Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2016 / 2017

Bot para evaluación colaborativa de ejercicios orales

Memoria del proyecto

DATOS DE LA ALUMNA O DEL ALUMNO	DATOS DEL DIRECTOR O DE LA DIRECTORA
NOMBRE: **Pablo**	NOMBRE: **JUAN ANTONIO**
APELLIDOS: **Uribe Bastero**	APELLIDOS: **PEREIRA VARELA**
	DEPARTAMENTO: **LENGUAJES Y SISTEMAS INFORMÁTICOS**
FDO:	FDO:
FECHA: **03-11-2016**	FECHA: **02-11-2016**

Índice

Índice de figuras

Índice de tablas

1. Introducción

En el sistema educativo se lleva realizando el mismo tipo de sistema de evaluación desde hace mucho tiempo. El profesor realiza un examen escrito para el cual el alumno debe estudiar de un libro de texto. Este método puede ser tedioso y no representar realmente lo que sabe el alumno. Puede estudiar para un día y al día siguiente no acordarse de nada.

Como no podría ser de otra forma, la tecnología puede cambiar completamente este sistema. Utilizando un dispositivo que, hoy en día, todo el mundo tenemos. Un smartphone. A través del smartphone un profesor podría evaluar a un alumno a través de un sistema de preguntas y respuestas. Lo que se indica no sería muy distinto al sistema de exámenes impuesto hasta ahora, pero un smartphone tiene muchísimas más opciones que un bolígrafo y un trozo de papel. Tiene una cámara de fotos, puede reproducir vídeos, un sistema gps, bluetooth, micrófono, etc.. Existen miles de formas de aprovechar estas opciones, pero en este proyecto se centra en dos: la capacidad de poder grabar voz y la de compartir información a través de Internet en menos de 1 segundo.

1.1. Origen del proyecto

Al principio no tenía muy claro que tema abarcaría mi proyecto. Por ello, fui a preguntar a mi tutor qué le parecían algunas ideas que había tenido. Su respuesta fue muy receptiva y me indicó lo que conllevaría cada una de ellas. Al ver que yo no lo tenía muy claro me propuso un proyecto que el mismo había empezado para sus clases.

El proyecto surge de una asignatura que se realiza online, por lo que el método de aprendizaje es a través de vídeos e emails. Por lo que un sistema de preguntas para poder aprender la materia es muy interesante. Es interesante la opción de poder hacerlo desde el móvil mientras estás viajando en el metro o en el autobús

1.2. Motivaciones para la elección del proyecto

Un proyecto sobre un bot en Telegram. Yo no había trabajado nunca con Telegram pero siempre me ha gustado aprender cosas nuevas. Además, lo que más me motivó fue que el trabajo que yo realizara sería utilizado en su asignatura el año que viene. Y en la de todos los profesores que se apuntaran.

Dentro de unos años muchos profesores utilizarán los bots de la aplicaciones de mensajería como Telegram, Skype, Facebook Messenger o Whatsapp(cuando desarrolle su sistema de bots) para enseñar/evaluar a sus alumnos. Probablemente este proyecto no sea el que triunfe pero al menos habrá sido uno de los primeros.

2. Planteamiento inicial

En esta sección se presenta la descripción, los objetivos, el alcance y las herramientas del proyecto, así como la evaluación económica y de riesgos.

2.1. Descripción

Se pretende añadir la funcionalidad para el profesor de mandar preguntas a sus alumnos para que estos contesten con un grabación de voz. Esta grabación de voz debe ser evaluada por los compañeros de clase con una puntuación. Todo ello a través de Telegram.

2.2. Objetivos

El objetivo principal es la creación de un bot para el aprendizaje colaborativo entre alumnos. A partir de este objetivo surgen otros objetivos secundarios los cuales se exponen a continuación:

- **Web de administración**: el profesor tiene un dato mas a tener en cuenta de cada alumno para poner las notas o valorar como de mal/bien van sus alumnos en función de como van haciendo los ejercicios del bot. Además de poder gestionar preguntas y respuestas del bot.
- **Más funciones**: tiene que ser posible añadir más funcionalidades de manera sencilla
- **Multilenguaje**: se creará un archivo donde se guarden todas las frases para que si se necesitan traducir no haya que indagar en el código.

2.3. ¿Qué son los bots de mensajería instantánea o chatbots?

Un bot en mensajería es un programa informático que se comunica con el usuario a través de mensajes, normalmente de texto. La mayoría de veces los mensajes son reactivos, es decir, que el bot contesta a un mensaje del usuario. Sin embargo, los mensajes también pueden ser proactivos, lo que significa que el bot enviará un mensaje al usuario cuando suceda una determinada situación.

Las funcionalidades de los bots de mensajería son casi infinitas. Se pueden usar para todo tipo de cosas usando todas las posibilidades que te da un smartphone. Desde un "simple"bot, realmente útil, que realiza un encuesta. Por ejemplo, suele ser típico que un grupo de amigos no se ponen de acuerdo que día quedar para una comida. Para solucionar esto es posible crear un grupo en Telegram, meter al bot, poner los días posible para quedar y la gente vota que día es el que le viene mejor.

También es posible realizar juegos con los bots. Existe un bot que simula los juegos de recolección para ordenador, como el que estaba de moda antes, Farmville, o el más reciente Clash of Clans. A donde quiero llegar con estos ejemplos es que el límite de estos bots es tu imaginación.

2.4. Definiciones, acrónimos y abreviaturas

A continuación se enumeran los términos relacionados con la aplicación que se va a realizar.

- **iOS**: sistema operativo para móviles y *tablets* creado por Apple. La última versión estable es la 9.3.5.
- **Botfather**: bot de Telegram que te permite crear bots
- **App**: abreviatura de aplicación.
- **SDK**: siglas de *Software Development Kit* (Kit de Desarrollo de Software). En este proyecto se utilizara el SDK de Microsoft Bot Framework.

2.5. Herramientas

Aquí se detallan las diferentes herramientas que se utilizaran para el desarrollo del proyecto.

- **Node.js**: es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello).
- **Notepad++**: es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación.
- **FileZilla**: es un cliente FTP multiplataforma de código abierto y software libre, licenciado bajo la Licencia Pública General de GNU.

- **PuTTY**: es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre.
- **Microsoft Bot Framework**: SDK de Microsoft que permite la creación de bots multiplataforma.
- **Restify**: es un módulo de Node.js que permite crear servicios web REST.
- **TeXstudio**: programa para la edición de textos de carácter profesional mediante código LaTeX. Se usará para la realización de la memoria del proyecto.
- **MySQL**: es un sistema de gestión de bases de datos relacional.
- **PHP**: es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico
- **HTML**: hace referencia al lenguaje de marcado para la elaboración de páginas web
- **CSS**: es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML
- **ngrok**: herramienta para la creación de túneles seguros en localhost.
- **Bot Framework Channel Emulator**: emulador proporcionado por Microsoft para realizar pruebas.
- **Cacoo.com**: herramienta online para la realización de diagramas de clase, modelos de dominio, diagrama de casos de uso, etc...

2.6. Arquitectura

En la Figura 1 aparece reflejada la arquitectura de la aplicación, la cual viene detallada a continuación:

Base de datos

En la base de datos se guardarán las preguntas, respuestas, notas y todos los mensajes enviados tanto por el usuario como por el propio bot

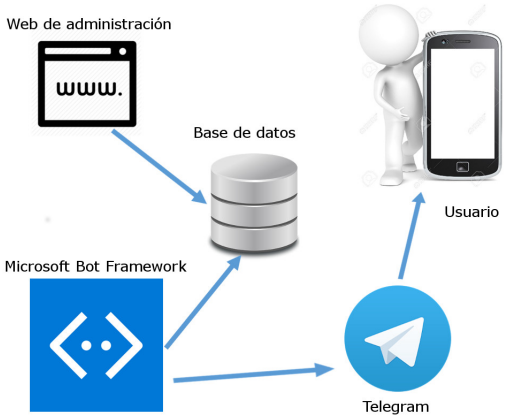

Figura 1: Arquitectura del bot

Web de administración

Desde la web de administración se podrán añadir más preguntas al bot y enviar tareas de evaluación colaborativa. Además, se podrán consultar todas las tareas realizadas por los alumnos: que preguntas han fallado/acertado, notas de voz enviadas, evaluaciones a sus compañeros, etc...

Microsoft Bot Framework

Se basa en un servidor restify de Node. En el están programadas todas las funcionalidades del bot. Accederá a la base de datos para recopilar la información necesaria y la enviará a la aplicación de mensajería correspondiente

Telegram

Por el momento el bot solo estará disponible en Telegram. Gracias a las posibilidades que nos da el SDK de Microsoft en un futuro se puede expandir a muchas otras plataformas (Skype, Facebook Messenger, Slack,...)

Usuario

El usuario tendrá que descargar en su smartphone una aplicación de mensajería(Telegram) y añadir el bot a sus chats. A partir de ese momento podrá comunicarse con el bot.

2.7. Alcance

Para definir el alcance de este proyecto se ha dividido la Estructura de Descomposición de Trabajo o EDT en 6 bloques: organización, aprendizaje, análisis y diseño, implementación y desarrollo, pruebas, y por último, documentación.

A continuación se detalla cada bloque:

1. **Organización**: comprende las tareas de planificación y instalación de software, además de la organización del proyecto.
2. **Aprendizaje**: tareas relacionadas con el estudio de las diferentes herramientas a usar en la aplicación.
3. **Análisis y diseño**: en este bloque se realizan tareas relacionadas con el análisis y diseño de la aplicación mediante diagramas de clase, secuencia...
4. **Implementación y desarrollo**: aquí se albergarán todas las tareas relacionadas con la implementación del bot.
5. **Pruebas**: las pruebas necesarias para el correcto funcionamiento del bot.
6. **Documentación**: tareas relacionadas a la creación y edición del DOP, memoria del proyecto y defensa.

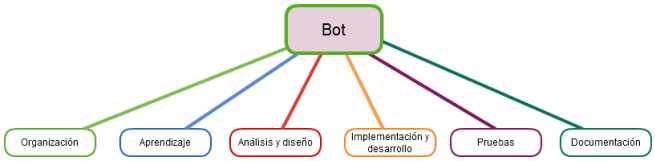

Figura 2: Diagrama EDT por bloques.

En las Figuras 2 y 3, se muestra la división por bloques mencionada previamente, así como la descomposición en tareas de cada bloque.

Todos los bloques son precedentes al siguiente, excepto por una excepción en el bloque de la documentación. Como se puede apreciar, en la Figura 3 se ha subdividido el bloque de documentación en otros dos bloques: DOP y memoria del proyecto. Las tareas relacionadas con el DOP se realizarán en paralelo a la organización y aprendizaje, mientras que las tareas de la memoria se realizarán tras el bloque de pruebas.

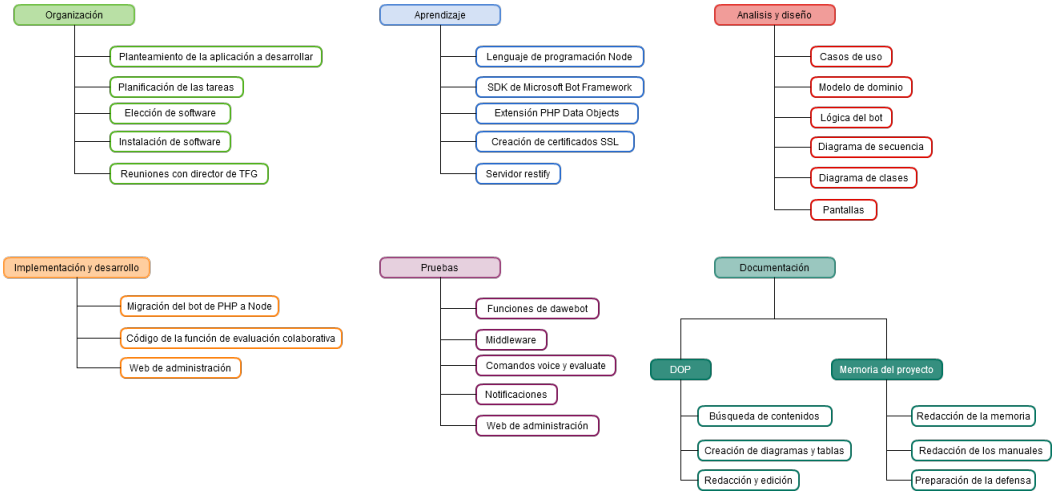

Figura 3: Diagrama EDT por tareas.

