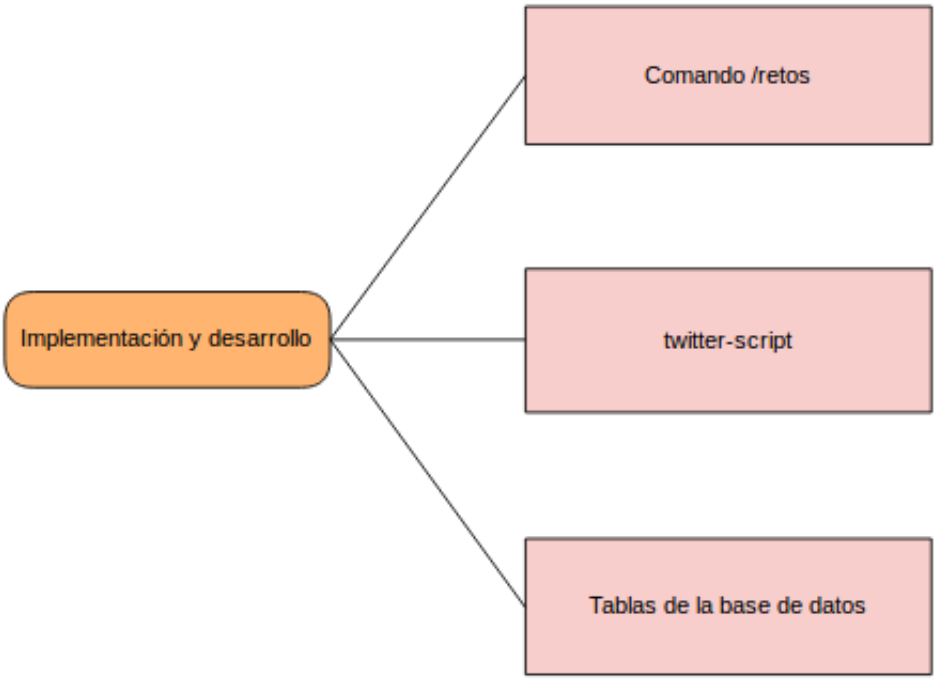

FIGURA 2.6: Diagrama EDT de la tarea de la implementación y desarrollo.

2.4.5 implementación y desarrollo

Este apartado contiene las tablas de la implementación y desarrollo del proyecto

El diagrama EDT muestra los puntos que componen la tarea de la implementación y desarrollo.

Se han dividido las tablas en 2 sub-apartados: la implementación y desarrollo del *bot* de Telegram y la implementación y desarrollo del *Twitter-script*.

implementación y desarrollo del *bot* de Telegram

A continuación, las tablas que desglosan el desarrollo e implementación del *bot* de Telegram.

Comando */retos*	
Descripción	Implementar y desarrollar el comando */retos* que utiliza muchas de las tecnologías utilizadas en el proyecto.
Recursos necesarios	PHP, biblioteca *longman Telegram-bot*, base de datos con información útil, cuentas en las páginas de los proveedores de los servicios de *Microsoft Cognitive Services* y *Twitter API*, *Google API*, FFmpeg, MySQL.
Duración estimada	100 horas

implementación y desarrollo del *Twitter-script*

A continuación, las tablas que desglosan el desarrollo e implementación del *Twitter script*.

Tablas personalizadas en la base de datos	
Descripción	implementación de las tablas personalizadas de la base de datos.
Recursos necesarios	MySQL, mysql-workbench.
Duración estimada	5 horas

Twitter-script	
Descripción	implementación y desarrollo del *Twitter script*, verificando que los retos que obtenemos no se repiten, ni se quedan sin solución.
Recursos necesarios	php, Twitterscrapper, MySQL.
Duración estimada	50 horas

2.4.6 Test

En este apartado se muestran los detalles de las pruebas realizadas durante el desarrollo de las distintas partes del proyecto, para verificar que funcione correctamente.

El diagrama EDT muestra los puntos que componen la tarea del desarrollo de test.

A continuación, las tablas que desglosan los apartados de la fase de testeo.

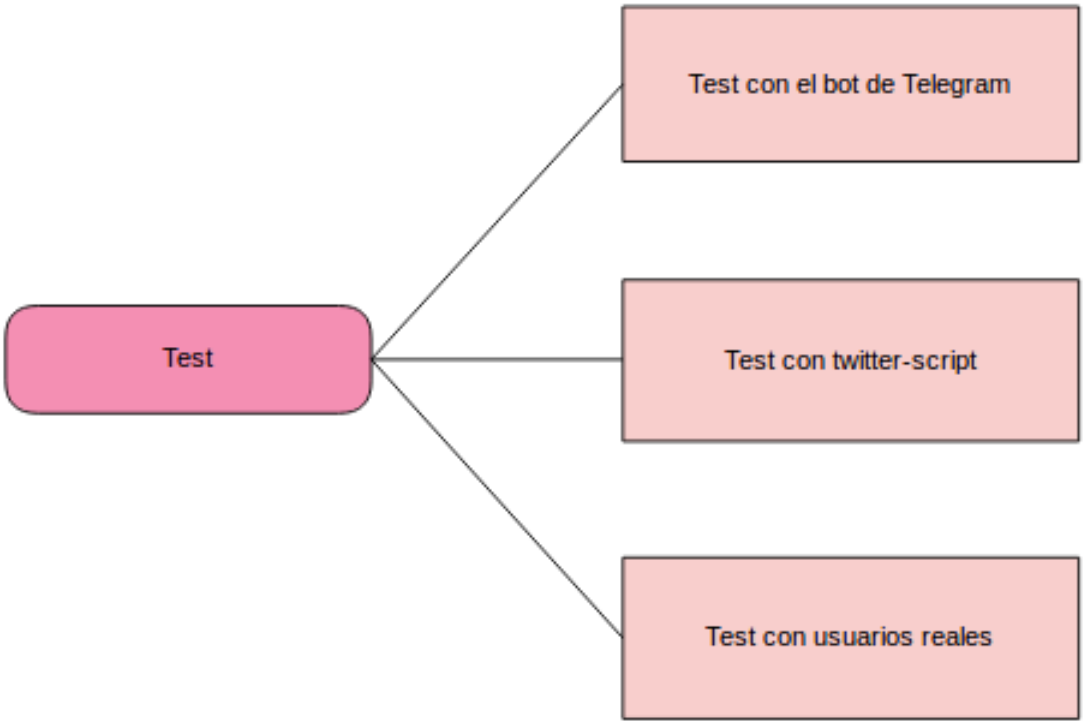

FIGURA 2.7: Diagrama EDT de la tarea del test.

Test con el *bot* de Telegram	
Descripción	Test realizados al *bot* de Telegram, antes de empezar a desarrollarlo, para probar las conexiones, y ver que nos podíamos comunicar correctamente, y durante el desarrollo para asegurar el funcionamiento correcto y detectar posibles errores de cualquier tipo.
Recursos necesarios	aplicación de Telegram de escritorio y de *Smartphone*, PHP, biblioteca *longman Telegram-bot*, base de datos con información útil, cuentas en las páginas de los proveedores de los servicios de *Microsoft Cognitive Services* y *Google Cloud Services*, FFmpeg, MySQL.
Duración estimada	20 horas

Test con el script de Twitter	
Descripción	Probar las llamadas a Twitter, y verificar que obtenemos lo que queremos, y que se guarda en la base de datos correctamente.
Recursos necesarios	PHP, MySQL(base de datos con información útil), *Twitter-scrapper*, cuentas en las páginas de los proveedores de los servicios de *Twitter API*, FFmpeg, .
Duración estimada	10 horas

Test con usuarios reales	
Descripción	Preguntar a los usuarios reales de Telegram acerca del funcionamiento del bot
Recursos necesarios	Usuarios externos.
Duración estimada	2 horas

2.4.7 documentación

Aquí se muestran las tareas de documentación del proyecto.

El diagrama EDT muestra los puntos que componen la tarea de la documentación.

A continuación, las tablas que desglosan los apartados de la fase de documentación.

documentación	
Descripción	creación de las figuras, tablas y bibliografía utilizadas en la memoria final del proyecto.
Recursos necesarios	Navegador web, editor de LaTeX, Cacoo.
Duración estimada	10 horas

Bibliografía	
Descripción	Búscar referencias de todo lo utilizado en el desarrollo a la hora de redactar la memoria final del proyecto.
Recursos necesarios	Navegador web, editor de LaTeX.
Duración estimada	5 horas

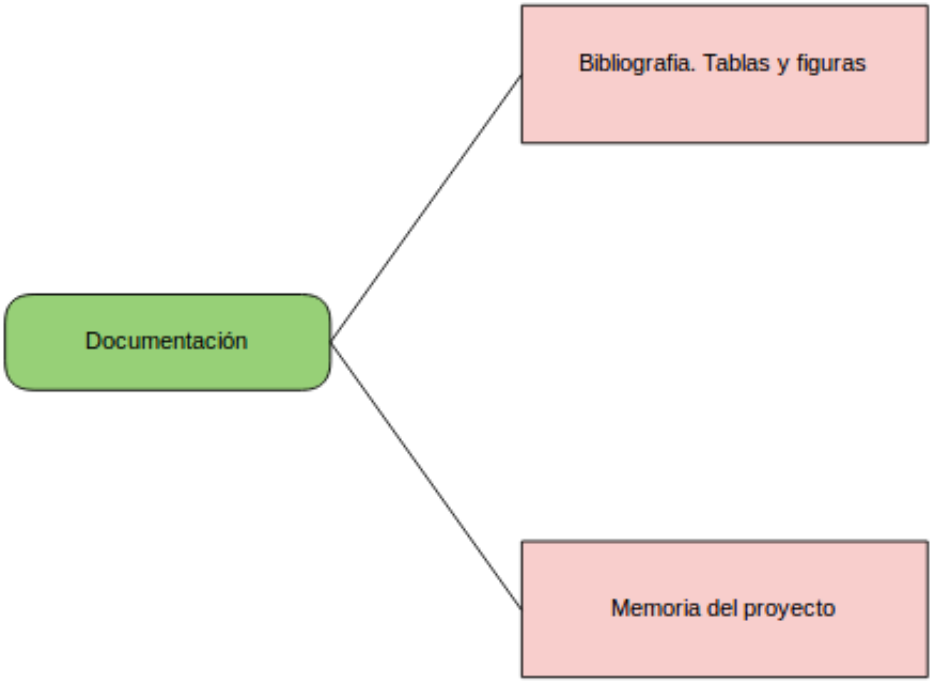

FIGURA 2.8: Diagrama EDT de la tarea de la documentación.

Memoria final de proyecto	
Duración estimada	32 horas
Descripción	Redactar la memoria final del proyecto. Explicar y organizar las ideas y los pasos realizados en un documento.
Recursos necesarios	Navegador web, editor de LaTeX.

2.4.8 Despliegue

Aquí de desglosan las tareas de despliegue del proyecto.

El diagrama EDT muestra los puntos que componen la tarea del despliegue.

A continuación, las tablas que desglosan los apartados de la fase de despliegue.

configuración del servidor de producción	
Duración estimada	5 horas
Descripción	Instalación y configuración del *software* utilizado en el servidor de producción.
Recursos necesarios	Conexión a *bots.ikasten.io* (servidor).

