Universidad del País Vasco Euskal Herriko Unibertsitatea

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

INDUSTRIA INGENIARITZA TEKNIKOKO ATALA
SECCIÓN INGENIERÍA TÉCNICA INDUSTRIAL

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2016 / 2017

TÍTULO DEL TFG

@myqueridobot, un bot educativo con preguntas de respuesta verbal

nombre del documento

Memoria del proyecto

DATOS DE LA ALUMNA O DEL ALUMNO	DATOS DEL DIRECTOR O DE LA DIRECTORA
NOMBRE Mikel	NOMBRE Juan Antonio
APELLIDOS Albóniga Vázquez	APELLIDOS Pereira Varela
	DEPARTAMENTO Lenguajes y Sistemas Informáticos
FDO.:	FDO.:
FECHA: 9-06-2017	FECHA: 9-06-2017

Anexo II

0. Resumen

Durante este proyecto se han desarrollado dos herramientas que permiten a los profesores ayudar a sus alumnos a desarrollar la competencia comunicativa.

La primera herramienta consiste en un *bot* implementado para la aplicación Telegram. Gracias a este *bot* y a los comandos que lo componen, los estudiantes pueden realizar *tests* sobre el temario de las asignaturas y ver sus estadísticas. Además, se les pueden plantear preguntas a las que tendrán que responder mediante grabaciones de voz, que posteriormente serán evaluadas mediante coevaluación entre alumnos.

La segunda herramienta creada, es el panel de administración del *bot*. Con él, los profesores podrán administrar usuarios, preguntas de voz o incluso consultar gráficos para observar el progreso de los estudiantes.

El *bot* ha sido creado a partir de la librería php-telegram-bot en PHP. En cuanto al panel de administración, está formado por ficheros en PHP, JavaScript y HTML. Las dos herramientas comparten una base de datos MySQL.

Índice

Índice de figuras

Índice de tablas

1. Definiciones y vocabulario técnico

Antes de comenzar con la introducción y el desarrollo del proyecto, es importante conocer algunos de los vocablos que se usan durante este documento. Para poder entender lo que se explica, es necesario conocer el significado de los siguientes tecnicismos:

- ***Bot***: Es la palabra robot acortada. Se refiere a un tipo de programa informático (*software*) autónomo que es capaz de llevar a cabo tareas concretas e imitar el comportamiento humano . Para mayor información: https://www.significados.com/bots/.
- ***Chatbot* o *bot* conversacional**: Es un programa (*software*) que simula mantener una conversación con una persona al responder automáticamente a las entradas realizadas por el usuario.
- **Telegram**: Es una aplicación de mensajería móvil y de escritorio basada en la nube, centrada en la seguridad y la velocidad. El enlace a su página web oficial es `https://telegram.org`.
- **PHP o *Hypertext Preprocessor***: Es un lenguaje de código abierto muy popular. Es especialmente adecuado para el desarrollo web y puede ser incrustado en HTML. Su página web oficial es: `http://php.net`.
- ***Feedback* o retroalimentación**: Es un método de control de sistemas que implica una respuesta o una reacción a una acción realizada. Para obtener más información: `https://www.significados.com/feedback/`.
- ***Warning* o advertencia**: En el ámbito técnico, las advertencias indican a los desarrolladores el mal uso de algún tipo de recurso sin que este llegue a generar un error.
- ***Log* o registro**: Son los archivos de texto utilizados normalmente para registrar accesos, incidencias, *warnings* o errores.
- **Lenguaje de programación interpretado**: Es un tipo de lenguaje de programación que necesita un programa informático (interprete) para ser analizado y ejecutado. Entre los más conocidos están Java, Python o Ruby.

- ***Front end***: Es la parte del *software* que interactúa con los usuarios.
- **JSON o *JavaScript Object Notation***: Es un formato de texto ligero utilizado para el intercambio de datos.
- **HTTP o *Hypertext Transfer Protocol***: Es el protocolo de comunicación que permite las transferencias de información en la *World Wide Web*.
- **Certificado SSL**: Es un documento electrónico que se une a una clave de identidad. Esta clave está formada por una dirección de correo electrónico, la empresa a la que pertenece y la ubicación. Estos certificados son la pieza clave de un proceso de autentificación digital.
- ***Callback***: En programación se denomina de esta manera a una retrollamada o devolución de llamada. Cuando una función termina de ejecutarse completamente, puede llamar a otra para que siga con la ejecución de la aplicación. Es muy usado en lenguajes de programación asíncronos como Node.js.
- ***WebHook***: Es un método de alteración de una página o aplicación web mediante *callbacks*.
- **REST o *Representational State Transfer***: Es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP. Permite la creación de servicios o aplicaciones que cualquier dispositivo que utilice el protocolo HTTP pueda utilizar.
- **Diagrama EDT o Estructura de Descomposición de trabajo**: Es un diagrama que consiste en la descomposición, de manera jerárquica, de las tareas de un trabajo o proyecto.
- **Migración de código**: Consiste en escribir el código programado en un lenguaje de programación diferente al que ya se tiene.
- **Conversación**: En los comandos de los *bots*, una conversación implica que el *bot* va a necesitar más de una respuesta del usuario para terminar la ejecución de un comando.
- ***Deadline***: Final del plazo otorgado para responder a una pregunta de voz.
- ***Big data***: Es un concepto que hace referencia a conjuntos de datos tan grandes, que aplicaciones informática tradicionales de procesamiento de datos no son suficientes para tratar con ellos.

- **AJAX o *Asynchronous JavaScript And XML***: Es una técnica de desarrollo web que permite la interacción asíncrona de JavaScript entre el cliente y el servidor.
- **SSH o *Secure SHell***: Es un protocolo que permite el control absoluto de una maquina remota mediante comandos. Se usa el puerto TCP/22 por defecto.

2. Introducción

Los métodos de enseñanza han evolucionado a lo largo de las últimas décadas. Las nuevas tecnologías han ayudado a esta evolución, pero aún quedan algunas competencias que, siendo imprescindibles a nivel universitario, son difíciles de desarrollar por los alumnos. Este es el caso de la competencia comunicativa. Esta aptitud engloba tanto la comunicación escrita como la oral, en la cual, los estudiantes, sobre todo del ámbito ingenieril, suelen tener dificultades. Los alumnos de ingeniería son uno de los ejemplos más claros, ya que, a la larga, tendrán que explicar a personas no técnicas algunos aspectos de lo que han estudiado.

Hay profesores que buscan nuevos métodos de enseñanza para que los alumnos aprendan a exponer sus proyectos, ideas, soluciones... de manera que una persona sin conocimiento de la materia pueda entender en que consisten. El uso de exposiciones orales a la hora de presentar proyectos es una de las metodologías más antiguas y, a su vez, una de las mas utilizadas, pero, ¿es este método eficaz?¿Hay algún otro con el que se puedan obtener mejores resultados? Hay docentes que tratan de encontrar las respuestas a estas preguntas probando nuevas técnicas de evaluación o utilizando nuevas herramientas. Una de esas nuevas herramientas que se están poniendo a prueba, son los *bots*.

José María Sánchez, un periodista madrileño, comenta en uno de sus artículos del periódico ABC [Sánchez, 2017] que, durante el año 2016, el 51,8 % del tráfico red mundial fue generado por *bots*. Este dato no es raro, ya que muchas empresas utilizan *bots*, ya sean como vendedores de los productos o servicios que ofrecen o como sistemas de atención al cliente, comunicándose entre ellos y produciendo tráfico en la red. Su disponibilidad las 24 horas del día los ayuda a ser los generadores de tráfico de red número uno. Viendo que estas herramientas tienen tanto impacto y un gran éxito entre grandes empresas, ¿por qué no probar a utilizarlas como medio de aprendizaje?

[Heller et al., 2005] documenta la prueba realizada con un *chatbot* que simula una conversación con Sigmund Freud. Gracias a él, alumnos de psicología de la universidad de Athabasca en Alberta (Canadá), adquieran conocimientos sobre las teorías del psicoanalista de forma entretenida. Después de hablar durante 10 minutos, a los 53 estudiantes del experimento se les realizó un cuestionario. En él, el 68 % de los encuestados respondieron que les gustaría repetir la experiencia. De hecho, uno de los estudiantes dijo lo siguiente: "*It was pretty cool the way it felt like I was actually interacting with Freud... he's deceased though, yeah, but the picture, the fast answers...*

made me pay attention to the answers alot more than if I had been simply reading a text written by someone else. Plus it was cool to feel like I could voice my own opinion with the most well-known psychoanalyst of all time". Después de leer esto, el uso de un *bot* educativo parece una buena idea.

Ahora lo único que queda por resolver es, ¿cómo adaptar un *bot* para que los usuarios puedan desarrollar su capacidad de expresión?

El objetivo de este Trabajo de Fin de Grado (TFG) es crear un *chatbot* en Telegram, con el que los estudiantes sean capaces de mejorar su capacidad de expresión.

2.1. Origen del proyecto

El proyecto original fue idea de Juanan Pereira, docente en la Universidad del País Vasco (UPV/EHU). El profesor Pereira imparte asignaturas *online*, basadas en el aprendizaje mediante ejercicios, trabajos, videos de YouTube y *tests* sobre el temario.

Juanan estaba decepcionado con e-Gela[1], la plataforma principal de la universidad que los alumnos usan para hacer los *tests*. Esto se debe a que e-Gela no funciona muy bien en los navegadores de dispositivos con una pantalla reducida como son los teléfonos móviles. Al tratar de encontrar una solución a su disconformidad, Telegram anunció el comienzo del uso de *bots* en su sistema de mensajería instantánea. Viendo en ello una oportunidad, y comenzó a desarrollar un *bot* con el que se pudieran realizar los *tests* de las asignaturas que imparte.

Después de terminar de desarrollar el apartado de los *tests*, se dio cuenta de que a este *bot* se le pueden añadir muchísimas funcionalidades, que pueden ayudar a los estudiantes a desarrollar sus capacidades. Al ver que los *bots* aceptan grabaciones entre los mensajes que pueden recibir, se planteó utilizar esta funcionalidad para desarrollar la capacidad comunicativa de sus alumnos. Además se le ocurrió que podía utilizar esta idea para simular preguntas que se le podrían llegar a hacer a sus alumnos en una entrevista de trabajo. Al ser un proyecto con un gran contenido y al saber que se probaría en sus clases lo propuso como TFG a algunos de sus alumnos.

Cuando el profesor Pereira le propuso al alumno Pablo Uribe Bastero este proyecto, Pablo quedó entusiasmado con la idea de que lo que realizara del proyecto se probaría en clases reales durante su realización. Desarrolló el proyecto en un lenguaje de programación diferente al usado hasta el momento. Planteando el proyecto de manera que el *bot* pudiera integrarse en Telegram y en otros tipos de sistemas de mensajería como Skype, Facebook Messenger, etc. Pero esto acarrea algunos problemas que se comentarán durante este documento, y que fueron una de las razones por las que se decidió desechar lo que Pablo realizó.

Aun así Pablo realizó un gran trabajo desarrollando en su *bot* todas las funcionalidades con las que finalizó su TFG.

[1]Web e-Gela: `https://egela1617.ehu.eus`.

2.2. Descripción y situación del proyecto

Este TFG es la continuación y mejora del proyecto "Bot para evaluación colaborativa de ejercicios orales", realizado por Pablo Uribe Bastero durante el año 2016. Como ya se ha comentado anteriormente, el proyecto consiste en el desarrollo de un *bot* con el que los alumnos puedan aprender y lleguen a mejorar su capacidad de expresión. Para ello el profesor propondrá preguntas relacionadas con el temario de las asignaturas que imparta. Los estudiantes responderán a esas preguntas mediante grabaciones de voz, las cuales, serán evaluadas por coevaluación entre los alumnos. Como apoyo al material, el *bot* también proporcionará *tests* a los alumnos sobre el temario de las asignaturas, para que estudiar no parezca una tarea tan pesada.

El *bot* realizado para este proyecto es un *chatbot* de Telegram llamado "@myqueridobot". Su base ha sido desarrollada utilizando la librería "php-telegram-bot"[2]. Como casi todos los *bots* de Telegram, @myqueridobot consta de comandos o palabras clave con los que se ejecutan las acciones que el usuario desee.

Hasta el momento existen diez comandos propios de @myqueridobot, que, unidos a los comandos genéricos de todos los *bots*, forman un conjunto de diecisiete instrucciones ejecutables, de los cuales sólo seis pueden ser ejecutados por los administradores del *bot*.

Además de los comandos, se ha realizado una página web que funciona como panel de administración. En ella, el profesor puede crear, modificar o eliminar grupos de alumnos y preguntas de voz. También contiene un apartado de opciones que el profesor puede ajustar a su gusto. Además, en esta web, se generan gráficos que pueden ser interesantes para el profesor.

[2]Dirección web de la librería: `https://github.com/php-telegram-bot/core`.

2.3. Motivaciones para la elección del proyecto

Hay cinco motivos principales que me han llevado a la elección de este proyecto.

Curiosidad

Nunca antes había planteado el funcionamiento de un *bot*. Cuando Juanan expuso en una presentación temas de los proyectos que ofrecía, este tema llamó especialmente mi atención. Me empecé a plantear las preguntas ¿cómo funciona un *bot* internamente?¿Qué se necesita saber para poder hacerlo? Por curiosidad empecé a investigar por mi cuenta en qué consiste su desarrollo. Al gustarme lo que vi, decidí que desarrollaría un *bot* en el TFG.

Me pareció un reto interesante

Cuando me dirigí a Juanan para preguntarle sobre los proyectos disponibles sobre *bots*, él me propuso este. Me comentó un poco cómo funcionan los *bots* en Telegram y me mostró lo que se había hecho anteriormente en el proyecto. El hecho de coger un proyecto que alguien había empezado, comprenderlo y mejorarlo, y además de eso, empezar sin saber casi nada al respecto, me pareció un gran reto a superar. Esto me motivó a escoger el proyecto, aunque me decepcioné un poco al tener que desechar todo lo referente al código que se había hecho hasta el momento.

Aprendizaje continuo

Durante el grado solo había utilizado algunas de las herramientas que utilizaría en el proyecto en una ocasión y, como son las más utilizadas para el desarrollo de páginas web, el proyecto me pareció útil para aprender a usarlas. Durante todo el proyecto me he dedicado a investigar y a aprender a usar nuevas herramientas, lo cual me ha dejado satisfecho en cuanto a esta motivación.

El problema a solventar me afecta

Soy una persona a la cual le cuesta expresarse. El hecho de que el proyecto ayude a mejorar esta capacidad me pareció muy interesante, ya que no solo me ayudaría a mí, sino que podría ser útil a otros estudiantes con el mismo problema. Además, el hecho de que siguiera con este problema a comienzos

del cuarto curso del grado me pareció algo grave, así que vi este proyecto como una forma de solventarlo.

Se ha puesto a prueba en clases reales

Todo proyecto tiene el fin de llegar a ser utilizado, y el hecho de que este se haya puesto en marcha en una clase, es, sin lugar a dudas, lo que más me motivó a la hora de escogerlo. Desde el mismo momento en el que se tuvo una versión completa del código, se puso a prueba en una de las asignaturas que yo mismo estaba cursando durante ese año. Lo cual fue algo que me pareció fascinante, ya que no solo he jugado el rol de desarrollador del proyecto, sino que también he sido parte del primer grupo en probarlo como un producto final. He desarrollado un proyecto que podría llegar a comercializarse, y del cual, he tenido experiencia como cliente. Es por ello que, en mi opinión, esta razón fue lo que más me motivó a escogerlo como TFG.

3. Planteamiento inicial

En esta sección se presentan los objetivos, las herramientas utilizadas, el alcance, la planificación temporal y la evaluación económica y de riesgos que se han realizado para este proyecto. También se explica, brevemente, la arquitectura que se utiliza para una mejor comprensión del funcionamiento de un *bot* de Telegram.

3.1. Objetivos

Los objetivos del proyecto se han dividido en tres secciones. De esta manera son más fáciles de identificar y de comprobar su cumplimiento. Además, al ser la continuación de un proyecto del que se desechó lo realizado, en este se han recogido algunos de los objetivos de su antecesor.

Bot de Telegram

En esta sección se muestran los objetivos principales que ha de cumplir el *bot*. Su completo desarrollo se ha basado en los siguientes objetivos:

- **Funcionalidades completas y sin fallos**: Al ser un *chatbot* que tiene como base una serie de comandos, esos comandos han de estar completos. Han de realizarse todos los comandos necesarios para que los usuarios puedan interactuar con el *bot*, recibiendo siempre el *feedback* necesario. No se permite que generen ningún tipo de error o *warning* en los *logs*, ya que esto puede afectar al desarrollo de este proyecto, pero también puede llegar a repercutir en otros.

- **Comandos claros y fáciles de recordar**: Las palabras clave que ejecutan las acciones del *bot* han de ser intuitivas, cortas y fáciles de recordar. Los vocablos tienen que relacionarse fácilmente con lo que ejecutan, sin necesidad de leer su descripción.

- **Ejecución de comandos sencilla**: Cuando el usuario ejecute un comando, en todo momento debe saber lo que tiene que hacer para seguir con el proceso de lo que el *bot* ejecute. Si el usuario ha de pulsar un botón, escribir algún identificador o enviar una grabación para responder, en todo momento recibirá una explicación de lo que se le pide. El *bot* ha de ser claro a la hora de explicar qué es lo que necesita para seguir la ejecución de sus comandos.

- **Seguridad**: Este apartado no se refiere al cifrado de los mensajes, ya que Telegram lo hace por defecto. Este objetivo se refiere a que ningún extraño pueda utilizar los comandos del *bot*. Solo tendrán acceso a ellos los estudiantes que se registren previamente.

Panel de administración

En este apartado se muestran los objetivos que ha de cumplir la página web que contiene el panel de administración del *bot*. Se han planteado los siguientes objetivos:

- **Utilidad y facilidad de uso**: Tiene que tener todos los apartados que un profesor requiera para la correcta administración del *bot*. Todas la funcionalidades del panel de administración han de ser simples y sencillas.
- **Información siempre disponible**: Siempre se le mostrará información necesaria. Ya sea mediante paneles, tablas o mensajes, el usuario de la web siempre recibirá *feedback* o la información que SE necesite en cada momento. También se harán comprobaciones como preguntar si realmente se desea eliminar algún elemento o si está seguro de querer cambiar las opciones del *bot*.
- **Uso de gráficos**: Tendrá un apartado en el que el profesor podrá consultar gráficos, que mostrarán información sobre el progreso de los alumnos y sobre el uso del *bot*. De esta manera el profesor podrá tener cierto control sobre si el *bot* está siendo usado o no y, en algunos casos, realizar estudios sobre cómo afecta esto a las calificaciones de los alumnos.

Generales y de documentación

En este apartado se muestran los objetivos comunes para los dos apartados anteriores (el *bot* y el panel de administración). También se presentarán los objetivos que se quieren alcanzar con la documentación y la memoria del proyecto.

- **Documentación bien definida**: Ya que este proyecto va a ser utilizado en clases reales, la documentación ha de estar bien definida y en apartados específicos. De esta manera, si surge algún problema o error, siempre se podrá consultar este documento o los comentarios en

el código para poder encontrar fácilmente lo que ha fallado. Además, esto hará más fácil las modificaciones y actualizaciones que se quieran realizar en el futuro.

- **Preparado para modificaciones y nuevas funcionalidades**: El *bot* y su panel de administración deben ser fáciles de modificar y se tiene que poder añadir nuevas funcionalidades sin problemas. Además, tiene que estar siempre preparado para las nuevas actualizaciones que se crean de la librería utilizada para el desarrollo del *bot* (php-telegram-bot).

3.2. Herramientas utilizadas

Aquí se detallan las diferentes herramientas y librerías que se utilizarán para el desarrollo del proyecto. También se explica, de manera resumida, en qué partes del proyecto se plantea usar cada una de las herramientas descritas.

- **Apache 2**: Es un servidor HTTP de código abierto. Gracias a su integración con PHP, se pueda acceder al *bot* y al su panel de control vía Internet. Su página oficial es: `https://httpd.apache.org/`.

- **Bitbucket**: Es un sistema de control de versiones que consta de repositorios privados gratuitos. Desde este sistema se puede clonar lo que exista en un repositorio para poder modificarlo en local y después subir el nuevo código a ese repositorio. Es una herramienta muy útil para proyectos en los que participa mucha gente, ya que en todo momento todos los desarrolladores pueden acceder a un código actualizado con una simple instrucción. El código del proyecto se guardará en un repositorio privado de este sistema. Su página oficial es: `https://bitbucket.org/product`.

- **Cacoo**: Es una aplicación para la creación de diagramas compartibles. Permite realizar diagramas, prototipos e incluso crear planos de edificios. El poder modificar un diagrama a tiempo real y permitir la coedición la hacen una herramienta muy útil. Con esta aplicación se realizarán los prototipos del panel de administración. Su página oficial es: `https://cacoo.com/lang/es/home`

- **Composer**: Es una herramienta para la gestión de dependencias en PHP. Gracias a ella se pueden instalar y actualizar las librerías que un proyecto necesite con solo su declaración en un archivo JSON. Será utilizado para instalar y mantener actualizadas todas las librerías usadas por el *bot*. Su página oficial es: `https://getcomposer.org/`.

- **Dia**: Es un programa gratuito para la creación de diagramas. Se usará para realizar todos los diagramas del proyecto referentes al desarrollo de código. Su página oficial es: `http://dia-installer.de/index.html.es`.

- **GanttProject**: Es un programa para la creación de diagramas Gantt de manera sencilla. Con solo crear el diagrama Gantt, este programa es capaz de crear un diagrama PERT del proyecto que se esté realizando. Los diagramas Gantt de este proyecto serán realizados con GanttProject. Su página oficial es: `http://www.ganttproject.biz/`.

- **Git**: Es un sistema de gestión de versiones de código abierto. Está integrado en sistemas como GitHub, Bitbucket o GitLab y se puede usar en casi todo tipo de plataformas. Es el programa que usará para compartir las diferentes versiones del código con otros potenciales desarrolladores. Su página oficial es: `https://git-scm.com/`.

- **Google Chrome**: Es un navegador web desarrollado por Google, compilado con base en varios componentes e infraestructuras de desarrollo de aplicaciones de código abierto. Es el navegador en el que será desarrollado el panel de administración del *bot*.

- **HTML**: También conocido como *HyperText Markup Language*, es un lenguaje de programación basado en etiquetas. Este lenguaje es ejecutado por el navegador, que es el encargado de interpretar el código y mostrar los elementos de las páginas web. El panel de administración del *bot* utilizará este lenguaje para mostrar los componentes en la web.

- **JavaScript**: Es un lenguaje de programación interpretado que es principalmente usado en páginas web dinámicas del lado del cliente. Gran parte del *front end* del panel de administración del *bot* será implementado en JavaScript.

- **MySQL**: Es el sistema de gestión y administración de bases de datos de código abierto más usado en el mundo. La base de datos del proyecto se creará con este sistema.

- **MySQL Workbench**: Es una herramienta visual para trabajar con MySQL. Tiene funciones de desarrollo, administración de usuarios y da la posibilidad de hacer copias de seguridad.
- **Overleaf**: Overleaf es una herramienta de escritura y publicación colaborativa *online* de LaTeX y Rich Text. Con ella, se puede ver en directo cómo se va modificando el fichero PDF resultante del código que se escribe. Además, ofrece control de versiones del documento. La memoria del proyecto será redactada en LaTeX con esta herramienta web. Su página oficial es: `https://www.overleaf.com/`.
- **PHP**: Es un lenguaje de código abierto que originalmente se diseñó para el desarrollo de páginas web dinámicas. Es la base de este proyecto, ya que el *bot* entero será desarrollado en este lenguaje y también se usará en el panel de administración.
- **php-telegram-bot**: Es una librería en PHP que se usa como base para el desarrollo de *bots* de Telegram. Su página oficial es: `https://github.com/php-telegram-bot/core`.
- **Plotly**: Es una plataforma *online* que permite la creación de gráficos interactivos. Es una aplicación muy usada en *business intelligence* y *data science* por tener la capacidad de mostrar grandes cantidades de datos en los gráficos que crea. En el panel de control del *bot* se usará Plotly en el apartado de gráficos. Su página oficial es: `https://plot.ly/`.
- **Putty**: Es un cliente SSH. Es una herramienta gratuita y que además tiene una versión *portable*. Esta herramienta será utilizada para tomar el control del servidor y realizar las acciones convenientes en cada momento. Ya sea para analizar los *logs* o para actualizar todo el código relacionado con proyecto. Su página oficial es: `http://www.putty.org/`.
- **Telegram**: Es una aplicación gratuita de mensajería móvil y de escritorio basada en la nube. Posee un sistema de *bots* junto a muchas funcionalidades interesantes como una nube de almacenamiento para cada usuario. El *bot* realizado para este proyecto será desarrollado para usarse en esta aplicación. Su página oficial es: `https://telegram.org/`.
- **Sublime Text 2**: Es un editor de texto preparado para la edición de código. Ofrece resaltado de sintaxis y soporte multiplataforma. Esta herramienta será utilizada para desarrollar todo el código, tanto del *bot* como del panel de control. Su página oficial es: `https://www.sublimetext.com/`.

3.3. Arquitectura

En este apartado, se explica cómo es la arquitectura de un *bot* de Telegram y el proceso que se lleva a cabo a la hora de llamar a un comando en uno de estos *bots*. También se muestra el funcionamiento que se plantea realizar para el panel de administración del *bot*.

Bot de Telegram

Para hacer funcionar un *bot* solo se necesita un servidor con Apache2, un certificado SSL, una base de datos MySQL, una de las librerías disponibles para el desarrollo de *bots* en Telegram y el interprete del lenguaje de programación de la librería. Como en este proyecto se usará PHP, el ejemplo de esta sección se hará con ese lenguaje de programación.

Antes de comenzar, en la Figura 1 se muestra un esquema que se seguirá para realizar la explicación de la arquitectura.