2.7.1. Organización

En este primer apartado, se exponen las tareas relacionadas con la organización y puesta a punto de todo lo necesario para la correcta elaboración del proyecto.

Planteamiento de la aplicación a desarrollar
Paquete de trabajo: Organización. **Duración**: 1 horas.
Descripción: en esta tarea se realizará una lluvia de ideas para decidir que tipo de aplicación se va a desarrollar.
Recursos necesarios: ideas.

Planificación de las tareas
Paquete de trabajo: Organización. **Duración**: 5 horas.
Descripción: planificar todas las tareas a realizar para el correcto desarrollo del bot.
Salidas/Entregables: tareas a realizar durante el proyecto.
Recursos necesarios: Cacoo.com y hoja de cálculo de Excel.

Elección de software
Paquete de trabajo: Organización. **Duración**: 5 horas.
Descripción: elegir que herramientas se van a utilizar para la elaboración del proyecto. Se escogerá la versión de la herramienta que sea mas apropiada dado que pueden darse incompatibilidades entre versiones.
Salidas/Entregables: herramientas a utilizar.
Recursos necesarios: ordenador de trabajo y conexión a Internet.

Instalación de software
Paquete de trabajo: Organización. **Duración**: 5 horas.
Descripción: instalación del software específico necesario para la realización del proyecto.
Salidas/Entregables: correcta instalación del software.
Recursos necesarios: ordenador de trabajo y conexión a Internet.

Reuniones con la empresa o el director del TFG
Paquete de trabajo: Organización. **Duración**: 25 horas.
Descripción: reuniones periódicas con el director del TFG para llevar un control sobre la planificación y realización del proyecto así como para establecer alguna modificación al mismo si se considera oportuno.
Salidas/Entregables: cambios a realizar en la aplicación o gestión del proyecto.
Recursos necesarios: lugar tranquilo en el que poder reunirse.

2.7.2. Aprendizaje

A continuación, se exponen las tareas centradas en el aprendizaje de los programas que serán utilizados para la realización del proyecto.

Aprendizaje de Node.js
Paquete de trabajo: Aprendizaje. **Duración**: 15 horas.
Descripción: estudio del lenguaje Javascript en un entorno Node.js.
Recursos necesarios: Servidor con Node.js y Notepad++.

Aprendizaje del SDK de Microsoft Bot Framework
Paquete de trabajo: Aprendizaje. **Duración**: 20 horas.
Descripción: estudio y análisis del SDK de Microsoft, su implementación en Node.js.
Recursos necesarios: Servidor Node.js, Notepad++ y documentación de Microsoft

Aprendizaje de PDO
Paquete de trabajo: Aprendizaje. **Duración**: 5 horas.
Descripción: aprendizaje sobre el uso de PDO en PHP.
Recursos necesarios: Servidor Apache y Notepad++.

Aprendizaje de creación de certificados SSL.
Paquete de trabajo: Aprendizaje. **Duración**: 3 horas.
Descripción: aprendizaje sobre la creación de certificados SSL.
Recursos necesarios: Letsencrypt.

Aprendizaje de creación de servidor Restify.
Paquete de trabajo: Aprendizaje. **Duración**: 3 horas.
Descripción: aprendizaje sobre la creación de servidor Restify.
Recursos necesarios: servidor con Node.js.

2.7.3. Análisis y diseño

En este apartado aparece las tareas relacionadas a la creación de los diferentes diseños de la aplicación.

Análisis y diseño de casos de uso
Paquete de trabajo: Análisis y diseño. **Duración**: 6 horas.
Descripción: realizar los diferentes casos de uso correspondientes a la aplicación.
Salidas/Entregables: casos de uso.
Recursos necesarios: Cacoo.com.

Análisis y diseño del modelo de dominio
Paquete de trabajo: Análisis y diseño. **Duración**: 6 horas.
Descripción: realizar el modelo de dominio correspondiente a la aplicación.
Salidas/Entregables: modelo de dominio.
Recursos necesarios: Cacoo.com.

Análisis y diseño de la lógica del bot
Paquete de trabajo: Análisis y diseño. **Duración**: 6 horas.
Descripción: realizar la lógica del bot mediante un diagrama de estados.
Salidas/Entregables: diagrama de estados.
Recursos necesarios: Cacoo.com.

Análisis y diseño del diagrama de clases
Paquete de trabajo: Análisis y diseño. **Duración**: 6 horas.
Descripción: realizar el diagrama de clases correspondiente a la aplicación teniendo como referencia la lógica implementada.
Salidas/Entregables: diagrama de clases.
Recursos necesarios: Cacoo.com.

Análisis y diseño de los diagramas de secuencia
Paquete de trabajo: Análisis y diseño. **Duración**: 16 horas.
Descripción: realizar los diagramas de secuencia correspondientes a la aplicación teniendo como referencia la lógica implementada .
Salidas/Entregables: diagramas de secuencia.
Recursos necesarios: Cacoo.com.

Análisis y diseño de pantallas
Paquete de trabajo: Análisis y diseño. **Duración**: 6 horas.
Descripción: diseñar las diferentes pantallas del menú que tendrá el bot.
Salidas/Entregables: pantallas del bot.
Recursos necesarios: Cacoo.com.

2.7.4. Implementación y desarrollo

A continuación se exponen las diferentes tareas que forman parte de la implementación y desarrollo del proyecto.

Migración del bot de PHP a Node.js
Paquete de trabajo: Implementación y desarrollo. **Duración**: 45 horas.
Descripción:se migrará el código de PHP a Node.js de la manera más óptima posible.
Salidas/Entregables: código del bot en Node.js.
Recursos necesarios: código antiguo, servidor con Node.js y Notepad++.

Código de la función de evaluación colaborativa
Paquete de trabajo: Implementación y desarrollo. **Duración**: 45 horas.
Descripción:se implementará la funcion de responder con una nota de voz y la de que los compañeros puedan evaluar esta misma.
Salidas/Entregables: código del bot.
Recursos necesarios: servidor con Node.js y Notepad++.

Web de administración
Paquete de trabajo: Implementación y desarrollo. **Duración**: 35 horas.
Descripción: se creará una web de administración para la gestión del bot.
Salidas/Entregables: código de la web.
Recursos necesarios: servidor Apache, MySQL y Notepad++.

2.7.5. Pruebas

En esta sección se detalla las diferentes pruebas que se ejecutarán una vez realizada la implementación del bot.

Pruebas de las funciones de dawebot
Paquete de trabajo: Pruebas. **Duración**: 10 horas.
Descripción: realización de las pruebas pertinentes para comprobar el correcto funcionamiento de las antiguas funcionalidades de dawebot.
Salidas/Entregables: posibles fallos surgidos.
Recursos necesarios: emulador de Microsoft.

Pruebas del Middleware
Paquete de trabajo: Pruebas. **Duración**: 3 horas.
Descripción: comprobar el correcto funcionamiento de la función de guardado de los mensajes tantp enviados como recibidos
Salidas/Entregables: posibles fallos surgidos.
Recursos necesarios: emulador de Microsoft.

Pruebas de los comandos voice y evaluate
Paquete de trabajo: Pruebas. **Duración**: 10 horas.
Descripción: comprobar el correcto funcionamiento de los nuevos comandos voice y evaluate.
Salidas/Entregables: posibles fallos surgidos.
Recursos necesarios: emulador de Microsoft y smartphone con Telegram.

Pruebas de notificaciones
Paquete de trabajo: Pruebas. **Duración**: 3 horas.
Descripción: comprobar la llegada de las notificaciones cuando tienes una pregunta por contestar o una respuesta por evaluar.
Salidas/Entregables: posibles fallos surgidos.
Recursos necesarios: smartphone con Telegram.

Pruebas de la web de administración
Paquete de trabajo: Pruebas. **Duración**: 16 horas.
Descripción: comprobar el correcto funcionamiento de todas las funciones web.
Recursos necesarios: acceso a Internet.

2.7.6. Documentación

Por último, se detallan las tareas relacionadas con la documentación del proyecto. Se ha querido dividir en 2 secciones, una dedicada al documento de objetivos del proyecto, y la otra, a la memoria. De esta forma, se consigue una mejor organización de este paquete de trabajo.

2.7.6.1. Documento de Objetivos del Proyecto

También conocido como DOP, es un documento muy importante a la hora de desarrollar un proyecto, en él, se detallan los objetivos del mismo, así como la planificación temporal, herramientas a utilizar u otros aspectos técnicos.

Búsqueda de contenidos para el DOP
Paquete de trabajo: Documentación - DOP. **Duración**: 6 horas.
Descripción: búsqueda de información necesaria para la elaboración del DOP acerca de las herramientas a utilizar y la tecnología a implementar.
Salidas/Entregables: enlaces de interés.
Recursos necesarios: ordenador con conexión a Internet, herramientas que se van a utilizar.

Creación de diagramas y tablas del DOP
Paquete de trabajo: Documentación - DOP. **Duración**: 10 horas.
Descripción: generar aquellas tablas y diagramas referentes a la planificación temporal u otros aspectos incluidos en el DOP. Los diagramas referidos al diseño de la aplicación no se incluyen en esta tarea.
Salidas/Entregables: tablas y diagramas del DOP.
Recursos necesarios: Cacoo.com, LibreOffice y GanttProject.

Redacción y edición del DOP
Paquete de trabajo: Documentación - DOP. **Duración**: 20 horas.
Descripción: redacción del documento de objetivos del proyecto así como la incorporación de los diagramas y tablas elaborados en la anterior tarea.
Salidas/Entregables: documento de objetivos del proyecto.
Recursos necesarios: TeXstudio.

2.7.6.2. Memoria del proyecto

La memoria del proyecto recoge el desarrollo del proyecto, problemas surgidos durante la realización del mismo, las conclusiones y líneas futuras que se pretendan implementar. También se incluyen los manuales necesarios para una correcta utilización de la aplicación.

Redacción de la memoria
Paquete de trabajo: Documentación - Memoria. **Duración**: 30 horas.
Descripción: redacción y edición de la memoria del proyecto.
Salidas/Entregables: memoria del proyecto.
Recursos necesarios: Latex, diagramas generados, aplicación desarrollada, herramientas utilizadas.

Redacción de los manuales
Paquete de trabajo: Documentación - Memoria. **Duración**: 15 horas.
Descripción: redacción y edición del manual de usuario así como una guía de como instalar el bot para una nueva asignatura.
Salidas/Entregables: manual de usuario y guías.
Recursos necesarios: Latex.

Preparación de la defensa
Paquete de trabajo: Documentación - Memoria. **Duración**: 30 horas.
Descripción: realización y preparación de la defensa del proyecto para su posterior presentación.
Salidas/Entregables: diapositivas presentación, esquema para presentar.
Recursos necesarios: proyecto, memoria, Dropbox.

En la siguiente tabla se muestran las tareas, su duración y las dependencias de cada una. Como se puede observar en la Tabla 1 el número de horas totales del proyecto asciende a 365 horas.

Nº de tarea	Nombre de la tarea	Predecesoras	Duración(horas)
0	**Bot para evaluación colaborativa de ejercicios orales**		**365**
1	**Organización**		**41**
2	Planteamiento de la aplicación a desarrollar		1
3	Planificación de las tareas	2	5
4	Elección de software	3	5
5	Instalación de software	4	5
6	Reuniones con el directo de TFG	2	25
7	**Aprendizaje**		**46**
8	Node.js	5	15
9	SDK de Microsoft Bot Framework	5,8,12	20
10	Extensión PHP Data Objects	5	5
11	Creación de certificados SSL	5	3
12	Servidor Restify	5	3
13	**Análisis y diseño**		**46**
14	Casos de uso	8,9	6
15	Modelo de dominio	14	6
16	Lógica del bot	15	6
17	Diagrama de secuencia	8,9,17	16
18	Diagrama de clases	8,9,16	6
19	Pantallas	8,9,10,18	6

Tabla 1: Dependencias y duración de las tareas (1).