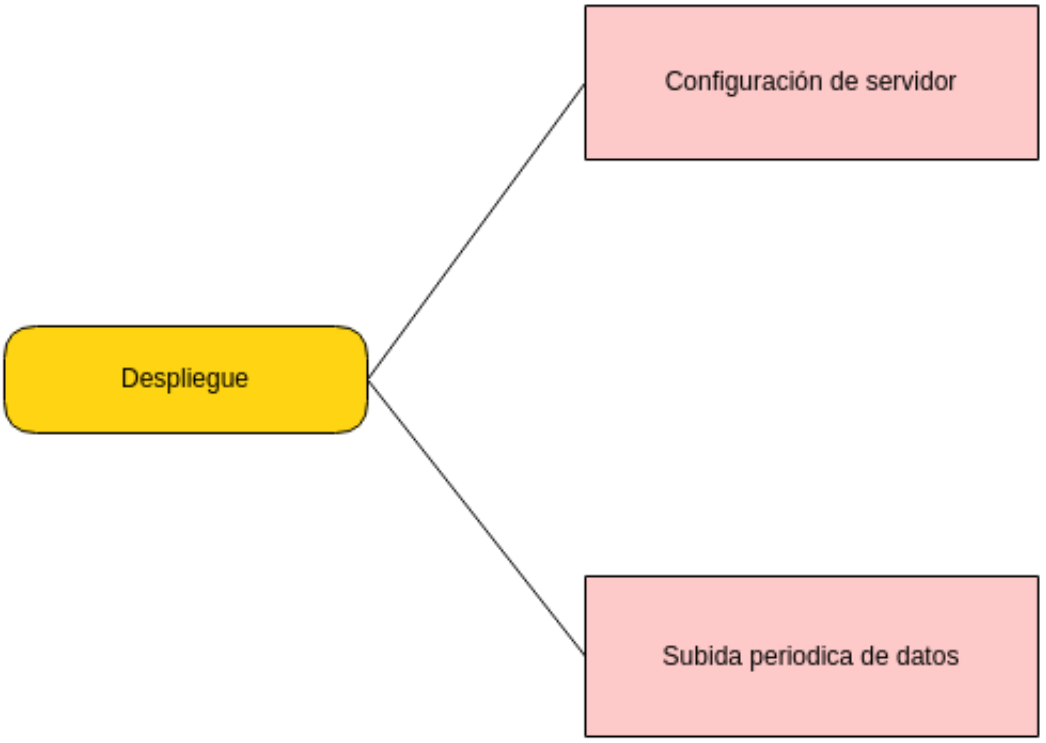

FIGURA 2.9: Diagrama EDT de la tarea del despliegue.

Subida periódica de datos	
Duración estimada	2 horas
Descripción	Subida periódica del código y base de datos al servidor de producción.
Recursos necesarios	Conexión al servidor *bots.ikasten.io*.

2.5 Planificación temporal

En la figura 2.10 se se ha hecho un diagrama de barras laterales, y este muestra las duraciones, tanto la duración que se creía que iba a durar la tarea y la duración final.

Una vez que se tiene una visión general de las tareas, se puede ver que en todas las tareas se tardó mas que lo que se estimó, y procederemos a hacer un diagrama de Gantt, como podemos ver en las figuras 2.11 y 2.12. Este contempla todas las subtareas que se han realizado a lo largo del proyecto, y cual ha sido el tiempo real que se ha tardado en hacer cada una, así como entre que fechas se realizaron

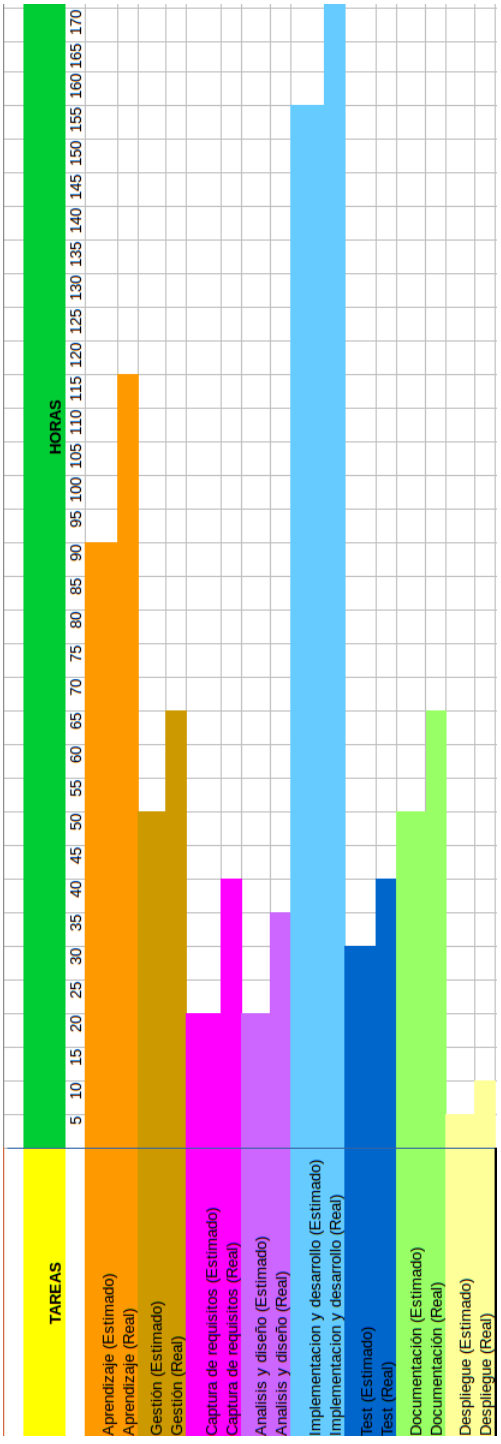

FIGURA 2.10: Gráfico que muestra la comparativa entre el tiempo a invertir calculado para cada tarea del proyecto en la planificación temporal y el tiempo real invertido.

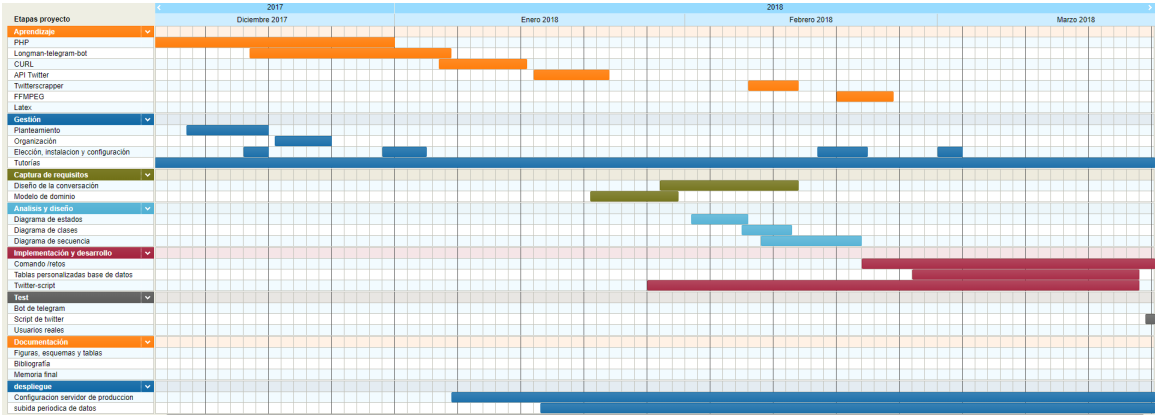

FIGURA 2.11: Gráfico que muestra el tiempo real invertido, en los meses diciembre, enero, febrero y marzo.

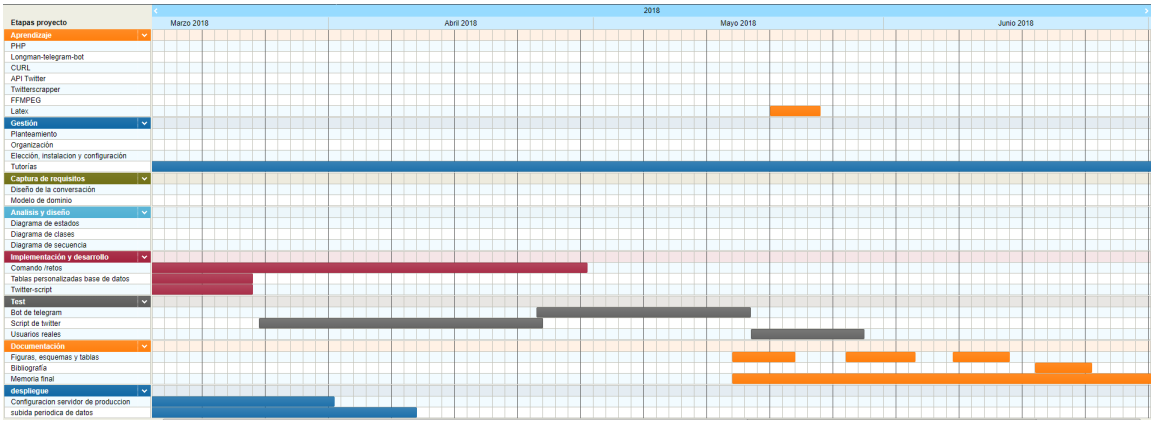

FIGURA 2.12: Gráfico que muestra el tiempo real invertido, en los meses abril, mayo y junio.

Chapter 3

Captura de requisitos

El bot tiene que estar diseñado de forma que satisfazca las necesidades del usuario. Para ello, se deben cumplir unos requisitos, tanto en el *bot* de Telegram como en la recogida de datos de *Twitter*, de tal forma que se le facilite la tarea al usuario.

3.1 Jerarquía de actores

En este proyecto se pueden identificar dos tipos de actores: el usuario del bot, y el cron de el twitter-script. El primero es el que interactúa directamente con el bot, y el segundo es un script programado para que interactúe con Twitter y obtenga las diferentes preguntas y respuestas.

3.2 Casos de uso

Los casos de uso de la figura 3.2 muestran las distintas acciones que puede realizar un usuario. Se distinguirán dos comandos: el comando */retos* y el comando */cancel*.

- ***/retos*:** Cuando el usuario ejecute este comando, se iniciara una conversación con el bot. Este, tratara de reconocer lo que el usuario quiere hacer, y que retos/preguntas le gustaría resolver. En primera instancia, el bot pedirá al usuario si quiere ver que opciones (comandos) tiene, o si quiere empezar directamente con las pruebas. Cuando el usuario empieza con los retos, se le proporcionaran una serie de

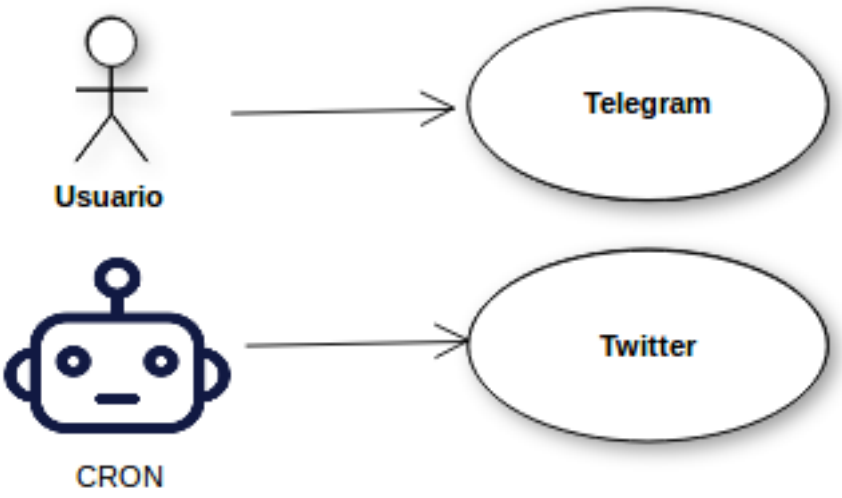

FIGURA 3.1: Jerarquía de actores.