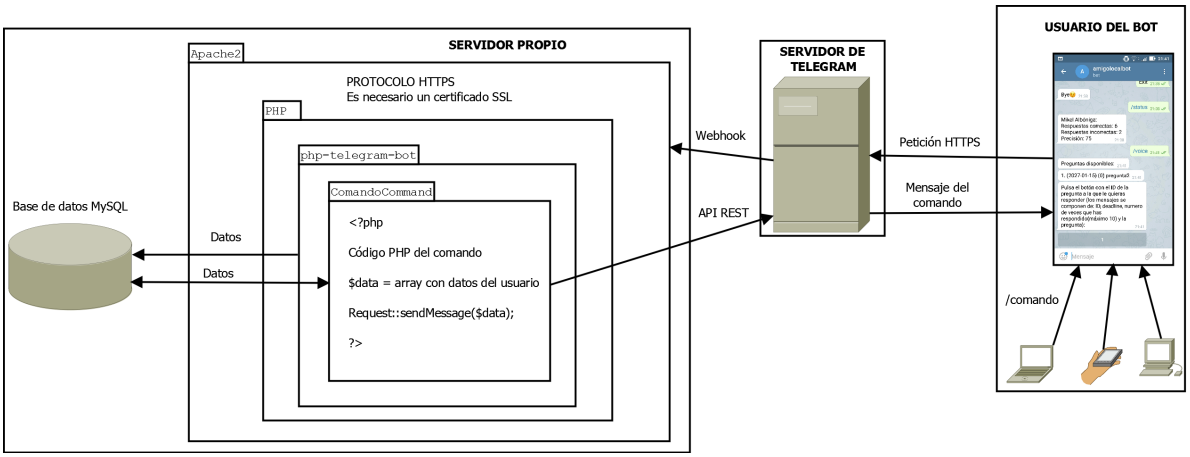


Figura 1: Esquema de la arquitectura de un *bot* de Telegram.

Cuando un usuario envía un comando a un *bot* de Telegram desde cualquier dispositivo, este es enviado a los servidores de Telegram. Al instante, Telegram realiza una petición HTTPS al *WebHook* del servidor que tenga la custodia de ese *bot*.

Todos los mensajes de Telegram son cifrados usando el protocolo HTTPS para las comunicaciones entre todos los dispositivos. Es por ello que el servidor necesita tener un certificado SSL que autentifique su identidad. Además, para añadir mayor seguridad, Telegram utiliza otro protocolo denominado

MTProto[3] para el encriptado interno de los mensajes.

Cuando la petición HTTPS llega al servidor, Apache2 la recoge y la delega al intérprete de PHP. Este detecta que la gestión de la petición pertenece a la librería php-telegram-bot. En este punto empieza la ejecución del código del comando, pero antes se guardan algunos datos del mensaje y del usuario en la base de datos.

Aunque existen comandos que no requieren el uso la base de datos durante su ejecución, antes de llamarlos, la librería guarda en varias tablas un registro del usuario, el *chat* y los mensajes que se le envían al *bot*.

Normalmente todos los comandos terminan con el envío de algún mensaje al usuario que inició la conversación. En la librería php-telegram-bot, se usa el método *sendMessage* de la clase *Request* para enviar el mensaje. Al llamar a este método, es necesario pasarle un parámetro con los datos del mensaje y el *chat* al que se va a enviar. Estos datos van encapsulados en una petición HTTPS que llega a los servidores de Telegram por medio de su *API REST*.

Por último, los servidores de Telegram se encargan de enviar la respuesta que recogieron de la petición HTTPS al usuario que ejecutó el comando.

Panel de administración del *bot*

En cuanto al panel de administración, solo es necesario tener un servidor con Apache2, la base de datos en MySQL y el intérprete de PHP. En la Figura 2 se puede apreciar el esquema del funcionamiento del panel de administración que se creará en este proyecto.

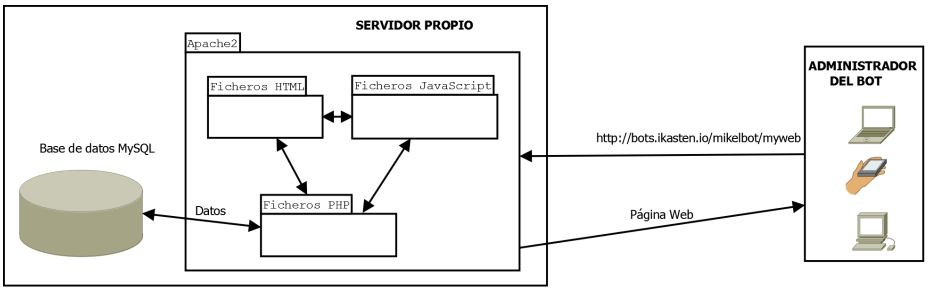


Figura 2: Esquema de la arquitectura del panel de administración.

[3]MTProto de Telegram: `https://core.telegram.org/mtproto`.

Cada vez que el administrador intenta acceder al panel, se realiza una petición HTTP a, en este caso, nuestro servidor. La petición es recogida por Apache2 y esta sistema delega el control al *script* HTML o PHP que se quiera mostrar en el navegador.

Existen tres tipos de ficheros que componen casi cualquier página web: ficheros JavaScript, ficheros PHP y ficheros HTML. Entre los tres tipos se pueden pasar información, pero solo los ficheros PHP tienen acceso a la base de datos. Estos pueden extraer información de la base de datos, introducirla, modificarla o eliminarla dependiendo de lo que se ejecute en el panel de administración.

Como la base de datos es compartida por el panel de administración y el *bot*, esto permite al administrador el control del *bot* desde la página web.

3.4. Alcance

Para definir el alcance de este proyecto se ha realizado un diagrama EDT. Se han identificado siete bloques principales: aprendizaje, organización, captura de requisitos, análisis y diseño, implementación y desarrollo, pruebas y documentación.

1. **Aprendizaje**: Bloque al que pertenecen las tareas relacionadas con el estudio de las diferentes herramientas usadas durante el proyecto.

2. **Organización**: Bloque que contiene las tareas necesarias para el buen desarrollo del proyecto.

3. **Captura de requisitos**: Bloque que agrupa las tareas que recogen los datos necesarios para reconocer las funcionalidades a realizar durante el desarrollo del proyecto.

4. **Análisis y diseño**: En este bloque se encuentran las tareas que se encargan de describir gráficamente el código del proyecto.

5. **Implementación y desarrollo**: Este bloque alberga todas las tareas que impliquen la escritura de código en cualquier tipo de lenguaje de programación.

6. **Pruebas**: Bloque que recoge las tareas con los métodos utilizados para la comprobación del correcto funcionamiento de las funcionalidades desarrolladas para el proyecto.

7. **Documentación**: A este último bloque pertenecen las tareas que requieran la redacción de información sobre el proyecto. Se realizará a lo largo de todo el proyecto.

Cada bloque del EDT es precursor al siguiente descrito en la lista, exceptuando el aprendizaje y la documentación. Las tareas de estos dos bloques se desarrollan en paralelo a las demás durante todo el proyecto.

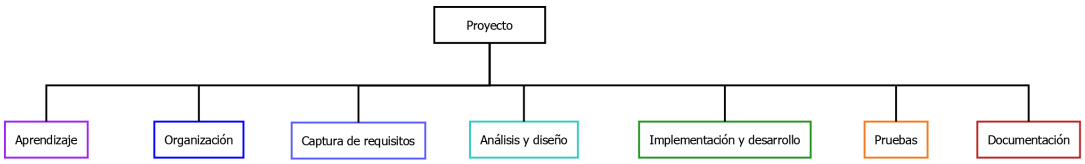


Figura 3: Diagrama EDT por bloques de nivel 1.

La Figura 3 muestra la división del proyecto en los bloques mencionados previamente.

A continuación se mostrarán detalladamente los datos sobre la planificación de cada uno de los bloques junto con las tablas y figuras correspondientes.

3.4.1. Aprendizaje

En este apartado se muestran los detalles de las tareas relacionadas con el aprendizaje del uso de los programas y librerías que se usarán en el proyecto.

La Figura 4 muestra el apartado del EDT en el que se pueden observar las tareas a realizar durante el aprendizaje. En las siguientes tablas se muestran los detalles de cada tarea.

Aprendizaje de PHP
Paquete de trabajo: Aprendizaje. **Duración estimada**: 30 horas.
Descripción: Estudio de la sintaxis y el uso del lenguaje de programación PHP.
Salidas/Entregables: N/A.
Recursos necesarios: Sublime Text 2, manual de PHP y PHP7.0.

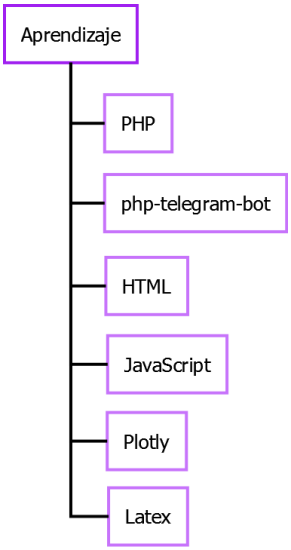


Figura 4: Diagrama EDT del bloque de aprendizaje.

Aprendizaje de php-telegram-bot
Paquete de trabajo: Aprendizaje. **Duración estimada**: 10 horas.
Descripción: Estudio del uso de la librería php-telegram-bot para la creación del nuevos comandos y entidades.
Salidas/Entregables: Comandos de prueba.
Recursos necesarios: Sublime Text 2, manual de PHP, PHP7.0 y apartados Wiki y problemas de la librería en GitHub.

Aprendizaje de HTML
Paquete de trabajo: Aprendizaje. **Duración estimada**: 5 horas.
Descripción: Estudio de la sintaxis y el uso del lenguaje de programación HTML.
Salidas/Entregables: N/A.
Recursos necesarios: Navegador Google Chrome y página web `https://www.w3schools.com/`.

Aprendizaje de JavaScript
Paquete de trabajo: Aprendizaje. **Duración estimada**: 15 horas.
Descripción: Estudio de la sintaxis y uso del lenguaje de programación JavaScript.
Salidas/Entregables: N/A.
Recursos necesarios: Navegador Google Chrome y página web `https://www.w3schools.com/`.

Aprendizaje de Plotly
Paquete de trabajo: Aprendizaje. **Duración estimada**: 5 horas.
Descripción: Aprender a usar la librería Plotly.js y a crear gráficos interactivos con ella. Los datos usados para esta tarea serán extraídos de una base de datos.
Salidas/Entregables: Gráfico de prueba.
Recursos necesarios: Página web `https://plot.ly/`.

Aprendizaje de LaTeX
Paquete de trabajo: Aprendizaje. **Duración estimada**: 30 horas.
Descripción: Aprendizaje del uso y sintaxis del lenguaje LaTeX para la redacción de textos, creación de tablas, inserción de imágenes, etc.
Salidas/Entregables: Documento de prueba en LaTeX.
Recursos necesarios: Página web `https://www.overleaf.com` y manuales de LaTeX.

3.4.2. Organización

Aquí se expondrán los detalles de todas las tareas que serán necesarias para el correcto desarrollo del proyecto.

La Figura 5 muestra la descomposición de las tareas del bloque de organización. A continuación se detallan las especificaciones de cada tarea y su duración estimada.

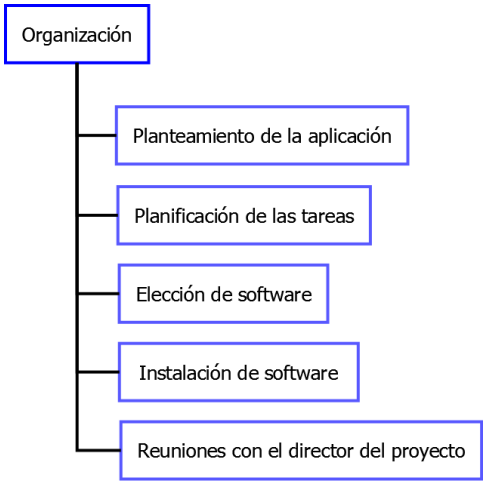


Figura 5: Diagrama EDT del bloque de organización.

Planteamiento de la aplicación
Paquete de trabajo: Organización. **Duración estimada**: 3 horas.
Descripción: En esta tarea se redactará un listado con las ideas de las nuevas funcionalidades que se puedan añadir al proyecto.
Salidas/Entregables: Documento con una lista de las ideas aplicables al proyecto.
Recursos necesarios: Papel y lápiz.

Planificación de las tareas
Paquete de trabajo: Organización. **Duración estimada**: 5 horas.
Descripción: Identificación, descripción y planificación temporal de todas las tareas que tendrán que realizarse en el proyecto.
Salidas/Entregables: Diagrama EDT con las tareas a realizar junto con un documento en el que se detalla cada una de las tareas.
Recursos necesarios: Papel, lápiz y Dia.

Elección de software
Paquete de trabajo: Organización. **Duración estimada**: 2 horas.
Descripción: Selección de las herramientas *software* que se usarán en el proyecto.
Salidas/Entregables: Documento con el listado de los nombres de las herramientas a usar.
Recursos necesarios: Papel y lápiz.

Instalación de software
Paquete de trabajo: Organización. **Duración estimada**: 5 horas.
Descripción: Instalación de las herramientas necesarias para el desarrollo del proyecto.
Salidas/Entregables: N/A.
Recursos necesarios: Ordenador con conexión a Internet.

Reuniones con el director del proyecto
Paquete de trabajo: Organización. **Duración estimada**: 25 horas.
Descripción: Reuniones que se realizarán durante el desarrollo del proyecto con el director del mismo. En las reuniones se tratarán temas como problemas, modificaciones en la planificación o funcionalidades del proyecto, con el fin de tener un control periódico.
Salidas/Entregables: Un documento con las soluciones, modificaciones o tareas pendientes de las que se habla durante la reunión.
Recursos necesarios: N/A.

3.4.3. Captura de requisitos

En este apartado se presenta la planificación de las tareas que recogen los requisitos a cumplir por las herramientas desarrolladas durante el proyecto.

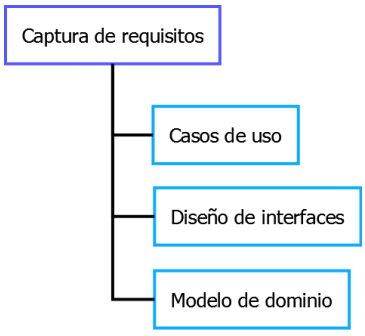


Figura 6: Diagrama EDT del bloque de captura de requisitos.

En la Figura 6 se muestra el desglose de tareas de esta sección. A continuación se muestran la información de la planificación de cada una de las tareas.

Casos de uso
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 2 horas.
Descripción: Identificar y recoger en un diagrama los casos de uso correspondientes al *bot* y al panel de administración.
Salidas/Entregables: Diagrama de casos de uso.
Recursos necesarios: Papel, lápiz y Dia.

Diseño de interfaces
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 25 horas.
Descripción: Creación de prototipos digitales y de papel de las interfaces del panel de administración.
Salidas/Entregables: Prototipos a papel y prototipos digitales.
Recursos necesarios: Papel, lápiz y Cacoo.

Modelo de dominio
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 2 horas.
Descripción: Identificar las entidades necesarias para después, crear el diagrama del modelo de dominio de las herramientas desarrolladas en este proyecto.
Salidas/Entregables: Diagrama del modelo de dominio.
Recursos necesarios: Papel, lápiz y Dia.

3.4.4. Análisis y diseño

En esta sección se muestran las tareas contenidas por el bloque de análisis y diseño.

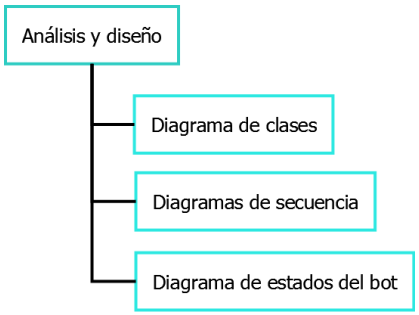


Figura 7: Diagrama EDT del bloque de análisis y diseño.

En la Figura 7 se puede contemplar la descomposición de tareas de este apartado. Ahora se expondrán los detalles y la planificación de cada una de esas ellas.

Diagrama de clases
Paquete de trabajo: Análisis y diseño. **Duración estimada**: 4 horas.
Descripción: Desarrollo del diagrama de las clases que necesitará el *bot* para cumplir con los objetivos del proyecto.
Salidas/Entregables: Diagrama de clases.
Recursos necesarios: Papel, lápiz y Dia.

Diagramas de secuencia
Paquete de trabajo: Análisis y diseño. **Duración estimada**: 10 horas.
Descripción: Realización de los diagramas de secuencia de algunos de los comandos principales del *bot*.
Salidas/Entregables: Diagramas de secuencia.
Recursos necesarios: Papel, lápiz y Dia.

Diagrama de estados
Paquete de trabajo: Análisis y diseño. **Duración estimada**: 2 horas.
Descripción: Diseño del diagrama que muestra los diferentes estados del *bot* durante la ejecución de sus diferentes comandos.
Salidas/Entregables: Diagrama de estados.
Recursos necesarios: Papel, lápiz y Dia.

3.4.5. Implementación y desarrollo

En esta sección se muestran las tareas a realizar durante la implementación y desarrollo del proyecto.

Tal y como se muestra en la Figura 8, este apartado se divide en dos bloques principales. A continuación se describe cada uno de los bloques y se expone la planificación de las tareas que agrupan.

3.4.5.1. Desarrollo del *bot*

En esta sección se muestra la planificación realizada sobre las tareas relacionadas con el desarrollo de las funcionalidades del *bot*.

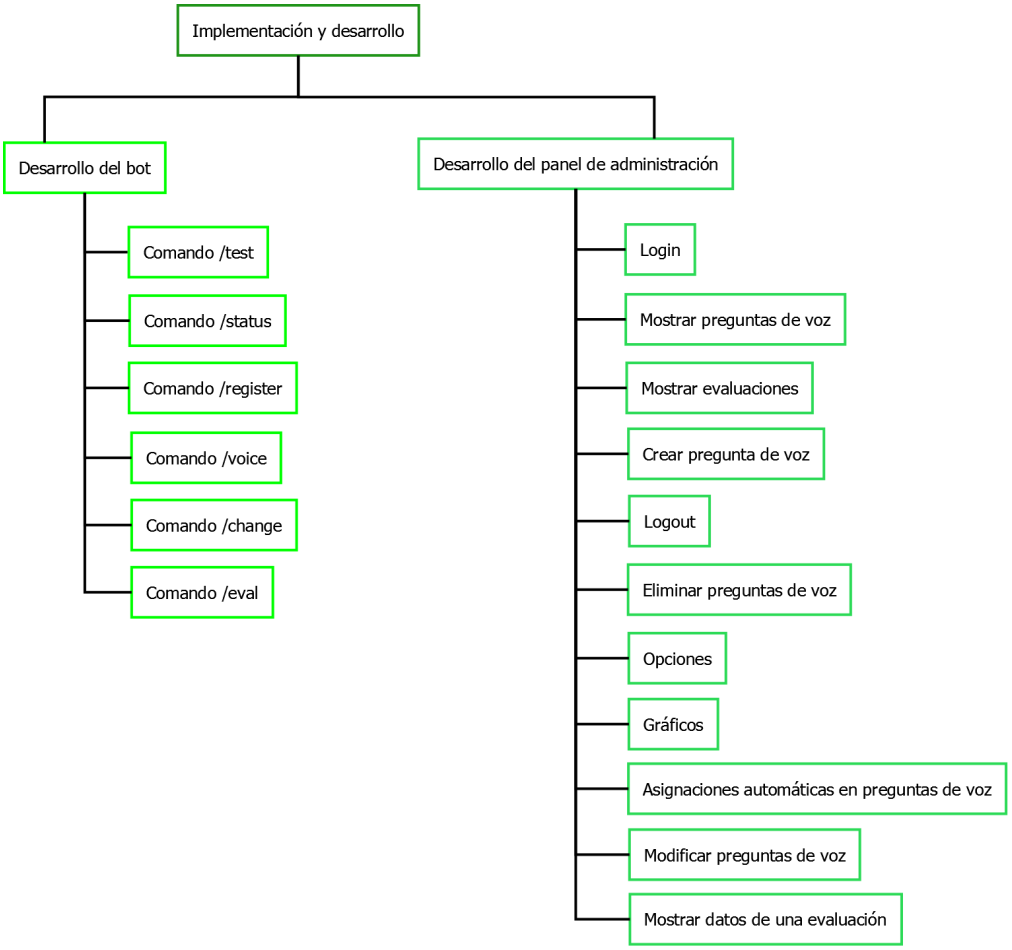


Figura 8: Diagrama EDT del bloque de implementación y desarrollo.

Comando /test
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*. **Duración estimada**: 20 horas.
Descripción: Implementación de los métodos y clases necesarias para el correcto funcionamiento del comando /test usado por el *bot*. Se podrán realizar los distintos *tests* que el profesor haya introducido en la base de datos.
Salidas/Entregables: Ficheros PHP con el código necesario para la correcta ejecución del comando /test.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.

Comando /status
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*. **Duración estimada**: 2 horas.
Descripción: Implementación de los métodos y clases necesarias para el correcto funcionamiento del comando /status usado por el *bot*. Este comando mostrará el número de respuestas correctas, número de respuestas incorrectas y porcentaje de acierto del usuario que lo ejecute.
Salidas/Entregables: Ficheros PHP con el código necesario para la correcta ejecución del comando /status.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.

Comando /register
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*. **Duración estimada**: 5 horas.
Descripción: Implementación de los métodos y clases necesarias para el correcto funcionamiento del comando /register usado por el *bot*. Los usuarios se podrán registrar como estudiantes gracias a este comando.
Salidas/Entregables: Ficheros PHP con el código necesario para la correcta ejecución del comando /register.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.

Comando /voice
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*. **Duración estimada**: 25 horas.
Descripción: Implementación de los métodos y clases necesarias para el correcto funcionamiento del comando /voice usado por el *bot*. Con este comando se le mostrará al usuario que lo ejecute, las preguntas de voz que tenga disponibles para responderlas. Después de seleccionar una el *bot* pedirá al usuario que la conteste mediante una grabación de voz.
Salidas/Entregables: Ficheros PHP con el código necesario para la correcta ejecución del comando /voice.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.

Comando /change
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*. **Duración estimada**: 7 horas.
Descripción: Implementación de los métodos y clases necesarias para el correcto funcionamiento del comando /change usado por el *bot*. Cada alumno puede dar más de una respuesta a una pregunta de voz. Es por ello que, con este comando, se podrá seleccionar la respuesta que se desea que los demás alumnos evalúen.
Salidas/Entregables: Ficheros PHP con el código necesario para la correcta ejecución del comando /change.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.

Comando /eval
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*. **Duración estimada**: 10 horas.
Descripción: Implementación de los métodos y clases necesarias para el correcto funcionamiento del comando /eval usado por el *bot*. Con este comando los estudiantes podrán evaluar las respuestas de voz de otros alumnos.
Salidas/Entregables: Ficheros PHP con el código necesario para la correcta ejecución del comando /eval.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.

3.4.5.2. Desarrollo del panel de administración

En este apartado se muestran las tareas agrupadas en el bloque de implementación y desarrollo que son dedicadas a la creación de las funcionalidades del panel administrativo del *bot*.

Login
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 2 horas.
Descripción: Implementación del *login* en el panel de administración. Se creará una sesión para que el usuario no tenga que volver a introducir sus datos, a excepción de la caducidad de dicha sesión o de la salida de la web mediante el método de salida.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

Mostrar preguntas de voz
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 15 horas.
Descripción: Desarrollo de una página web en la que se muestren todos los datos relevantes de las preguntas de voz creadas por los profesores. Los datos aparecerán en las columnas de una tabla. También contendrá botones con diferentes funcionalidades.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

Mostrar evaluaciones
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 4 horas.
Descripción: Desarrollo de la página web que muestre una tabla con los datos de las respuestas de voz de los alumnos a las preguntas de voz. Entre los datos de la tabla se mostrarán los audios que contienen la respuesta y los criterios que se han evaluado.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

Crear pregunta de voz
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 20 horas.
Descripción: Implementación de la página web que permita introducir los campos necesarios para la creación de una pregunta de voz. También se podrán seleccionar los alumnos que se desee que respondan a la pregunta y a sus evaluadores.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

Logout
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 1 hora.
Descripción: Implementación de los ficheros que destruyan la sesión que el usuario del panel de administración tenía iniciada.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

Eliminar preguntas de voz
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 2 horas.
Descripción: Desarrollo de una página web que permita la eliminación de preguntas de voz ya creadas. Antes de la supresión de cualquier pregunta se pedirá una confirmación para poder realizar la acción con seguridad.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

Opciones
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 5 horas.
Descripción: Desarrollo de una página web que permita la modificación de las opciones del *bot*. Algunas de las posibles opciones es el permitir la autoevaluación de los alumnos o la selección de la escala en la que se evalúa.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

Gráficos
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 15 horas.
Descripción: Implementación de una página web que permita la creación de gráficos con los datos almacenados en la base de datos. Se quieren realizar algunos gráficos por defecto y permitir al usuario crear otros con los parámetros que desee.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2, Plotly y Sublime Text 2.

Asignación automática de preguntas de voz
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 15 horas.
Descripción: Implementar los métodos necesarios para realizar asignaciones automáticas de preguntas de voz a alumnos. Se quieren realizar tres tipos de asignaciones, una aleatoria, otra según las notas de los alumnos y una última que asignaría la pregunta a todos los alumnos.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

Modificar preguntas de voz
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 10 horas.
Descripción: Creación de una página web con la que se permita modificar los datos de alguna de las preguntas de voz creadas anteriormente.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

Mostrar datos de una evaluación
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración. **Duración estimada**: 6 horas.
Descripción: Implementación de una página web que muestre la respuesta de un alumno a una pregunta de voz, junto con una tabla que contenga las calificaciones recibidas de otros alumnos.
Salidas/Entregables: Ficheros con el código del desarrollo de esta funcionalidad del panel de administración.
Recursos necesarios: Google Chrome, Apache2 y Sublime Text 2.

3.4.6. Pruebas

En esta sección se exponen las diferentes pruebas a realizar para comprobar el correcto funcionamiento del *bot* y del panel de administración.

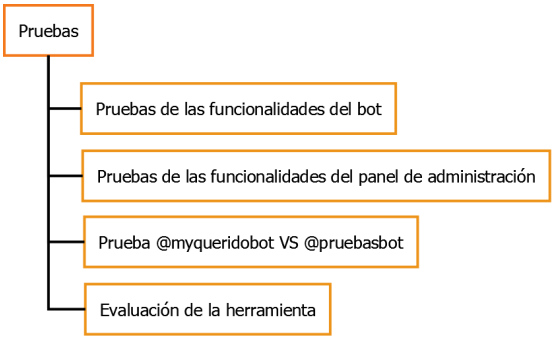


Figura 9: Diagrama EDT del bloque de pruebas.