20	**Implementación y desarrollo**		**125**
21	Migración del bot de PHP a Node.js	5,9,15	45
22	Código de la función de evaluación colaborativa	21	45
23	Web de administración	10	35
24	**Pruebas**		**42**
25	Funciones de dawebot	21	10
26	Middleware	21,22	3
27	Comandos voice y evaluate	22	10
28	Notificaciones	22	3
29	Web de administración	23	16
30	**Documentación**		**111**
31	*DOP*		*36*
32	Búsqueda de contenidos	3	6
33	Creación de diagramas y tablas	32	10
34	Redacción y edición	33	20
35	*Memoria del proyecto*		*75*
36	Redacción de la memoria	22,27,34	30
37	Redacción de los manuales	36	15
38	Preparación de la defensa	37	30

Tabla 2: Dependencias y duración de las tareas (2).

2.8. Planificación temporal

Una vez que se han definido las tareas que se realizarán en el transcurso del proyecto queda situarlas en el tiempo mediante la creación de un diagrama Gantt.

Se ha establecido la jornada laboral en 5 horas al día dando un total de 25 horas semanales. Los fines de semana se descansará siempre y cuando los plazos lo permitan y no haya tareas pendientes de terminar.

En la Figura 4 aparece representado el diagrama Gantt del proyecto. Como se puede observar, se ha decidido dejar una semana aproximadamente entre bloques de tareas y entre la preparación de la memoria y la defensa. De esta forma, se consigue un margen de espacio para prevenir posibles problemas que pudieran surgir durante la realización del proyecto.

Por último, y como se ha mencionado anteriormente, el bloque de la documentación se ha dividido en dos secciones. Una es el DOP que se realiza al principio del proyecto y otra la memoria, la cual se empieza a realizar una vez acabadas las pruebas.

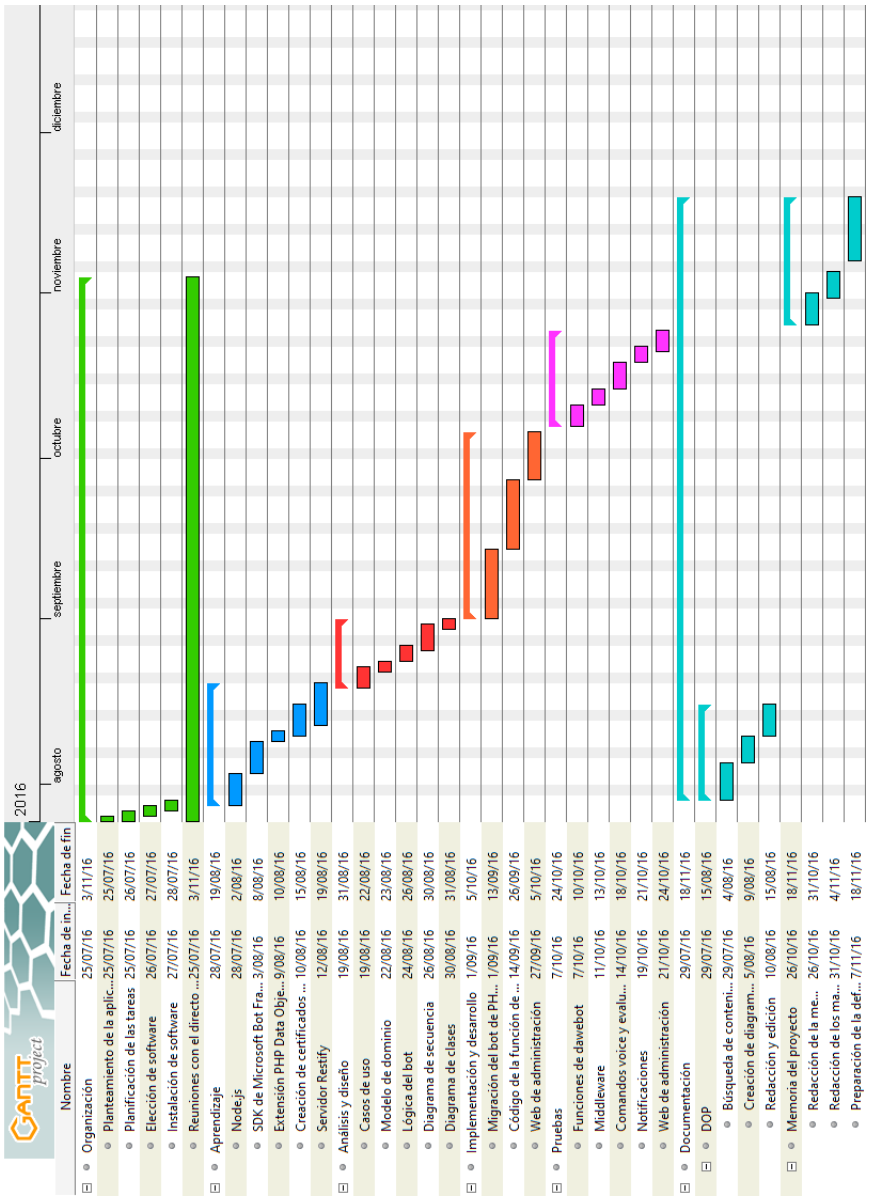

Figura 4: Diagrama Gantt

2.9. Evaluación económica

Al tratarse de un trabajo de fin de grado no se espera reporte económico con su realización. No obstante, se puede calcular el valor económico de la aplicación mediante sus gastos directos e indirectos por si en un futuro se pretendiera comercializar.

2.9.1. Mano de obra

Según lo publicado en el BOE [9] el Miércoles 10 de Febrero de 2016 (Sec III - Anexo 1, Anexo 2) se define el sueldo mensual a 12 pagas de un Programador en 1.384,09 €. El salario por hora de trabajo se puede obtener mediante la siguiente ecuación:

$$Sueldo/hora = \frac{Sueldo\ mes}{Semanas/mes * Días\ trabajo/semana * Horas\ trabajo/día} \quad (1)$$

Un mes tiene, por lo general, 4 semanas de duración, y la jornada laboral está estipulada a 5 días de trabajo a la semana y 8 horas al día. Despejando la ecuación quedaría:

$$Sueldo/hora = \frac{1384,09}{4 * 5 * 8} = \frac{1384,09}{160} = 8,65 \quad (2)$$

Dando un total de 8,65 € por cada hora trabajada.

La duración estimada del proyecto asciende a 365 horas de las cuales 46 son dedicadas al aprendizaje de las herramientas a utilizar. Esas horas no se van a tener en cuenta a la hora de calcular los costes de mano de obra por lo que:

$$Horas\ totales - Horas\ aprendizaje = 365 - 46 = 319\ horas \quad (3)$$

Y multiplicado por el sueldo/hora establecido anteriormente dá un total de:

$$Sueldo = Horas\ trabajo * Sueldo/hora = 319 * 8,65 = 2759,35 \quad (4)$$

El gasto económico en lo referente a la mano de obra sería de **2759,35 €**.

2.9.2. Gasto de software

Al tratarse de un proyecto de fin de grado y poder contar con las licencias de estudiante en las herramientas que lo requerían, o bien por haber usado software libre, el gasto en este apartado es **0 €**.

2.9.3. Gasto de hardware

En este apartado se tratarán los aspectos económicos relacionados con la amortización del material electrónico que se usará en la realización del proyecto.

2.9.3.1. Ordenador portátil HP Pavilion g6

La vida útil que se estima para este portátil es de unos 4 años (48 meses) y su precio fue unos 700 € aproximadamente.

Su amortización mensual es:

$$Amortización\ mensual = \frac{Precio}{Vida\ útil} = \frac{700}{48} = 14,58 \quad (5)$$

El resultado es 14,58 €/mes y como el ordenador va a ser usado durante todo el proyecto (6 meses):

$$Amortización\ total = Amort.\ mensual * Meses = 14,58 * 6 = 87,5 \quad (6)$$

Dando un total de **87,5 €** en lo referente a la amortización del ordenador.

2.9.3.2. Dispositivo móvil iPhone 6 128GB

El precio de este dispositivo fue de 700 € y su vida útil es de 2 años (24 meses) por lo que la amortización mensual quedaría:

$$Amortización\ mensual = \frac{Precio}{Vida\ útil} = \frac{700}{24} = 29,16 \quad (7)$$

El resultado es 29,16 €/mes, y el dispositivo móvil será usado durante el desarrollo y las pruebas de la aplicación que ocupan un periodo de unos 2 meses. Por tanto la amortización total del móvil sería:

$$Amortización\ total = Amort.\ mensual * Meses = 29,16 * 2 = 58,33 \quad (8)$$

La amortización del móvil relativa a este proyecto es de **58,33 €**.

2.9.3.3. Suma de gastos

La suma de todos los gastos relacionados con la amortización de hardware es:

$$Gastos = Amort.ordenador + Amort.móvil = 87,5 + 58,33 = 145,83 \quad (9)$$

Dando un total de **145,83 €** en este apartado.

2.9.4. Gastos indirectos

A continuación se exponen los gastos derivados del proyecto que no tienen una relación directa con el desarrollo del mismo.

2.9.4.1. Luz e Internet

El aula de empresa en la que se va a realizar el proyecto está dentro de la escuela, se utilizarán las tomas de corrientes de la propia facultad para conectar el ordenador, y la red eduroam para la conexión a Internet, por lo que los gastos en este apartado son **0 €**.

2.9.4.2. Bono mensual de metro Bilbao

Para llegar a la escuela se tomará el metro Bilbao[1] tanto para la ida como para la vuelta. Se ha optado por coger el bono mensual, dado que resulta mas económico, junto con el descuento Gazte[2] por ser menor de 26 años que son 34 € al mes y viajes ilimitados entre 2 zonas.

El proyecto se realizará desde Enero de 2016 hasta Junio del mismo año dando un total de 6 meses así que:

$$Gasto\ en\ Bonos\ de\ Metro = Meses * Precio/mes = 6 * 34 = 204 \quad (10)$$

Los gastos derivados en este apartado son de **204 €**.

2.9.4.3. Dietas

Con objeto de perder el menor tiempo posible con viajes de ida y vuelta entre la universidad y el domicilio particular para comer, se llevará la comida previamente preparada en casa a la universidad. Para un correcto transporte de la comida, se comprará un porta alimentos cuyo precio es **30 €**.

2.9.4.4. Suma de gastos

La suma de todos los gastos indirectos es:

$$Gastos = Luz/Internet + Bono\ Metro + Dietas = 0 + 204 + 30 = 234 \quad (11)$$

Dando un total de **234 €** en esta sección.

[1] Pagina oficial metro Bilbao: https://www.metrobilbao.eus/

[2] Mas info: https://www.metrobilbao.eus/utilizando-el-metro/billetes/21/billete/141

2.9.5. Gastos totales

En este ultimo apartado de la evaluación económica, se calcula el gasto total que origina la realización del proyecto.

En la Tabla 2.9.5 aparece reflejada la suma de todos los gastos mencionados previamente.

Tipo de Gasto	**Gasto ocasionado (en €)**
Mano de obra	2759,35 €
Gasto en software	0 €
Gasto en hardware	145,83 €
Gastos indirectos	234 €
Total	**3139,14 €**

Tabla 3: Costes totales de la aplicación

Dando un total de **3139,14 €** por el completo desarrollo de la aplicación.

2.9.6. Posibilidades de negocio

En un principio este bot está orientado a un trabajo de fin de grado sin fines lucrativos. Estará disponible a todos los profesores que estén dispuestos a utilizarlo.

2.10. Riesgos

En este apartado se hablará acerca de los posibles riesgos que pueden surgir durante la realización del proyecto, su plan de prevención y la probabilidad de que sucedan.

Se tendrá en cuenta que hay diferentes tipos de riesgos que dependiendo de su gravedad puedan afectar de una mayor o menor manera al desarrollo del proyecto.

A la hora de calcular el impacto se ha estipulado la jornada laboral a 6 horas al día por lo que 1 día equivale a 6 horas(3 por la mañana y 3 por la tarde).

2.10.1. Problemas con eduroam

Algunos días la conexión a eduroam de la universidad es limitada o nula por lo que no sería posible avanzar en algunos aspectos del proyecto tales como la subida de imágenes al servidor de Vuforia para su evaluación de contraste o no disponer del trabajo actualizado porque la última versión se encuentra en Dropbox.