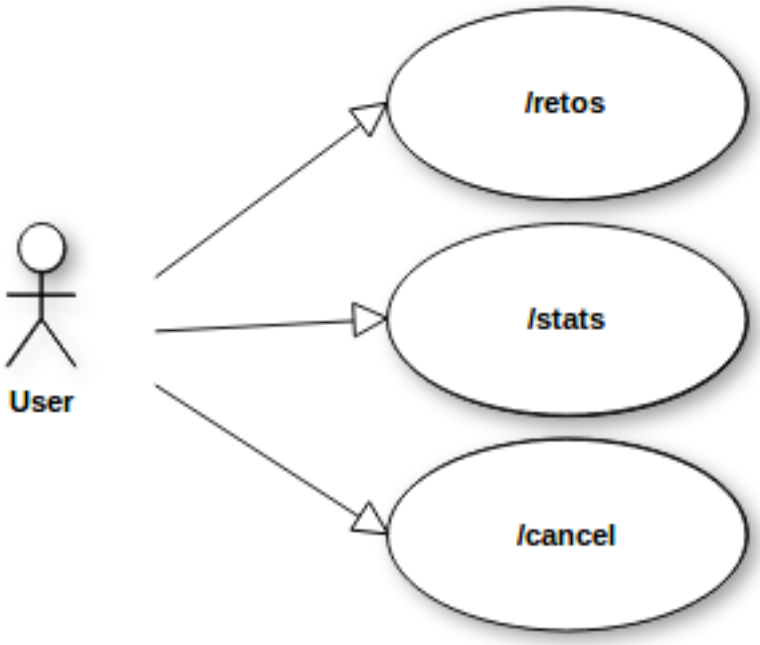

FIGURA 3.2: Casos de uso.

preguntas, incluyendo botones para no tener que teclear mucho. Una vez que el bot sabe que preguntas/retos proporcionarle al usuario, este ira resolviéndolos pudiendo enviar una nota de voz, y así poder utilizar un lenguaje natural en vez de estar escribiendo. Esto también proporciona que el usuario pueda practicar con la pronunciación.

- ***/stats*:** Este comando lo usará el usuario únicamente si quiere ver sus estadísticas con respecto a todas los retos que ha realizado: numero de preguntas correctas, numero de preguntas incorrectas y porcentaje de acierto.

- ***/cancel*:** La ejecución de este comando implica la cancelación de cualquier otro comando que pudiera estar en curso. Aunque el comando */ask* va a tener pocos estados en la mayoría de los casos, la existencia de una comando para cancelar toda actividad es siempre útil.

3.3 Diseño del *twitterscraper / twitter API*

El diseño de Twitterscraper es un poco sencillo, ya que uńicamente lo llamamos una vez, para recoger los tweets mas antiguos.

En lo que a TwitterAPI se refiere, se ha diseñado de forma que la recogida de tweets sea semanal, de tal forma que aprovechémos el servicio de TwitterAPI una única vez a la semana, ya que este nos devuelve los tweets de los últimos 7 días, y así obtendremos todos los tweets haciendo las menores llamadas posibles.

En la figura 3.3, podemos observar como el cron programado en en servidor local se comunica con Twitter mediante la API o la libreria twitterscrapper, y estos guardan la informacion que devueven en una base de datos MySQL.

Objetivos para el *twitter-script*

- **script robusto**: El twitter-script deberá de ser robusto con respecto a cambios en la API, o con respecto a pequeños cambios en la base de datos, es decir, los problemas que puedan surgir en el futuro, deberían de ser fácilmente solucionables.

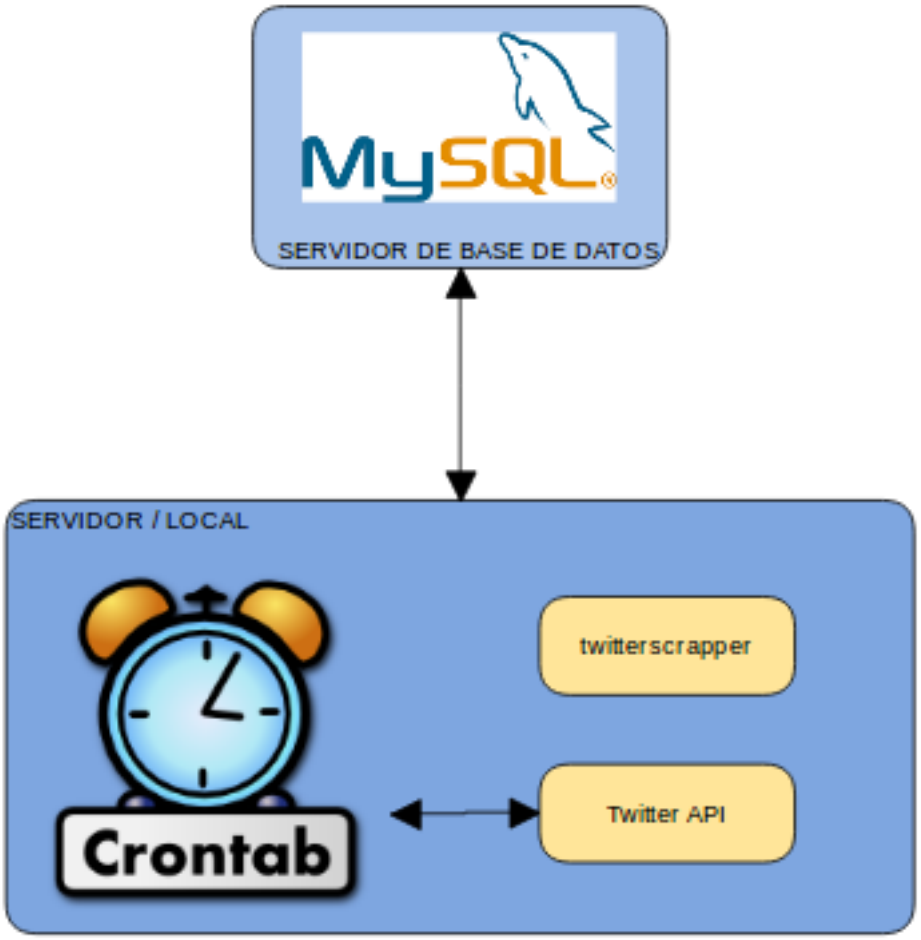

FIGURA 3.3: Diseño del twitter-script

3.4 Diseño de los resultados

Cuando el usuario este contestando las preguntas que se le plantean, una vez que ya sabe si la ha respondido bien o mal, este le planteara la opción de ver otras respuestas disponibles, si es que la ha respondido correctamente, y la opción de revelarle todas las respuestas si no ha contestado bien. Asimismo, cuando use las notas de voz para enviar audio, si la traducción no ha sido correcta, el bot le enviara lo que el le ha entendido, así el usuario sabrá si su pronunciación no ha sido la correcta. El usuario tendrá un comando (/stats), que le ofrecerá la opción de ver su porcentaje de aciertos, así como el numero de aciertos y el numero de fallos.

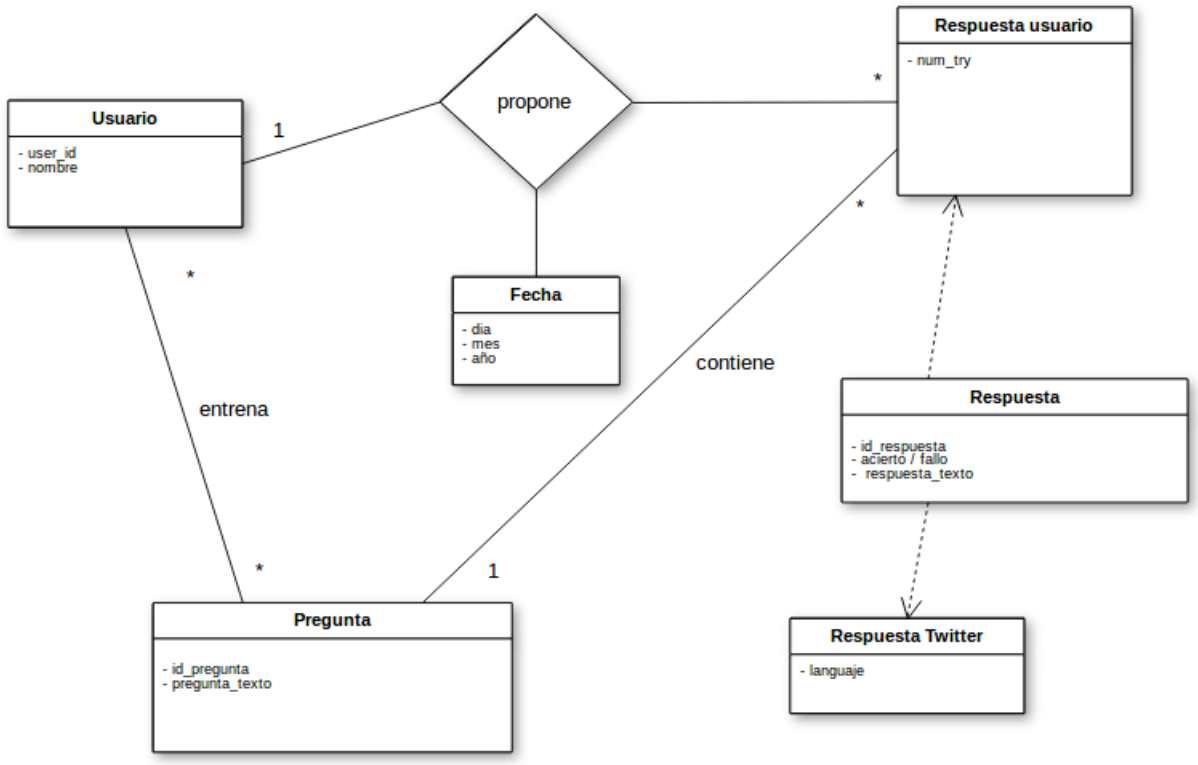

FIGURA 3.4: Modelo de dominio

3.5 Modelo de dominio

El modelo de dominio sirve para representar conceptualmente y mostrar como se relacionan las diferentes entidades del proyecto.

A continuación se describirán las diferentes entidades que se han encontrado para este proyecto.