En la Figura 9 se muestra el desglose de las tareas de este bloque. Como se puede observar en la imagen, no solo se realizaran pruebas del funcionamiento, sino que se realizarán algunas pruebas contra @pruebasbot, el desarrollado en el proyecto "Bot para evaluación colaborativa de ejercicios orales", realizado por Pablo Uribe Bastero. Además, la herramienta se pondrá a prueba en una asignatura de la Universidad del País Vasco, la cual, también se documentará.

Pruebas de las funcionalidades del bot
Paquete de trabajo: Pruebas. **Duración estimada**: 25 horas.
Descripción: Realización e implementación de las pruebas necesarias que comprueben el correcto funcionamiento de las funcionalidades del *bot*.
Salidas/Entregables: Ficheros con el código que realicen las pruebas del funcionamiento automáticamente.
Recursos necesarios: Telegram, librería php-telegram-bot y ordenador y *smartphone* con conexión a Internet.

Pruebas de las funcionalidades del panel de administración
Paquete de trabajo: Pruebas. **Duración estimada**: 35 horas.
Descripción: Realización de las pruebas necesarias para comprobar que todas las funcionalidades de la web de administración del *bot* funcionan correctamente.
Salidas/Entregables: N/A.
Recursos necesarios: Apache2 y ordenador con conexión a Internet.

Prueba @myqueridobot VS @pruebasbot
Paquete de trabajo: Pruebas. **Duración estimada**: 1 hora.
Descripción: Se realizará una prueba que comprobará el tiempo que tarda cada *bot* en responder a un usuario.
Salidas/Entregables: Tabla que contiene los tiempos que ha necesitado cada *bot* para responder al usuario.
Recursos necesarios: Telegram y ordenador y *smartphone* con conexión a Internet.

Evaluación de la herramienta
Paquete de trabajo: Pruebas. **Duración estimada**: 5 horas.
Descripción: Todo lo desarrollado durante este proyecto será puesto a prueba durante el curso 2016-2017 en la asignatura "Desarrollo de Aplicaciones Web Enriquecidas". Es una asignatura del grado de Ingeniería Informática de Gestión y Sistemas de Información en la Universidad del País Vasco. Después de realizar la prueba, se presentarán algunas conclusiones sobre los resultados obtenidos acerca de la problemática que se intenta solventar con este proyecto.
Salidas/Entregables: Conclusiones sobre los resultados obtenidos.
Recursos necesarios: Primera versión del proyecto completo.

3.4.7. Documentación

Por último, se detallan las tareas relacionadas con la documentación del proyecto. Este bloque se dividirá en dos secciones para, de esta manera, separar la documentación relacionada con la programación de la documentación de la memoria del proyecto.

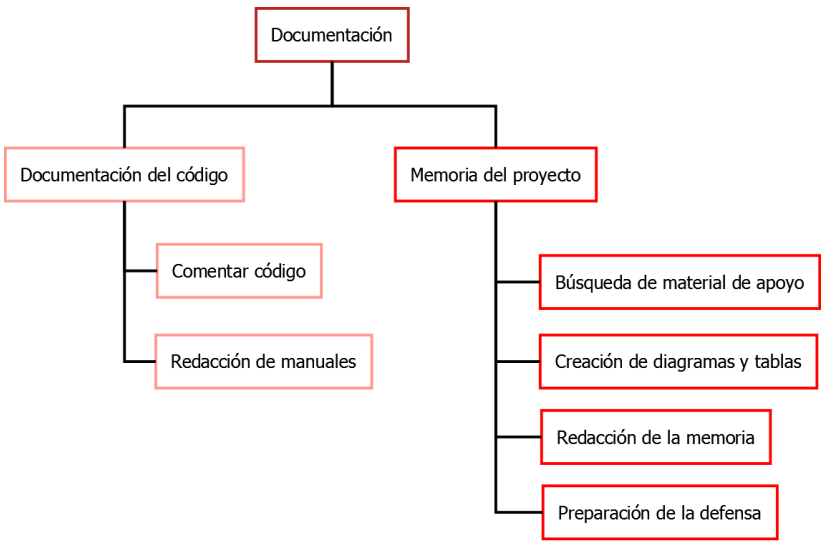


Figura 10: Diagrama EDT del bloque de documentación.

En la Figura 10 se muestra la división mencionada anteriormente junto con la descomposición de las secciones creadas. A continuación se describirá cada una de ellas y se mostrará la planificación realizada para cada tarea.

3.4.7.1. Documentación del código

En este apartado se recogen las tareas de explicar el código y redactar los manuales de uso e instalación de todo lo desarrollado durante el proyecto.

Comentar código
Paquete de trabajo: Documentación - Documentar código. **Duración estimada**: 15 horas.
Descripción: Explicación, mediante comentarios, de las clases, métodos e incluso lineas del código programadas.
Salidas/Entregables: Ficheros de código comentados.
Recursos necesarios: Sublime Text 2.

Redacción de manuales
Paquete de trabajo: Documentación - Documentar código. **Duración estimada**: 10 horas.
Descripción: Redacción de los manuales de uso e instalación del software relativo al proyecto.
Salidas/Entregables: Manuales.
Recursos necesarios: Sublime Text 2.

3.4.7.2. 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.

Búsqueda de material de apoyo
Paquete de trabajo: Documentación - Memoria del proyecto. **Duración estimada**: 5 horas.
Descripción: Búsqueda de artículos, libros o documentos que tengan alguna relación con las herramientas usadas durante este proyecto.
Salidas/Entregables: Bibliografía.
Recursos necesarios: Ordenador con conexión a Internet.

Creación de tablas y diagramas
Paquete de trabajo: Documentación - Memoria del proyecto. **Duración estimada**: 3 horas.
Descripción: Creación de las tablas y diagramas, independientes de los apartados de casos de uso y análisis y diseño, que se usarán en este documento.
Salidas/Entregables: Tablas y diagramas.
Recursos necesarios: Dia, Overleaf y GanttProject.

Redacción de la memoria
Paquete de trabajo: Documentación - Memoria del proyecto. **Duración estimada**: 50 horas.
Descripción: Redacción de la memoria del proyecto.
Salidas/Entregables: Memoria del proyecto.
Recursos necesarios: Overleaf.

Preparación de la defensa
Paquete de trabajo: Documentación - Memoria del proyecto. **Duración estimada**: 25 horas.
Descripción: Elaboración y preparación de la presentación de la defensa del proyecto.
Salidas/Entregables: Presentación *Power Point*.
Recursos necesarios: Proyecto y documentación.

3.4.8. Resumen de la planificación realizada

En las siguientes tablas se muestran el nombre, duración y las dependencias de cada una de las tareas identificadas. Como se puede observar en la Tabla 3, el número de horas totales que se estima para la duración del proyecto asciende a 510 horas.

ID	Nombre de la tarea	Predecesores	Duración estimada
	Aprendizaje		95
1	PHP	-	30
2	php-telegram-bot	1	10
3	HTML	-	5
4	JavaScript	-	15
5	Plotly	4	5
6	Latex	-	30
	Organización		40
7	Planteamiento de la aplicación	-	3
8	Planificación de las tareas	7	5
9	Elección de software	8	2
10	Instalación del software	9	5
11	Reuniones con el director del proyecto	7	25
	Captura de requisitos		29
12	Casos de uso	8	2
13	Diseño de interfaces	11	25
14	Modelo de dominio	12,13	2
	Análisis y diseño		16
15	Diagrama de clases	14	4
16	Diagramas de secuencia	15	10
17	Diagrama de estados del bot	12	2

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

	Implementación y desarrollo		156
18	Desarrollo del bot		69
19	Comando /test	1,2,16,17	20
20	Comando /status	1,2,16,17	2
21	Comando /register	1,2,16,17	5
22	Comando /voice	1,2,16,17	25
23	Comando /change	1,2,16,17	7
24	Comando /eval	1,2,16,17	10
25	Desarrollo del panel de administración		87
26	Login	1,3,4,14	2
27	Mostrar preguntas de voz	1,3,4,14	15
28	Mostrar evaluaciones	1,3,4,14	4
29	Crear pregunta de voz	1,3,4,14	20
30	Asignaciones automáticas en preguntas de voz	1,3,4,14	7
31	Modificar pregunta de voz	1,3,4,14	2
32	Logout	1,3,4,14	1
33	Eliminar preguntas de voz	1,3,4,14	5
34	Opciones	1,3,4,14	15
35	Gráficos	1,3,4,5,14	10
36	Mostrar datos de una evaluación	1,3,4,14	6

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

	Pruebas		66
37	Pruebas de las funcionalidades del bot	18	25
38	Pruebas de las funcionalidades del panel de administración	25	35
39	Prueba myqueridobot VS pruebasbot	37	1
40	Evaluación de la herramienta	37,38	5
	Documentación		108
	Documentación del código		25
41	Comentar el código	18,25	15
42	Redacción de manuales	37,38	10
	Memoria del proyecto		83
43	Búsqueda de material de apoyo	7	5
44	Creación de diagramas y tablas	8	3
45	Redacción de la memoria	6,41,42,43,44	50
46	Preparación de la defensa	45	25
	TOTAL	**45**	**510**

Tabla 3: Dependencias y duración de las tareas (3).

3.5. Planificación temporal

Una vez se conocen las tareas a realizar, hay que planificar como se desarrollarán durante el tiempo que se ha planeado que dure el proyecto. Para ello se ha realizado el diagrama Gantt que se muestra en la Figura 11. En él, se pueden observar las diferentes los diferentes bloques mencionados en el alcance del proyecto (Sección 3.4). Como bien se menciona en esa sección, los bloques de implementación y de documentación han sido divididos para facilitar el reconocimiento de las tareas.

Durante este proyecto se quiere simular un horario de un trabajador a media jornada que trabaja 28 horas semanales. Como el proyecto comienza el día 20 de noviembre de 2016, y se ha planificado su finalización el día 7 de julio de 2017, hay un periodo de más o menos siete meses y medio. Suponiendo que un mes tiene más o menos cuatro semanas, se han realizado los siguientes cálculos:

$$Número\ de\ meses * Número\ de\ semanas * Horas\ de\ trabajo = 7{,}5 * 4 * 28 = 840\ horas \quad (1)$$

De las 840 horas resultantes, hay que quitar los días festivos, eventos y fines de semana, en los cuales no se trabajará, a menos que sea necesario para la finalización del proyecto en el plazo previsto. Se ha estimado que el total de horas de trabajo que se perderán durante esos días será de 70 horas. Teniendo en cuenta esto, las horas disponibles para trabajar en el proyecto quedarían de la siguiente manera:

$$Horas\ totales - Horas\ festivas = 840 - 70 = 770\ horas\ disponibles\ para\ trabajar \quad (2)$$

Teniendo en cuenta que la estimación del número de horas totales invertidas en el proyecto será de 510 horas (Sección 3.4.8), se puede apreciar que hay tiempo de sobra para poder entregar el proyecto a tiempo. Además, tal y como se muestra en el Gantt, muchas de las tareas se realizarán en paralelo a otras, ya que las dependencias no lo impiden.

Dicho todo lo referente a la planificación temporal, ya solo queda aclarar un pequeño detalle. Como podrás apreciar, en el Gantt, el tiempo empieza a correr desde septiembre y no desde noviembre como se ha comentado en esta sección. Esto es debido a que este proyecto es la continuación de "Bot para evaluación colaborativa de ejercicios orales", el cual, será defendido por Pablo

Uribe Bastero ante el tribunal correspondiente el 19 de noviembre del 2016. Como este proyecto no empezará a desarrollarse hasta pasada esa fecha, se usará ese lapso de tiempo para realizar tareas de aprendizaje. Para ello, el director del proyecto ha recomendado la instalación de algunas herramientas. Es por ello que la instalación de herramientas comienza antes que la elección de las mismas, ya que serán utilizadas para el aprendizaje del funcionamiento y desarrollo de un *bot* de Telegram.

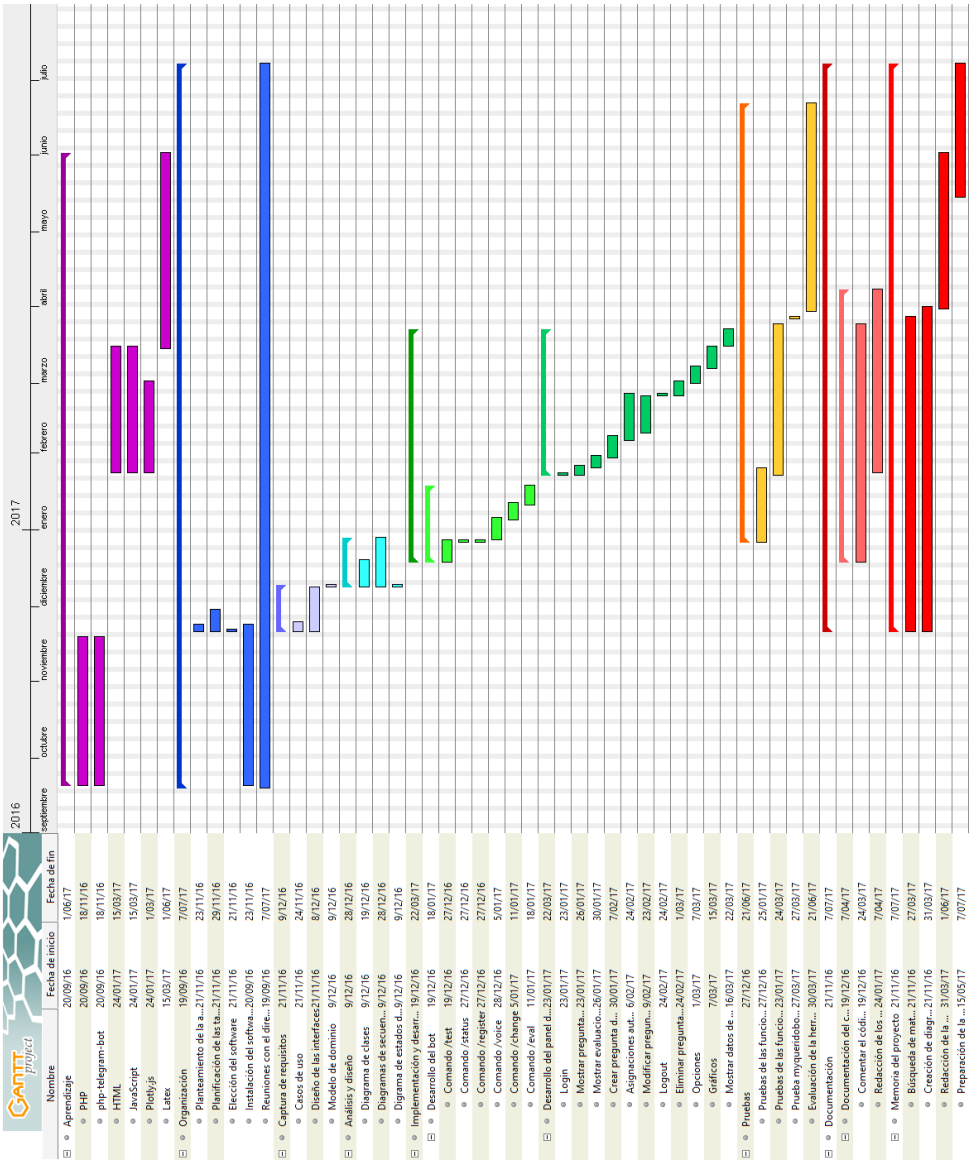


Figura 11: Diagrama Gantt

Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Gantt/Gantt.png`.

3.6. Evaluación económica

Aunque este proyecto se trate de un TFG del que no se espera obtener ningún beneficio económico, se ha realizado la siguiente valoración económica por si en un futuro se llegara a comercializar.

3.6.1. Mano de obra

En la publicación de tablas salariales del BOE el miércoles 15 de marzo de 2017, se indica que el sueldo neto de un programador informático es de 1.436,62€ mensuales. El salario por hora de trabajo se puede obtener mediante la siguiente ecuación:

$$Sueldo/hora = \frac{Sueldo\ mes}{Semanas/mes * Días\ trabajo/semana * Horas\ trabajo/día} \tag{3}$$

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:

$$Sueldo/hora = \frac{1384,09}{4 * 5 * 8} = \frac{1{,}436,62}{160} = 8,98 \tag{4}$$

Dando un total de 8,98€ por cada hora trabajada.

En la planificación del proyecto, se ha estimado que se trabajarán 510 horas, de las cuales, 95 se dedicarán al aprendizaje del uso de las herramientas que se utilizarán. Esas horas no se van a tener en cuenta a la hora de calcular los costes de mano de obra por lo que:

$$Horas\ totales - Horas\ aprendizaje = 510 - 95 = 415\ horas \tag{5}$$

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

$$Sueldo = Horas\ trabajo * Sueldo/hora = 415 * 8,98 = 3{,}726,7 \tag{6}$$

El gasto económico en lo referente a la mano de obra sería de 3.726,7€.

3.6.2. Gasto de *software*

Como todas las herramientas que se piensan utilizar para el desarrollo del proyecto son gratuitas, en este apartado el gasto en herramientas *software* será de 0€. En este caso, no se está teniendo en cuenta que la universidad ha proporcionado el servidor en el que se han realizado las pruebas del *bot* y del panel de administración.

Aunque a la hora de comercializar lo que se desarrolle en el proyecto el servidor lo pagará el cliente, el haber realizado las pruebas en uno también ha generado un coste extra que se tendrá en cuenta.

Después de mirar varias herramientas, se tendría en cuenta que se hubiese usado un servidor de Microsoft Azure de un coste de más o menos 100€. Además de eso, es necesario un certificado digital para que el servidor pueda realizar las conexiones HTTPS que necesita el *bot*. Eso añade un coste de unos 10€.

El gasto total en *software* sería de 110€/año.

3.6.3. 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.

3.6.3.1. Ordenador portátil Asus F550C

La vida útil que se estima para este portátil es de unos 5 años (60 meses) y su precio fue unos 700€ aproximadamente.

Su amortización mensual es:

$$Amortización\ mensual = \frac{Precio}{Vida\ útil} = \frac{700}{60} = 11,67 \quad (7)$$

El resultado es 11,67€/mes y como el ordenador va a ser usado durante todo el proyecto (8 meses):

$$Amortización\ total = Amort.\ mensual * Meses = 11,67 * 8 = 93,36 \quad (8)$$

Dando un total de 93,36€ en lo referente a la amortización del ordenador.

3.6.3.2. Dispositivo móvil ASUS Zenfone 2 Laser

El precio de este dispositivo fue de 180 € y su vida útil es de 2 años (24 meses) por lo que la amortización mensual quedaría:

$$Amortización\ mensual = \frac{Precio}{Vida\ útil} = \frac{180}{24} = 7,5 \quad (9)$$

El resultado es 7,5 €/mes, y el dispositivo móvil será usado durante el desarrollo de todo el proyecto (8 meses). Por tanto la amortización total del móvil sería:

$$Amortización\ total = Amort.\ mensual * Meses = 7,5 * 8 = 60 \quad (10)$$

La amortización del móvil relativa a este proyecto es de 60 €.

3.6.3.3. Suma de gastos

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

$$Gastos = Amort.ordenador + Amort.móvil = 93,36 + 60 = 153,36 \quad (11)$$

Dando un total de 153,36 € en gasto *hardware*.

3.6.4. Gastos indirectos

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

3.6.4.1. Luz e Internet

Durante el desarrollo del proyecto también se realiza un gasto en electricidad e Internet. Como este gasto puede llegar a variar mucho dependiendo del coste de la luz y de la tarifa de Internet que se tenga, se da por hecho que el gasto total de este gasto indirecto suma aproximadamente 150 €.

3.6.5. Gastos totales

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

En la Tabla 4 aparece reflejada la suma de todos los gastos mencionados previamente. Dando un total de 4.140,06 € por el completo desarrollo del proyecto.

Tipo de gasto	***Gasto ocasionado (en €)***
Mano de obra	**3.726,7 €**
Gasto en *software*	**110 €**
Gasto en *hardware*	**153,36 €**
Gastos indirectos	**150 €**
Total	**4.140,06 €**

Tabla 4: Costes totales del proyecto

3.6.6. Posibilidades de negocio

Aunque las herramientas que se desarrollen en este TFG no se piensan comercializar, en caso de que se quisiera llegar a sacar algún beneficio económico de ellas, se ha planteado la siguiente opción.

Como las herramientas que se desarrollen tienen un propósito educativo, estas se podrían llegar a vender a entidades como universidades o incluso a empresas que quieran dar formación a sus empleados. Esto implica un mercado bastante amplio, en el cual se podrían llegar a vender licencias de las herramientas que se van a desarrollar por un módico precio de 13.550 €. Esto implica que con solo una venta de la herramienta a una de las entidades, el proyecto ya obtendría un beneficio de 9.409,94 € (sin tener en cuenta los riesgos). Todas las demás ventas serían beneficios.

3.7. 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 en mayor o menor grado al desarrollo del proyecto.

A la hora de calcular el impacto, se ha estipulado la jornada laboral de entre 4 y 8 horas al día.

3.7.1. Enfermedad o lesión

Consiste en la indisponibilidad por una causa médica, ya sea derivada del trabajo o por causas ajenas a este.

Prevención

- **Derivadas del trabajo**:
  - Mantener una postura correcta a la hora de trabajar.
  - Mantener la distancia correcta frente al monitor.
  - Realizar descansos periódicamente.
- **No derivadas del trabajo**:
  - Dormir las horas recomendadas por los expertos en salud.
  - Realizar actividad deportiva, sin ser de riesgo.

Plan de contingencia

- Acudir al médico para diagnosticar el alcance de la lesión o enfermedad.
- En caso de que el diagnóstico sea favorable, se continuará con el trabajo. Siempre intentando descansar más tiempo del que se hacía hasta el momento.
- En caso de que el diagnóstico sea desfavorable, se alargará el tiempo necesario para recuperarse y poder continuar con el trabajo.

Probabilidad

- Diagnóstico favorable. Baja: 15 %.
- Diagnóstico desfavorable. Muy baja: 5 %.

Impacto

- Diagnóstico favorable. 0,15 x 2 días = 2,4 horas. Impacto medio.
- Diagnóstico desfavorable. 0,05 x 7 días = 2,8 horas. Impacto medio.

3.7.2. Problemas con el equipo informático

Daños imprevistos, tanto de hardware como de software, debido a acciones relacionadas con el trabajo, transporte de material o agentes externos.

Prevención

- Mantener el equipo en unas condiciones ambientales adecuadas.
- Realizar una revisión mensual del equipo.
- Realizar copias de seguridad periódicamente y en varios dispositivos de almacenamiento.

Plan de contingencia

- ***Software***: Se procederá a restaurar la última copia de seguridad guardada y se procederá a realizar el trabajo que se haya perdido.
- ***Hardware***: Se procederá a usar otro ordenador, en el cual se volverán a instalar los programas requeridos y se pasará la copia del trabajo realizado.

Probabilidad

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

Impacto

- Fallo de *software*. 0,15 x 1 día = 1,2 horas. Impacto bajo.
- Fallo de *hardware*. 0,05 x 1 día = 0,4 horas. Impacto bajo.

3.7.3. Problemas con las herramientas de desarrollo

Es complicado predecir los problemas que pueden surgir con los programas que se usarán durante el proyecto, sobre todo sabiendo que se tiene experiencia del uso de gran parte de ellos. Aun así es necesario tenerlos en cuenta.

Prevención

- Tener una buena curva de aprendizaje con los programas.
- Conocer los lugares de donde extraer información. Entre ellos manuales y páginas oficiales de las herramientas.

Plan de contingencia

- Buscar en Internet la solución al problema.
- Aplicar los conocimientos que se tienen sobre las herramientas para intentar solventar el problema.

Probabilidad

- Encontrar un problema y dar con la solución adecuada. Media: 45 %.

Impacto

- Solución no aparente. 0,45 x 3 días = 10,8 horas. Impacto medio.

4. Antecedentes

En este capítulo, se van a mencionar los proyectos en los que nos hemos basado para el desarrollo. En el proyecto pionero, se desarrolló un *chatbot* conocido como @dawebot [Pereira, 2016]. Su funcionalidad principal es poder responder preguntas tipo *test* sobre el temario de la asignatura Desarrollo de Aplicaciones Web Enriquecidas (DAWE).

La otra referencia es el proyecto que precede a este. En él se desarrollo un *bot* denominado @pruebasbot (fue una continuación y mejora de @dawebot).

4.1. @dawebot

Es el resultado de un proyecto desarrollado por Juanan Pereira en PHP. @dawebot consta de siete comandos propios y tres genéricos, creados para los siguientes propósitos:

- **Aprendizaje autónomo**: Permite realizar *tests* sobre el temario de DAWE. Al ser una asignatura no presencial, implica que los estudiantes tendrán que aprender por su cuenta todo el temario. Gracias a los *tests*, esta tarea puede ser bastante más rápida y entretenida de lo que parece. Además, el tener una pequeña explicación por cada pregunta ayuda a solucionar las dudas que surjan.
- **Movilidad**: Hoy en día, gracias a los *smartphones*, todo aquel que tenga uno con una tarifa que le permita el acceso a Internet, puede interactuar con el *bot*, dónde y cuando quiera. Como ya es común que los estudiantes usen este tipo de dispositivos para comunicarse, también admite la posibilidad de que aprendan con ellos en cualquier lugar.
- **Registro de datos**: Todos los mensajes que se le envíen a @dawebot son guardados en una base de datos, utilizados para posteriores estudios. Esto también permite a los estudiantes tener un control de como llevan la asignatura gracias al comando /status, que envía las estadísticas del usuario que lo ejecuta.

Con los datos obtenidos con @dawebot, se ha realizado un estudio con la intención de observar los resultados de los alumnos que han utilizado este *bot*. Se intentaba, y todavía se intenta, demostrar que el uso del *bot* mejora el aprendizaje. Eso haría que los estudiantes obtuviesen mejores notas, pero no se ha conseguido demostrar nada hasta el momento, ya que se disponen de pocos datos que analizar.

4.2. @pruebasbot

Como ya se ha mencionado en los apartados anteriores, @pruebasbot, junto con su correspondiente panel de administración, fueron desarrollados con el planteamiento de mejorar @dawebot. Para ello, Pablo, migró el código de @dawebot de PHP a Node.js. Esto se realizó para hacerlo compatible con una herramienta desarrollada por la empresa Microsoft, que permite el uso del *bot* en otros programas diferentes a Telegram como Skype, Facebook Messenger, etc.

Después de realizar las tareas de migración, se tanteó la posibilidad de añadir al *bot* una funcionalidad para que los estudiantes, tengan que hablar para responder a algunas preguntas que este les plantee. Con esta idea se desarrollaron otras funcionalidades como la coevaluación y algunos apartados del panel de administración.