Prevención

- Tener el trabajo sincronizado en el ordenador y en Dropbox.

Plan de contingencia

- Realizar la parte del trabajo que no requiera conexión a Internet.
- En caso de que el problema persista durante varios días se ira avanzando con esa parte del proyecto en el domicilio particular.

Probabilidad

- Quedarse sin conexión a Internet en la universidad. Baja: 5 %.

Impacto

- Caída del servidor eduroam. 0,05 x 1 día = 0,3 horas. Impacto bajo.

2.10.2. Fallos en el ordenador

Puede darse el caso de que el ordenador empiece a dar fallos tanto de *software* como de *hardware* poniendo en peligro la información del trabajo que se encuentre en ese momento en el ordenador sin copia de seguridad.

Prevención

- Disponer de un buen antivirus en el ordenador.
- Ser cauto a la hora de instalar programas o descargarse cierto tipo de archivos.
- Conservar el ordenador en las mejores condiciones posibles.

Plan de contingencia

- Si es un problema de *software* se intentará buscar una solución por Internet. En caso de que el fallo sea imposible de arreglar se volverá a dejar el ordenador como salido de fábrica salvando la máxima información posible.
- Si resulta un fallo de *hardware* se llevara el ordenador a un experto para su arreglo. Se deberá buscar un ordenador sustituto para perder el menor tiempo posible.

Probabilidad

- Problema de *software*. Baja: 15 %.
- Problema de *hardware*. Baja: 5 %.

Impacto

- Fallo de *software*. 0,15 x 1 día = 0,9 horas. Impacto bajo.
- Fallo de *hardware*. 0,05 x 1 día = 0,3 horas. Impacto bajo.

2.10.3. Problemas con Node.js, Microsoft Bot Framework, Apache, MySQL, PHP, Bot Framework Channel Emulator

Es complicado predecir los problemas que pueden surgir con los programas mientras se está realizando el bot, y más aún cuando no se está familiarizado con ninguna de estas aplicaciones.

Prevención

- Tener una buena curva de aprendizaje con los programas.
- Conocer las mejores páginas y foros especializados para consultar y extraer información.

Plan de contingencia

- Para un problema sencillo, se procederá a buscar en Internet el código de error que arroja el programa y coger la mejor respuesta para poder arreglar el fallo.
- Si el problema resulta un poco más complicado, se avanzará el proyecto por otra vía para volver a intentar solventarlo en otro momento.
- En caso de que sea un problema de difícil solución se procederá a pedir ayuda en los foros de desarrolladores del Microsoft Bot Framework con objeto de perder el menor tiempo posible.

Probabilidad

- Problema fácil solución. Media: 50 %.
- Problema solución intermedia . Media: 25 %.
- Problema difícil solución. Baja: 5 %.

Impacto

- Solución fácil. 0,5 x 1 día = 3 horas. Impacto medio.
- Solución intermedia. 0,25 x 1 día = 1,5 horas. Impacto medio.
- Solución difícil. 0,05 x 1 día = 0,3 horas. Impacto bajo.

2.10.4. Falta de inspiración

La productividad de un trabajador no es la misma todos los días. Esto puede deberse a diversos factores tales como problemas emocionales, cansancio o la incapacidad de solventar un problema surgido en un determinado momento.

Prevención

- Dormir 8 horas al día e intentar llevar un estilo de vida tranquilo.
- Ser benévolo a la hora de establecer los plazos del proyecto.

Plan de contingencia

- Si resulta ser un problema emocional o cansancio, se cogerá la tarde libre para relajarse, y así coger fuerzas para proseguir al día siguiente con el proyecto.
- Para el caso de bloqueo en algún punto concreto del proyecto se procederá a empezar a hacer otra parte del proyecto independiente a la anterior con objeto de no perder tiempo.

Probabilidad

- Problema emocional o cansancio. Baja: 15 %.
- Bloqueo en proyecto. Media: 25 %.

Impacto

- Problema emocional o cansancio. 0,15 x 1 día = 0.9 horas. Impacto bajo.
- Bloqueo en proyecto. 0,25 x 1 día = 1,5 horas. Impacto medio.

3. Antecedentes

En este bloque, se detallarían los programas que comparten características con el proyecto a desarrollar. Hablaremos de @dawebot [5], el chatbot en el que nos basaremos para desarrollar nuestro proyecto.

3.1. @dawebot

@dawebot es un sistema de estudio que nació en base a dos razones:

- Movilidad: el uso del smartphone como forma de estudio para poder estudiar donde se quiera
- Facilidad: se usan las aplicaciones de mensajería (Telegram, Whatsapp, Skype, etc..) como base para el desarrollo del sistema de estudio. Lo que no requiere más aplicaciones instaladas ni aprender una interfaz para cada una de ellas.

Pero, ¿qué es @dawebot? es un chatbot de Telegram programado en PHP que te permite acceder y contestar a una serie de preguntas de tipo test ordenadas por temas. Todas las estadísticas son guardadas en la base de datos lo que te permite realizar una serie de gráficas muy interesantes: quiénes son los mejores/peores alumnos, cuándo lo utilizan más los alumnos, en qué temas se ha fallado/acertado y más... Además, también permite al propio alumno ver cuántas preguntas ha acertado y fallado con un simple comando.

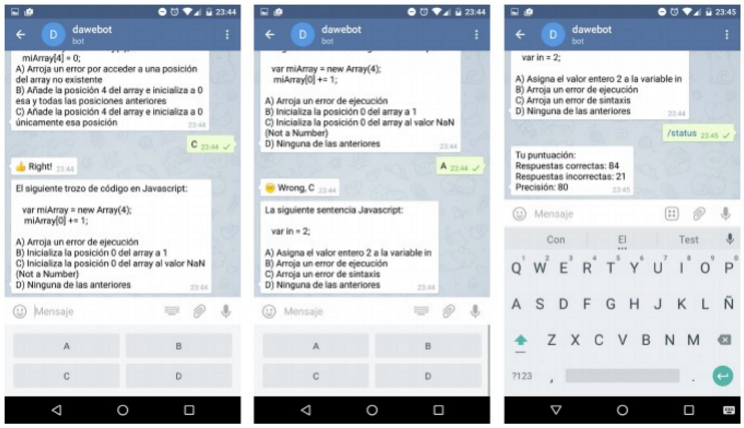

Figura 5: Pantallazos del uso de @dawebot

El bot a desarrollar en este proyecto se migrará el código de @dawebot de PHP a un SDK desarrollado por Microsoft. Después de migrar todas las funcionalidades de este, se comenzará con el desarrollo de nuevas.

4. Captura de requisitos

En esta sección, se van a tratar los diferentes requisitos a tener en cuenta para un correcto desarrollo y funcionamiento de la aplicación. Se tratará de satisfacer, en lo máximo posible, las necesidades de los usuarios.

4.1. Migración del bot de PHP a Node.js

Es un requisito no funcional, pero imprescindible para poder seguir avanzando en el desarrollo del bot. Por una serie de razones que se explicarán más tarde en la sección 5.1, el bot se debe migrar de PHP usado en @dawebot a un SDK desarrollado por Microsoft.

4.2. Jerarquía de actores

El bot no necesita registro previo y todas las funcionalidades están accesibles para cualquier usuario. Por ello, solo se ha definido un tipo de actor aparte del administrador que accederá desde la web de administración. Se puede ver en la Figura 6.

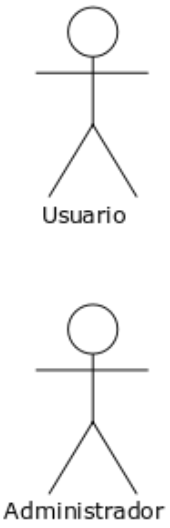

Figura 6: Jerarquía de actores

4.3. Casos de uso

Los casos de uso se utilizan para describir los pasos que deben hacer los actores para realizar acciones dentro del sistema. En este caso existen dos tipos de actores, como ya hemos mencionado antes, Usuario y Administrador.

4.3.1. Usuario

Su diagrama de casos de uso correspondiente se muestra en la Figura 7.

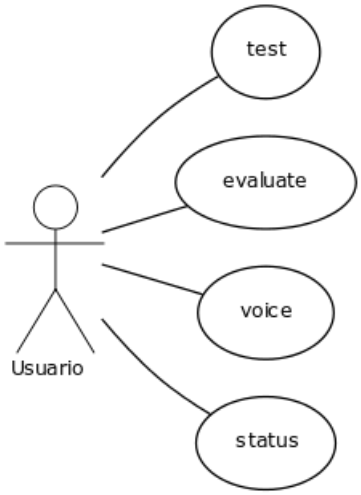

Figura 7: Diagrama de casos de uso de Usuario

- **voice**: el usuario selecciona qué pregunta, si es que tiene alguna, quiere responder con una nota de voz.
- **test**: el usuario selecciona el tema sobre el que quiere responder y se le hace una pregunta de tipo test sobre este mismo.
- **status**: se le permite al usuario visualizar el número de preguntas correctas(del comando test) e incorrectas que lleva hasta el momento.
- **evaluate**: el usuario selecciona qué nota de voz, si es que tiene alguna, quiere evaluar.

4.3.2. Administrador

Su diagrama de casos de uso correspondiente se muestra en la Figura 8.

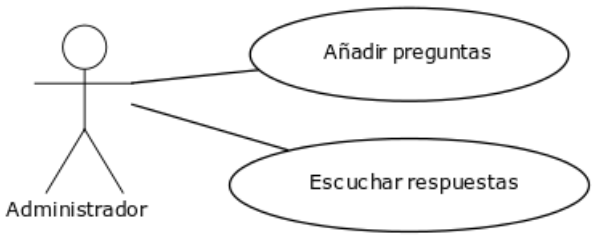

Figura 8: Diagrama de casos de uso de Administrador

- **Añadir preguntas**: permite al administrador añadir preguntas para que los alumnos las contesten con una nota de voz, para que luego sean evaluadas por sus compañeros. Además, se puede seleccionar tanto una fecha de inicio como una de fin.
- **Oír respuestas**: permite al administrador escuchar las notas de voz y ver las notas propuestas por los alumnos.

4.4. Modelo de dominio

El modelo de dominio representa de una forma conceptual aquellas entidades que forman parte de la aplicación y su relación entre ellas.

La descripción de cada entidad viene detallada a continuación:

- **botConnectorBot**: entidad que representa el alma del bot. Aquí es donde se configuran el certificado SSL, la url y puerto en el que escuchará el bot, los comandos y las notificaciones.
- **Diálogo**: entidad que representa los comandos que tiene el bot (test, status, voice, evaluate)
- **Settings**: entidad que representa la configuración del bot.
- **Prompt**: entidad que representa los textos del bot.
- **DB**: entidad que representa la conexión a la base de datos.

- **Gamer**: entidad que representa un jugador, es decir, en este caso un alumno.

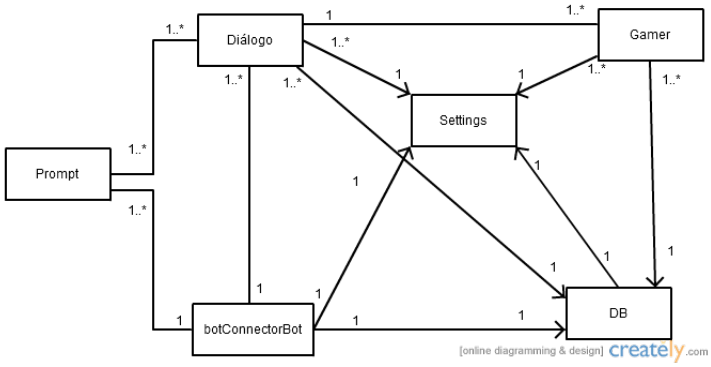

Figura 9: Modelo de dominio

Según la documentación de Node.js [8], en Node.js los archivos y los módulos tienen una correspondencia uno a uno. Por esa razón, la mayoría de las cardinalidades son 1:1.

La relación con Settings, botConnectorBot y DB es siempre uno por lo anteriormente explicado. Diálogo y Gamer tienen una relación 1 - 1..* porque solamente el comando test es el que accede a esta entidad.