Usuario

Esta entidad modela al usuario o cliente del bot de Telegram. Este entrena con preguntas y propone respuestas, contiene los atributos donde se contemplan los detalles generales del usuario de Telegram: nombre y apellidos, identificador de usuario, numero de teléfono etc.

Pregunta

Todas las preguntas que el bot plantea al usuario se almacenan en una base de datos. Contienen: identificador de pregunta, el texto de la pregunta.

Chapter 4

Análisis de antecedentes

Aprender y practicar inglés, nunca ha sido tarea fácil, y mucho menos si nos tenemos que mover de casa e ir a clases etc. Hoy en día, tenemos constancia de que existen numerosas aplicaciones para este tipo de propósitos que son de gran ayuda. Duolingo o Babbel son un ejemplo de ellas.

Duolingo, es una aplicación bastante completa que sirve para aprender mas de un idioma, y que contiene una interfaz bastante intuitiva. Son diferentes ejercicios que funcionan como un juego, ya que cada vez que fallas alguna pregunta pierdes vida. Nuestro bot, contiene estas características en mayor o menor medida, ya que ofrece una interfaz intuitiva, y se utiliza para responder preguntas, en nuestro caso traducciones. Además de ello, Duolingo no ofrece a los usuarios, tanto en la aplicación web como en los bots, la opción de responder las preguntas mediante voz en vez de texto. Edugalbot en cambio, si contempla esta opción.

Babbel, es otra aplicación parecida a Duolingo, pero basada en vocabulario, es decir, traducir palabras y asociarlas a grupos de palabras etc.

Estos ejemplos, son aplicaciones que hay que instalar en un dispositivo. Nuestro bot, ofrece la opción a los usuarios de Telegram de poder practicar ingles sin la necesidad de instalar ninguna aplicación de terceros, con mayor comodidad para el usuario.

FIGURA 4.1: Duolingo bots

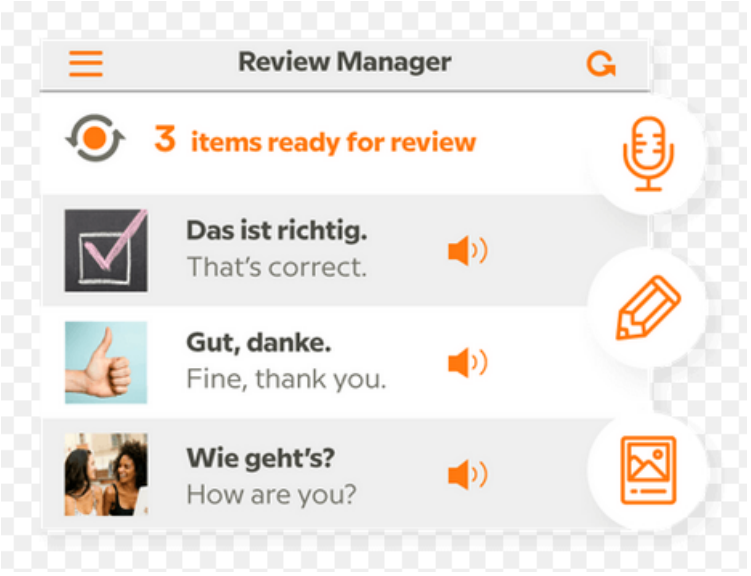

FIGURA 4.2: Babbel

Chapter 5

Análisis y diseño

En este apartado se mostraran y explicarán los diagramas fundamentales que hay que realizar en cualquier proyecto: diagrama de secuencia, diagrama de clases y diagrama de estados.

Antes de explicar como se han diseñado los métodos que utiliza el bot para mantener una conversación, se procederá a explicar el concepto maquina de estados.

5.1 Conversación con el bot

Una conversación trata del dialogo entre dos o mas sistemas que se comunican entre si intercambiándose información. Los chatbots tienen diferentes estados en una conversación: conversación activa, parada y cancelada.

Cuando un usuario ejecuta un comando, se inicia una conversación con el bot, y por lo tanto se intercambia información entre el usuario y el bot, puesto que el bot responde al usuario enviándole la información que se le ha requerido.

Se distinguen tres estados posibles para una conversación: activa, parada, cancelada. Una conversación pasa de estado activa a parada cuando el usuario deja de responder al bot, y este se quede en modo parada, ya que ni el bot ni el usuario han concluido la conversación. Una conversación puede pasar de activa a cancelada cuando el usuario ejecuta el comando */cancel*, y el bot da por concluida la conversación.

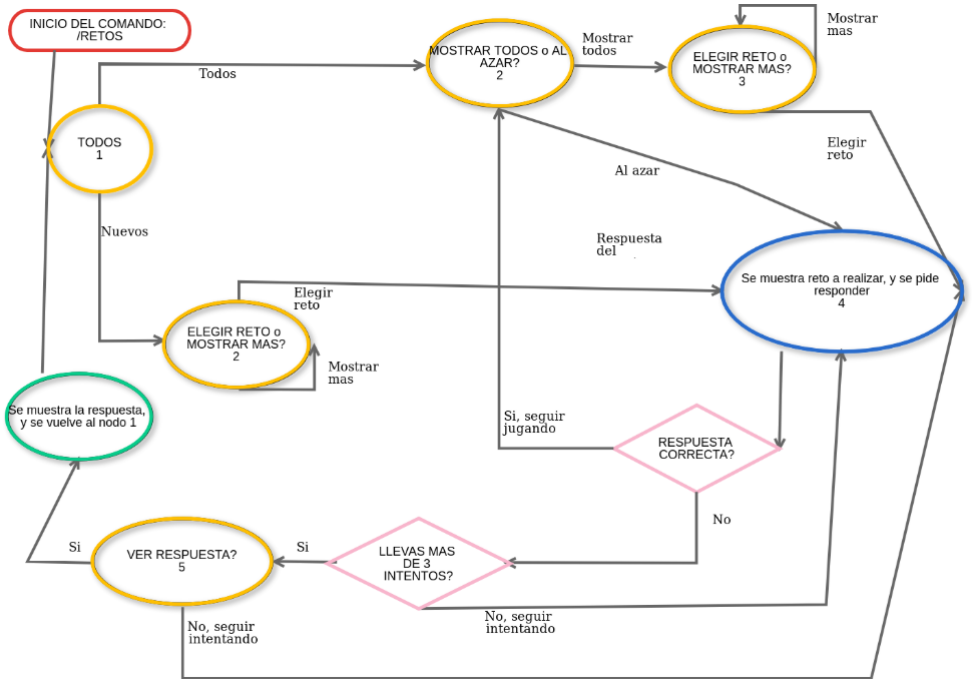

FIGURA 5.1: Diagrama de Estados.

5.2 Diagrama de estados

Una máquina de estados es una estructura de programa que nos sirve para determinar el comportamiento de algo en base al estado en el que se encuentre. Para cada estado por tanto se tendrá un comportamiento. Este sistema funciona mediante entradas y salidas, y las salidas no solo dependen de las entradas actuales, sino que también pueden depender de las entradas mas antiguas.

Para poder guardar un histórico de los datos que han entrado en una conversación, y mantenerlos accesibles, usaremos un objeto llamado notes, que se asocia a la conversación cuando esta se inicia. Aquí es donde se podría guardar toda la información que se quisiera. para poder utilizarla posteriormente.

El bot de Telegram utiliza este sistema y por lo tanto debemos hacer un diagrama de estados que define el flujo de estados de el bot que se lleva a cabo en la ejecución del comando /retos. En el diagrama de estados de la figura 5.1, se puede apreciar que el objetivo del *chatbot* es responder a una serie de preguntas, y obtener una respuesta del bot, tanto positiva como negativa, y que el flujo continúe.

Como se puede apreciar en el diagrama, el bot tiene que pasar por varios estados antes de que el bot responda. En este caso son los distintos botones los que definen en que estado estamos en cada momento, por lo que no hay confusión de estados en ningún caso, cada botón que se pulse, ira al estado que le prosigue.

5.3 Diagrama de clases

El diagrama de clases sirve para mostrar la estructura del sistema como podemos ver en la figura 5.2, definiendo las clases los métodos y atributos, y las relaciones entre estas.

El diagrama de clases tiene algunas clases de la biblioteca, las clases que se usan para este proyecto, y otras clases que se han creado. La clase RetosCommand, es la principal de este diagrama, pues es donde se ha desarrollado la mayor parte del proyecto. Las clase DB es una clase de la propia librería de Longman/TelegramBot, y se ha tenido que modificar, o mas bien, añadir, funciones especificas para nuestro propósito. Las restantes, son clases que hemos tenido que usar, para generar las pantallas, generar las notas de la conversación, etc.

5.4 Diagramas de secuencia

Un diagrama de secuencia muestra como interactúan un conjunto de objetos a través del tiempo mientras se esta ejecutando un proceso.

A continuación se mostrara un diagrama de secuencia, el mas completo, con respecto a las acciones que hace el bot. No se han hecho dos diferentes, ya que la trayectoria seria la misma en todos los casos, solo cambia cuando el usuario envía un audio o contesta mediante escritura. Cuando un usuario envía un audio, el flujo varia un poco y es un poco mas extenso que cuando lo hace de forma escrita, por lo que se ha elegido mostrar ese diagrama.

Para empezar, el usuario ejecuta el comando /retos, y el bot le ofrece dos botones. El usuario pulsa en el botón TODOS para elegir entre todos los retos disponibles que hay en la base de datos. Acto seguido, se le pregunta si queremos mostrar todos los retos, o elegir uno al azar, y el usuario elige al azar (mostrar todos solo seria un paso previo), y