Aunque el proyecto se quiso probar en una de las asignaturas de la Universidad del País Vasco, no se llegó a utilizar porque se comenzó el desarrollo de este proyecto. Al final el código de @pruebasbot se acabó desechando, pero se tomó como referencia para realizar las herramientas que se han desarrollado en este TFG.

4.3. Situación actual del proyecto

Este proyecto se centrará en migrar el código de las nuevas funcionalidades desarrolladas para @pruebasbot a PHP. Después, se tratará de añadir nuevos comandos a @myqueridobot, el *bot* que se desarrollará durante este proyecto.

Con algunos cambios en las funcionalidades de @pruebasbot, se quiere permitir a los alumnos enviar más de una respuesta a una pregunta de voz. Por ello, se quiere permitir al estudiante escoger la grabación que realmente desea que se evalúe. El comando /change que se desarrollará durante este proyecto se encargará de dar esta funcionalidad a @myqueridobot.

También se desean añadir dos funcionalidades principales al panel de control del *bot*. Una que permita la selección automática de los estudiantes que respondan y evalúen las preguntas de voz, para que el profesor no tenga que hacerlo de manera manual en caso de tener muchos alumnos. La otra funcionalidad es para permitir la creación de gráficos con los datos recogidos por el *bot*. De esta manera, se facilitará la recogida de datos para realizar las investigaciones oportunas, o simplemente para llevar un control de los estudiantes.

4.4. Comparativa

La Tabla 5 muestra la comparativa de las funcionalidades de los tres *bots* mencionados en este capítulo.

Funcionalidades ofrecidas	***@dawebot***	***@pruebasbot***	***@myqueridobot***
Comando para realizar *tests*	✓	✓	✓
Comando para consultar estadísticas	✓	✓	✓
Comando para responder preguntas de voz	✗	✓	✓
Comando para evaluar respuestas de voz	✗	✓	✓
Comando para cambiar respuesta de voz seleccionada	✗	✗	✓
Comando para ver calificaciones de respuesta de voz	✗	✗	✓
Comando de registro	✗	✗	✓
Realizado en PHP	✓	✗	✓
Realizado en Node.js	✗	✓	✗
Funciona en múltiples aplicaciones	✗	✓	✗
***Testeado* en pruebas reales**	✓	✗	✓

Tabla 5: Comparativa de funcionalidades de los *bots*.

5. Captura de requisitos

En este capítulo, se recogen los requisitos que ha de cumplir la herramienta desarrollada durante el proyecto, de manera que se satisfagan las necesidades de los usuarios que la utilicen.

Cabe destacar que durante el desarrollo, se han realizado cambios en todos los diagramas. Esto se ha debido a los diferentes factores que han afectado al proyecto. En las siguientes secciones, se mostrarán los diagramas relacionados a la captura de requisitos y sus modificaciones. Estas últimas, han sido provocadas por las nuevas necesidades que no se habían tenido en cuenta en el planteamiento inicial (Sección 3).

El cliente Juanan Pereira, profesor de la asignatura en la que se está realizando la evaluación de la herramienta, identificó algunas necesidades imprevistas. Esas nuevas necesidades se exponen como modificaciones en esta sección.

5.1. Jerarquía de actores

Como se comentó en el apartado de objetivos (Sección 3.1), la seguridad es algo a tener en cuenta en este proyecto. Es por ello que el *bot*, tiene un comando de registro oculto. Esto es para que un usuario no identificado no pueda acceder a los comandos del *bot*. Para poder utilizarlos, hay que registrarse con el comando /register y meter los datos que el *bot* nos pida. De está manera nos habremos registrado como estudiante en la herramienta.

Para poder utilizar el panel de administración, es necesario estar registrado como profesor en la herramienta. Como se puede apreciar, con lo comentado en esta sección, se puede llegar a intuir que existen dos tipos de actores. En la Figura 12 se muestran estos dos tipos a los que se les ha denominado “Estudiante“ y “Profesor“.

5.2. Casos de uso

En este apartado, se muestran en diagramas las posibles acciones que puede tomar cada tipo de usuario mientras utiliza las herramientas desarrolladas durante este proyecto. Como se ha mostrado en la sección anterior (Sección 5.1), se han identificado dos tipos de actores, Estudiante y Profesor, los cuales se se describen a continuación junto con sus diagramas de casos de uso. Los detalles de las modificaciones se explicarán el capítulo de desarrollo

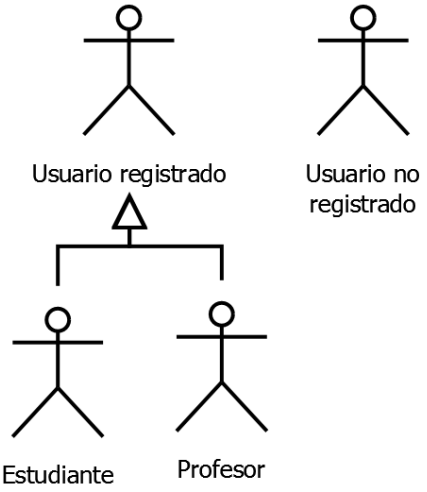


Figura 12: Jerarquía de actores

(Sección 8).

5.2.1. Estudiante

Este es el tipo de usuario registrado capaz de usar todos los comandos del *bot*. Si el usuario no está registrado e intenta ejecutar algún comando, el *bot* le mandará un mensaje de error indicándole que no está registrado.

A este usuario se le modificaron y añadieron algunas posibles acciones más. Es por ello que, en primer lugar, se presentará la primera versión del diagrama de casos de uso que se realizó, y después, se dará paso a la última versión del diagrama. En cada diagrama, se explicarán las posibles acciones a realizar por el usuario Estudiante.

En esta primera versión del diagrama, ilustrado en la Figura 13, se muestran las siguientes acciones que puede ejecutar un usuario Estudiante:

- **/test**: Cuando un Estudiante ejecuta este comando, se le envía un listado de botones con los *tests* disponibles. Cuando pulse un botón, se le enviará la primera pregunta del *test* y se le mostrarán los botones con las posibles respuestas a la pegunta. Al responder, se le indicará si la respuesta ha sido correcta o no y el *bot* mostrará las opciones pertinentes al final de cada pregunta.
- **/status**: Al ejecutar este comando, el Estudiante recibirá un mensaje

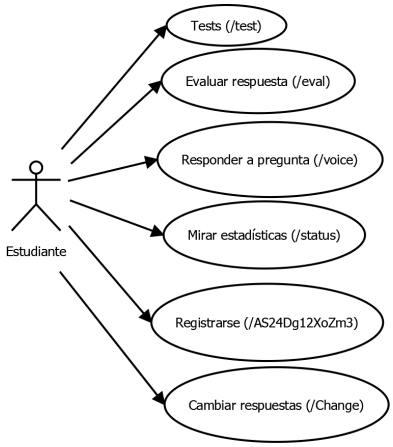


Figura 13: Diagrama de casos de uso inicial de Estudiante

con las estadísticas de los resultados en los *tests* que ha realizado hasta el momento.

- **/AS24Dg12XoZm3**: Al comienzo, este era el comando de registro. En la última versión se hace más intuitivo y se le añade mayor seguridad.
- **/voice**: Este comando envía al usuario el listado de preguntas de voz que tiene asignadas para poder responderlas. Al seleccionar una, se le enviará la pregunta seleccionada para que conteste mediante una grabación de voz.
- **/eval**: Al ejecutar este comando, el usuario recibe una lista con las respuestas de voz de otros estudiantes para evaluarlas. Al escoger una de las respuestas, al usuario se le enviarán mensajes con la pregunta que se quiere responder con la grabación de voz, la grabación de voz, el criterio a evaluar y un último mensaje que indica la escala en la que se evalúa. En caso de tener que evaluar más de un criterio, al Estudiante se le enviará uno detrás de otro hasta que se terminen de evaluar todos.
- **/change**: Los Estudiantes pueden enviar más de una grabación como respuesta a una pregunta de voz. Este comando les permite seleccionar la respuesta que desean que sea evaluada por otros usuarios Estudiante.

En la última versión realizada se han modificado algunos de los comandos que vienen por defecto con la librería php-telegram-bot, se ha modificado el comando de registro de la primera versión y se ha creado un nuevo comando. Las modificaciones han sido reflejadas en la última versión del diagrama de casos de uso (Figura 14).

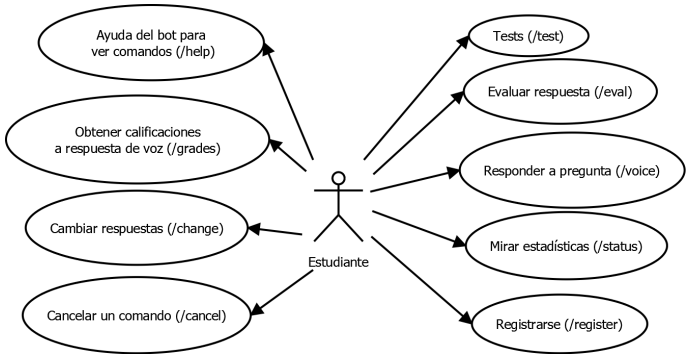


Figura 14: Diagrama de casos de uso final de Estudiante

Los cambios respecto a la primera versión han sido los siguientes:

- **/register**: El comando de registro se modificó para ser más fácil de recordar. Aunque sea un comando fácil de descubrir aún estando oculto, la seguridad está en lo que ejecuta después. Al ejecutar este comando se le pide al usuario un identificador único de su profesor. Este identificador es un *String* que puede llegar a los veinte caracteres. Después de escribir el identificador correctamente, se le pide al usuario que escoja el grupo de estudiantes al que pertenece.

- **/help**: Este es uno de los comandos que venían por defecto con la librería php-telegram-bot (versión 0.42.0 y anteriores). Este comando envía al usuario el listado de comandos disponibles del *bot*, junto con una pequeña descripción de lo que hace. Tuvo que modificarse para poder cambiar el lenguaje del *bot* y para ocultar los comandos del administrador y de registro.

- **/cancel**: Con este comando se puede cancelar la ejecución de cualquier comando que se lleva a cabo mediante una conversación. Este comando también venía por defecto con la librería. Tuvo que modificarse para eliminar el mensaje que enviaba por defecto.

- **/grades**: Este es un nuevo comando que permite al estudiante conocer las notas que ha recibido alguna de sus respuestas de voz. Para ello el *bot* le envía al Estudiante un listado con las preguntas que ha respondido mediante una grabación de voz y que alguien ya ha calificado. Al escoger una, se le enviará al estudiante un listado con las calificaciones que ha recibido la grabación y, si han sido necesarias, las grabaciones de voz de sus compañeros explicando el por qué de la nota que ha recibido.

Entre estos comandos no se ha mencionado el cambio del comando /start, ya que solo ha sido modificado para poder cambiar el idioma del nuevo mensaje que envía al usuario. Este comando solo se ejecuta una vez.

Puede que en este apartado los comandos de *bot* se hayan explicado de una manera un tanto difusa. Por tanto, se recomienda echarle un vistazo al diagrama de estados del *bot* en la Sección 6.2. De esta manera, se podrá entender un poco mejor el proceso que sigue cada comando.

5.2.2. Profesor

Pertenecen al grupo de usuarios Profesor aquellos que estén registrados como profesor en la herramienta. Se les permite el uso del panel de administración, pero para poder acceder a él, deben introducir correctamente su nombre de usuario e identificador de Telegram.

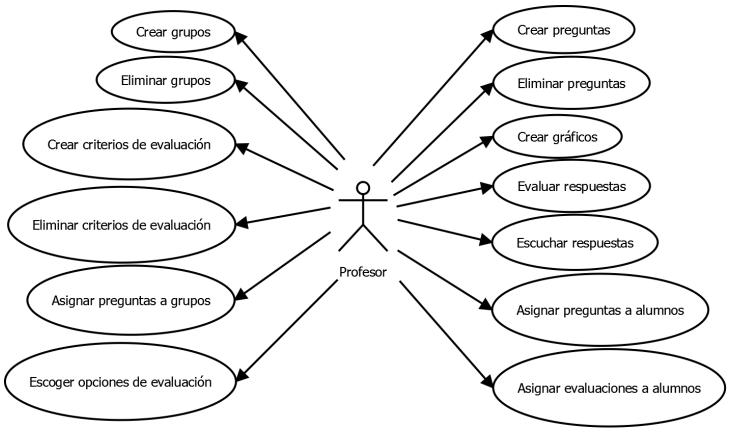


Figura 15: Diagrama de casos de uso inicial de Profesor

En la Figura 15, se muestra la primera versión de los casos iniciales. A continuación, se detalla cada una de las posibles acciones que podía realizar un usuario Profesor en el panel de administración:

- **Crear preguntas**: Los usuarios Profesor, pueden crear nuevas preguntas de voz introduciendo los datos necesarios. De este apartado solo hay dos datos obligatorios, el texto en el que se formula la pregunta y el *deadline*.
- **Eliminar preguntas**: Después de realizar las comprobaciones necesarias, permite al Profesor eliminar las preguntas de voz seleccionadas.

- **Crear gráficos**: El panel de administración permite a los usuarios Profesor crear gráficos con los datos obtenidos del *bot*. Los datos muestran su uso y el desarrollo que han tenido los alumnos del usuario Profesor en el tiempo.
- **Evaluar respuesta**: En un principio, se quería hacer que el Profesor pudiera evaluar las grabaciones que los estudiantes envían como respuesta a las preguntas de voz. La idea se acabó descartando porque podría repercutir en los estudios que se quieran realizar sobre las evaluaciones de los estudiantes.
- **Escuchar respuesta**: El usuario Profesor puede escuchar las respuestas de voz que han enviado los estudiantes a las diferentes preguntas de voz. Al realizar esta acción, al Profesor también se le muestran los datos de evaluación de esa respuesta.
- **Crear grupos**: El profesor puede crear grupos de alumnos para, en caso de tener muchos alumnos, poder realizar una administración adecuada. Este planteamiento se ha sacado de las conclusiones adquiridas del artículo [Espinosa, 2016].
- **Eliminar grupos**: Además de la creación, también permite se le permite al Profesor eliminar los grupos creados. Se eliminan después de realizarse las comprobaciones necesarias.
- **Crear criterios de evaluación**: Al crear una pregunta, el Profesor puede seleccionar los criterios que se quieren evaluar en las respuestas a esta pregunta. Es por ello que, durante la creación de la pregunta, también se pueden crear los criterios que el profesor crea oportunos.
- **Eliminar criterios de evaluación**: Tal y como se pueden crear los criterios de evaluación, también se pueden eliminar. Siempre se hacen las comprobaciones oportunas antes de la supresión de un criterio.
- **Asignar preguntas a grupos**: Al crear una pregunta se tienen que seleccionar los grupos a los que se les va a asignar. Algunos de los alumnos de esos grupos recibirán la pregunta para responderla.
- **Asignar preguntas a alumnos**: El Profesor puede escoger los estudiantes que quiere que reciban la pregunta de voz. Tiene varias maneras de selección. Puede pulsar un botón para hacer una selección aleatoria de estudiantes, que es la opción por defecto. Puede hacer que se seleccionen todos los estudiantes mediante otro botón, o puede seleccionarlos de manera manual.

- **Asignar evaluaciones a alumnos**: Por cada alumno que se selecciona, hay que escoger los evaluadores que calificarán su respuesta. Para seleccionarlos, existen las mismas opciones que en la asignación de preguntas a alumnos.
- **Escoger opciones de evaluación**: El profesor puede cambiar algunas de las opciones del *bot*, como el lenguaje que utiliza o la escala de evaluación que acepta. Gran parte de las opciones modificables tienen como objetivo la evaluación de preguntas de voz.

En la última versión, se han añadido nuevas acciones posibles y se han modificado los nombres de algunas de las ya mencionadas para hacerlas más inteligibles. En la Figura 16 se pueden apreciar todas las modificaciones realizadas en el diagrama de casos de uso.

Figura 16: Diagrama de casos de uso final de Profesor

Entre las nuevas acciones se encuentran las siguientes:

- **Escoger opciones de evaluación**: En el *bot* se ha realizado un comando de administrador. Con él, el Profesor puede ejecutar una instrucción en la base de datos directamente desde el *bot*.

- **Asignar grupo a alumnos**: El profesor puede coger a un usuario cualquiera, que como mínimo haya mandado un mensaje al *bot*, y agregarle a uno de los grupos de alumnos que estén creados. De esta manera, el profesor también tiene un método para registrar a los usuarios como estudiantes.
- **Eliminar alumnos de grupo**: Tal y como se pueden agregar estudiantes a grupos, también se pueden eliminar después de haber realizado las comprobaciones pertinentes.
- **Modificar preguntas**: Ahora se permite la modificación de preguntas de voz ya creadas. Gracias a esta nueva opción ahora se pueden quitar las asignaciones de preguntas y evaluaciones realizadas con anterioridad.
- **Escuchar explicaciones sobre evaluación**: Después de la evaluación de todos los criterios, los alumnos pueden enviar una grabación de voz con la explicación de como se ha evaluado la respuesta de voz. En el panel de administración, el Profesor, puede escuchar todas esas explicaciones enviadas por los alumnos por cada una de las respuestas de voz.
- **Ver información de los estudiantes**: El profesor puede ver a quien evalúa y quien es evaluado por cada estudiante.

5.3. Diseño de interfaces

En esta sección se muestran los prototipos realizados para el panel de administración del *bot*. Todos los prototipos fueron aprobados por el cliente, antes de comenzar a realizar el modelo de dominio (Seccion 5.4). También hay que decir que la versión final del panel, no ha quedado como se propone en los prototipos, ya que se añadieron y descartaron algunas funcionalidades durante la implementación del proyecto.

Para la realización de prototipos, se pasó por dos fases fundamentales. A continuación se muestra cada una de las fases con algunas de las imágenes de los prototipos realizados.

5.3.1. Prototipos a papel

En este apartado se muestran los prototipos realizados mediante lápiz y papel. Se ha utilizado este método de creación de prototipos por ser económi-

co y fácilmente maleable. Además, como no cuesta nada la modificación, el cliente no ha dudado en pedir algunas modificaciones durante su desarrollo.

Los prototipos son muy simples, ya que tampoco se quería pasar demasiado tiempo con detalles. En las Figuras 17, 18, 19 y 20 se muestran los diseños realizados con algunas explicaciones en ellos.

Los prototipos que aparecen en la Figura 17, son los diseños de las interfaces de la creación de grupos y de las opciones del panel de administración. La interfaz de la creación de grupos, consiste en un campo de texto, para introducir el nombre del nuevo grupo, y una tabla con las preguntas de voz existentes. Las preguntas se le podrían asignar al grupo directamente desde esta ventana. En cuanto a las opciones, se plantearon unas cuantas opciones que poner en la interfaz junto con algún elemento de selección.

En la Figura 18, se muestran los diseños de la creación de una pregunta y del listado de evaluaciones. En la creación de preguntas se muestran todos los campos necesarios para la creación de preguntas. Entre ellos el texto de la pregunta, el *deadline*, una tabla con selección de criterios y la tabla en la que se seleccionan los estudiantes a los que se les asigna la pregunta. En esa tabla, hay un campo que abre una pequeña ventana con el listado de evaluadores del alumno sobre el que se ha hecho *click*. La lista de evaluaciones, es una simple tabla con los campos que se han estimado necesarios.

La Figura 19 muestra el diseño de una ventana de evaluación y el apartado de gráficos del panel de administración. En la ventana de evaluación se puede apreciar un pequeño reproductor con el que se puede escuchar la grabación, un apartado para que el profesor pudiera calificar la respuesta y dos tablas. Una contiene la media que ha obtenido en cada criterio esa respuesta, y la otra, un desglose de todas las evaluaciones que se han realizado sobre esa respuesta. En el apartado de gráficos, se muestran botones con los que se crean los gráficos por defecto. Debajo de esto está la sección en la que el usuario puede crear sus propios gráficos introduciendo algunos datos. De este apartado, el campo de la calificación fue suprimido por la alteración de datos para investigaciones. En cuanto a los gráficos, se descartó el apartado de dejar al usuario crear sus propios gráficos. Esto se debió a la falta de tiempo y a la complejidad del algoritmo.

Por último, en la Figura 20 se muestra el listado de grupos y el listado de preguntas. Estas dos interfaces consisten en tablas con los datos que se han creído necesarios y de algunos botones para añadir funcionalidad.