5. Elección de lenguajes y tecnologías

Una correcta selección de las herramientas a utilizar se antoja fundamental para un correcto desarrollo del proyecto. Una mala elección, puede suponer una gran perdida de tiempo, por posibles incompatibilidades de formatos a la hora de importar archivos a otras herramientas, o por resultar la herramienta mucho mas compleja de utilizar de lo que se pensó en un primer momento.

A continuación se exponen las diferentes tecnologías usadas en el proyecto y los motivos de su utilización.

5.1. Telegram PHP vs Microsoft Bot Framework

Antes de empezar con nada relacionado con el proyecto había qué decidir que SDK habría que utlizar. El bot original estaba programado en PHP y solo era compatible con Telegram, por lo que se tiene en cuenta también el SDK de Microsoft.

Las principales ventajas de realizar el bot en PHP eran que gran parte del bot estaba ya programado y con un año de pruebas reales con una clase. Sin embargo, el SDK de Microsoft ofrece grandes ventajas. Por ejemplo, es compatible con varias aplicaciones de mensajería incluida Telegram (Slack, Facebook Messenger, Email, Skype, Kik y algunas más) y probablemente añada mas aplicaciones en un futuro, seguramente Whatsapp cuando lance la característica de los bots. Además de la compatibilidad con múltiples aplicaciones de mensajería, otra de las grande ventajas trabajar con Microsoft es saber que tendrás actualizaciones frecuentes y soporte por si tienes algún problema.

Analizando las ventajas y desventajas de ambos SDK, valoré que el SDK de Microsoft era la mejor opción para el desarrollo de este proyecto.

5.2. Node.js vs C#

Una vez elegido el SDK de Microsoft, este te da la opción de poder usar Node.js o C#. La elección del lenguaje ha sido por comodidad. Jamás he programado en C#, por lo que el aprendizaje de este lenguaje me llevaría un largo período de tiempo. Aunque tampoco había programado nunca en Node.js, sí que lo había hecho en Javascript. Por ello, la elección mas lógica me pareció desarrollar en Node.js

6. Análisis y diseño

Por un lado se va a realizar un análisis de los elementos que requieren una mención especial dentro del bot, con objeto de entender mejor su funcionamiento. Por el otro, se van a explicar los diagramas de estados, clases y secuencias que se han realizado para su posterior desarrollo.

6.1. Análisis de los elementos del bot

En este apartado se va a intentar explicar de una forma sencilla, aquellos elementos internos del bot que no se han detallado a lo largo de la memoria, pero que son indispensables para un correcto funcionamiento de la aplicación.

6.1.1. BotBuilder

BotBuilder es la principal clase del SDK de Microsoft, como ya había mencionado antes, es el alma del bot. En esta se configura el certificado SSL, los comandos, los cronjobs, el Middleware y la url en la que escuchará el bot.

6.1.2. Certificado SSL

Para realizar una conexión segura con una web se utiliza el protocolo https en vez del regular http. Para ello se necesita una certificado emitido por una Autoridad de Certificación(AC). Let's Encrypt es una de ellas y proporciona certificados temporales gratuitos renovables cuando agotan. Se ha utilizado esta AC para la conseguir una conexión segura tanto para el bot como para la web de administración

6.1.3. Comandos

Estos son las opciones que tiene el usuario para poder realizar acciones. Hasta el momento tenemos lo siguientes, aunque en un futuro se podrían añadir más:

- **test**: comando para empezar las preguntas tipos test por temas.
- **status**: comando para saber el número de respuestas correctas y erróneas.
- **voice**: comando para responder a las preguntas de voz pendientes.
- **evaluate**: comando para evaluar las respuestas de voz pendientes.

6.1.4. Cronjobs para notificaciones

Un cronjob es una función que se ejecuta cada cierto tiempo. Para este bot utilizamos dos cronjobs para mandar notificaciones al alumno:

- **voiceAnswer**: cuando el profesor añade una pregunta de voz que un alumno la conteste, este Cronjob se encarga de avisar al alumno que tiene una pregunta pendiente. Se ejecuta cada minuto.
- **voiceGrades**: cuando el alumno contesta a una pregunta con una nota de voz, los alumnos que deben evaluarla son notificados con un mensaje. Se ejecuta cada minuto.

6.1.5. Middleware guardando conversaciones

El SDK de Microsoft proporciona Middleware para realizar lo que se desee con los mensajes antes de enviarlos. Se ha utilizado esta funcionalidad para guardar todos los mensajes enviados y recibidos por el bot para que si existe algún problema se pueda verificar si el bot ha estado funcionando y enviando los mensaje correctamente.

6.1.6. Messaging endpoint

Para configurar la url de escucha del bot se utiliza un servidor Restify con el certificado SSL mencionado antes.

6.2. Diagrama de estados

En este apartado, se detallan los diferentes estados de la aplicación y su relación entre ellos. El diagrama de estados correspondiente a la aplicación se muestra en la Figura 10.

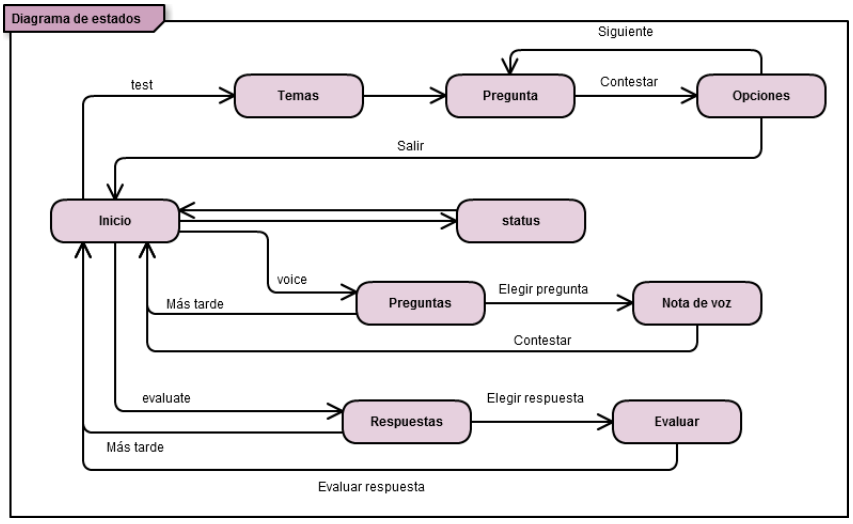

Figura 10: Diagrama de estados

- **Inicio**: es el primer estado al iniciar una conversación con el bot. Para pasar al siguiente estado se debe llamar a uno de los comandos disponibles.

- **Temas**: se debe seleccionar uno de los temas que se dan a elegir.
  - **Pregunta**: se realiza una pregunta que se debe contestar.
  - **Opciones**: se dan las opciones de Salir(volver al estado Inicio), Siguiente(pasar a la siguiente pregunta) o si está disponible preguntar la razón y detalles por los que la respuesta es correcta o incorrecta.
- **Status**: se muestran las respuestas correctas y erróneas hasta el momento.
- **Preguntas**: se muestran las preguntas pendientes de contestar.
  - **Nota de voz**: una vez elegida una pregunta se debe mandar una nota de voz con la respuesta.
- **Respuestas**: se muestran las respuestas de notas de voz pendientes de evaluar.
  - **Evaluar**: una vez elegida una respuesta se debe evaluar esta misma.

6.3. Diagrama de clases

Un diagrama de clases es un buen método de ver la estructura de una aplicación orientada a objectos. En el aparecen los atributos y métodos de cada entidad así como su relación.

En la Figura 11 se muestra el diagrama de clases correspondiente al bot.

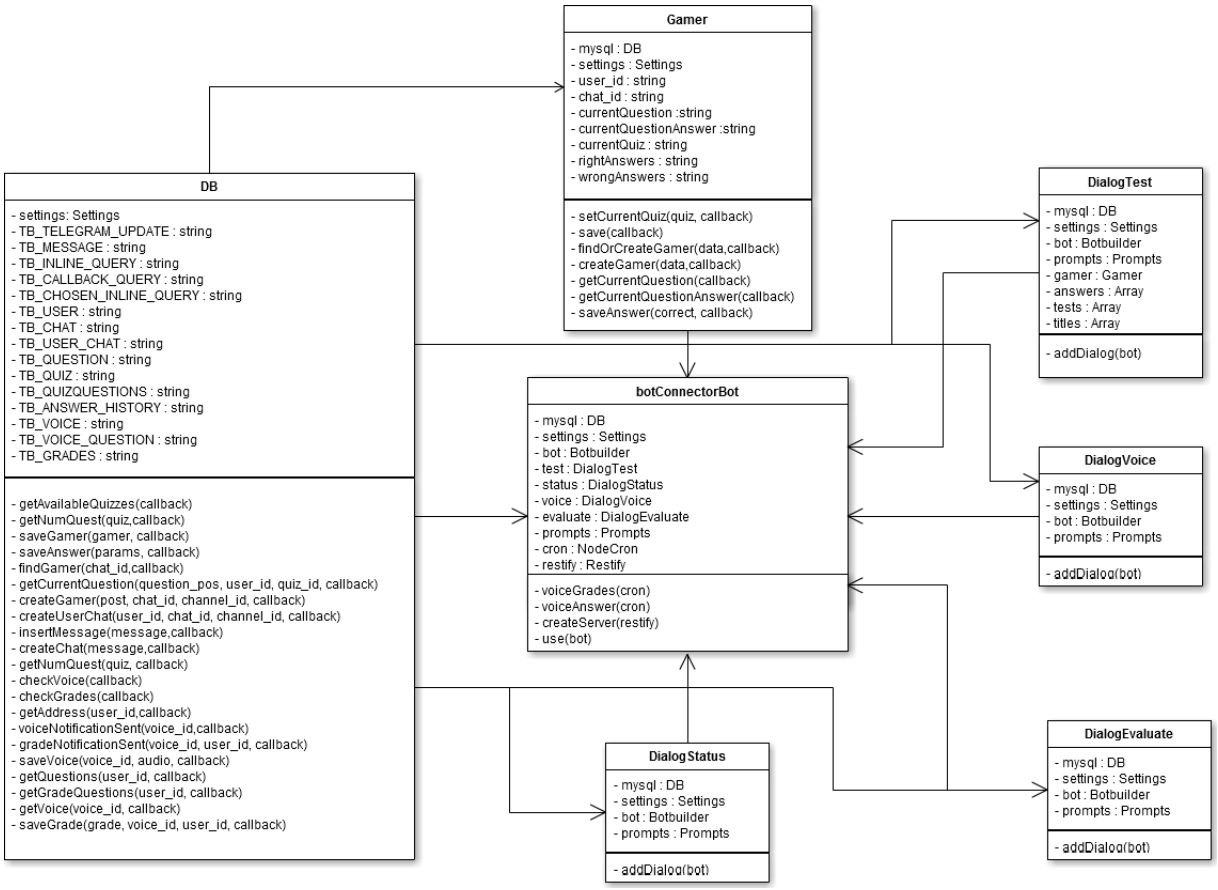

Figura 11: Diagrama de clases

- **botConnectorBot**: es la clase principal, que se encarga de llamar a todos los métodos en el momento adecuado.
- **DB**: gestiona todas las llamadas a base de datos.
- **DialogTest**: clase donde llama al invocar el comando test.
- **DialogStatus**: clase donde llama al invocar el comando status.

- **DialogVoice**: clase donde llama al invocar el comando voice.
- **DialogEvaluate**: clase donde llama al invocar el comando evaluate.
- **Gamer**: gestiona el funcionamiento de las estadísticas y las preguntas de un jugador cuando invoca el comando test.

6.4. Diagramas de secuencia

Se han creado los diagramas de secuencia que pertenecen a cada uno de los comandos importantes. Se ha omitido el comando status por ser una simple llamada a base de datos.

El comando test (Figura 12) primero consulta la base de datos en busca de los temas disponibles. Después de que el usuario haya elegido uno de ellos, realizará una pregunta con sus respectivas respuestas. Una vez que el usuario haya respondido se le dará la opción de pasar a la siguiente pregunta (loop) o Salir y acabar con el diálogo. Al responder también se le dará la opción de preguntar porque la respuesta es correcta o errónea, aunque no se ha puesto en el diagrama de secuencia para que quedase claro el comportamiento del comando test.