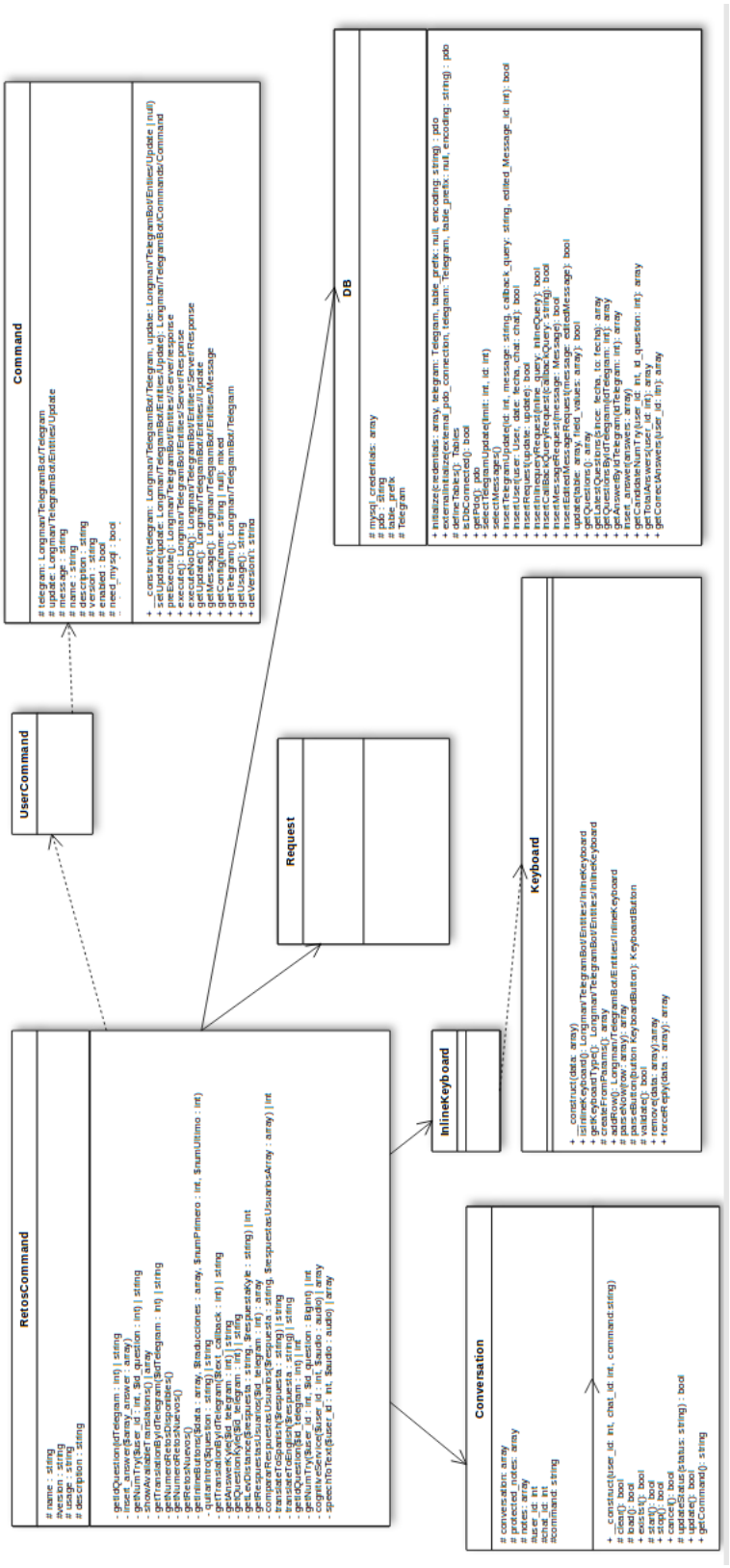

FIGURA 5.2: Diagrama de clases.

cuando pulsa se le muestra la pregunta que va a traducir. Entonces el usuario responde mediante una nota de audio

A continuación en las figuras 5.3 y 5.4, se observan los pasos que da el bot hasta enviar su respuesta.

Cuando el usuario responde con una nota de audio, se llama a los servicios de cognitiveServices y speechToText para procesar ese audio. Una vez que se tiene el audio procesado, es decir, en texto, se procede a comparar la respuesta del usuario con la respuesta original (la de Kyle). Si la respuesta tiene una distanciaLevhenstein[1] menor a 5, entonces la respuesta es correcta, y se guarda en la base de datos la respuesta del usuario, y se le muestra a este que su traducción ha sido correcta, junto con un botón diciendo que siga jugando. Si la distancia levhenstein es mayor que 5, entonces la respuesta no se da por valida, y procedemos a compararla con las respuestas de los usuarios de Twitter. Estas respuestas no son del todo fiables, por lo que necesitaremos por lo menos 3 respuestas exactas para determinar si la respuesta dada por el usuario es correcta, si hay 3 respuestas con distancia levhenstein menor a 3, entonces el bot da la respuesta por buena. En caso de que no se haya acertado con las respuestas de los usuarios de Twitter, se procederá a usar la API de GoogleTranslate, de modo que traduciremos las dos frases, tanto la respuesta de Kyle como la respuesta del usuario del bot, entonces obtendremos con la distancia levhenstein si esta respuesta es correcta o no, en caso de que no sea correcta, se le notifica al usuario que su respuesta no ha sido correcta. Si el usuario ha repetido este reto mas de 3 veces, entonces se le mostrara la opción de ver la respuesta, en caso de que sean menos de 3 veces los intentos realizados, no tendrá la opción de ver respuesta, y únicamente tendrá la opción de repetir la traducción, hasta que la falle 3 veces o mas.

El bot desarrollado puede pasar por 5 estados, definidos a continuación.

Estado 1

El bot ha iniciado la ejecución del comando, y se le propondrá al usuario a ver si quiere realizar entre todos los retos, o únicamente entre los nuevos

[1]Distancia Levhenstein: distancia entre palabras o en otras palabras, el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra

Estado 2

El bot recibe la respuesta del usuario, y le muestra la opción de elegir entre mostrar los retos, o que sea al azar

Estado 3

Dependiendo de el callback del estado 2, nos mostrara por pantalla el reto a realizar, o una lista de retos disponibles, si es al azar, saltará al estado 5

Estado 4

El estado en el que elegimos nosotros el reto a realizar, por lo que se le muestra al usuario el reto.

Estado 5

Aqui el usuario ya ha contestado, y se hacen todas las comprobaciones correspondientes, y el bot responde al usuario con una buena nota, o mala.

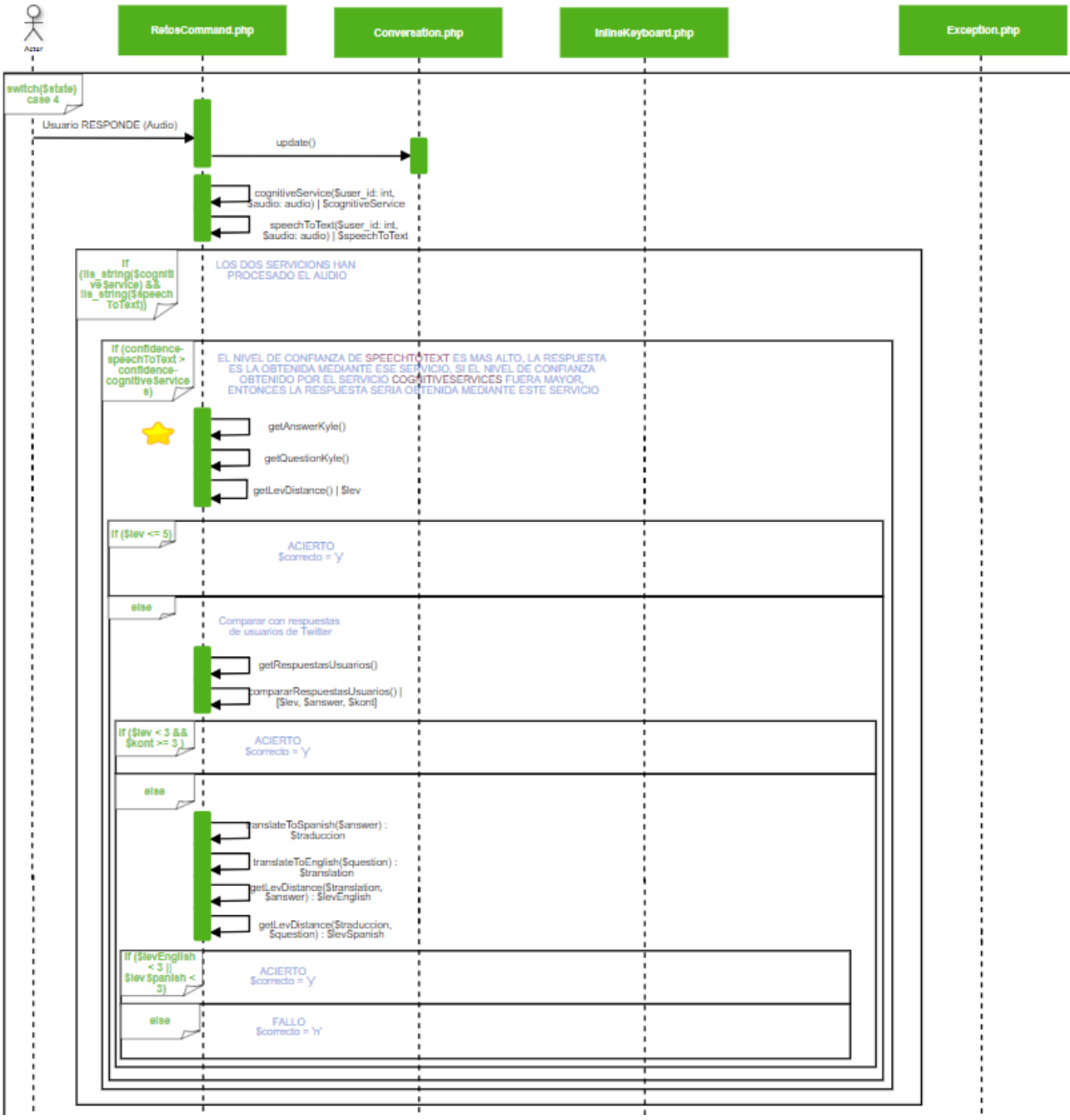

FIGURA 5.3: Diagrama de secuencia 4. El usuario ha contestado mediante audio, por lo que ahora se procederá a llamar a los servicios de speechToText y cognitiveServices para procesar el audio y cambiarlo a texto