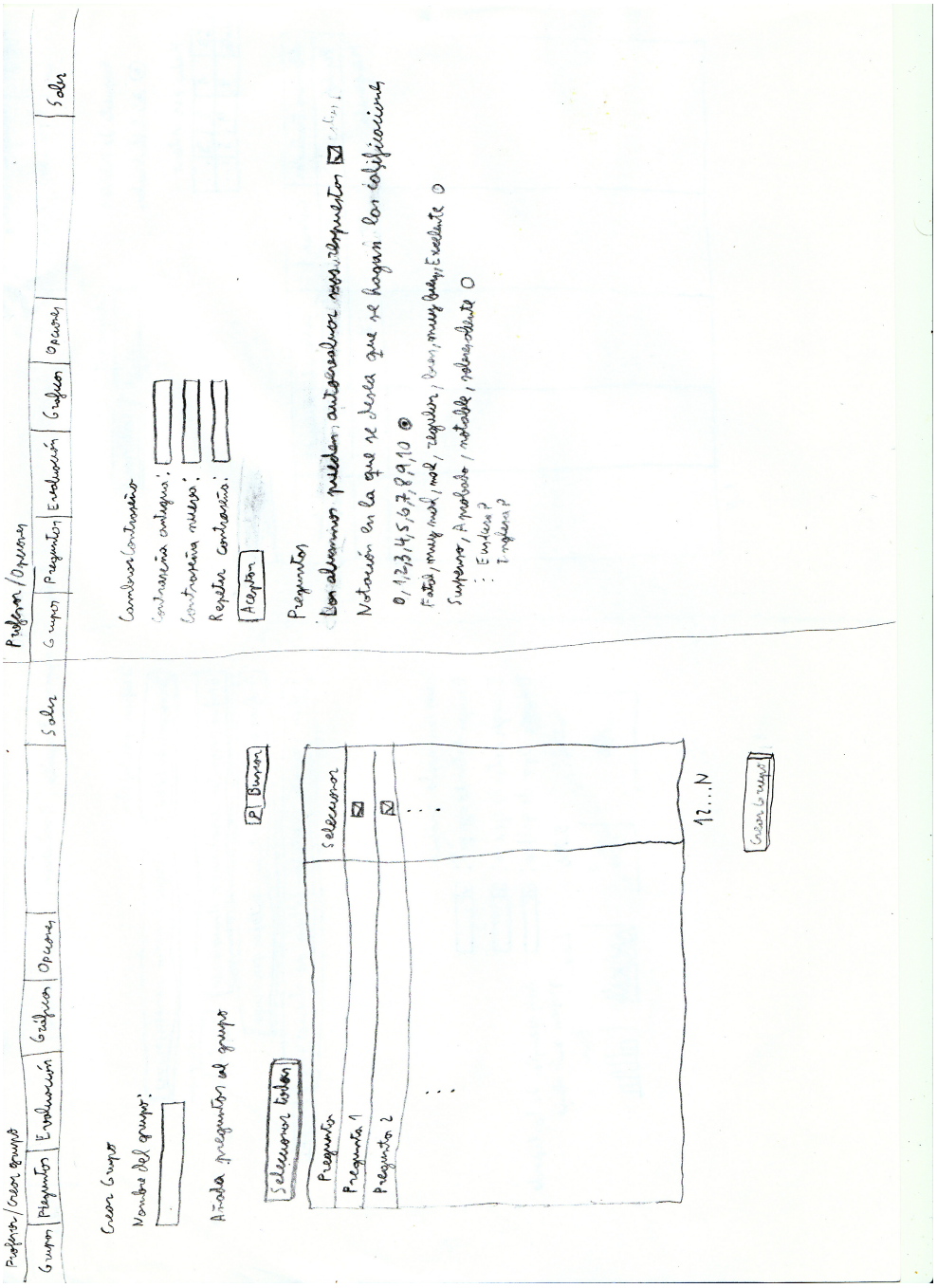


Figura 17: Prototipo a papel de creación de grupos y opciones

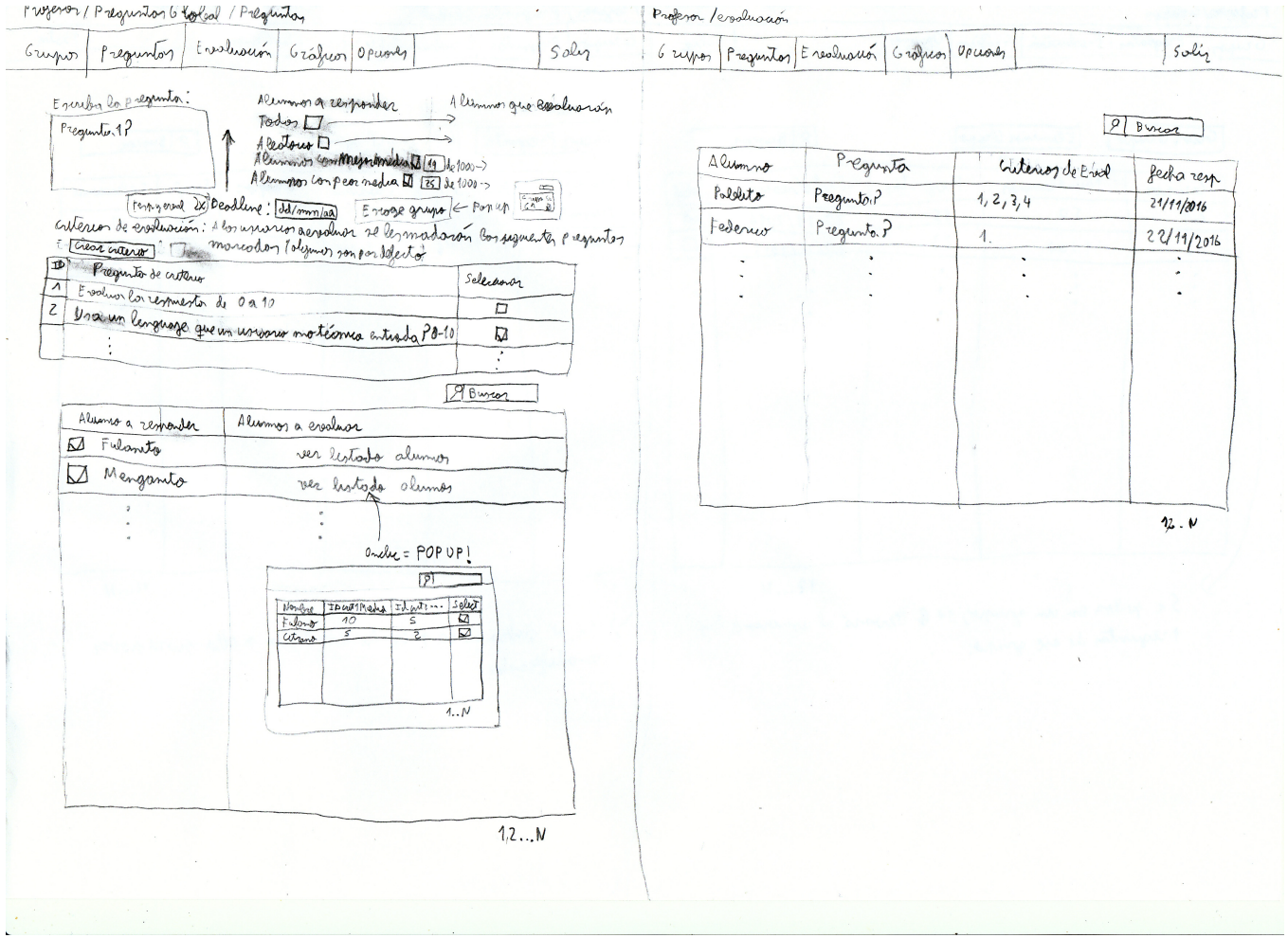


Figura 18: Prototipo a papel de creación de preguntas y listado de evaluaciones

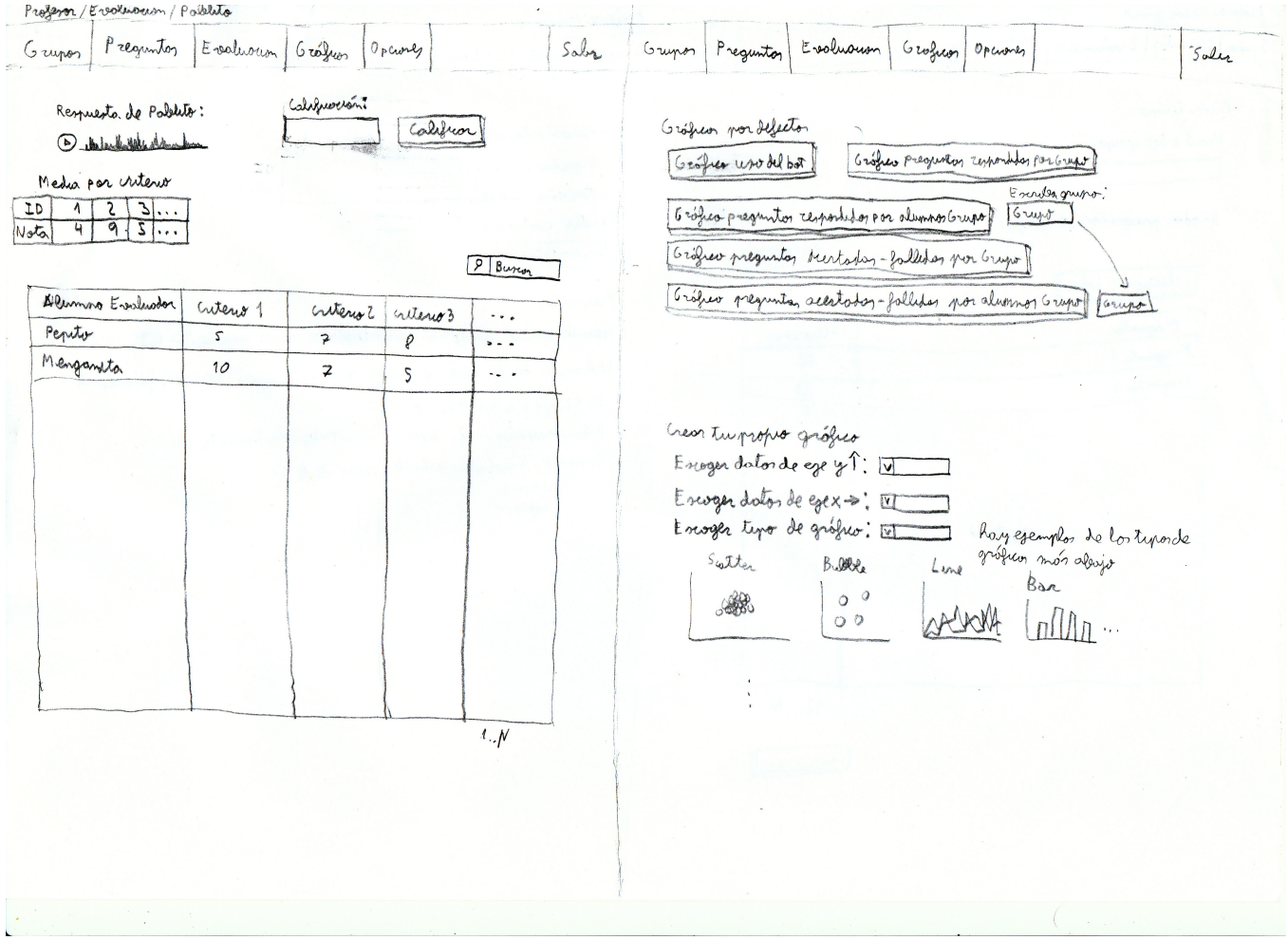


Figura 19: Prototipo a papel de una evaluación y apartado de gráficos

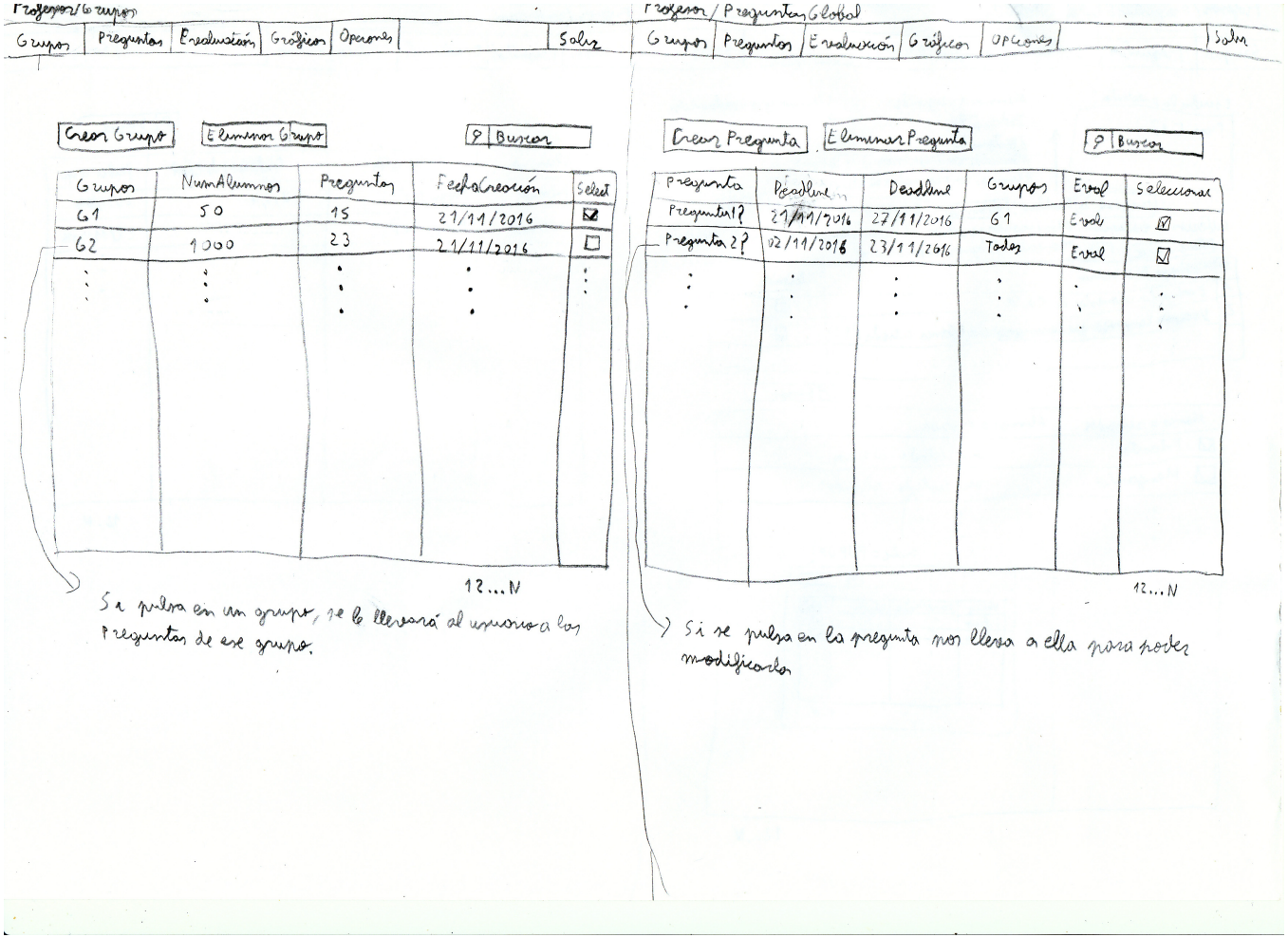


Figura 20: Prototipo a papel del listado de preguntas y listado de grupos

5.3.2. Prototipos digitales

Después de la realización de prototipos a papel, se pasaron los diseños a un formato digital, para que el cliente pudiera comprobar como quedarían al exponerlos en un navegador. Estos también fueron aprobados después de algunas pequeñas modificaciones solicitadas por el cliente.

Para no cargar el documento con las imágenes de los prototipos digitales, estos se muestran en la siguiente dirección web:

- `http://bots.ikasten.io/mikelbot/Imagenes`

En estos prototipos todas las ventanas de las diferentes funcionalidades quedan más detalladas. En ellos se aprecian los dos cambios más significativos. La selección de criterios ahora se hace con cajas de selección y, en cuanto a las tablas con textos muy extensos, se ponen puntos suspensivos después de los 50 caracteres. Mediante el atributo "*title*" de HTML, se muestra todo lo que debería estar escrito. Esto se activa al pasar el cursor por encima del texto con puntos suspensivos.

5.4. Modelo de dominio

El modelo de dominio es un modelo que representa, de manera conceptual, las entidades, relaciones y atributos relacionados con un problema específico. En esta sección, se muestra el modelo de dominio desarrollado durante este proyecto. Solo se mostrará la última versión, ya que desde la primera solo se ha añadido una entidad, junto con sus relaciones.

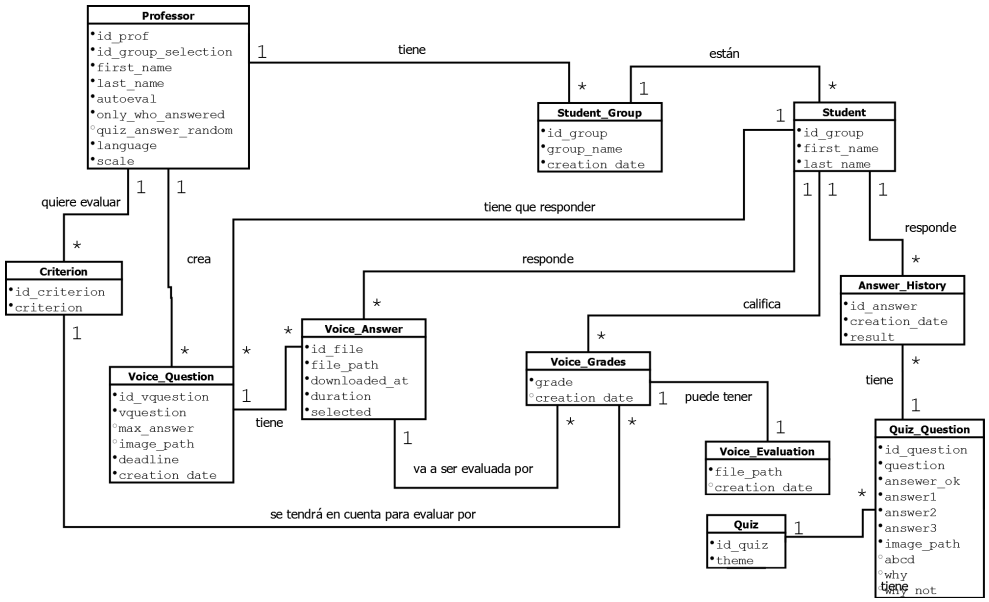


Figura 21: Diagrama de modelo de dominio

A continuación, se describe cada una de las entidades que aparecen en la Figura 21:

- ***Professor***: Es la entidad que representa al profesor. Esta entidad guarda, el nombre completo e identificador de Telegram del profesor, un identificador para poder relacionarlo con los grupos de estudiante que ha creado, y todos los demás atributos son las opciones que se ofrecen en el panel de administración del *bot*.
- ***Student_Group***: Es la entidad que encarna un grupo de estudiantes. En ella se guarda un identificador, el nombre del grupo, el identificador del profesor que lo creó y la fecha en la que se creó el grupo.
- ***Student***: Es la encarnación de un alumno que utiliza el *bot*. Solo se guarda su identificador, el identificador del grupo al que pertenece y su nombre completo de Telegram.
- ***Quiz***: Representa el *test* de un tema de la asignatura que el profesor imparte. Con guardar un identificador y el nombre del tema que contiene es suficiente.
- ***Quiz_Question***: Representa una pregunta de uno de los *tests*. En esta entidad se guarda el identificador del tema al que pertenece, un identificador propio, el texto de la pregunta, la respuesta correcta, tres respuestas incorrectas, una imagen que pueda enviarse junto a la pregunta, la posición por defecto en la que se encuentra la respuesta correcta (A, B, C o D), un texto explicando por qué la respuesta correcta es correcta y otro texto explicando por qué las demás respuestas son incorrectas.
- ***Answer_History***: Esta entidad simboliza una respuesta de un alumno a una pregunta de *test*. Se guardan todos los datos que la relacionen con la pregunta y el alumno, pero además, se guarda la fecha en la que se dio la respuesta y si esa respuesta fue correcta o no.
- ***Criterion***: Es la encarnación de un criterio de evaluación. En esta entidad se guarda un identificador, el identificador del profesor que creó el criterio y el texto que contiene el criterio a evaluar.
- ***Voice_Question***: Es la entidad que representa una pregunta de voz. En ella se guarda un identificador, el identificador del profesor que creó la pregunta, el texto de la pregunta, el máximo de respuestas que un alumno puede dar a esa pregunta, el *path* de una imagen que se quiera enviar con la pregunta, la fecha de creación y la fecha del *deadline*.

- ***Voice_Answer***: Representa la respuesta de un alumno a una pregunta de voz. Es por ello que es necesario guardar el identificador del estudiante que la ha creado y el identificador de la pregunta que responde. Estos datos se guardan para mantener las relaciones con otras entidades, pero también es imprescindible que guarde el *path* donde está almacenada la grabación de voz con la respuesta, un identificador, el registro de tiempo en el que se descargó la respuesta, la duración de la grabación y si es la respuesta seleccionada para ser evaluada o no.

- ***Voice_Grades***: Esta entidad simboliza una calificación de un criterio de evaluación que le ha dado un alumno a una respuesta de voz. Para representar todas estas relaciones, la entidad guarda el identificador del usuario evaluado, el identificador de usuario evaluador, el identificador de la pregunta que se ha respondido con la respuesta, el identificador de la respuesta y el identificador del criterio evaluado. Además de eso se guardará la nota que el evaluador ha dado al evaluado teniendo en cuenta el criterio que se evalúa.

- ***Voice_Evaluation***: Esta entidad representa una grabación de voz que envía el un evaluador a un evaluado explicando por qué ha recibido las calificaciones que ha recibido. Por ello se guarda el identificador de la respuesta que se ha evaluado, el identificador del evaluador y el *path* al fichero con la grabación.

Todas las relaciones que existen en el modelo de dominio de este proyecto son del tipo 1-*. Esto implica que, por ejemplo, un profesor tiene muchos grupos, que en un grupo hay muchos alumnos, que en un *test* tiene muchas preguntas, etc.

Todos los nombres de las entidades y de los atributos están en inglés ya que la base de datos, las clases y todo lo que implique implementación de código en cualquier lenguaje de programación, se desarrollará en este idioma. Esto se debe a que al director del proyecto le gustaría llegar a distribuir el producto final de este proyecto en otros países. Por si eso ocurre, todo lo referente a programación se ha preparado de manera que en cualquier otro país, se pueda seguir desarrollando la herramienta creada durante el proyecto.

6. Análisis y diseño

Aparte de el diagrama de clases, diagrama de estados y diagramas de secuencia, en esta sección, se expone una explicación de algunos de los elementos usados en los comandos, que se han de tener en cuenta a la hora de desarrollar un *bot* de Telegram.

6.1. Análisis de elementos

En este apartado se explican de una forma sencilla, aquellos elementos internos, que tienen que entenderse para el correcto desarrollo de un comando de cualquier *bot* de Telegram.

6.1.1. Conversación

Cuando un comando necesita de más de una interacción con el usuario, se crea una conversación. Al igual que en la vida real, en una conversación se intercambian datos hasta que uno de los interlocutores decide terminarla. También es posible cancelar una conversación sin haberla terminado. Dicho esto, se puede intuir que una conversación tiene los siguientes tres estados:

- **Parada**: Cuando un comando no necesite interactuar más con el usuario, la conversación se cambia a este estado. Se utiliza para indicar que la conversación ha terminado correctamente y que ya no es necesario volver a ejecutar el comando. Por lo menos hasta que el usuario lo vuelva a invocar.
- **Cancelada**: Una conversación puede ser cancelada por el usuario con el comando genérico /cancel. Cuando el usuario llama a este comando la conversación termina y no se vuelve a ejecutar el código del primer comando.
- **Activa**: Mientras la conversación se encuentre en este estado, cada vez que el usuario envíe un mensaje al *bot*, se ejecutará el comando que comenzó la conversación. La ejecución podrá obtener el mensaje enviado por el usuario, ya sea para guardarlo ,o para extraer datos de él.

En la Figura 22 se muestra cambio el estado de la conversación de activa a parada en su parte superior. En la parte inferior se muestra la forma en la que se cambia el estado a cancelada. Pero esta es una simple representación para poder explicar su funcionamiento. En realidad es más complejo que como aparece en la Figura 22.

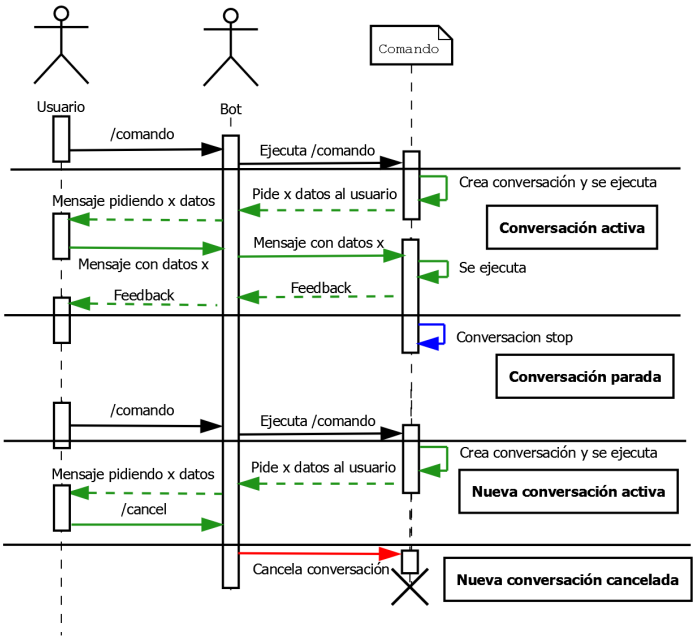


Figura 22: Exposición gráfica de una conversación

6.1.2. Máquina de estados

Es un modelo de comportamiento de un sistema que depende de los elementos que entren (entradas) y que salgan (salidas) de él. Las maquinas de estados están compuestas por conjuntos de estados, que son los intermediarios entre las entradas y las salidas. La máquina necesita las entradas para saber en qué estado se encuentra y, de esa manera, generar las salidas correspondientes.

En el caso de los comandos del *bot* que utilizan conversaciones, la maquina de estados es fundamental. Cada vez que el usuario envía un mensaje al *bot*, el comando necesita conocer el punto de la conversación en el que se encuentran. Ese punto, es un estado. Pero esto no se refiere a si la conversación está activa, parada o cancelada, sino que cada comando tiene sus propios estados.

En cada uno de los diferentes estados del comando, este ejecutará una secuencia de código u otra, para enviarle al usuario el mensaje (salida) correspondiente.

Para intentar hacerlo más simple, se ha creado la Figura 23. En ella, se muestra un comando en sus diferentes estados durante una conversación.

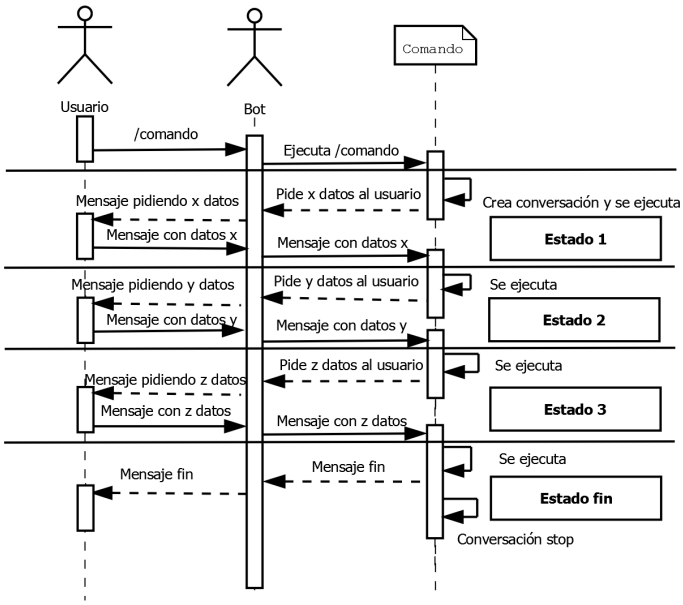


Figura 23: Explicación gráfica de estados durante una conversación

6.1.3. Notas de una conversación

Cada conversación que se crea, consta de una variable en la que se guardan datos obtenidos durante la conversación. Cada vez que el usuario envía un mensaje durante una conversación, el *bot* ejecuta completamente un comando. Cuando esto pasa, el *bot* ha perdido todos los datos anteriores al mensaje que acaba de recibir.

Las notas se utilizan para guardar los datos que se vayan a necesitar en los diferentes momentos de la conversación. Su uso más común es para guardar el estado en el que se encuentra el comando, para así, poder recuperarlo en cada ejecución y llamar al código que este estado requiera ejecutar.

6.1.4. *Inline button* y *callback query*

Los *bots* de Telegram utilizan dos tipos de botones:

- ***Keyboard button***: Este tipo de botones hacen que el teclado del dispositivo se oculte, para así dar paso a un teclado con botones que contienen un texto predefinido por el programador. Al pulsar uno, simplemente se envía un mensaje con el texto que pone en el botón. Básicamente, hace que el usuario no tenga que escribir ese texto y además incita al usuario a pulsarlos. Gracias a esto se evitan muchas molestias al usuario.

- ***Inline button***: Estos botones aparecen directamente en la zona de los mensajes. Pueden ser más estéticos, pero son algo complicados a la hora de implementar sus funcionalidades. Al pulsar uno de estos botones, se realiza una llamada a un comando de sistema denominado "*callback query*". Este comando ha de encargarse de gestionar las funciones del botón que se ha pulsado.

En la Figura 24 se muestran los dos tipos de botones. Los botones con el texto *"Next question"* y *"Exit"* son botones de tipo *Keyboard button*, en cambio el botón que contiene el texto *"Test JavaScript Básico (8)"* es un *Inline button*.

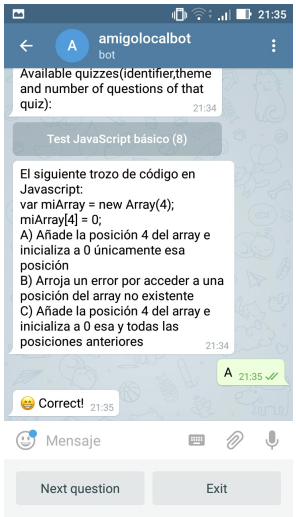


Figura 24: Tipos de botones usados en Telegram

6.2. Diagrama de estados

En este apartado, se muestran y explican los diferentes estados por los que pasa el *bot* creado en este proyecto. En el diagrama expuesto en la Figura 25, se pueden apreciar las situaciones en las que se encuentra el *bot* durante la ejecución de cada uno de sus comando.

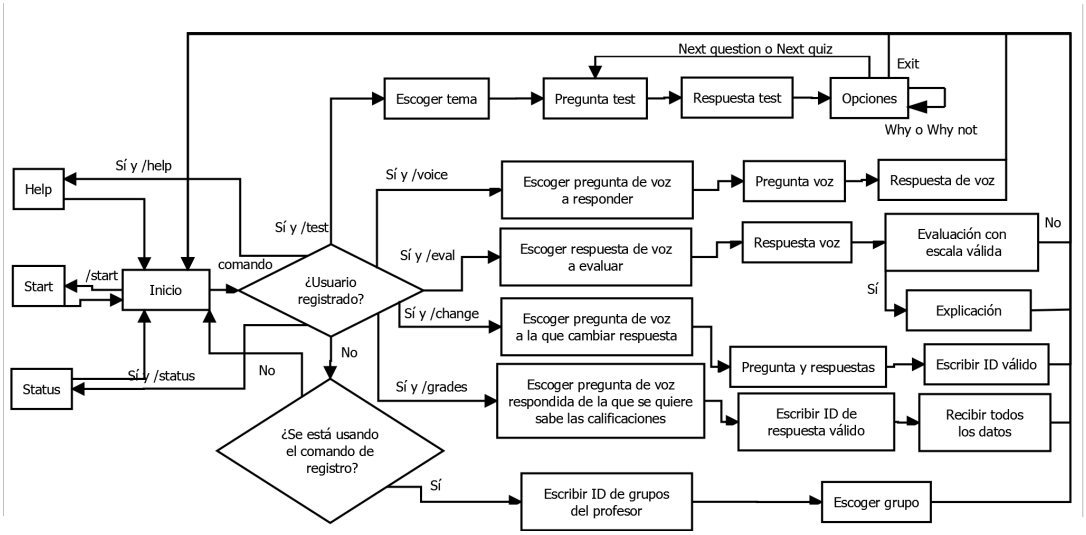


Figura 25: Diagrama de estados del bot

- **Inicio**: Estado inicial que se da al abrir el *chat* con el *bot* en Telegram. Para llegar a los demás estados, el usuario ha de ejecutar uno de los comandos del *bot*
- ***Start***: En este estado, el *bot* envía un mensaje de bienvenida al usuario.
- ***Help***: Si el usuario está correctamente registrado, el *bot* envía un listado con los comandos disponibles junto con una pequeña descripción.
- ***Status***: Si el usuario se encuentra registrado, en este estado, el *bot* le envía las estadísticas de su evolución en los *tests*.
- **Escoger tema**: Cuando es usuario ejecuta el comando /test estando correctamente registrado, se le manda un listado de los temas de los cuales puede realizar un *test*.
- **Pregunta *test***: Después de que el usuario ha escogido un tema, se le envía una pregunta de *test* sobre ese tema.