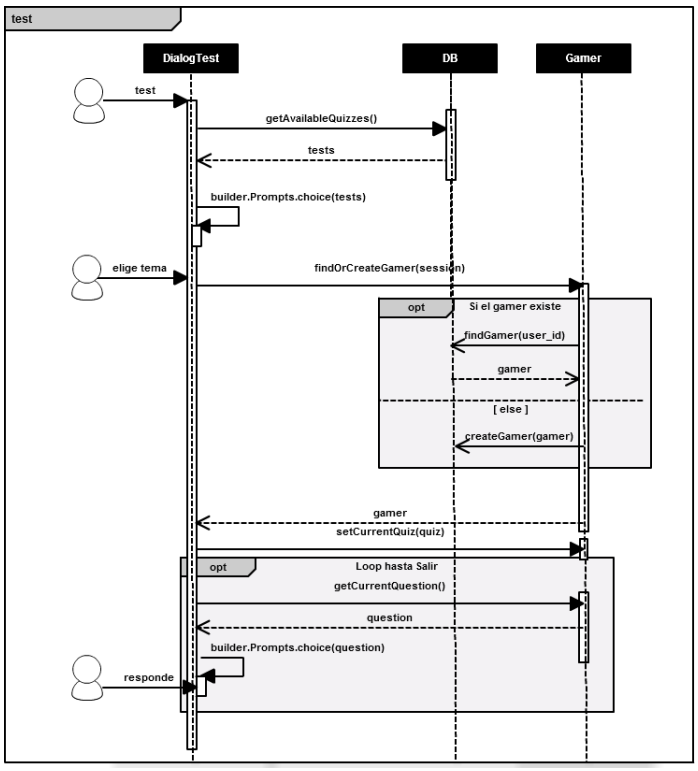

Figura 12: Diagrama de secuencia del comando Test

El comando voice (Figura 13) primero consulta si el alumno tiene preguntas por responder. Una vez que el alumno haya elegido una pregunta, deberá mandar una nota de voz. Finalmente, la nota de voz será guardada en la base de datos.

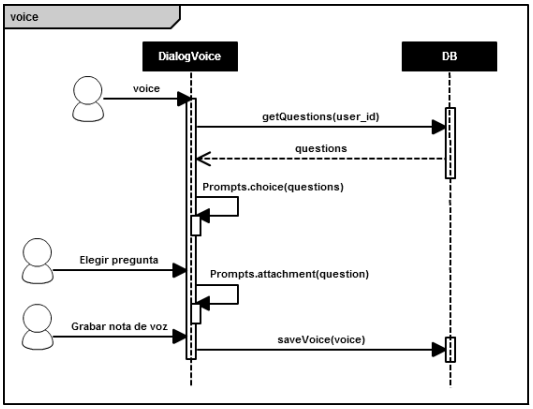

Figura 13: Diagrama de secuencia del comando Voice

El comando evaluate (Figura 14) primero consulta si el alumno tiene respuestas sin evaluar. Una vez que el alumno haya elegido una de estas, se le mandará nota y deberá evaluar la respuesta. Por último, la nota será almacenada en la base de datos.

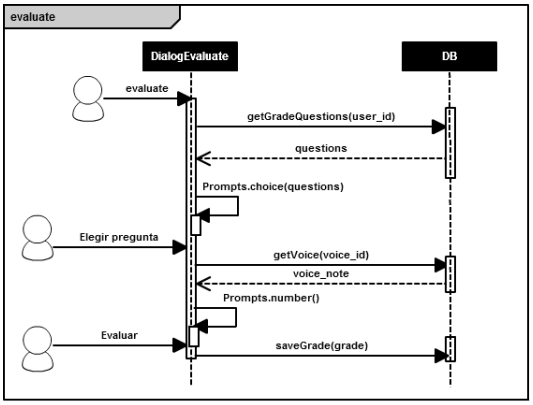

Figura 14: Diagrama de secuencia del comando Evaluate

7. Desarrollo

En esta sección se van a explicar de una forma sencilla los pasos realizados para la elaboración de la aplicación. También se expondrá la evolución que han tenido los diseños iniciales a la hora de adaptarlos al diseño final del bot.

7.1. Desarrollo inicial del bot

En un principio había que migrar todo el código ya programado en PHP a Node.js. Al ser Node.js un lenguaje asíncrono orientado a eventos y no a objetos, la lógica debía cambiar por completo. Esto nos daba la oportunidad de coger todas las cosas buenas de PHP y re-diseñar lo que se pudiese mejorar.

El cambio de usar el SDK en PHP al de Microsoft también supuso algún cambio en la base de datos para la gestión de usuarios pues para cada plataforma (Telegram, Skype, Slack,etc..) se usa un sistema distinto y Microsoft los unifica de forma bastante útil.

Por lo demás, partió como base de un bot de ejemplo ofrecido por el SDK de Microsoft y se fue modificando este mismo para el desarrollo de nuestro bot. La idea inicial era crear una clase para la gestión del bot y sus características. Aparte de esta clase debía haber una para cada comando, una para la base de datos y otra más para el alumno(la idea de esta se sacó la versión en PHP).

7.2. Desarrollo final del bot

Las principales clases del diseño inicial y final apenas variaron. Tan solo se añadieron dos clases más. La de configuración donde se guardan los datos de acceso a la base de datos, los datos del bot para las notificaciones, si se deben activar o no los logs y las respuestas para las preguntas tipo test (A,B,C,D). Y la de textos, donde se definen todos los textos con los que responderá el bot en vez de definirlos dentro del código.

Lo más complicado de todo el desarrollo del bot fue tener que aprender Node.js. Acostumbrado a programar en lenguajes orientados a objetos como Java, PHP u Objective-C, aprender un nuevo lenguaje orientado a eventos fue una tarea ardua. Aunque una vez que aprendí las bases, el desarrollo fue mucho mas llevadero.

7.2.1. Pantallazos

Comando test

En la figura 15 se puede observar como se llama al comando test que te da la opción de elegir un tema.

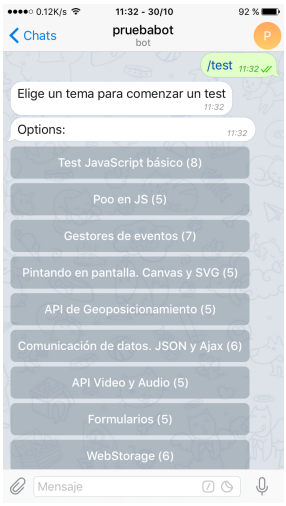

Figura 15: Comando test

Después de haber elegido un tema, aparece la pregunta con sus respuestas a elegir (Figura 16).

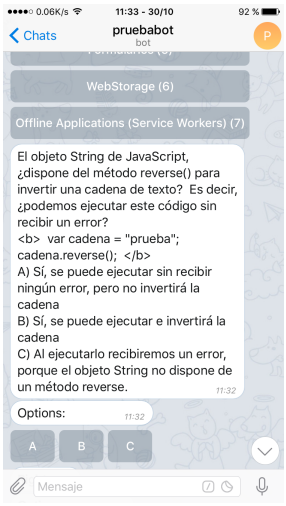

Figura 16: Comando test

Finalmente, tras haber respondido, te informa si has acertado o fallado y te da distintas opciones(Figura 17).

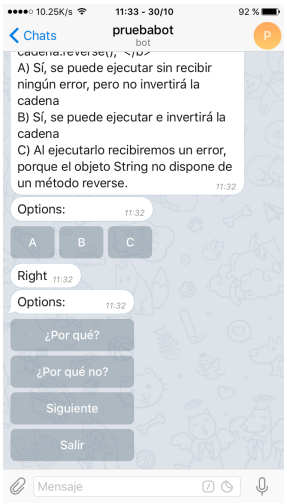

Figura 17: Comando test

Comando status

En la figura 18 se ve como se llama al comando status que muestra las estadísticas.

Figura 18: Comando status

Comando voice

En la Figura 19 se puede observar como se llama al comando voice que muestra las preguntas pendientes.

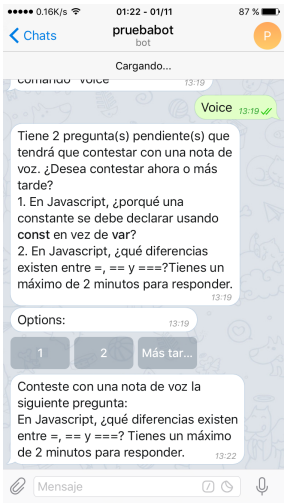

Figura 19: Comando voice

Tras elegir una de las preguntas, se requiere enviar una nota de voz (Figura 20).

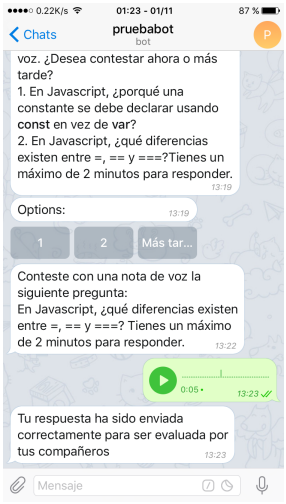

Figura 20: Comando voice

Comando evaluate

En la Figura 21 se ve como al principio el bot nos ha notificado que tenemos una nota de voz pendiente de evaluar. Ejecutamos el comando evaluate y este nos muestra las preguntas de las notas de voz pendientes.

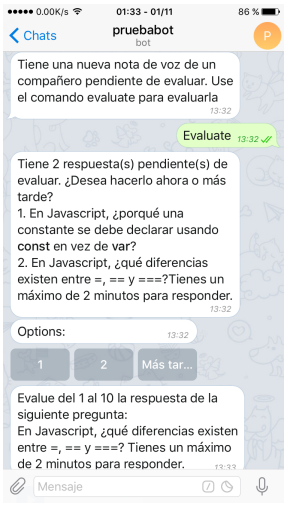

Figura 21: Comando evaluate

Una vez se ha elegido una de ellas, te envía la nota de voz para que la escuches y la evalúes (Figura 22).

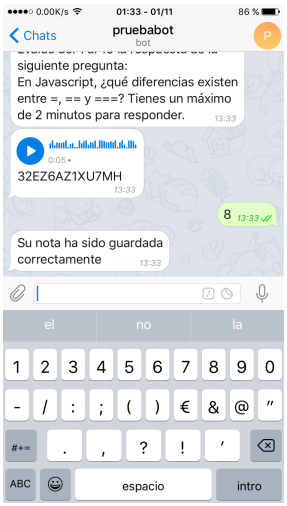

Figura 22: Comando evaluate

Web de administración

A continuación se muestran las pantallas de la web de administración. En la Figura 23 se ve el listado de preguntas enviadas junto con la fecha límite para responder, las evaluaciones no tienen fecha límite.

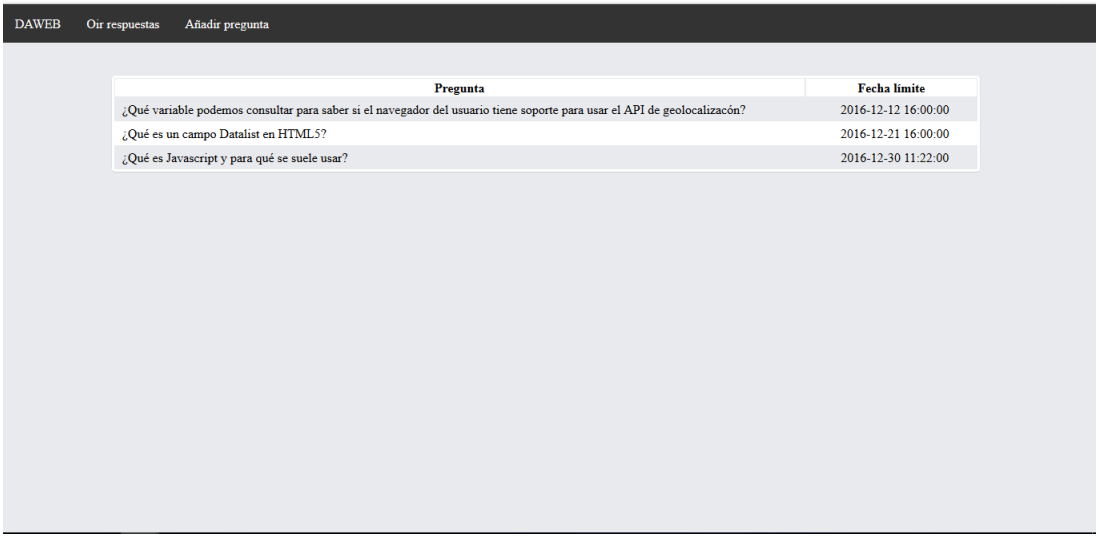

Figura 23: Web de administración

En la Figura 24 se pueden ver todas la respuestas y evaluaciones realizadas sobre una pregunta en concreto. A la izquierda se ven los alumnos que deben responder a la pregunta con su nombre, apellido y usuario de Telegram. Las notas de voz se pueden escuchar con un reproductor en la propia web sin necesidad de descargarse lo archivos. A la derecha observamos todos los alumnos que deben evaluar una nota de voz.