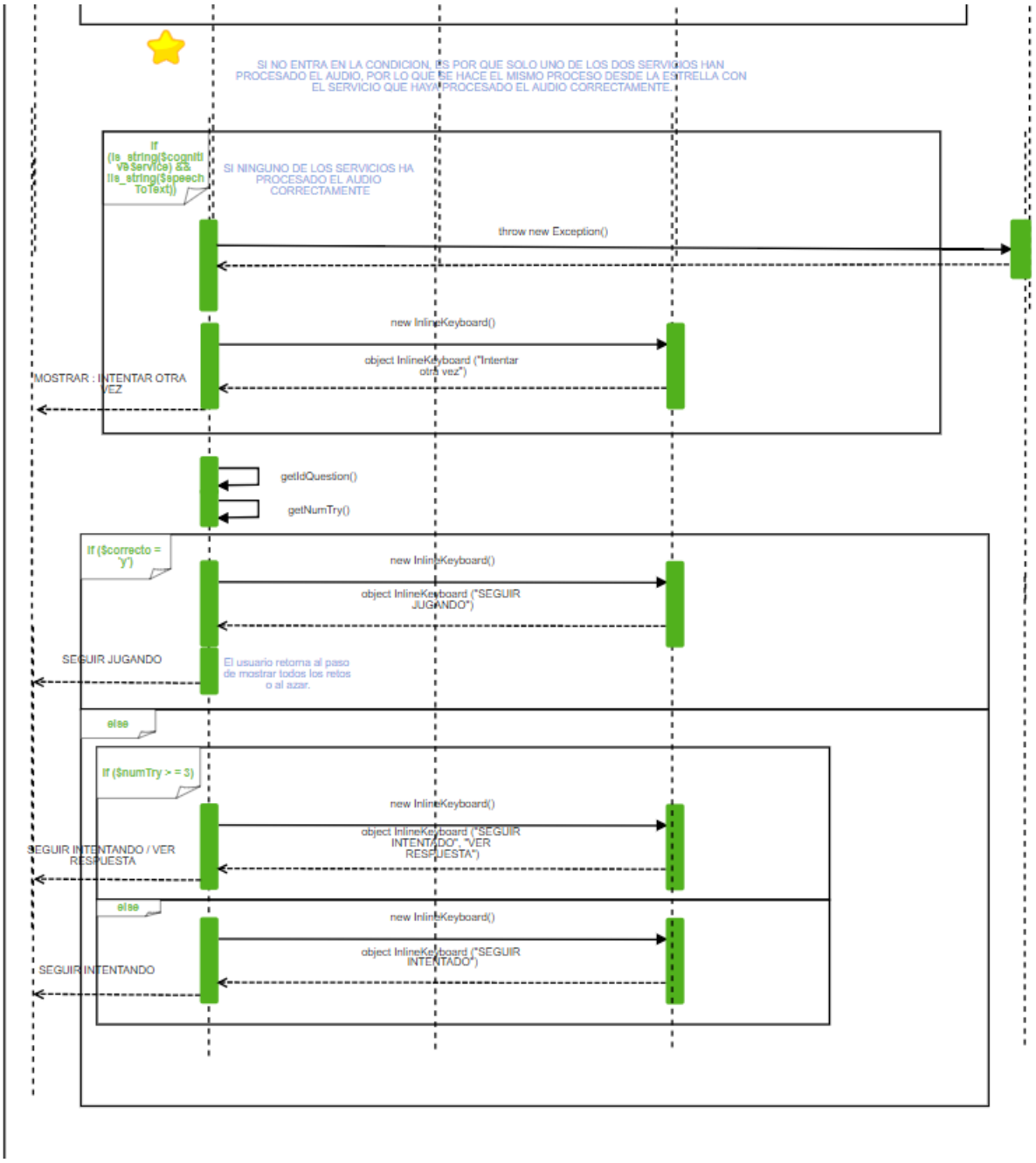

FIGURA 5.4: Diagrama de secuencia 4 (continuación). Lo que hace si el audio es ilegible, y además, se hace la comprobación de si la respuesta del usuario es correcta o incorrecta.

Chapter 6

Desarrollo

En este capitulo se van a detallar todos los pasos que se han seguido a la hora de desarrollar el proyecto. También se explicaran los obstáculos encontrados en el camino, y todo lo referente para resolverlos. El desarrollo se dividirá en dos secciones, por una parte tendremos el desarrollo del Twitter script, y por otra parte el desarrollo del bot de Telegram.

Tal y como se ha explicado anteriormente, el Twitter script sirve para recoger los diferentes retos, por lo que primero se termino de desarrollar este (a pesar de que más tarde sufriría algunas modificaciones debido a los obstáculos que se presentan en el desarrollo), y después el bot. Antes de empezar a desarrollar el bot en su totalidad, se implementó un bot de prueba, que serviría de base y guía para su versión final.

6.1 Necesidades del cliente

Antes de comenzar a explicar el desarrollo, es conveniente tener una vision general de las necesidades del cliente, es decir, de los requerimientos que el bot debe cumplir.

Estas necesidades son: Por una parte, que el bot sea fácil y rápido de usar por parte del usuario, y por otra parte, se necesita que cada pregunta acepte diferentes de respuestas posibles y en un futuro posibilidad de añadir diferentes idiomas.

Facilidad de uso del *chatbot*

Este proyecto estará abierto al publico, por lo que se requiere que su uso sea lo mas sencillo posible, además de ofrecerle rapidez al usuario para llegar a las preguntas que quiere responder. Por esto se han utilizado herramientas sencillas de cara al usuario como botones, de manera que tenga que teclear lo menos posible, y esto haga cómoda y rápida su utilización, además de utilizar tecnologías para poder usar el lenguaje natural de las personas que es la voz.

Capacidad de variantes en las respuestas

Para traducir una frase, la mayoría de las veces, siempre hay diferentes traducciones posibles, por lo que este proyecto tratara de comprender la mayor cantidad de respuestas posibles, y así poder ofrecerle al usuario mas opciones de respuesta.

Capacidad de integrar diferentes idiomas

Actualmente el bot incluye diferentes traducciones en ingles del usuario de Twitter @imkylemillar. Este usuario de Twitter solamente propone traducciones del castellano al ingles. En un futuro, se podrían añadir mas idiomas tanto para preguntar como para responder, obteniéndolas de diferentes fuentes, por lo que el bot debe estar programado con la opción de poder añadirle mas idiomas, y así poder abarcar un publico mas amplio.

6.2 Desarrollo de la base de datos

La base de datos del proyecto, se ha basado en la base de datos que ofrece la biblioteca de PHP longman-telegram-bot. Esta, almacena los mensajes y chats de los usuarios que utilizan el bot. A pesar de que esa base de datos es muy completa y de gran utilidad, para nuestro propósito tuvimos que añadir mas tablas. En nuestro caso, tenemos que mostrar una serie de traducciones disponibles, y estas las obtenemos de esas tablas añadidas.

La figura 6.1 muestra las tablas que se incluyen en la biblioteca *longman telegram-bot* y la figura 6.2 muestra las tablas añadidas para el funcionamiento del proyecto.

La información que recogemos de Twitter mediante Twitter-script se introducen en las siguientes tablas: questions, original-answers, candidate-answers, Twitter-user.

A continuación, se procederá a mostrar la función de cada tabla.

Tabla questions

En esta tabla se almacenan las preguntas y la respuesta original propuesta por el creador de la pregunta, que en este caso es el usuario de Twitter @imkylemillar.

Tabla original-answers

En esta tabla se almacenan las respuestas que los distintos usuarios de Twitter responden a la pregunta formulada. Esta tabla se utiliza para comprobar estas respuestas con la del usuario que este usando el bot, para poder ampliar las respuestas posibles hacia la pregunta.

Tabla candidate-answers

Esta tabla contiene las respuestas de los usuarios del bot, y si la han respondido correctamente o no, para después poder sacar estadísticas de cada usuario.

Tabla Twitter-user

Esta tabla contiene los usuarios de Twitter de donde se recogen las preguntas, en este caso solo hay un único usuario. Esta tabla no es muy necesaria en nuestro caso, pero en un futuro si se quisieran añadir mas usuarios de Twitter de donde recoger las preguntas seria mas fácil de adaptar.

FIGURA 6.1: Diagrama entidad relación de las tablas de la base de datos correspondientes a la biblioteca *longman telegram-bot*.

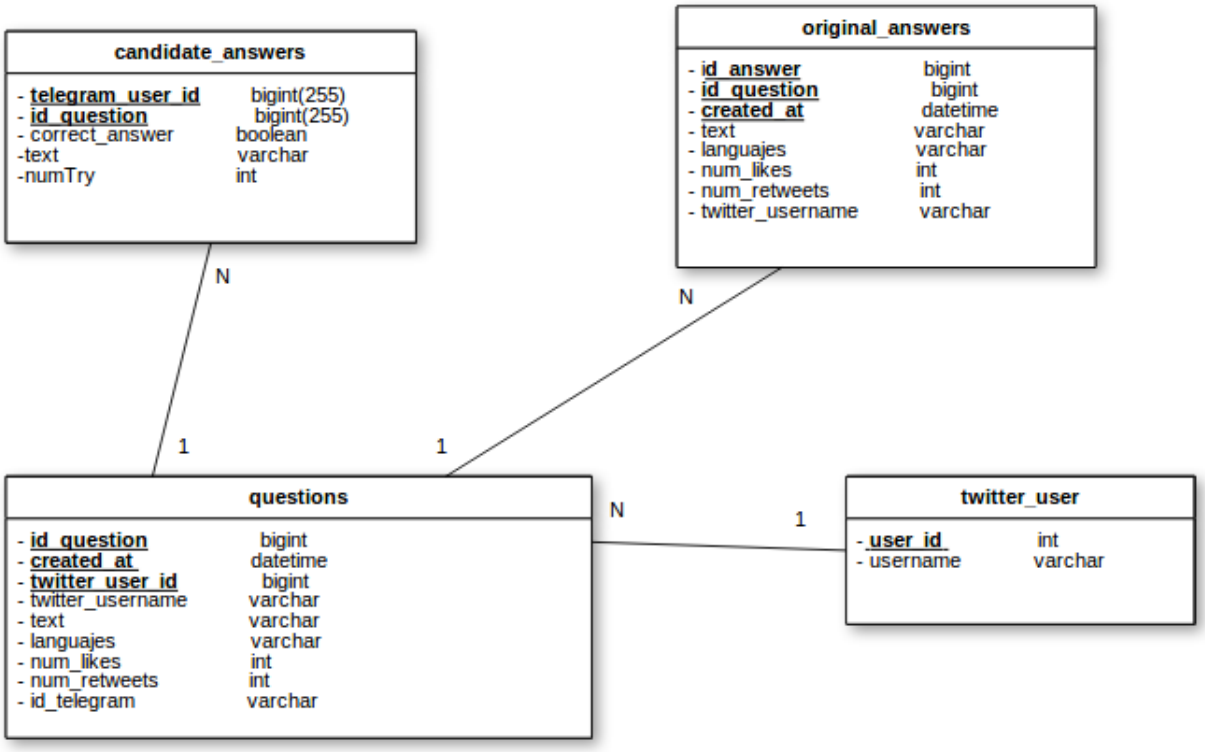

FIGURA 6.2: Diagrama entidad relación de las tablas nuevas creadas para almacenar las preguntas y las respuestas

6.3 Desarrollo del *Twitter-script*

En esta sección se va a proceder a detallar todo lo relacionado con la implementación y el desarrollo del Twitter-script.

Hoy en día, existen varias bibliotecas o paquetes que se han desarrollado para Twitter, y que ofrecen diferentes métodos para poder comunicarnos con la API de Twitter. Lo primero que se hizo, fue elegir la biblioteca que usaríamos y con la que desarrollariamos a lo largo del proyecto. En este caso, se eligió entre dos paquetes: abraham/Twitteroauth y J7mbo/Twitter-api-php. Después de varias pruebas haciendo conexiones e intentando obtener algunos datos, nos dimos cuenta que abraham/Twitteroauth era la mas sencilla de usar.

Una vez que se entiende como usar la biblioteca de forma básica, procedemos a realizar nuestro propósito, que es obtener las preguntas que @imkylemillar propone.

Para empezar, hay que decir que @imkylemillar es un usuario de Twitter que, a diario, propone frases en castellano para traducirlas al ingles. Pasadas unas horas desde la

propuesta del reto en Twitter, el profesor Kyle Millar suele publicar una respuesta posible para todos aquellos usuarios de Twitter que se han atrevido a responder. Antes de empezar el proyecto, ya sabíamos de la existencia de este usuario, y de su trayectoria en Twitter, por lo que nos basamos en esa información para desarrollar el bot.

Cuando conseguimos obtener los tweets de @imkylemillar, es decir, las traducciones posibles con su respuesta, se detecto que no se obtenían mas de 15 retos diferentes. Esto se debe a que Twitter solamente ofrece una API standard que se pueden obtener los tweets de los últimos 7 días. Para hacer pruebas simples son suficientes, pero se quedaban cortas a lo largo del desarrollo del bot, por lo que se procedió a utilizar Twitterscrapper[1], una biblioteca con la que se pueden obtener tweets de cualquier fechas.

Una vez que obtenemos un tweet de @imkylemillar, procedimos a obtener las respuestas que otros usuarios proponen a dicho tweet. En este caso hubo bastantes problemas a la hora de obtener las respuestas, ya que en la documentación de la API no se mostraba claramente como acceder a ellas, y se tuvo que programar una función para obtenerlas.