- **Respuesta *test***: Cuando un usuario responde a una pregunta de *test*, el *bot* le responde directamente si la respuesta ha sido correcta o incorrecta. Ademas le envía al usuario las posibles acciones que puede realizar.
- **Opciones**: En este punto, el usuario escoge una opción, que puede ser salir, la cual le lleva al estado Inicio, puede pedir ir a la siguiente pregunta o siguiente *test* (esto depende de si el *test* ha finalizado) o puede volver al mismo estado. Esto último es en caso de que quiera saber por qué la respuesta correcta es correcta o por qué las demás respuestas son incorrectas.
- **Escoger pregunta de voz a responder**: Se llega a este estado después de haber comprobado que el usuario está registrado como estudiante. Cuando el usuario llama al comando /voice, recibe un listado con las preguntas de voz que tiene asignadas y que todavía no ha pasado su *deadline*.
- **Pregunta voz**: Al escoger una pregunta de voz, al usuario se le envía la pregunta y se le pide que responda mediante una grabación de voz.
- **Respuesta de voz**: Cuando el usuario envía la respuesta de voz, el *bot* la descarga en una carpeta del servidor en el que se encuentra. Después envía al usuario al estado inicial después de enviarle un mensaje como *feedback*.
- **Escoger respuesta de voz a evaluar**: Después de comprobar que el usuario está correctamente registrado, el comando /eval le envía una lista de preguntas. Por cada una de esas preguntas, existe una respuesta de voz de un compañero del usuario esperando a ser evaluada por este.
- **Respuesta voz**: En este estado se le envía al usuario la pregunta, respuesta de voz y el criterio de evaluación que ha escogido evaluar del listado del estado anterior.
- **Evaluación con escala válida**: Cuando el usuario envía la calificación de todos los criterios, el *bot* le pregunta si quiere dar una explicación de voz con las evaluaciones. Si el usuario se niega, lo manda al estado inicial. De lo contrario le envía al estado Explicación.
- **Explicación**: En este estado se le pide al usuario una grabación de voz. En ella, ha de explicar cual ha sido la razón de dar las calificaciones que otorgadas a la respuesta de voz evaluada. Después de descargar la grabación, se envía al usuario al estado inicial.

- **Escoger pregunta de voz a la que cambiar la respuesta**: Al terminar de comprobar que el usuario está registrado. Si lo está y el comando que se ha llamado ha sido /change, el *bot* le envía un listado de las preguntas de voz que ya han sido respondidas. Esas son las preguntas en las que el usuario puede cambiar su respuesta seleccionada.

- **Pregunta y respuestas**: En este estado, al usuario se le envía la pregunta y las grabaciones de voz con las que respondió. Para seleccionar una, el usuario ha de enviar un mensaje con el identificador de la respuesta que quiere seleccionar para ser evaluada. El identificador se envía junto a la respuesta.

- **Escribir ID válido**: Cuando el usuario escoge una respuesta, el *bot* guarda los datos y envía al usuario al estado inicial.

- **Escoger pregunta de voz respondida de la que se quiere saber las calificaciones**: Cuando el usuario está correctamente registrado y ha ejecutado el comando /grades, se le envía un listado de preguntas en las cuales su respuesta ha sido evaluada.

- **Escribir ID de respuesta válido**: Para poder seleccionar una de las preguntas, el usuario ha de escribir el identificador de esta. Al hacerlo, se le envía la pregunta que se ha evaluado y la respuesta de voz que escogió para que fuera evaluada.

- **Recibir todos los datos**: En este estado se le envía al usuario todos los datos de la evaluación. Un listado con todas las calificaciones que ha recibido su respuesta y otro con las explicaciones de voz que se dieron durante las evaluaciones.

- **Escribir ID del profesor**: En este estado se le pide al usuario que introduzca el identificador de grupos de su profesor. Esto es en el caso que se haya invocado el comando /register.

- **Escoger grupo**: Después de que el usuario escriba un identificador correcto, el *bot* le envía el listado de los grupos en los que se puede integrar. Al escoger uno de los grupos, los datos se guardan y se envía al usuario al estado inicial.

6.3. Diagrama de clases

Es un diagrama en el que se representa la estructura de un sistema. Para ello, se muestran las clases, métodos, atributos y relaciones de manera estática.

Durante el desarrollo del proyecto se realizaron bastantes modificaciones en el diagrama de clases. Esto se debió a las nuevas necesidades que el cliente, Juanan Pereira, identificó y a los sucesivos cambios de versión que sufrió la librería php-telegram-bot. Durante el proyecto esta librería ha pasado de la versión 0.35, que fue desde la que se comenzó a desarrollar, a la versión 0.42, en la que se ha dado por finalizado el apartado de desarrollo del proyecto. En la Figura 26 se muestra la versión final del diagrama de clases correspondiente al código del *bot*.

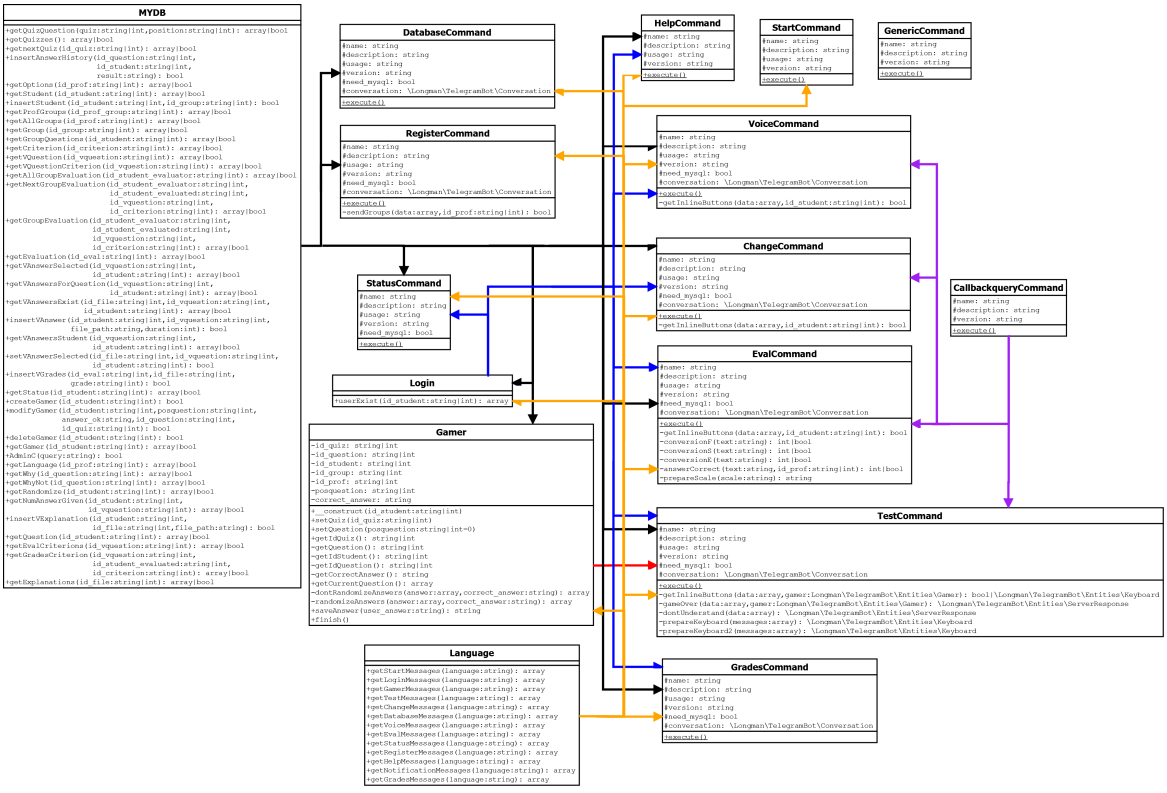


Figura 26: Diagrama de clases
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Analisis/DiagramaClases.png`.

A continuación se procede a explicar cada una de las clases del diagrama:

- ***MYDB***: Esta clase es la encargada de realizar las conexiones con la base de datos de MySQL en el servidor. Todos los métodos de esta clase son utilizados por las demás. Ya sea para insertar, modificar, seleccionar o eliminar elementos de la base de datos.

- ***Login***: Esta clase se encarga de comprobar si el usuario que está accediendo al *bot* esté registrado como estudiante. Casi todos los comandos hacen uso de esta clase. Gracias a ella se mantiene la seguridad en el *bot*.

- ***Language***: Todos los comandos usan esta clase. Esta clase está conectada a los diferentes ficheros que contienen los textos de los mensajes en diferentes idiomas. Cada vez que un comando necesita enviar un mensaje, tiene que llamar a algún método de esta clase pasando como parámetro el nombre del fichero. Por ejemplo, si se quieren los textos en español, el parámetro será *Spanish*.

- ***DatabaseCommand***: Esta clase permite la ejecución del comando de administrador /database.

- ***RegisterCommand***: Es la clase que contiene los métodos necesarios para la ejecución del comando de registro /register.

- ***HelpCommand***: Esta clase es genérica de la librería php-telegram-bot, pero a sido modificada debido a los requisitos del cliente. La clase implementa todos los métodos necesarios para que el comando /help envíe un mensaje, al usuario que lo ejecute, con un listado de los comandos disponibles.

- ***StartCommand***: Con esta clase, se permite enviar al usuario un mensaje de bienvenida al ejecutar el comando /start.

- ***GenericCommand***: Esta es una clase genérica de la librería php-telegram-bot. Ha sido modificada debido a un error que lanzaba al ejecutar algunos de los nuevos comandos que se han desarrollado durante el proyecto. Esta clase implementa los métodos que se llaman por defecto cuando se hace una llamada a un comando inexistente. Simplemente envía un mensaje avisando al usuario que el comando introducido no existe.

- ***VoiceCommand***: Esta clase contiene los métodos necesarios para la ejecución del comando /voice, que permite al usuario responder las preguntas de voz que tiene asignadas.

- ***ChangeCommand***: Gracias a las funciones de esta clase, se permite al usuario cambiar su respuesta de voz seleccionada para que esta sea la que se evalúe.

- ***EvalCommand***: Cuando el usuario ejecuta el comando /eval, se ejecutan los método de esta clase. Estos, permiten al usuario evaluar las respuestas de voz que se tengan asignadas.

- ***TestCommand***: Los métodos de esta clase se ejecutan al llamar al comando /test. Este comando permite al usuario realizar los *tests* que el *bot* tenga disponibles.

- ***Gamer***: Esta clase se usa como intermediaria en los *tests*. Guarda y prepara los datos necesarios para que el comando /test funcione correctamente. Entre los datos que guarda se encuentran el tema del que se esta realizando el *test* y la pregunta que se está realizando en cada momento. También se encarga de dar una posición (A, B, C o D) a las respuestas de cada pregunta. Además se ocupa de la interacción con la base de datos, para mandar guardar en ella el historial de respuestas dadas por el usuario que esté haciendo el *test*.

- ***StatusCommand***: Gracias a las funciones que esta clase implementa, el usuario puede ver las estadísticas obtenidas en los *tests* mediante el comando /status.

- ***GradesCommand***: Los métodos de esta clase implementan la funcionalidad del comando /grades, que permite al usuario ver las calificaciones que ha recibido una de sus respuestas de voz.

- ***CallbackqueryCommand***: Este comando es ejecutado al pulsar un *Inline button*. Se encarga de dar funcionalidad a todos los botones de este tipo que aparecen durante la ejecución de los comandos del *bot*.

Todas las clases descritas en este apartado han sido creadas o modificadas durante el desarrollo del proyecto. También es importante decir que detrás del funcionamiento del *bot*, hay muchas clases que implementan las funcionalidades que este tiene por defecto. Pero, al no haber realizado ninguna modificación en ellas, no se han añadido al diagrama de clases.

También es necesario comentar que no se ha realizado ningún diagrama de clases del panel de administración. Esto es debido a que no utiliza ninguna clase, solo son ficheros de tipo .js, .php y .html que contienen algunos métodos sueltos y el código necesario para que la página web funcione correctamente.

6.4. Diagramas de secuencia

Los diagramas de secuencia tienen como objetivo representar gráficamente la interacción entre los objetos de un sistema. Para este proyecto, se han creado los diagramas de los comandos que se consideran los más complejos e importantes del *bot*.

Los diagramas de secuencia que se muestran en las Figuras 27, 28 y 29, son las versiones finales. Se tuvieron que realizar algunos cambios en sus estructuras debido a las nuevas necesidades del cliente y a algunas mejoras de eficiencia.

Como ya se ha comentado en este capítulo, los comandos se componen de diferentes estados. Para simplificar la explicación de los comandos se detallará lo que se realiza en cada estado.

6.4.1. Comando */test*

En las Figuras 27 y 28, se muestra el diagrama de secuencia del comando /test. Antes de que se ejecute el código que contienen los estados que forman este comando, se crea la conversación con el usuario y se crea el *Gamer*. Como se ha comentado en la sección anterior, el *Gamer* se utiliza como un intermediario que guarda datos sobre el *test* que el usuario realice. Además durante este diagrama se detallaran algunas de sus funciones.

Durante la creación del *Gamer*, se llama a la función del *Login* para comprobar si el usuario está registrado como estudiante. En caso de no estarlo el comando termina su ejecución. Después de comprobar que el usuario es un estudiante, se empiezan a ejecutar los siguientes estados:

- **Estado 0**: En este estado se realiza una búsqueda de los *tests* disponibles. Después, se crea una lista de *Inline buttons* asociados a cada uno de esos *tests*. Para terminar se añaden los datos necesarios a las notas de la conversación y se guardan al llamar al método update().

- **Estado 1**: Antes de llegar a este estado, el usuario ha tenido que pulsar uno de los *Inline buttons* asociado a un *test*. Al realizar esa acción se llama a la clase *Callbackquery*. Esta clase se encarga de gestionar que botón se ha pulsado y añade algunos datos a las notas de la conversación con el usuario. Después, desde esa clase, se realiza la llamada al comando /test. Este recibe todos los datos y, de ellos, guarda en el *Gamer* el identificador del *test* que el usuario quiere realizar.

- **Estado 2**: Durante este estado se recogen los datos de la pregunta de *test* del tema que escogió el usuario. Después, se busca en la base de datos la opción del profesor que indica si se quiere que las respuestas del *test* se distribuyan de manera aleatoria en las posiciones A, B, C y D. En caso de que la opción que ha devuelto la base de datos nos indique que está activada, se llama a un método que sitúa las respuestas en posiciones aleatorias. En caso de no estar activada la opción, primero se posiciona la respuesta correcta en su posición por defecto y después las incorrectas se sitúan en las demás posiciones libres. Por último, se modifican los datos de la base de datos y las notas de la conversación. En este estado se le envía la pregunta al usuario para que la responda.

- **Estado 3**: Cuando el usuario responde a una pregunta de *test*, se entra en este estado. Primero se comprueba que el usuario haya pulsado uno de los botones que se le mandaron para responder. Cuando se ha comprobado que ha pulsado uno, se comprueba si la respuesta que va asociada al botón es la correcta. Después, se le manda un mensaje al usuario indicándolo. Para terminar, se prepara un teclado de botones con las opciones que puede realizar el usuario y se modifican los datos de las notas de la conversación.

- **Estado 4**: Cuando el usuario pulsa una de las opciones después de una pregunta, este estado ejecuta el código correspondiente. Si se pulsa el botón que indica que se quiere ir a la siguiente pregunta, se vuelve al estado 2 para que le envíe otra pregunta. Si se pulsa el botón en el que pone "Por qué" (depende del idioma), se le envía al usuario un mensaje sacado de la base de datos que explica por qué la respuesta correcta es correcta. Lo mismo pasa con el botón "Por qué no", exceptuando que en este caso el mensaje indica por qué las respuestas incorrectas no son correctas. En caso de pulsar la opción "Salir", se termina la ejecución del comando y se para la conversación.

- **Estado 5**: Este estado es igual al anterior, exceptuando que este se da solo cuando se han terminado las preguntas del último *test* disponible.

Es por ello que ya no está la opción de ir a la siguiente pregunta.

- **Estado 6**: Este estado también se parece mucho al estado 4. En este caso la pregunta que se contestó antes de llegar aquí, fue la última pregunta sobre un tema. Es por ello que en este apartado se tienen las opciones de repetir el *test* o de pasar al siguiente. Este último hace que se ejecute el código del estado 7.
- **Estado 7**: Este estado se encarga de preparar el siguiente *test*. Para ello, se consigue el *test* actual y se busca el siguiente en la base de datos. Para terminar, se modifican las notas de la conversación.

6.4.2. Comando */voice*

En la Figura 29 se muestra el diagrama de secuencia del comando /voice. Al comenzar la ejecución de este comando solo se crea la nueva conversación con el usuario. Después comienza a ejecutarse el código de su maquina de estados.

- **Estado 0**: En este estado, primero se mira si el usuario esta registrado como estudiante. Después, se hace una llamada al método que prepara el listado con los *Inline buttons*. Cada uno estos está asociados una de las preguntas de voz que el usuario tiene asignadas. Por último, se le envía al usuario un listado de mensajes con las preguntas de voz que puede responder y otro con los *Inline buttons* y se modifican las notas de la conversación.
- **Estado 1**: Cuando el usuario pulsa un *Inline button*, la clase *CallbackqueryCommand* se encarga de gestionar las funciones de ese botón. Para ello, añade elementos a las notas de la conversación y ejecuta el comando /voice. Cuando el comando recibe los datos de las notas, busca en la base de datos la pregunta que el usuario quiere responder. Para terminar, el *bot* le envía al usuario la pregunta junto a un mensaje que le indica que responda mediante una grabación de voz.
- **Estado 2**: Para llegar a este último estado, el usuario tiene que enviar una grabación de voz. Cuando lo hace, se realizan las comprobaciones necesarias para confirmar que es una grabación. Después, se recogen los datos del usuario y de la pregunta de la base de datos. Por último, se descarga la grabación en una carpeta del servidor y si todo ha ido bien, se introducen los datos en la base de datos.

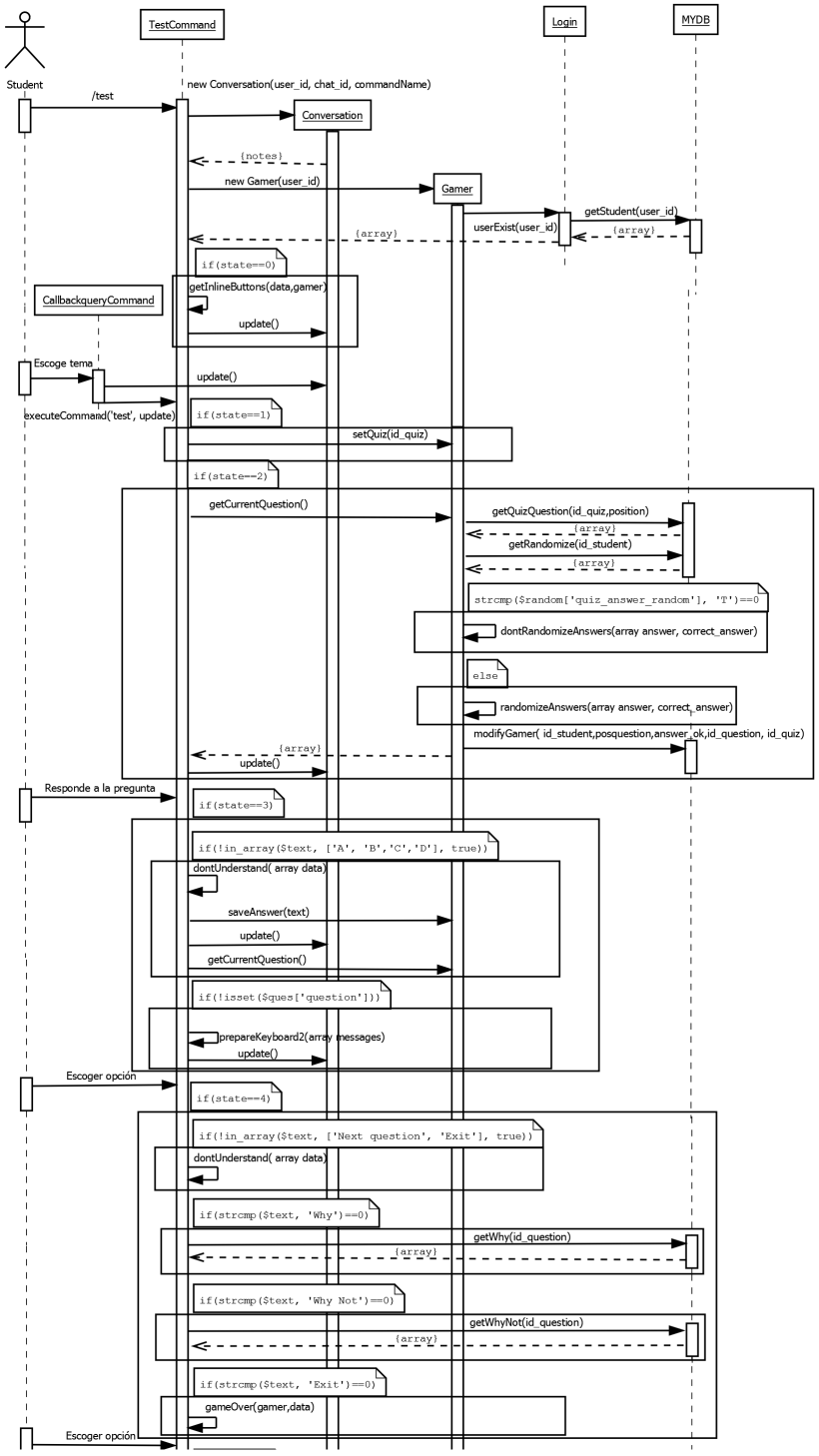


Figura 27: Diagrama de secuencia del comando /test (1)

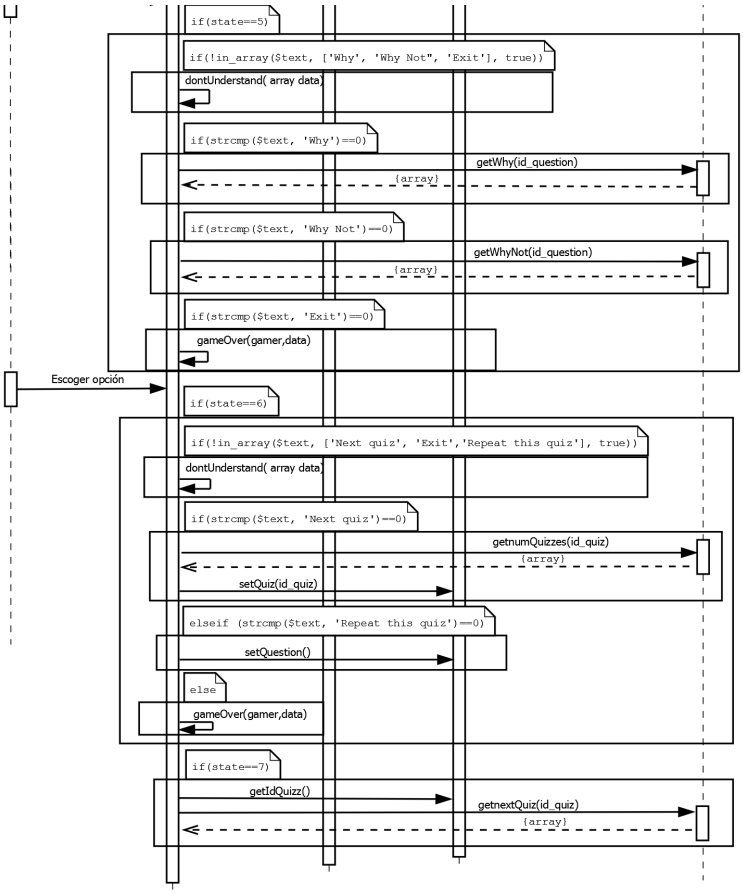


Figura 28: Diagrama de secuencia del comando /test (2)

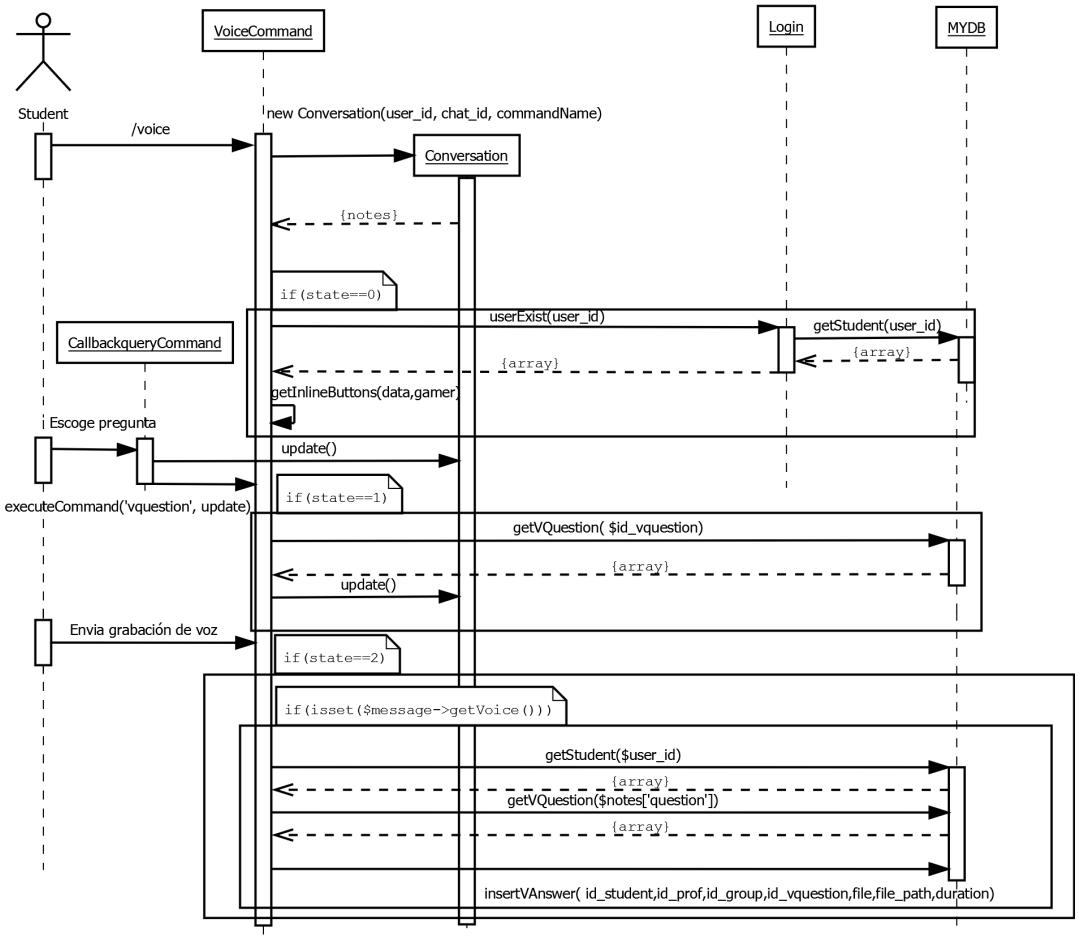


Figura 29: Diagrama de secuencia del comando /voice

7. Elección de lenguajes y tecnologías

A la hora de desarrollar un proyecto, todo el mundo sabe cuán importante es el hecho de escoger las herramientas adecuadas. La mala selección de estas, no solo implica la pérdida de tiempo de todos los integrantes del proyecto, sino que puede suponer la ruina económica de este o, en casos extremos, de una empresa completa.