Figura 24: Web de administración

En la Figura 25 se ve el formulario desde donde se envía una nueva pregunta. Para la fecha límite se ha programado un calendario con JQuery para un uso mas intuitivo de esta funcionalidad. También se ha programado un checkbox que marca y desmarca todos los alumnos para que respondan/evalúen. Cuando uno de los alumnos que debe responder la pregunta no está marcado se esconden los evaluadores para quitar contenido basura de la pantalla.

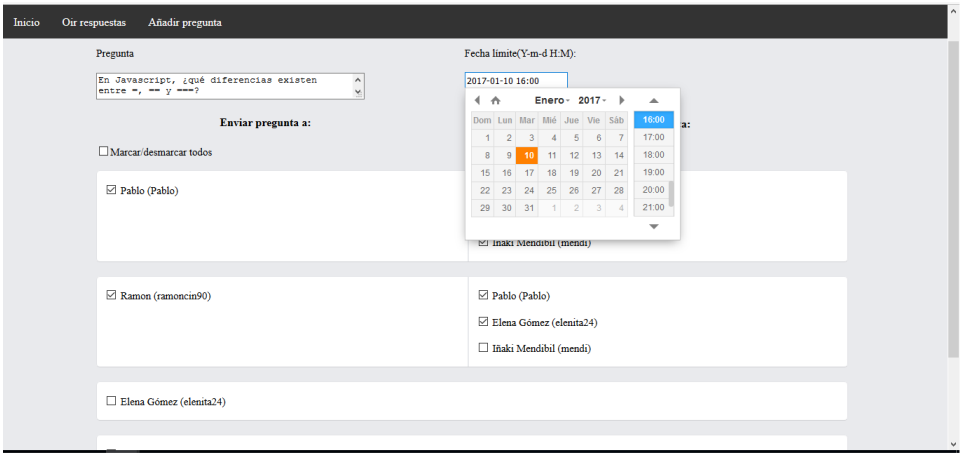

Figura 25: Web de administración

8. Pruebas

En esta sección se van a exponer las pruebas que se han realizado para comprobar el correcto funcionamiento de la aplicación.

8.1. Pruebas de la migración

En primer lugar, se exponen las pruebas realizadas para asegurar la correcta migración del bot a SDK de Microsoft.

Prueba 1 de la migración
Descripción de la prueba: llamar al comando status.
Salida esperada: aparecen el número de respuestas falladas y acertadas hasta le momento.
Resultado obtenido: éxito.

Prueba 2 de la migración
Descripción de la prueba: llamar al comando test.
Salida esperada: aparecen los temas disponibles para realizar un test.
Resultado obtenido: éxito.

Prueba 3 de la migración
Descripción de la prueba: elegir un tema de los propuestos por el comando test.
Salida esperada: aparece la pregunta con las respuestas posibles.
Resultado obtenido: éxito.

Prueba 4 de la migración
Descripción de la prueba: funcionan como deberían los botones ¿Porque? y ¿Porque no?.
Salida esperada: los botones ¿Porque? y ¿Porque no? muestran el texto de base de datos correspondiente y muestran las opciones Siguiente y Salir.
Resultado obtenido: éxito.

Prueba 5 de la migración
Descripción de la prueba: funciona como debería el botón Siguiente.
Salida esperada: aparece la siguiente pregunta.
Resultado obtenido: éxito.

Prueba 6 de la migración
Descripción de la prueba: funciona como debería el botón Salir.
Salida esperada: sales del diálogo y vuelves al estado inicial.
Resultado obtenido: éxito.

8.2. Pruebas del Middleware

Las pruebas de esta sección están dirigidas al correcto funcionamiento del registro de toda la actividad del bot.

Prueba 1 del Middleware
Descripción de la prueba: enviar un mensaje al bot.
Salida esperada: el texto se guarda en la base de datos, así como el usuario que lo ha enviado y el identificador del chat en el que se ha mandado.
Resultado obtenido: éxito.

Prueba 2 del Middleware
Descripción de la prueba: recibir un mensaje del bot.
Salida esperada: el texto se guarda en la base de datos, así como el usuario al que se ha enviado y el identificador del chat en el que se ha mandado.
Resultado obtenido: éxito.

8.3. Pruebas de los comandos evaluate y voice

Estas pruebas han sido diseñadas para comprobar el corrector funcionamiento de los comandos que componen las funciones de evaluación colaborativa del bot.

Prueba 1 de los comandos evaluate y voice
Descripción de la prueba: llamar al comando voice.
Salida esperada: aparecen las preguntas pendientes de responder, si es que existe alguna.
Resultado obtenido: éxito.

Prueba 2 de los comandos evaluate y voice
Descripción de la prueba: elegir una de las preguntas pendientes de responder.
Salida esperada: aparece la pregunta y te pide una nota voz para responder a dicha pregunta.
Resultado obtenido: éxito.

Prueba 3 de los comandos evaluate y voice
Descripción de la prueba: mandar una nota de voz cuando se pide.
Salida esperada: se notifica si se ha subido correctamente y se sale del diálogo.
Resultado obtenido: éxito.

Prueba 4 de los comandos evaluate y voice
Descripción de la prueba: pulsar el botón salir.
Salida esperada: salir del diálogo.
Resultado obtenido: éxito.

8.4. Pruebas de las notificaciones

Prueba 1 de las notificaciones
Descripción de la prueba: añadir una nueva pregunta a la base de datos para responder con una nota de voz.
Salida esperada: se envía una mensaje a los alumnos correspondientes avisándoles que tienen una pregunta pendiente.
Resultado obtenido: éxito.

Prueba 2 de las notificaciones
Descripción de la prueba: responder a una pregunta con una nota de voz.
Salida esperada: se notifica a los alumnos correspondientes que deben evaluar la nota de voz de un compañero.
Resultado obtenido: éxito.

8.5. Pruebas de la web

Prueba 1 de la web
Descripción de la prueba: entrar en la página de visualización de las respuestas.
Salida esperada: aparecen todas las notas de voz(se pueden escuchar) y sus respectivas notas.
Resultado obtenido: éxito.

Prueba 2 de la web
Descripción de la prueba: añadir una pregunta.
Salida esperada: se añade correctamente la pregunta a la base de datos, con sus respectivos registros de evaluación y respuesta.
Resultado obtenido: éxito.

9. Conclusiones

En esta última sección, se va a realizar una comparativa entre la planificación que se estimo a comienzos del proyecto y la que finalmente ha sido, posibles lineas futuras para el bot, licencias y por último, una reflexión final.

9.1. Análisis entre planificación estimada y real

A continuación se exponen los cambios entre la planificación de tareas estimada en el alcance del proyecto (Sección 2.7) y la que finalmente ha sido.

9.1.1. Cambios respecto al alcance

Los bloques de tareas (Figura 2) se han mantenido iguales, por lo que se van a explicar las diferencias internas de cada bloque.

Organización

En este bloque, no se ha realizado ningún cambio en las tareas. Finalmente el planteamiento de la aplicación requirió menos tiempo y se efectuaron menos reuniones con el director del TFG de las que se preveían, por lo que el computo real de tiempo en este bloque ha sido menor del estimado.

Aprendizaje

Al igual que en bloque anterior no se ha realizado ningún cambió sobre las tareas y el tiempo real ha sido algo menor del estimado.

Análisis y Diseño

En este bloque, se ha añadido una nueva tarea, rediseñar los diagramas de la aplicación, que se ha realizado entre el desarrollo inicial y final del bot, los cuales se muestran en el bloque de implementación y desarrollo.

Rediseño de diagramas
Paquete de trabajo: Análisis y diseño. **Duración**: 14 horas.
Descripción: rediseñar los diagramas del bot aplicando los cambios.
Salidas/Entregables: Diagramas adaptados.
Recursos necesarios: Cacoo.com.

Al añadir esta tarea, el tiempo real ha sido mayor que el estimado para este bloque.

Implementación y desarrollo

El tiempo estimado ha sido un poco menor del que se ha necesitado en finalmente, aunque es una diferencia de apenas unas pocas horas.

Pruebas

Al final, no se han realizado las pruebas de automatización debido a falta de tiempo, por lo que el tiempo real en este bloque de tareas ha sido menor del estimado.

Documentación

Respecto al documento de objetivos del proyecto se han mantenido las tareas, y el tiempo real ha sido igual que el estimado.

Por otro lado, al bloque de la memoria del proyecto se le ha añadido la tarea de revisión del documento que no se puso, en un primer momento, en el alcance (Sección 2.7.6).

Revisión del documento
Paquete de trabajo: Documentación. **Duración**: 20 horas.
Descripción: revisar el documento de objetivos del proyecto y la memoria en busca de errores y solventarlos.
Salidas/Entregables: documento revisado y listo para su entrega.
Recursos necesarios: TeXstudio, Cacoo.com, LibreOffice, GanttProject, diccionarios online y la aplicación desarrollada.

Esta tarea ha supuesto un incremento en el numero de horas invertidas frente a las estimadas para este bloque.

9.1.2. Reevaluación económica

Tal y como se observa en la Figura **??**, se han trabajado unas pocas horas más de las estimadas en el alcance de este proyecto. Dado que se trata de un pequeño exceso de horas trabajadas, se ha decidido mantener la evaluación económica inicial (Sección 2.9) dando un total de **3139,14 €** por el completo desarrollo de la aplicación.

9.1.3. Rediseño de los diagramas de planificación

En las siguientes páginas aparece la estructura de descomposición de trabajo actualizada de este proyecto, así como el diagrama Gantt final de este proyecto.

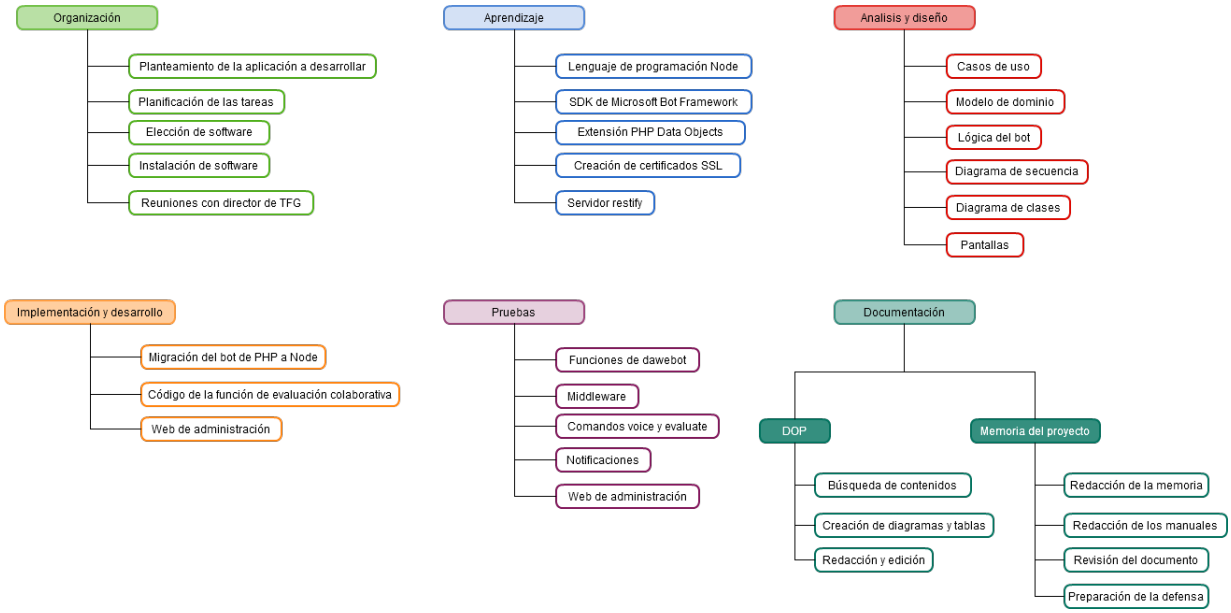

Figura 26: Diagrama final EDT por tareas

Nº de tarea	Nombre de la tarea	Predecesoras	Duración(horas)
0	**Bot para evaluación colaborativa de ejercicios orales**		**373**
1	**Organización**		**41**
2	Planteamiento de la aplicación a desarrollar		1
3	Planificación de las tareas	2	5
4	Elección de software	3	5
5	Instalación de software	4	5
6	Reuniones con el directo de TFG	2	25
7	**Aprendizaje**		**46**
8	Node.js	5	15
9	SDK de Microsoft Bot Framework	5,8,12	20
10	Extensión PHP Data Objects	5	5
11	Creación de certificados SSL	5	3
12	Servidor Restify	5	3
13	**Análisis y diseño**		**46**
14	Casos de uso	8,9	6
15	Modelo de dominio	14	6
16	Lógica del bot	15	6
17	Diagrama de secuencia	8,9,17	16
18	Diagrama de clases	8,9,16	6
19	Pantallas	8,9,10,18	6