Mientras desarrollabamos el script, se procede a modificar la base de datos varias veces, implementando los diferentes diseños que realizábamos, hasta quedarnos con el definitivo.

6.4 Desarrollo del *chatbot*

En este apartado, que es la parte principal del proyecto, se explicara como se ha desarrollado el bot, es decir, que pautas se han dado, como ha sido el seguimiento y mostraremos los obstáculos que se iban superando a lo largo del desarrollo.

Antes de empezar a desarrollar el bot, se tuvo que aprender a utilizar el API de Telegram para desarrollo de chatbots, es decir, a partir de un bot desarrollado por el docente Juanan Pereira, del que se tuvo que estudiar a fondo su comportamiento, y de los comandos predeterminados que ofrece la biblioteca longman-telebram-bot, se empezó a desarrollar un pequeño comando, el cual solo nos devolvía exactamente lo que le habíamos mandado, es decir, un comando echo. Una vez que se entendió el funcionamiento básico de los comandos sencillos, intentamos programar un comando un poco mas difícil, usando la api de open-weather[2], que nos devolviera la temperatura que hace en este preciso momento

[1] solo se uso una vez, y es algo que se programo únicamente para propósito educacional.
[2] https://openweathermap.org/api

en la ciudad que el usuario eligiera. Cuando ya tuvimos la formación necesaria para cubrir las necesidades que se nos proponían, empezamos a programar nuestro chatbot, aunque no sin complicaciones, que iremos explicando a lo largo de este apartado.

Tras terminar de desarrollar el Twitter-script, se procedió a mantener una reunión con el tutor del proyecto, y en esta reunión se determinaron las bases en las que se asentaría el proyecto, y como este debería estar diseñado para satisfacer las necesidades de los usuarios. La base de datos ya estaba definida en este punto del desarrollo, y tenia los datos necesarios para nuestras necesidades, por lo que se implemento el comando /retos que serviría como base de lo que iba a ser la tarea definitiva. Este comando en principio solo mostraba todas las traducciones posibles. Y una vez que conseguimos mostrar todas las traducciones (las que existían en la base de datos), queríamos acceder a una en concreto, la que el usuario eligiera. En un principio para acceder a una frase, lo que se hizo fue crear otro comando: /traducir. A este comando habría que meterle como parámetro el numero de la frase que el usuario quisiera traducir, y ahí empezaría el proceso.

Una vez que se programo el comando /traducir, y conseguimos desarrollar el proceso de traducción y respuesta del bot, nos reunimos de nuevo con el tutor del proyecto. En esta reunión, se aclaro la funcionalidad exacta que debía tener el chatbot, y se vio que no era del todo claro el proceso de acceder a las frases, ya que había que utilizar dos comandos por separado: uno para ver las traducciones con el numero de traducción, y otro para acceder a la traducción que el usuario quiera utilizando dicho numero. Esta complejidad hacia que un usuario común no supiera como utilizar el bot, por lo que se busco la manera de ofrecerle al usuario una forma mas sencilla de acceder a la frase que quiera traducir: InlineQueryButtons.

Los InlineQueryButtons, son botones que se imprimen en la pantalla y no en el teclado, es decir, un botón que no desaparece al ocultar el teclado, lo podemos ver en la figura 6.3. Para poder introducir este tipo de botones dentro del desarrollo, tuvimos que empezar por aprender a utilizarlos, y como decirle que nos devuelva una cosa u otra. Cuando ya aprendimos a obtener el resultado que nos devolvían estos botones, cambiamos el formato del chatbot, y esta vez lo hicimos todo en un solo comando. Para ello creamos otro comando: /retos, que englobaría todo el proceso de traducción, desde mostrar las

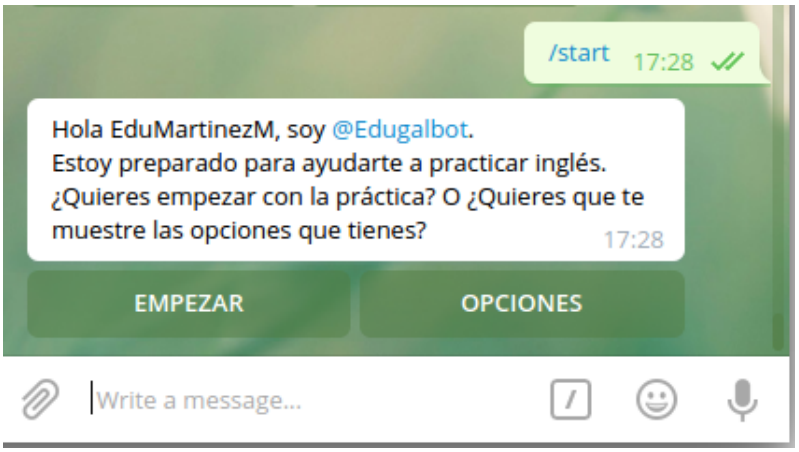

FIGURA 6.3: Figura InlineQueryButtons

traducciones, hasta elegir la traducción a realizar, responder y que el bot conteste.

Lo que se hizo fue utilizar estos InlineQueryButtons para que el usuario tuviera que teclear lo menos posible, e intentar simplificar al máximo la comunicación con el bot. Por eso se decidió hacer la conversación en base a estos. Cuando queremos mostrar todas las traducciones, mostramos 10 frases, cada una en su botón, y un botón mostrar mas, que te ofrecerá otras 10 alternativas. Si llegamos al final, y no hay mas traducciones, el botón "Mostrar más" no aparecerá. Si el usuario quiere elegir una opción tendrá que pulsar encima del botón de la frase, y tendrá que contestar.

Tras terminar el desarrollo básico del comando, de manera que el usuario conteste al bot vía texto, se procedió a cubrir otra necesidad, una de las mas importantes, definida por el director del proyecto, y es que, como ya hemos explicado anteriormente, el bot, debería de ser capaz de interactuar mediante comandos de voz. Para esto, se procedió a utilizar varias tecnologías que nos ofrecen dos de las empresas mas grandes a nivel informático: GOOGLE y MICROSOFT.

Cuando un usuario envía un mensaje de voz, para contestar una pregunta, estos mensajes de audio, se guardan como ficheros en los servidores de Telegram. Es en este punto cuando obtuvimos un pequeño problema, y es que los audios, se guardan en Telegram como .ogg, y ni GOOGLE ni MICROSOFT aceptan en sus APIS este tipo de ficheros para poder convertirlos posteriormente en texto. Lo que se hizo para superar el problema es utilizar la herramienta FFMPEG para convertir esos archivos .ogg a otros tipos de archivos que se aceptarían; como lo son .wav y .flac. Se ha convertido a estos dos tipos de archivos

ya que Microsoft solo acepta .wav, y con respecto a Google, nos dimos cuenta que la conversión se hacia mejor cuando se le transmitía un fichero .flac.

Para hacer la transmisión de ficheros, tanto para descargarnos los audios de Telegram, como para utilizar las APIS de conversión de audio a texto, se utilizo la herramienta CURL.

Se obtuvo definitivamente un sistema completo en el que el chatbot pueda hacer consultas completas, enviadas por el usuario tanto por mensaje de audio, como en formato de texto. Las respuestas que se obtienen de los usuarios, se comparan siempre con las respuestas de la base de datos, tanto de los usuarios que contestan en Twitter al tweet original, como la respuesta original propuesta por el mismo usuario que propone la pregunta. Al hacer unas cuantas pruebas con el bot, nos dimos cuenta de que no era del todo fiable a la hora de responder, ya que a veces podría haber mas de una respuesta posible; por ejemplo, Google Translate traducía algunas frases de una manera diferente a la traducción original, por lo que se nos ocurrió utilizar la API de Google Translate, para poder abarcar mas respuestas posibles.

Google translate nos ofrece una API a la cual podremos llamar para traducir de un idioma a otro, en nuestro caso del ingles al castellano y viceversa. Lo que hicimos para que el bot pudiera comparar entre mas respuestas posibles fue utilizar esa API para traducir la respuesta del usuario al castellano, y la respuesta original al ingles, por lo que de esa manera, tendríamos mas opciones con las que comparar.

Una vez que se consiguió comparar todas las respuestas, y obtener la respuesta del bot, se realizo una ultima reunión con el director. En esta reunión se decidió añadirle otra funcionalidad al bot, y es que cada usuario que utiliza el bot tiene una cuenta de Telegram, o por lo menos un identificador que lo distingue de cualquier otro usuario que también este usando el bot. Se llego a la conclusión de que sería de gran ayuda añadir la función de sacar estadísticas al usuario, es decir, que mediante otro comando: /stats, por ejemplo, cada usuario pudiera acceder a sus estadísticas con respecto a las respuestas que el ha propuesto, de tal forma que el usuario pueda hacerse una idea del nivel de ingles que tiene. Esto se realizo en un periodo corto de tiempo, ya que solo tuvimos que añadir una tabla en la base de datos en la cual introduciríamos todas las respuestas de los usuarios, junto con la pregunta y si la respuesta es correcta o no lo es.

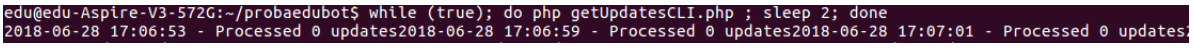

FIGURA 6.4: Comando para ejecutar el fichero getUpdatesCLI.php cada 2 segundos

Para que el proyecto fuera lo mas profesional posible, el director del proyecto nos recomendó crear un segundo bot de Telegram, de tal manera que tuviéramos dos bots, uno para desarrollar la estructura, y hacer todas las pruebas que se necesiten, y otro que sería el bot de producción. El bot de desarrollo se instalo en local, es decir en un ordenador personal, por lo que necesitábamos ejecutar un fichero (getUpdatesCli.php) cada vez que queríamos que el bot procesase una respuesta que le enviábamos. Lo que se hizo para que este proceso de ejecución no fuese molesto, mediante un simple bucle infinito, ejecutábamos ese fichero cada 2 segundos, así se hacia automáticamente sin necesidad de ejecutarlo a mano, lo podemos observar en la figura 6.4. Por otro lado, en el bot de producción, se programo un webhook, de tal forma que era Telegram el que buscaba en sus servidores automáticamente si ha habido algún mensaje, y entonces el bot lo procesaba.

Para poder manejar ambos bots de manera efectiva y cómoda, se creo un repositorio en BitBucket, al cual se subían los cambios que se hacían en el bot de desarrollo, solamente cuando estaba todo correctamente desarrollado, y se descargaban los cambios desde ese repositorio, en el bot de producción, de tal manera que la tarea de introducir los desarrollos en el bot de producción fuera tarea sencilla.

Chapter 7

Tests

El desarrollo del bot, hay que testearlo, y realizar varias pruebas para asegurarnos de que funciona correctamente. Para ello, se han utilizado dos métodos de testeo: Pruebas con usuarios reales, y testing automático.

7.1 Pruebas con usuarios reales

Estas pruebas, se han hecho una vez que teníamos la mayoría de funcionalidades del proyecto programadas, y cuando había una comunicación usuario-bot funcional.