Al tratarse de un proyecto en el que toda la realización del producto final implica la implementación de código, es realmente importante hacer una selección de los lenguajes de programación, librerías y programas a utilizar. Una correcta elección de estos puede ahorrar muchos problemas y horas de trabajo. Es por ello que en este proyecto se plantearon varios lenguajes con los que trabajar. En esta sección se muestran los planteamientos que se realizaron a la hora de escoger un lenguaje de programación.

7.1. Elección del SDK

SDK son las siglas de *software development kit*, que hacen referencia al conjunto de herramientas que un desarrollador de *software* utiliza a la hora de crear aplicaciones para un sistema concreto.

Al comienzo de este proyecto se plantearon dos opciones, usar alguno de los SDK que el propio Telegram ofrece o usar el SDK que usó Pablo, el desarrollador del proyecto predecesor a este. Pablo desarrollo su *bot* con *Microsoft Bot Framework*, un SDK que ofrece la empresa Microsoft, que permite que el *bot* desarrollado sea compatible con otras aplicaciones de mensajería. Además, ofrece actualizaciones cada poco tiempo y tiene un gran soporte técnico.

Cuando me plantearon esta opción pensé que podría ser una buena herramienta. Para comprobar cómo acabó el proyecto de Pablo, estuve presente en la defensa de su TFG. Cuando vi lo lento que funcionaba el *bot*, me eché para atrás con la idea de usar el mismo SDK que él utilizó. En la sección de pruebas (Sección 9.3) se pueden llegar a apreciar el tiempo medio que tarda cada uno de sus comandos.

Descartada esta idea, estaba claro que el SDK que usaría para el desarrollo del *bot*, sería uno de los que se ofrecen en la página de Telegram[4]. En este punto, el problema era cuál usar. Pero para ello, antes necesitaba saber qué lenguaje de programación utilizaría.

[4]SDKs de Telegram: `https://core.telegram.org/bots/samples`

7.2. Elección del lenguaje de programación

Cuando le pregunté a Juanan Pereira, el director de este proyecto, por los proyectos de TFG en los que se desarrollan *bots*, me comentó que él solo trabajaba con PHP, Python y Node.js. Estos son tres de los lenguajes en los que se pueden desarrollar los *bots* de Telegram, pero en total hay doce posibles.

El hecho de que Juanan solo trabajara con tres, me ayudó a reducir las opciones a esos tres lenguajes. De esta manera, podría obtener información de él en caso de tener algún error del que no encontrara la solución.

Las opciones se redujeron a una en un santiamén. Se escogió PHP como lenguaje de programación por dos motivos principalmente. El primero es que hasta el día de la presentación, Juanan me mandó tareas para aprender cómo funciona un *bot* de Telegram. Él, me envió unos tutoriales que realizó para los estudiantes que quieren trabajar con *bots* de Telegram. En estos tutoriales se usa un SDK de PHP, y como además, los comandos que realicé para aprender los desarrollé en este lenguaje, decidí que lo usaría para implementar el *bot* del proyecto.

La otra razón por la que escogí PHP fue porque Juanan me pasaría la parte que él tenía desarrollada del *bot* en este lenguaje de programación. Aunque al final no se pudo utilizar ese código por diferencias de versiones, me ayudo a escoger PHP como lenguaje de programación. Y de ello también escogí la librería php-telegram-bot (que en aquella época se denominaba Akalongman).

7.3. Programa para la conversión de ficheros de audio

Durante el desarrollo, surgió un problema inesperado que se comentará en profundidad en el capítulo de desarrollo (Sección 8). Este problema provocó la búsqueda de una forma de convertir un fichero de audio con el formato .oga a .ogg.

Cuando se comentó este problema en una reunión con el director del proyecto, este me recomendó el uso del programa FFmpeg. Este programa se puede usar como reproductor de música, pero también tiene implementadas funcionalidades de conversión de tipos. Indagué acerca de este programa y encontré una librería en PHP con el mismo nombre. Gracias a esa librería se puede interactuar con el programa FFmpeg directamente mediante código en PHP.

Como se planteaba fácil la conversión de ficheros, decidí usar esta librería para solucionar el problema que tenía. Además, una de las ventajas que le encontré a este programa es que permite la conversión de ficheros que necesitaba. Ya que no hay muchos programas que interactuen con ficheros .oga.

7.4. Librería para la creación de gráficos

Cuando comencé con el proyecto, el director y cliente , Juanan, me comentó que una de las necesidades que tenía claras era la de poder realizar gráficos de los datos que el *bot* guardase. Por ello, me recomendó el libro [Dale, 2016]. En este libro, Kyran Dale cuenta como combinar el lenguaje de programación Python y la librería D3 para la visualización de datos.

D3 es una librería gratuita hecha en JavaScript con la que se pueden crear gráficos interactivos. Se pueden realizar desde gráficos de barras hasta lo que puedas llegar a imaginar.

Buscando en Internet, encontré la página web de la librería[5]. Los ejemplos que hay en esa página me dejaron fascinado. Uno de sus ejemplos, es un gráfico circular en el que se representan teclas de piano. El caso es, que ese piano se puede tocar haciendo *click* en las teclas. ¡Y todo ello creado con D3!

Aparte de esta librería también busqué otras opciones como Charts.js o Highcharts para tener más referencias. Pero mientras buscaba, recordé el nombre de la aplicación con la que Juanan había realizado algunos de los gráficos que enseñó como prototipo. Los había hecho con Plotly, una aplicación web gratuita con la que se pueden realizar gráficos interactivos. Buscando un poco acerca de esta aplicación encontré [Ali et al., 2016] y [Prakash, 2016], dos artículos científicos que hablan del uso de Plotly para mostrar gráficos de *big data*.

Teniendo toda esta información, y después de saber que Plotly, internamente, está implementado con la librería D3, me decidí por la utilización de una librería de esta aplicación para mostrar los gráficos en el panel de administración del *bot*.

[5]Página oficial de D3: `https://d3js.org/`

7.5. Tablas del panel de administración

Cuando realicé los prototipos digitales y se los mostré al director, le comenté que las tablas que aparecen en el panel de administración quería que se pudieran ordenar y se pudiera buscar en ellas. Al decirle esto, él me recomendó usar la librería DataTables[6].

DataTables es una librería implementada con JQuery, una librería de JavaScript que facilita la interacción con los elementos del lenguaje HTML. DataTables permite una mayor interacción con tablas, ya que les añade muchísimas funcionalidades. Permite repartir la tabla en páginas para que esta no sea gigantesca, búsquedas y ordenación del contenido por defecto.

Viendo que usando esta librería casi todo lo referente a las tablas estaba hecho, decidí que la usaría para el desarrollo de las tablas de datos del panel de administración. Además, al mirar su página web, vi que tenía muchos ejemplos y un gran soporte.

[6]Página web oficial de DataTables: `https://www.datatables.net/`

8. Desarrollo

En este capítulo se van a detallar los pasos que se siguieron a la hora de desarrollar el *bot* y su panel de administración web. Además, se detallará los pasos seguidos para implementar las necesidades que el cliente ha ido encontrando durante el desarrollo del proyecto. También se explicarán los problemas encontrados y las modificaciones que han sido necesarias para solucionarlos.

Como durante el desarrollo del proyecto han surgido nuevas necesidades y problemas técnicos que no se habían tenido en cuenta, el alcance y la planificación han sido severamente alterados. Estas modificaciones se detallan en el apartado de conclusiones (Sección 10).

8.1. Necesidades del cliente

Antes de comenzar a comentar cuál fue el proceso que siguió el desarrollo, se han de explicar las necesidades que el director y cliente, Juanan Pereira, identificó mientras se realizaba el diseño y la implementación del proyecto. Para ello, primero se explicarán las que afectaron al desarrollo del panel de administración web y después las que afectaron al *bot*. Aunque el desarrollo de las herramientas fue en orden inverso, es necesario explicarlo en este orden para poder entender las funcionalidades del *bot*, ya que el panel de administración es el pilar que lo sostiene.

8.1.1. Panel de control web

El panel de control, es una página web desde la que se pueden controlar las opciones, los alumnos, las preguntas de voz y las evaluaciones que se realizan con el *bot*. Para poder hacerlo, se han desarrollado las siguientes funcionalidades a petición del cliente:

- ***Login***: Para entrar al panel de administración del *bot*, se ha de estar registrado como profesor en la herramienta. Esta funcionalidad comprueba los datos que se introducen en los campos del *login* con los datos almacenados en la base de datos. Si el usuario está registrado, le deja entrar a su panel de administración y crea una sesión que caduca al de 24 minutos por defecto (esto puede ser alterado en las opciones de PHP). De esta manera, si el usuario no cierra su sesión o la sesión no caduca, al volver a entrar a la página del *login* no tendrá que volver

a introducir los datos, ya que se le reenvía directamente a una página dentro del panel de administración.

En el caso de que el usuario haya introducido mal algún dato o que no esté registrado, se muestra un mensaje de error indicando que el usuario o contraseña son incorrectos.

Para poder entrar en el panel de administración, hay que poner el nombre de usuario de Telegram (first_name en la tabla user) y el identificador de Telegram como contraseña. Para conocer estos datos, el usuario se puede dirigir a la base de datos y mirarlos o ejecutando el comando /whoami con en *bot*. Este comando venía por defecto con la librería php-telegram-bot hasta la versión 0.42. /whoami permite al usuario conocer su nombre y apellido, su identificador y su nombre de usuario de Telegram.

- ***Logout***: Tal y como existe un *login*, se ha creado un *logout*. Consiste en destruir la sesión iniciada en el *login*.
- **Mostrar todas las preguntas de voz creadas por el usuario**: Para mostrar todas las preguntas, se ha creado una tabla con los siguientes campos:
  - ***ID***: Identificador que se le ha dado en la base de datos a la pregunta.
  - ***Question***: 50 primeros caracteres de la pregunta. Al pasar el cursor por encima se puede ver la pregunta completa gracias al atributo *title* de HTML.
  - ***Creation date***: Fecha de la creación de la pregunta.
  - ***Deadline***: Fecha en la que termina el plazo para responder a la pregunta.
  - ***Students who anwer***: Lista de los nombres de los estudiantes que tienen que responder a la pregunta. Como en todos los listados de las tablas del panel de administración, en este campo solo aparecen los 50 primeros caracteres, pero si se pasa el cursor por encima del campo se muestra la lista completa. Este campo se añadió un tiempo más tarde que la verdadera creación de la tabla a petición del cliente.
  - ***Number of students who answer***: Número de estudiantes que tienen que responder a la pregunta. Este campo se añadió junto al anterior para no tener que contar los alumnos que han de responder a las preguntas.

- ***Groups that answer***: Está formado por un listado de los nombres de los grupos a cuyos alumnos se les ha asignado la pregunta.
- ***Criterions to evaluate***: Este campo contiene un listado de los criterios que van a ser evaluados en las respuestas a la pregunta de voz. Aparecen todos los criterios separados por comas.
- ***Select***: En este campo hay un *checkbox* para seleccionar las preguntas a eliminar. Este campo está relacionado con los botones "*Select all*" y "*Unselect all*", que, cuando son pulsados, seleccionan o deselcionan todos los *checkbox* que hay en este campo de la tabla. Estos botones aparecen en todas las tablas en las que existe un campo "*Select*".

- **Mostrar todos los grupos de alumnos creados por el usuario**: Todos los grupos creados por el usuario aparecen en una tabla con los siguientes campos:

  - ***ID***: En este campo aparece el identificador que la base de datos le ha otorgado al grupo.
  - ***Group name***: Nombre del grupo. Por si es demasiado largo, solo se escriben los primeros 50 caracteres. Si se pasa el cursor por encima del campo aparece el nombre completo del grupo.
  - ***Number of students***: En este campo aparece el número de estudiantes que pertenecen al grupo.
  - ***Number of questions***: En este campo aparece el número de preguntas que tiene asignadas cada grupo
  - ***Creation date***: Este campo muestra la fecha en la que se creó cada grupo.
  - ***Select***: Campo con los *checkbox* para seleccionar las preguntas a eliminar.

- **Mostrar todas las evaluaciones de las respuestas a las preguntas de voz que el usuario creó**: Las respuestas de los estudiantes que han respondido a alguna de las preguntas de voz creadas por el usuario aparecen en una tabla. Solo aparecen las respuestas que han sido seleccionadas para ser evaluadas y que algún alumno ya ha evaluado.

  Los campos de la tabla son los siguientes:

  - ***ID***: Campo con los identificadores que la base de datos ha otorgado a cada respuesta de voz.

- ***Answer***: En este campo hay un elemento de tipo audio con el que se pueden escuchar las respuestas de los estudiantes a las preguntas de voz.
- ***Question***: En este campo se encuentran escritas las preguntas de voz a las que se responde con la grabación que se encuentra en el campo "*Answer*". Como estas pueden ser muy extensas, se muestran solo los 50 primeros caracteres y, si se pasa el cursor por encima, se muestran las preguntas completas.
- ***Creation date***: En este campo se muestra la fecha de en la que se creó la grabación de voz del campo "*Answer*".
- ***Number of evaluations***: Este campo muestra el número de evaluaciones que se le ha realizado a cada respuesta. Es decir, el número de estudiantes que han evaluado esa respuesta de voz.
- ***Criterions to evaluate***: Este campo contiene un listado de todos los criterios que se han tenido que evaluar en cada respuesta. Todos ellos separados por comas.

- **Crear preguntas de voz**: En esta funcionalidad, al usuario se le permite introducir algunos de los campos necesarios para la creación de una pregunta. Entre ellos se encuentra el texto de la pregunta, la introducción del *deadline* y del número máximo de respuestas que cada alumno puede dar a la pregunta de voz.

- **Subir imagen al servidor**: Esta necesidad fue identificada a la mitad del desarrollo del panel de administración. Consiste en la subida de una imagen al servidor para que esta, se envíe a los alumnos junto con la pregunta de voz. Para poder realizar esta acción, en la base de datos se guarda el *path* a la imagen junto a la pregunta de voz.

- **Crear, eliminar y seleccionar criterios a evaluar en las respuestas de una pregunta de voz**: Esta necesidad fue identificada justo antes de la realización de los prototipos a papel. Consiste en que el usuario pueda crear y eliminar criterios que se evaluarán en las respuestas de las preguntas de voz. En una pregunta de voz se puede seleccionar más de un criterio para evaluar, pero como mínimo ha de escogerse uno.

- **Seleccionar los grupos con los alumnos a los que se les asignará la pregunta de voz**: Para esta funcionalidad se ha creado una tabla, en la cual, se pueden seleccionar los grupos de alumnos a los que se les

quiere asignar la pregunta que se está creando o modificando. La tabla contiene los siguientes campos:

- ***ID***: Campo con los identificadores que la base de datos ha asignado a cada grupo.
- ***Group name***: Campo que contiene los nombres de los grupos que forman el cuerpo de la tabla.
- ***Number of students***: Número de estudiantes que tienen los grupos de alumnos.
- ***Check group members***: Este campo contiene el texto "*Click here to check the group members*". Al hacer *click* sobre este campo, aparece una pequeña ventana con una tabla que contiene los datos de los estudiantes que pertenecen al grupo de la fila en la que se ha pulsado. Esa tabla contiene el identificador de Telegram del alumno, su nombre completo y el número de preguntas y evaluaciones que se le han asignadas en total.
- ***Select***: Este campo consta de un *checkbox* que permite seleccionar los grupos que participarán en la pregunta que se está creando o modificando.

- **Mostrar el listado de los estudiantes para la asignación y evaluación de preguntas de voz**: En esta necesidad entran en juego dos tablas con algunos campos similares. La tabla en la que se muestran los alumnos a los que se les puede asignar la pregunta tiene los siguientes campos:

  - **Select**: Este campo de la tabla contiene un *checkbox* con el que se seleccionan los estudiantes a los que se les quiere asignar la pregunta para que la respondan.
  - **ID**: Este campo contiene los identificadores de Telegram de los estudiantes.
  - **Student**: En este campo se encuentra el nombre y apellido de los estudiantes tal y como lo tienen en Telegram.
  - **Number of questions to answer (Unanswered questions/Assigned questions)**: Este campo contiene dos números separados por el carácter *slash*. El número a la izquierda del *slash* representa el número de preguntas que el alumno tiene asignadas y que no ha dado una respuesta. El de la derecha del *slash* representa el número total de preguntas que tiene asignadas un alumno.

- **Answered this question**: Este campo solo contiene los valores "*Yes*" o "*No*". Indica si el estudiante ya ha dado una respuesta a la pregunta que se está asignando.

  Este campo se añadió al realizar la funcionalidad de modificar una pregunta, ya que Juanan quería saber quien había respondido a las preguntas para asignarles unos evaluadores que también hubieran contestado a esa misma pregunta.
- **Evaluator list**:En este campo se muestra el mensaje "*Click here to see the evaluator list of this student*". Si se pulsa en él, se abre un *iframe* en el que aparece la tabla para escoger los evaluadores del alumno de la fila en la que se ha hecho *click*.

En la tabla de selección de los evaluadores aparecen los siguientes datos:

- **ID**: Campo que contiene los identificadores de Telegram de los estudiantes.
- **Student**: Nombre y apellido que los estudiantes tengan en la aplicación de Telegram.
- **Questions to answer**: Número de preguntas que tiene cada alumno por responder.
- **Answers to evaluate**: Campo que indica el número de respuestas que cada alumno tiene que evaluar.
- **Answered this question**: Campo que contiene "*Yes*" si el alumno ya ha contestado a la pregunta que se está creando o modificando.
- **Select**: Este campo está compuesto por un *checkbox* que permite la selección de los evaluadores de un alumno.

Gracias a estas dos tablas se puede ver la información sobre la asignación de las preguntas y los evaluadores de cada alumno.

- **Selección automática y manual de estudiantes para asignación de preguntas de voz**: Como se ha comentado en el punto anterior, en la tabla de asignación de la pregunta modificada o creada, existe un campo denominado "*Select*". Ese campo permite la selección manual de la asignación de la pregunta a diferentes estudiantes.

  Además del modo manual, existen varios botones encima de las tablas. Esos botones permiten la selección automática de estudiantes. Fue una petición de Juanan para, en caso de tener grupos grandes de estudiantes, no perder el tiempo seleccionando de uno en uno los alumnos.

  Los tres botones de asignación automática son los siguientes:

- **Random selection**: Al pulsar este botón, los estudiantes que tendrán que responder a la pregunta de voz que se esté creando o modificando, se seleccionan aleatoriamente.

  Aunque sí se puede considerar un método aleatorio, llega un punto en el que se tiene que jugar con la estadística para que se equilibren las proporciones. Es decir, si el método de selección siempre fuera aleatorio, podría llegar a haber estudiantes que tuvieran que realizar todas las preguntas que el profesor crease, mientras que otros no tendrían que responder a ninguna de ellas. Para que esto no suceda, se ha jugado un poco con las probabilidades de que a un alumno se le asigne una pregunta para responderla.

  En un principio el método de selección es aleatorio, con una probabilidad del 50 % de que a un estudiante se le asigne la pregunta de voz. Pero, cuando entre el alumno con más preguntas asignadas y el que menos preguntas tiene hay una diferencia de cuatro o más preguntas, las probabilidades varían. Cuando eso sucede se realiza una la siguiente regla de tres por cada estudiante:

  $$Probabilidad\ del\ estudiante = 1 - \frac{(Preguntas\ estudiante - min) * 0{,}95}{(max - min)} \tag{12}$$

  En la ecuación se muestran las siguientes variables:

  - **Probabilidad del estudiante**: Probabilidad de que al estudiante se le asigne la pregunta de voz.
  - **Preguntas estudiante**: Es el número total de preguntas que tiene asignadas hasta el momento el estudiante del que se quiere calcular la probabilidad.
  - **max**: Número de preguntas asignadas que tiene el alumno con mayor número de preguntas asignadas para responder.
  - **min**: Número de preguntas asignadas que tiene el alumno con menor número de preguntas asignadas para responder.

  Con la ecuación, se consigue que el usuario con mayor número de preguntas asignadas solo tenga un 5 % de probabilidad de que se le asigne la pregunta. Mientras que el que menos preguntas asignadas tiene, adquiere una probabilidad del 100 % de que se le asigne la pregunta de voz.

- **Select all**: Cuando se pulsa este botón se seleccionan todos los estudiantes de la lista.

- **Unselect all**: Cuando se pulsa este botón se desmarcan todos los *checkbox* de la tabla. Esto implica que se quita la selección a todos los estudiantes.

También hay que decir que cuando un estudiante es seleccionado para que se le asigne la pregunta, se le asignan sus evaluadores aleatoriamente de la lista de estudiantes que participan en la pregunta.

- **Selección automática y manual de estudiantes para evaluación de la respuesta de voz de otro estudiante**: En el iframe en el que se encuentra la tabla de evaluadores de un alumno, encontramos cuatro botones. Tres de ellos son iguales a los del punto anterior, pero tienen algunas diferencias en su ejecución a la hora de presionarlos.

  El botón "*Random selection*" recorre toda la lista de evaluadores y selecciona los evaluadores con un 50 % de probabilidad siempre. Pero antes de la asignación, se comprueba que el estudiante no haya sobrepasado el número máximo de evaluaciones que un alumno puede realizar en una pregunta de voz. Este valor se especifica en las opciones del profesor.

  En cuanto al botón "*Select all*" solo cambia la realización de la misma comprobación que en el botón "*Random selection*". El botón "*Unselect all*" no necesita realizar ninguna comprobación, así que su ejecución no cambia respecto al anterior punto.

  Aparte de los botones ya mencionados, para esta funcionalidad se ha creado un cuarto botón. "*Random selection of students who have already answered*" es el botón que realiza una selección aleatoria de los estudiantes que ya han respondido a la pregunta de voz. Este botón fue añadido posteriormente, cuando Juanan identificó la necesidad al intentar asignar los evaluadores durante la evaluación de la herramienta.

- **Modificación de una pregunta de voz**: Se permite la modificación de los parámetros que forman las preguntas de voz. Además de cambiar los parámetros, también se permite la modificación de los alumnos que han de responder a la pregunta y de sus evaluadores.

  Eso sí, hay que tener cuidado con los cambios que se realicen porque puede llegar a haber perdida de datos. Si a un alumno se le había asignado la pregunta, este responde, y después se modifica la pregunta quitando la asignación a ese estudiante, su respuesta y evaluaciones son eliminadas de la base de datos. Aunque la respuesta siga realmente guardada en el servidor, la base de datos perderá la referencia a ellos.

- **Supresión de una o varias preguntas de voz**: En el panel de administración, al profesor se le permite eliminar las preguntas de voz que ha creado. Para ello solo tiene que seleccionarlas en la tabla en la que aparecen todas las preguntas de voz y pulsar el botón "*Delete question*". Al realizar esta acción, se realiza una comprobación obligando a escribir una frase especifica al profesor para que las preguntas realmente se eliminen.

- **Creación y supresión de grupos de alumnos**: En la página en la que aparece la tabla con todos los grupos, hay un botón para eliminar los grupos seleccionados y otro para crear más grupos.

  El botón para eliminar los grupos funciona igual que el del punto anterior, solo cambia el mensaje que el profesor ha de escribir.

  El botón para crear grupos, crea un *iframe* en el que se ha de introducir el nombre del nuevo grupo. Al pulsar el botón de confirmación, el nuevo grupo aparece en la tabla de grupos.

- **Mostrar alumnos de un grupo**: Al hacer *click* en cualquier campo que no sea "*Select*" de una de las filas de la tabla, lleva al profesor otra página. En ella, se muestra el listado de los alumnos que pertenecen al grupo de la fila en la que se ha hecho *click*.

- **Añadir y eliminar alumnos a un grupo**: En la pantalla en la que se muestran todos los alumnos de un grupo, existe un botón para añadir más estudiante y otro para eliminar estudiantes de ese grupo.

  El botón para eliminar los estudiantes del grupo funciona igual que los de los puntos anteriores , solo cambia el mensaje que el profesor ha de escribir.

  El botón para añadir estudiantes, abre un *iframe* con una tabla en la que aparecen todos los usuarios, que en algún momento han escrito al *bot*, pero que no están registrados en ningún grupo. En esa tabla se pueden seleccionar los usuarios que se quieran agregar al grupo y al pulsar el botón "*save*" en ese *iframe*, los usuarios pasarán a ser estudiantes del grupo que aparece como título del *iframe*.