Figura 27: Comparativa horas estimadas y reales (1)

20	**Implementación y desarrollo**		**125**
21	Migración del bot de PHP a Node.js	5,9,15	45
22	Código de la función de evaluación colaborativa	21	45
23	Web de administración	10	40
24	**Pruebas**		**42**
25	Funciones de dawebot	21	10
26	Middleware	21,22	3
27	Comandos voice y evaluate	22	10
28	Notificaciones	22	3
29	Web de administración	23	16
30	**Documentación**		**111**
31	*DOP*		*36*
32	Búsqueda de contenidos	3	6
33	Creación de diagramas y tablas	32	10
34	Redacción y edición	33	20
35	*Memoria del proyecto*		*75*
36	Redacción de la memoria	22,27,34	30
37	Redacción de los manuales	36	15
38	Revisión del documento	36	3
39	Preparación de la defensa	37	30

Figura 28: Comparativa horas estimadas y reales (2)

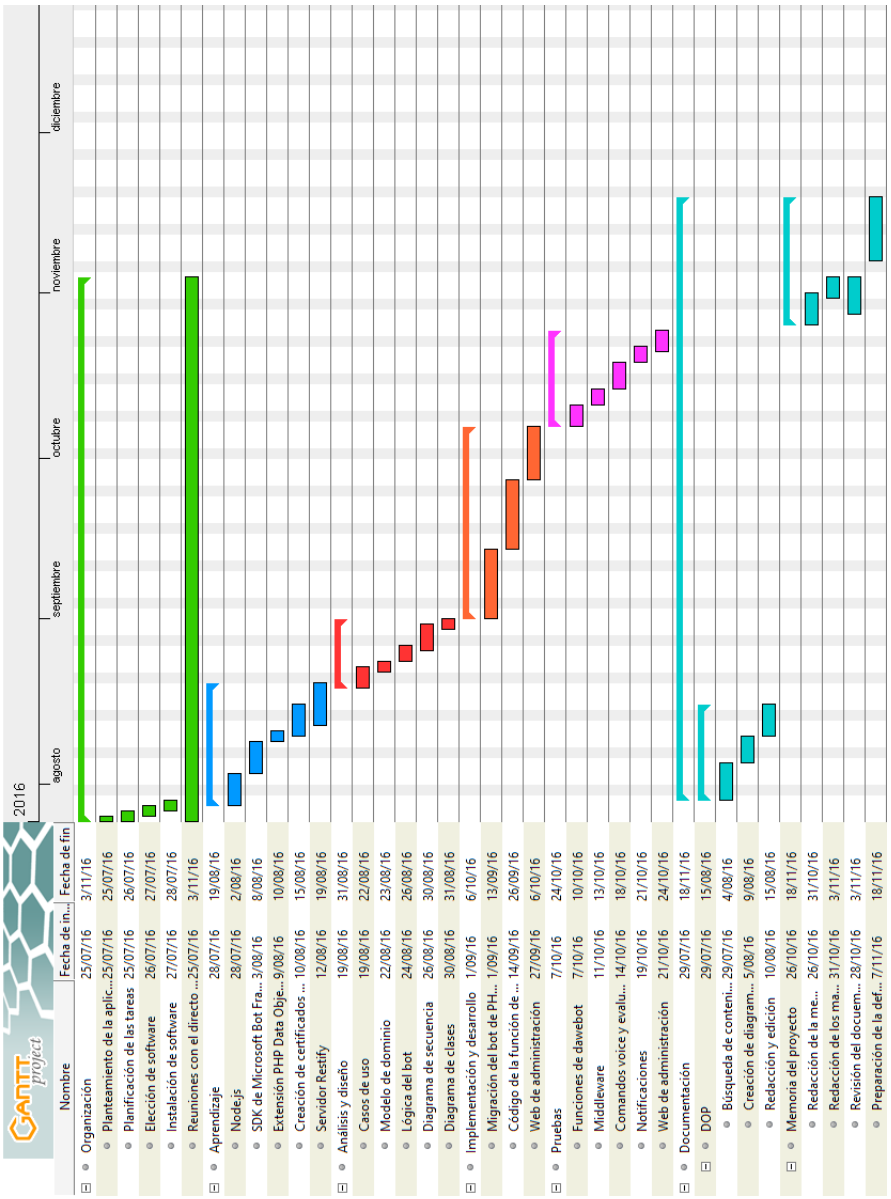

Figura 29: Diagrama Gantt final

9.2. Lineas futuras para el bot

En este apartado se van a reflejar las ideas surgidas para mejorar el bot en un futuro.

- **Nuevos comandos**: en un futuro se podrán añadir distintos tipos de comandos con formas completamente distintas de aprender. Usando cualquiera las funciones que nos permite un smartphone: cámara, gps, bluetooth, etc... Las posibilidades aquí son cuasi infinitas.
- **Web de administración**: la web de administración hasta el momento tan sólo tiene un par de funciones muy básicas. Se pueden poner estadísticas y gráficas por alumno, por clase, temas e incluso asignaturas
- **Traducción a varios idiomas**: gracias al sistema con el que está hecho se podría implementar la característica de que el alumno pueda elegir el idioma que prefiera con unos pequeños cambios.
- **Multiplataforma**: de momento solo se ha probado si funciona correctamente en Telegram. El SDK permite la multiplataforma por lo que se podría ampliar sin demasiado esfuerzo las plataformas funcionales

9.3. Licencias

En este apartado se verán, por un lado, las licencias que tienen aquellos recursos que se han utilizado para desarrollar el bot, y por el otro, el tipo de licencia que se le ha aplicado al bot.

9.3.1. Recursos utilizados

Los textos de las preguntas y las respuestas son propiedad del director del proyecto. Todos los demás recursos utilizados en el bot son de mi propiedad.

9.3.2. Bot

La licencia sobre los derechos de uso que se ha establecido para el bot es de GPLv3. `https://github.com/Hanoko/bot`

9.4. Reflexión personal

Para terminar con esta memoria de mi Trabajo de Fin de Grado me gustaría dedicar unas palabras de como ha sido mi experiencia universitaria a lo largo de estos años. Aún me acuerdo de mi primer día, llegué algo pronto ya que no había estado nunca allí. En realidad nadie había estado antes, era un nuevo edificio terminado apenas hacía un mes. Todo el mundo estaba perdido buscando su clase. Los primeros días algunos alumnos se sentaban en el suelo porque no había suficiente sillas en las aulas. La universidad parecía un desastre.

Tras un tiempo, uno se iba acostumbrando a la vida de universitario. Muchos trabajos, nuevos compañeros, menos horas de clase que en el colegio y las dos cosas que más he disfrutado en la universidad: estudiar lo que más me gusta y la libertad para decidir.

Pero todo ello ha llegado a su fin, comienza una nueva fase en mi vida. Por eso me gustaría agradecer a todo el personal de la UPV que ha conseguido que mi experiencia en la universidad haya sido posible.

M gustaría darle las gracias especialmente a mi director de proyecto Juanan Pereira, primero por proponerme este proyecto y segundo por apoyarme y guiarme en todo momento durante el desarrollo de este mismo.

Respecto a los bots dirigidos a la educación, veo un auténtico mundo de posibilidades. Ya es hora de que las nuevas tecnologías entren en las aulas.

Para concluir esta memoria, me gustaría agradecer a todas aquellas personas que han dedicado una parte de su tiempo a la lectura de este documento.

Bibliografía

[1] Documentación proporcionada por Microsoft - Disponible en https://docs.botframework.com/en-us/node/builder/overview/.

[2] nodeJs callbacks simple example. Disponible en StackOverflow: http://stackoverflow.com/questions/19739755/nodejs-callbacks-simple-example.

[3] Callback after all asynchronous forEach callbacks are completed. Disponible en StackOverflow: http://stackoverflow.com/questions/18983138/callback-after-all-asynchronous-foreach-callbacks-are-completed.

[4] I need a Nodejs scheduler that allows for tasks at different intervals [closed]. Disponible en StackOverflow: http://stackoverflow.com/questions/20499225/i-need-a-nodejs-scheduler-that-allows-for-tasks-at-different-intervals.

[5] Juanan Pereira. 2016. Leveraging chatbots to improve self - guided learning through conversational quizzes. TEEM'16 Proceedings. DOI: http://dx.doi.org/10.1145/3012430.3012625.

[6] Some behaviours with prompt.choice - Github. Disponible en: https://github.com/Microsoft/BotBuilder/issues/1299.

[7] [Telegram] - Prompt.choice button not working - Github. Disponible en: https://github.com/Microsoft/BotBuilder/issues/1042.

[8] Documentación de Node.js. Disponible en: https://nodejs.org/api/modules.html.

[9] Ministerio de Empleo y Seguridad Social - Boletín Oficial de Estado - Miércoles 10 Febrero 2016. Disponible en: https://www.boe.es/boe/dias/2016/02/10/pdfs/BOE-A-2016-1290.pdf.

A. Anexo: Manual de instalación

Este es el manual de instalación del bot.

A.1. Requisitos previos

En la siguiente lista aparece lo necesario para poder configurar correctamente el bot:

- **Servidor web Apache**: se debe disponer de un servidor web.
- **Base de datos MySQL**: base de datos MySQL instalada.
- **PHP**: debe estar instalado PHP junto con el paquete para usarlo con MySQL.
- **npm**: es necesario para instalar los módulos del bot.
- **nodejs**: es necesario para el funcionamiento del bot.
- **letsencrypt**: el servidor Restify del bot necesita una conexión segura https, y se aprovecha el certificado para la web de administración

A.2. Base de datos

Se importará el archivo ”bot.sql” en la base de datos MySQl. Este contiene la estructura de la base de datos.

A.3. Bot

Primero debes registrar tu bot en `https://dev.botframework.com` y seguir las instrucciones proporcionados por Microsoft. Una vez creado el bot, añade Telegram como canal y vuelve a seguir las instrucciones de Microsoft.

Para configurar el bot hay que copiar el contenido de la carpeta ”dawebot” a la ruta que desee. Una vez copiada, vaya hasta la carpeta y ejecute el comando ”npm install”. Este instalará todos los módulos necesarios para el correcto funcionamiento del bot. Una vez hecho esto, modifica el archivo ”settings.js” con los valores correctos.

Luego es necesario añadir el servidor restify como servicio de Ubuntu para que este se inicie por si solo cada vez que se reinicia el servidor. Para ello copia el archivo 'restify.service" (modificando las variables "ExecStart", "MICROSOFT_APP_ID" y "MICROSOFT_APP_PASSWORD") a la carpeta "/etc/systemd/system" y ejecuta:

"sudo systemctl enable /etc/systemd/system/restify.service"

"sudo systemctl start restify.service".

A.4. Web de administración

Para configurar la web de administración se debe copiar el contenido de la carpeta "daweb" a la ruta pública del servidor Apache, se modifica el archivo "db.php" con los datos correctos y ya está operativa.