Se ha elegido a diferentes usuarios para que utilicen el bot, y contesten a estas preguntas:

- ¿Te has sentido perdido en algún momento?, ¿Donde?
- ¿Ha sido agradable y fácil la conversación con el bot? (Nota del 1 al 5)
- ¿Recomendarías el bot a otra persona?
- ¿Que funcionalidades más se te ocurren para hacer el bot aun mas completo?

7.2 Testing automático

Para hacer el testing automático, se ha utilizado una biblioteca desarrollada por Telegram, y que básicamente se trata de una librería para construir clientes Telegram. Esta

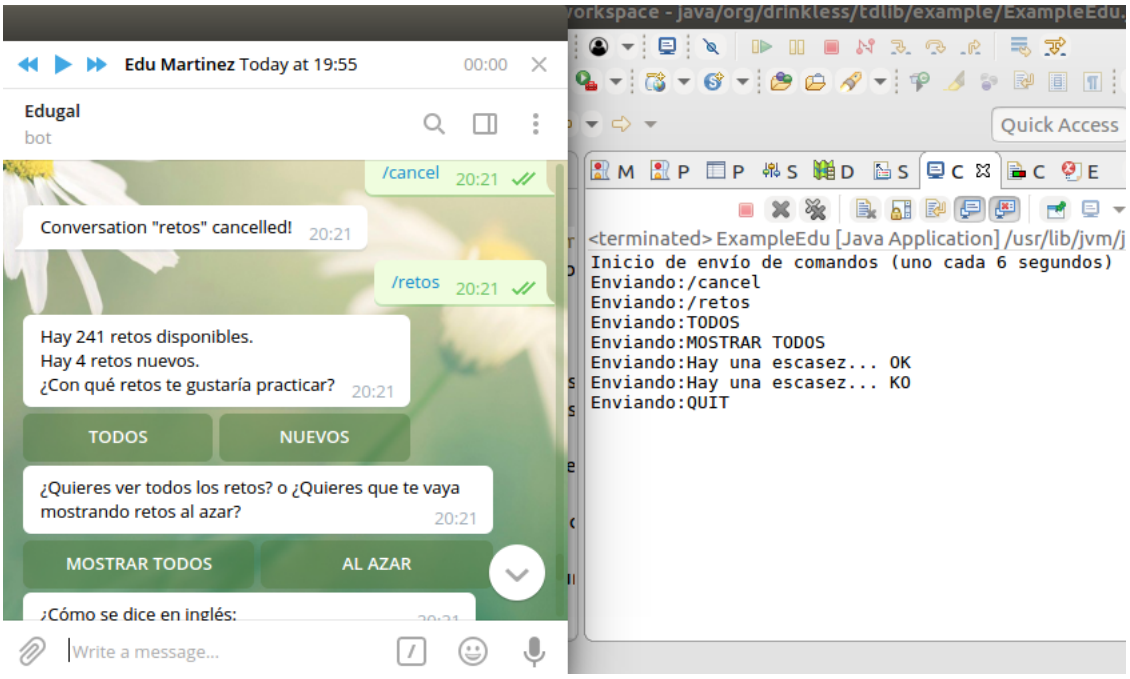

FIGURA 7.1: Ejemplo de ejecución del test automático junto con Telegram.

biblioteca, la utilizaremos de tal modo que se comunique con el bot, mandándole una serie de mensajes para que este responda.

Para crear el programa de testeo mediante esta librería, podremos clonarla en nuestra maquina mediante esta url: https://github.com/tdlib/td.git

El programa de testeo mediante esta librería puede ser programado en cualquier lenguaje que utilice las funciones de C. En este caso, hemos decidido usar el lenguaje JAVA, ya que teníamos conocimientos de ello.

En principio solo usamos esta librería para hacer un testing automático, por lo que hemos simulado un único caso de uso, con dos casos distintos de respuesta, es decir, el cliente Telegram elige una respuesta, y le manda al bot una respuesta que no es correcta, y el bot debería de contestar que NO es correcta y acto seguido se le manda una respuesta correcta, y en este caso el bot debería de decir que la respuesta es correcta.

Chapter 8

Conclusiones

En esta sección, se comparara entre la planificación que se hizo al principio, es decir, cuando la tarea no estaba desarrollada, y lo que se ha llegado a desarrollar. Se obtendrán una serie de conclusiones sobre el proyecto, los planes que puedan hacerse en el futuro y las conclusiones personales que se han obtenido.

8.1 Comparación entre la planificación previa al desarrollo y los objetivos logrados

En principio, cuando se empezó a desarrollar se siguió la planificación previa al desarrollo, y esto ha sido clave para el proyecto, aunque avanzando con el desarrollo, se han obtenido varios problemas, de los cuales algunos no han podido resolverse.

8.1.1 Objetivos

Cuando se estuvo planificando el proyecto, se establecieron algunos objetivos primarios que había que cumplir, y estos mayormente, se han conseguido. Pero hay varias razones por las que se sabe que los resultados no son del todo fiables. y es que para traducir del castellano al ingles, este tiene varias opciones, y ha sido difícil contemplarlas todas.

He aquí dos casos de poca fiabilidad:

Caso 1: Puede que tu respuesta sea correcta, pero no se ha propuesto ni por el usuario de Twitter (@imkylemillar), ni por GoogleTranslate, ni por otros usuarios de Twitter.

Caso 2: Puede que hayas usado abreviaturas en las palabras como "can't" o "haven't", y las respuestas que se contemplen, no estén escritas con abreviaturas.

8.1.2 Herramientas utilizadas

Inicialmente, se contemplo usar una serie de herramientas, que posteriormente, se ha tenido que ampliar.

twitterscrapper

Esta herramienta se ha tenido que usar porque Twitter limita los tweets que se pueden obtener con la API standard a solo 7 días, es decir, solo se pueden obtener los tweets de los últimos 7 días, por lo que para hacer pruebas simples nos podíamos valer de ello, pero para hacer pruebas mas detalladas y con mas información se quedaba corta. Por lo tanto, accedimos mediante esta herramienta a los tweets mas antiguos, así llenando la base de datos de información útil, y pudiendo realizar todas las pruebas necesarias.

FFmpeg

Esta herramienta la tuvimos que usar una vez empezado el desarrollo, ya que cuando empezamos a hacer pruebas para poder enviarle al bot una nota de audio en vez de texto, teníamos que procesar ese audio con los servicios de Microsoft de Cognitive services y los servicios de Google de SpeechToText. Estas herramientas procesan un tipo de audio, y de Telegram se obtiene otro tipo de audio, por lo que esta herramienta nos ayudo a la conversión de formatos de audio.

8.1.3 Planificación temporal

La duración de el desarrollo del bot ha sido mas larga que lo que se esperaba al inicio. Las diferentes herramientas que se quería usar requerían de un aprendizaje que ha sido mas largo de lo esperado, pero que a su vez, ha sido reconfortante haber podido aprender

un poco mas de cada una de ellas. También se pensaba en la planificación que la documentación iba a ser larga y costosa, y así ha sido. Por lo demás, las diferentes partes del proyecto se han asemejado bastante a la duración que se estimaba al principio.

8.2 Planes de futuro

Durante el desarrollo, se han tenido varias ideas y se han apuntado para una posible ampliación de desarrollo del bot. Este, además de lo que ofrece, podría ofrecer mas cosas, alguna de las cuales explicaremos a continuación.

8.2.1 Puntuación de los usuarios

El bot, en la actualidad, guarda el numero de intentos que se ha realizado para cada pregunta, pero solamente le ofrece al usuario la opción de ver sus estadísticas globales, es decir, las preguntas que ha realizado, las preguntas que ha acertado, y el porcentaje de acierto. En un futuro, se podría implementar una estrategia de "gamification", como puede ser añadir soporte para puntuar las acciones de los usuarios y así poder hacerlo mas interactivo, por ejemplo dándole puntos al usuario cada vez que acierte una pregunta, y por cada X preguntas acertadas, ofrecer una medalla.

8.2.2 Mostrar el audio

Actualmente el bot no ofrece la opción de escuchar el audio correcto, es decir, escuchar correctamente como se pronuncia, por lo que en el futuro poder ofrecerle al bot esta mejora seria de lo mas ideal, así el usuario sabrá como se pronuncian las palabras que no sepa pronunciar, y así ofrecerá un servicio bastante inteligente.

8.2.3 Proponer los retos que no se han intentado

El bot seria mas completo si se pudiera acceder a los retos que aun no se han intentado por parte del usuario, para así el usuario tenga claro que retos ha realizado y cuales no.

8.2.4 Añadir idiomas

En la actualidad el bot solo ofrece traducciones del castellano al ingles, pero estaría bien si se le pudiera añadir mas idiomas, y que el usuario eligiera a que idioma quiere traducir la frase.

8.3 Conclusión personal

Para finalizar, quisiera opinar sobre lo que este TFG me ha aportado, y es que, nunca sabes lo que es un proyecto de verdad hasta que te pones a ello, y te das cuenta de todos los problemas a los que te tienes que enfrentar.

Ha sido muy interesante aprender sobre todas las herramientas que se han usado. La programación en php me ha parecido muy útil, y hay mucha información sobre ello en internet, por lo que muchas dudas con respecto al lenguaje se pueden resolver rápidamente, otras no tanto.

La herramienta Curl me ha parecido interesante para transferir archivos, audios etc,. a los servicios de Google y Microsoft. Me he dado cuenta, que hay infinidad de herramientas, que cuando haces que interactúen entre si, intercambiándose información o realizando diferentes acciones, es gratificante ver todo el potencial que puede tener un programa.

Por otra parte, trabajar programando un bot, y haciendo la planificación de como debería de ser la conversación, como deberían de estar situados los botones etc,. ha sido motivador, ya que hemos tenido que pensar en como hacer que un usuario tenga una interacción agradable con una maquina.

Personalmente, este TFG me ha aportado mucho, en cuanto a desenvolverme por mi mismo, a la hora de buscar soluciones, y tengo que agradecer la ayuda de Juanan a la hora de instalar y configurar bibliotecas o herramientas, que me han dado algún que otro problema, y sin su ayuda, el tiempo invertido en el proyecto hubiese sido mayor.

Appendix A

Webs consultadas / Bibliografía

Wikipedia, *PHP*.

URL https://es.wikipedia.org/wiki/PHP

Google API, *SpeechToText, GoogleTranslate*.

URL-1 https://cloud.google.com/speech-to-text/

URL-12https://cloud.google.com/translate/

PHP, *Introducción a cURL*.

URL http://php.net/manual/es/intro.curl.php

Avtandil Kikabidze aka LONGMAN, *PHP telegram bot*.

URL https://packagist.org/packages/longman/telegram-bot

FFMPEG, *FFMPEG - converting audio*.

URL https://www.ffmpeg.org/

TDlib, *Testing the application.*

URL https://github.com/tdlib/td

chatbot, *Integrating call systems with chatbots.*

URL http://www.cys.cic.ipn.mx/ojs/index.php/CyS/article/viewFile/2868/2381

Twitter API, *API reference.*

URL https://developer.twitter.com/en/docs/tweets/search/api-reference