- **Mostrar las calificaciones que la respuesta de voz de un estudiante ha recibido**: En la tabla que contiene las respuestas de los estudiantes, si se hace *click* en alguno de los campos de la tabla, lleva al administrador a una página en la que se muestra: la pregunta que se ha contestado, el audio que reproduce la grabación que el estudiante ha dado como respuesta a la pregunta, una tabla que contiene la media

de las calificaciones en cada criterio y otra tabla que detalla por cada evaluador, qué calificación ha otorgado a la respuesta en cada criterio.

- **Permitir escuchar las explicaciones de voz de las evaluaciones realizadas por los alumnos**: Esta funcionalidad se añadió durante la evaluación de la herramienta. Juanan la propuso porque, en caso de que un estudiante ponga calificaciones bajas a una respuesta, normalmente a los alumnos les gusta explicar cuál ha sido el fallo cometido. Además es un gran *feedback* para el estudiante que propuso la respuesta. Es por ello que se añadió un apartado de explicaciones de voz en la tabla de calificaciones mencionada en el punto anterior.

  Si un estudiante envía una explicación de voz, esta aparecerá en el último campo de la tabla. En caso de que no se envíe ninguna explicación, en ese campo aparecerá el carácter "-".

- **Mostrar gráficos del uso del *bot* y del progreso de los alumnos**: El panel de administración tiene un apartado denominado "*Graphics*" en el que aparecen cuatro botones. Cuando se pulsa uno de esos botones, se abre una nueva pestaña en el navegador en la que se muestra un gráfico interactivo con los datos que se indican en el botón presionado.

  Al pulsar el botón "*Graphic of correct/wrong answers in quizzes per student*", en la nueva pestaña aparece un gráfico de barras con las respuestas que cada alumno ha dado en los *tests*. Por cada estudiante aparece una barra de color naranja, que indica las respuestas incorrectas, y una barra sobrepuesta a esa en color azul, que indica las respuestas correctas.

  Al presionar el botón "*Graphic of answered questions in quizzes per student*" aparece otro gráfico de barras. En este, las barras se dividen en colores, que se corresponden con los diferentes *tests* que tiene disponibles el *bot*. Por cada alumno se muestra el número de preguntas que ha respondido en cada *test*.

  El botón "*Graphic of the quizz answers given by students during the last 12 months*" muestra un gráfico temporal. En el, se puede observar el número de preguntas de *tests* que se han realizado por mes durante el último año.

  Al pulsar el botón "*Graphic of the quizz answers given per student during the last 12 months*" aparece un gráfico temporal. Este gráfico muestra cuantas preguntas de *test* ha realizado cada estudiante durante el último año.

- **Permitir cambiar el lenguaje del *bot***: En el apartado de opciones se puede cambiar el lenguaje que usa el *bot*. Los lenguajes por defecto son español e inglés, pero se pueden añadir más si se quiere. Más adelante se explica como añadir otro lenguaje.

- **Permitir la autoevaluación de los alumnos en preguntas de voz**: El administrador puede permitir que los alumnos autoevaluen sus respuestas a las preguntas de voz. Para ello solo hay que marcar la opción y guardar. Al hacer esto, cada vez que se vayan a seleccionar los evaluadores de un estudiante en una pregunta, el estudiante también podrá ser seleccionado para evaluarse a sí mismo.

- **Permitir cambiar las posiciones por defecto de la respuesta correcta en los *tests* del *bot***: Anteriormente ya se ha comentado esto. En el apartado de opciones existe la posibilidad de hacer que las respuestas de los *tests* varíen de posición o se queden estáticas. Esta última opción es útil en caso de que se quiera realizar un *test* durante una clase, ya que al estar las respuestas siempre en la misma posición, el profesor siempre sabe cuál es la posición en la que se encuentra la correcta.

- **Poder escoger el máximo de evaluaciones por alumno en una pregunta de voz**: En el apartado de opciones el administrador puede poner un límite de evaluaciones por alumno en las preguntas de voz. A la hora de escoger los evaluadores de un alumno, se comprobará que ese límite no ha sido sobrepasado antes de seleccionar a un alumno. Además, si se llega al máximo en algún alumno y se intenta seleccionar manualmente, saltará una alerta indicando que el alumno ha llegado al limite de evaluaciones que se le pueden asignar en esa pregunta.

- **Permitir cambiar la escala en la que se realizaran las evaluaciones de las respuestas de voz**: A petición de Juanan, se permite cambiar la escala de evaluación que los alumnos utilizarán para calificar las respuestas a las preguntas de voz. Aunque la escala se cambie, todas ellas se traducen a una puntuación de cero a diez. De esta manera, al administrador no le surgirán dudas en cuanto a la puntuación que recibe cada respuesta.

El *bot* permite las siguientes escalas:

- De cero a diez.
- De cero a cien.
- *Fatal, Very bad, Bad, Regular, Good, Very good* o *Excelent.*
- *Fail, Approved, Notable* o *Distinction.*
- De una a cinco estrellas.

- **Mostrar información de los estudiantes**: En el apartado "*Student Information*" del panel de administración, se muestra una tabla con todos los alumnos del administrador. Al hacer *click* en el campo "*Check evaluations*" de esa tabla, se puede ver quienes van a ser evaluados por el estudiante que se encuentra en la fila que se ha pulsado. La información se muestra en un *iframe* que contiene los siguientes datos en una tabla:

  - **ID**: Contiene el identificador de Telegram de los estudiantes que van a ser evaluados por el estudiante de la fila en la que se ha pulsado.
  - **Student name**: Nombre y apellido que tienen en Telegram los estudiantes.
  - **Question ID**: Identificador de la pregunta de voz en la que el estudiante mencionado en el título del *iframe* evaluará a cada estudiante de la tabla.
  - **Question**: Los cincuenta primeros caracteres de la pregunta. Si se pasa el cursor por encima se muestra la pregunta completa.
  - **Creation date**: Fecha en la que se asignó al estudiante como evaluador.
  - **Checked**: Indica en inglés si la fila se ha marcado como vista en algún momento. En "*Student information*", encima de la tabla que muestra los estudiantes, hay un botón con el que se puede marcar como visto toda la tabla. Eso se hace después de que al administrador le aparezca un *iframe* para confirmar la acción.

  Además de ver de quien es evaluador el estudiante, en la tabla de "*Student information*" también está el campo "*Check evaluators*". Haciendo *click* en este campo, aparece un *iframe* con una tabla en la que aparecen los evaluadores de la respuesta del estudiante de la fila en la que se ha pulsado. La tabla tiene los mismos campos que la anterior, pero esta vez son los evaluadores de las respuestas del estudiante los que aparecen en ella.

- **Poder ordenar y buscar elementos en todas las tablas del panel de administración**: Exceptuando la tabla de las medias de las calificaciones al mostrar una evaluación, todas las demás tablas son interactivas. Se pueden ordenar por campos o buscar algún tipo de elementos en ellas.

8.1.2. *Bot* de Telegram

Respecto al *bot*, no hubo muchas necesidades nuevas ya que se tenían a los *bots* comentados en el capítulo de antecedentes (Sección 4) como referencia para la creación del de este proyecto.

En el siguiente listado se detallan todas las funcionalidades del *bot* desarrollado en este TFG:

- **Escoger tema del que realizar un *test***: Con esta funcionalidad, se le envía al usuario un listado de botones. Cada botón, está relacionado con un tema del que el *bot* tiene preguntas de tipo *test* disponibles. Al escoger un tema, se le envía al estudiante la primera pregunta del *test* de ese tema.

- **Posicionamiento de las respuestas de *test***: Esta funcionalidad depende de las opciones que haya escogido el profesor. El profesor, puede escoger entre posicionar las respuestas de los *tests* en posiciones por defecto o hacer que estas se posicionen de manera aleatoria.

  Cuando escoge posicionarlas por defecto, el estudiante siempre recibirá las respuestas en unas posiciones predeterminadas. Para que esto ocurra, en la base de datos se guarda una posición en la que se encontrará la respuesta correcta. Esta posición puede ser A, B, C o D, dependiendo del número de respuestas que tenga la pregunta de *test*. La respuesta correcta se meterá en esa posición y las respuestas incorrectas se posicionaran en las posiciones sobrantes en el orden que se cojan de la base de datos.

  En el caso de las posiciones aleatorias, es como ello mismo indica. Las respuestas se posicionan en posiciones aleatorias. De esta manera los estudiantes no pueden aprenderse las posiciones de las respuestas de los *tests*.

- **Por qué y por qué no**: Cada pregunta de *test* tiene asociadas una respuesta correcta y entre una y tres respuestas incorrectas. El profesor, puede meter en la base de datos un texto explicativo que indica por qué

la respuesta correcta es correcta y otro explicando por qué las demás respuestas son incorrectas. Esto puede ayudar a solventar las dudas de los estudiantes sin necesidad de acudir al profesor.

- **Repetir *test*, pasar al siguiente o salir**: Al terminar un *test*, @pruebasbot y @dawebot, vuelven a enviar la primera pregunta del *test* que se está realizando. Es decir, se quedan en bucle en el mismo *test*. Para que esto no pasase en el *bot* de este proyecto, al terminar un *test*, se le dan diferentes opciones al estudiante para que haga lo que crea conveniente. El alumno puede repetir el mismo *test*, pasar al siguiente en caso de que exista o salir y dejar de hacer *tests*.

- **Imagen asociada a pregunta de *test***: Hay preguntas en las que es necesaria una imagen para poder entender a que se refiere la pregunta. Gracias a esta funcionalidad, se puede enviar una imagen junto a una pregunta de *test*. Esto se consigue guardando el *path* de la imagen en un atributo de la pregunta de *test* en la base de datos.

  @dawebot también tiene esta funcionalidad, pero en su caso, no esta del todo bien programada.

- **Ver estadísticas en *tests***: Gracias a esta funcionalidad, el usuario puede ver el número total de preguntas de *test* ha respondido correctamente, el número total de respuestas que ha respondido incorrectamente y cuál es su precisión.

  La precisión es el cálculo de el número de respuestas correctas entre el numero total de respuestas (correctas más incorrectas).

- **Escoger preguntas de voz a responder**: Con esta funcionalidad, se le envía al usuario un listado de mensajes con las preguntas que tiene asignadas. Después, se le envía otro listado con *Inline buttons* asociados a cada una de las preguntas. De esta manera se permite al usuario escoger cuál quiere responder. Antes de enviar los listados, se comprueba que no haya pasado el *deadline* de cada una de las preguntas y que el usuario no haya dado el máximo de respuestas permitidas en cada una de ellas.

- **Imagen asociada a pregunta de voz**: Tal y como es la funcionalidad de las imágenes en las preguntas de *test*, esa necesidad también es aplicable a las preguntas de voz.

  Además, a los profesores se les puede hacer interesante preguntar a sus alumnos cosas acerca de la imagen. Por ejemplo, imagina que un profesor de historia del arte quiere preguntar algo acerca de una escultura.

Que mejor forma de dejar que el alumno se explaye acerca de ella que mostrar en una imagen la escultura y preguntando: “Dime todo lo que sepas acerca de la escultura de la siguiente imagen.“.

Otro aspecto a tener en cuenta es que @myqueridobot, el *bot* realizado en este proyecto, es innovador en este aspecto, ya que ni @pruebasbot ni @dawebot han aplicado esta funcionalidad a las preguntas de voz.

- **Descargar en el servidor la respuesta de voz de una pregunta**: Cuando un estudiante responde a una pregunta de voz mediante una grabación, esta es descargada a una carpeta en el servidor. Si la descarga se ha completado correctamente, los datos son insertados en la tabla que corresponde de la base de datos y se deja esta respuesta como seleccionada para evaluar.

- **Registrar un estudiante**: Para esta funcionalidad, se tienen dos versiones.

  En la desarrollada para el cliente, tiene menor seguridad por una petición de este. Cuando el usuario ejecuta el comando de registro, se le registra como estudiante del primer grupo existente. De esta manera, se pierde seguridad a la hora de que cualquiera pueda llegar a registrarse con el *bot*. Lo único que hace falta es descubrir el comando oculto de registro. Esto se soluciona eliminando de manera manual los usuarios desconocidos con las funcionalidades del panel de administración.

  La segunda versión, se desarrollo de una manera bastante más segura. El profesor, al registrarse, tiene que meter en la base de datos un campo llamado “*id_group_selection*“. Este campo permite meter en el un *String* de como mucho veinte caracteres. Ese campo es el identificador que el usuario tiene que meter en uno de los pasos del registro para identificar a su profesor. Gracias a esto, el comando de registro se puede hacer tan seguro como complicado sea el identificador. Además, este método ayuda a que pueda haber más de un profesor, cada uno con sus grupos de alumnos, utilizando las herramientas desarrolladas en este proyecto.

- **Permitir cambiar la respuesta de voz seleccionada para evaluar en preguntas de voz**: Un estudiante puede tener más de una oportunidad para responder a una pregunta de voz. El profesor decide cuantas respuestas pueden dar los alumnos a una pregunta cuando esta es creada. Es por ello que se vio la necesidad de poder cambiar la respuesta que se quiere que los demás alumnos evalúen.

Cuando el alumno ejecuta el comando /change, se realizan las comprobaciones necesarias para comprobar cuales son las preguntas en las que se puede cambiar de respuesta. Después, se le envía un listado con esas preguntas para que escoja en cual se quiere cambiar la respuesta de voz. Cuando escoge una, se le envía la pregunta que ha escogido y todas las respuestas de voz que este estudiante ha dado a esta pregunta. En este punto el estudiante puede escoger la respuesta que quiere seleccionar enviando un mensaje con el identificador que se ha dado con la respuesta. Para terminar, se guarda todo en la base de datos si el identificador enviado por el alumno es correcto.

- **Ver todos los comandos disponibles (sin contar los de administrador y el comando de registro)**: Esta funcionalidad la han tenido todos los *bots* realizados con la librería php-telegram-bot por defecto hasta la versión 0.42. Pero no todos esos *bots* tienen un comando de registro que hay que ocultar. Es por ello que se tuvo que modificar el comando por defecto para, además de ocultar los comandos que solo el administrador puede usar, ocultar el comando de registro. Es una forma de prevenir que cualquiera pueda usar el *bot* a su antojo.

- **Escoger la respuesta de voz a evaluar**: A los estudiantes se les manda un listado con las posibles respuestas a evaluar. En ese listado solo aparecerán las respuestas de las preguntas a las que se les haya pasado la fecha del *deadline*.

- **Poder evaluar todos los criterios asociados a una respuesta de voz**: Cuando un alumno escoge una respuesta a evaluar, se le envía la pregunta a la que se responde, la grabación de voz dada como respuesta y el criterio a evaluar. Al dar una puntuación a la respuesta en ese criterio, se le enviará el siguiente criterio si es que se le ha asignado. En una respuesta se puede tener que evaluar más de un criterio, es por ello que se le envían uno detrás de otro al estudiante para que evalúe todos.

- **Permitir dar una explicación de voz a la evaluación realizada**: Cuando se terminan de evaluar todos los criterios comentados anteriormente, se le pregunta al estudiante si quiere dar una explicación de voz a las puntuaciones que se han dado en los diferentes criterios.

  Esta funcionalidad fue una de las últimas que se añadió al *bot*. Juanan identificó esta funcionalidad al darse cuenta de que si un alumno había recibido una mala nota en algún criterio, el no tendría oportunidad de saber por qué el evaluador puso esa nota. Por esta razón, y porque en

caso de reclamación Juanan no tendría ninguna información, se decidió realizar una funcionalidad que permitiera a los alumnos explicar las razones de las calificaciones proporcionadas a las respuestas de voz. Pero no es obligatorio dar la explicación.

- **Escoger la respuesta en la que se quieren ver las calificaciones recibidas**: En esta funcionalidad, se le envía al alumno un listado de las preguntas en las que su respuesta ha recibido alguna calificación.
- **Mostrar todas las calificaciones de una respuesta de voz**: Cuando el estudiante escoge una pregunta del listado enviado en el punto anterior, se le envía al usuario la pregunta y su respuesta de voz junto con un listado de los criterios que se han evaluado. Por cada criterio, aparece otro listado con las calificaciones que ha recibido la respuesta de voz en ese criterio.
- **Enviar todas las explicaciones de voz que los alumnos han dado a unas calificaciones**: Al terminar con el punto anterior, al usuario se le envía el listado de grabaciones con las explicaciones de las calificaciones que los evaluadores han dado a la respuesta de voz.

En este apartado también hay que comentar que se crearon dos ficheros para ejecutar mediante la herramienta Cron de Linux. Esos ficheros permiten lo siguiente:

- **Notificación de preguntas y evaluaciones disponibles**: Permite mandar notificaciones con el *bot* para avisar a los alumnos que tienen preguntas de voz asignadas o evaluaciones por realizar. Es recomendable que el fichero con esta funcionalidad se ejecute como mínimo una vez a la semana.
- **Falta de dos días para el *deadline***: Esta funcionalidad permite enviar notificaciones durante tres días seguidos. Esas notificaciones avisan a los estudiantes que las reciben de que existe al menos una pregunta de voz que no se ha respondido, de la cual faltan dos días para que termine el *deadline*. Se recomienda ejecutar el fichero con esta funcionalidad una vez al día.
- **Cambio de formato de los ficheros que contienen las grabaciones de voz**: Se creó un fichero PHP que, gracias la librería y al programa FFmpeg, permite cambiar los ficheros con las grabaciones de voz de .oga a .ogg.

8.2. Desarrollo de la base de datos

Durante el desarrollo del proyecto, la base de datos se ha tenido que ir modificando por dos razones principales. La primera, son todas las nuevas necesidades que se identificaban mientras se implementaba el proyecto. La segunda razón es por la librería php-telegram-bot.

Al ir añadiendo las nuevas funcionalidades, estas requerían de nuevos datos que no se veían reflejados en la base de datos. Es por ello que, necesidades como "Permitir dar una explicación de voz a la evaluación realizada" o "Permitir cambiar el lenguaje del *bot*" (Puntos 8.1.2 y 8.1.1), hicieron que se tuviera que modificar la base de datos para poder guardar la información que se necesitaba para implementarlas.

La librería php-telegram-bot también estuvo implicada en las modificaciones realizadas en la base de datos. La base de datos desarrollada durante este proyecto se compone de dos partes que están relacionadas por una tabla. Una parte está formada por las tablas de la base de datos que la librería proporciona en el fichero "structure.sql". La otra parte, es la que se ha desarrollado durante el proyecto para guardar los datos necesarios para la realización de todas las funcionalidades mencionadas anteriormente.

El problema con la parte de la base de datos de la librería a sido sus continuas actualizaciones. En algunas de las actualizaciones de la librería, se han realizado modificaciones en el fichero "stucture.sql". Como no tienen ningún sistema para que se automaticen los cambios de la base de datos que se realizan en ese fichero, hay que realizar las modificaciones a mano. Esto ha llevado bastante tiempo, ya que en cada actualización que se ha realizado en la librería, había que comprobar si se había cambiado algo en el fichero con la estructura y, si se modificó algo, aplicarlo manualmente en la base de datos del proyecto.

Este último problema se hubiera solucionado si los desarrolladores de la librería hubieran usado algún sistema que permita realizar los cambios en la base de datos de manera automática. También se hubiera ahorrado tiempo si en vez de una base de datos, el proyecto hubiera tenido dos. Separando la base de datos de la librería de la base de datos del proyecto, simplemente se tendría que cambiar la base de datos de la librería por la misma actualizada.

El diagrama de la base de datos con la que se ha finalizado el proyecto se muestra en la Figura 30. En él se muestra la partición de la base de datos en dos. La parte superior del diagrama está compuesto por las tablas que pertenecen a la librería php-telegram-bot. La parte inferior se compone de las tablas que se han desarrollado para este proyecto. Como se puede apreciar

en la Figura 30, la parte del diagrama de la base de datos del proyecto se corresponde mucho con el diagrama de modelo de dominio presentado en el apartado 5.4. A continuación se explican las tres tablas que diferencian la base de datos del modelo de dominio presentado.

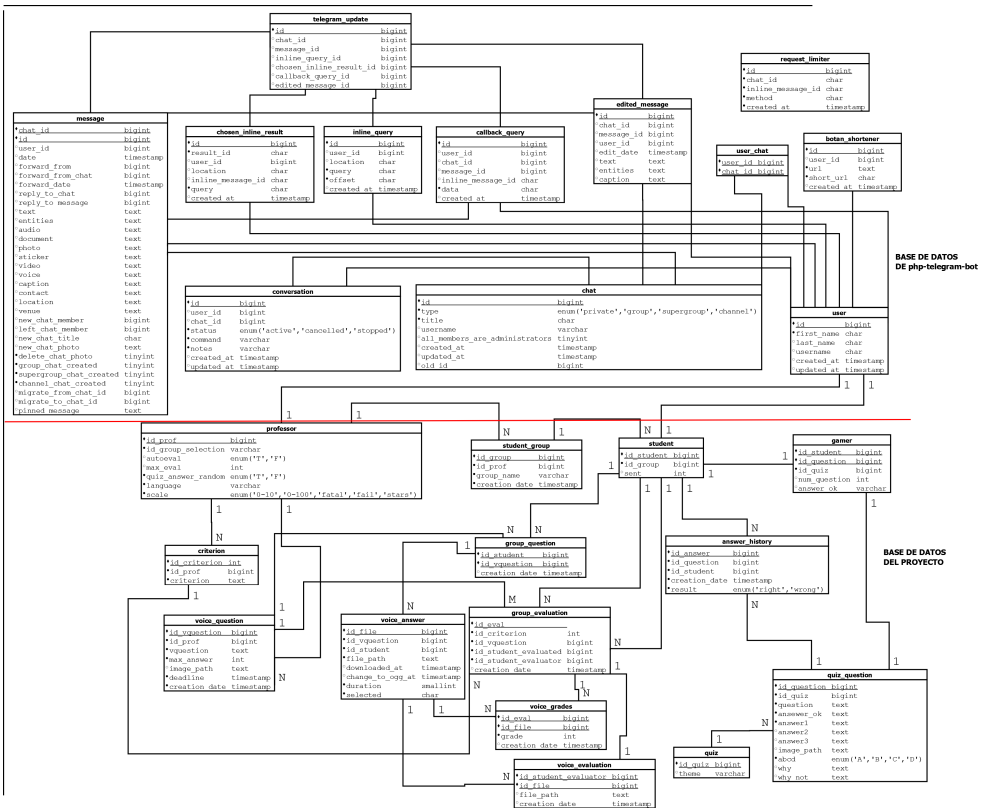


Figura 30: Diagrama del diseño de la base de datos.
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/ModeloDominio/DiagramaBD.png`.

- **group_evaluation**: En esta tabla se guarda la relación de evaluación entre alumnos. Quién evalúa a quién, en que pregunta y que criterio se ha de evaluar.

- **group_question**: Esta tabla guarda la relación entre los estudiantes y las preguntas de voz que se les han asignado.

- **gamer**: Esta tabla guarda la situación en la que se encuentra un estudiante mientras realiza un *test* con el *bot*. Es necesario guardar la información del *test* que se está realizando y de la pregunta en la que se encuentra el estudiante.

8.3. Desarrollo del *bot* de Telegram

En un principio, se pensó que el desarrollo del *bot* no llevaría tanto trabajo como el que realmente se ha realizado. Esto se debió a que antes de que Pablo Uribe presentara su TFG se pensó en seguir con lo que él había desarrollado. Pero como ya se ha comentado durante este documento, se desechó todo lo que él realizo por la lentitud con la que el *bot* respondía.

Después de desechar el código de Pablo, Juanan prestó el código que él tenía de @dawebot para este proyecto, pero esto tampoco se pudo llegar a utilizar. El código que tenía Juanan utilizaba la versión 0.31 de la librería php-telegram-bot (conocida como Akalongman en aquel entonces), mientras que cuando se empezó este proyecto, la librería ya iba por la versión 0.35. Había métodos que en la nueva versión estaban obsoletos, se había modificado la base de datos y había algún que otro error por el código. Por estas razones se decidió no utilizar lo que Juanan implementó y empezar todo de cero. Como se puede presuponer, esto afectó bastante a la planificación inicial, ya que en un principio solo se tenía planteado modificar un poco los comandos para hacer su código más intuitivo.

El *bot* se comenzó a desarrollar de manera local. La librería php-telegram-bot permite desarrollar *bots* sin necesidad de un certificado mediante un fichero con el nombre "getUpdatesCLI.php". Este fichero recibe todas las peticiones HTTPS que se le mandan al *bot* y realiza las acciones necesarias para que un comando se ejecute normalmente. Pero hay que ejecutar el fichero cada vez que un usuario envía un nuevo mensaje para que el *bot* conteste. Aunque esta manera de desarrollo es muy cómoda, también tiene sus inconvenientes, los cuales se comentarán después.

Mientras se desarrollaba el *bot* en local, Juanan comentó que había un problema con las grabaciones de voz. Cuando un usuario de Telegram realiza una grabación de voz y la envía, esta tiene el formato .oga. El problema con ese formato es que ni el elemento audio de HTML5 ni los dispositivos Android con versiones inferiores a la 6.0 pueden reproducir este tipo de ficheros. Esto quiere decir que ni dispositivos Android de más de un año ni el panel de administración podrían reproducir las respuestas de voz que grabarían los alumnos.

Para solucionar el problema con los .oga, lo más lógico es cambiar el formato por otro que si sea reconocido por todas las herramientas que se utilicen en el proyecto. Investigando un poco, se descubrió una librería en PHP que utiliza un programa con su mismo nombre, FFmpeg, que permite la conversión de ficheros de tipo .oga a .ogg. Claro que esto también llevó su

tiempo descubrir como funcionaba, ya que aunque los ficheros .oga utilizan la misma codificación que los .ogg, estos últimos tienen unas características que los .oga cambian. Para ponerlo de una manera sencilla, los archivos .oga son específicos de audio, mientras que los .ogg son más genéricos. Por ejemplo, los .oga usan un mayor número de canales de audio que los .ogg, lo cual hay que cambiar a la hora de realizar la conversión de archivos.

Después de aprender a usar la librería e implementar la conversión de los ficheros en el mismo comando, Juanan me comentó que FFmpeg usa muchos recursos a la hora de realizar la conversión y que si se hacían varias en paralelo podrían ralentizar el servidor. Por ello se me pidió sacar la parte de la conversión del comando y preparar un fichero que se ejecutase mediante Cron, el administrador de procesos de los sistemas Unix. Es por esto que se creó el fichero Cron que se ha mencionado en la funcionalidad "Camio de formato de los ficheros que contienen las grabaciones de voz".

Otro de los ficheros preparados para usar con Cron que se pidió, fue el fichero "Notification.php". Este fichero permite que el *bot* envíe notificaciones a los estudiantes que tienen preguntas de voz por responder o que tienen alguna evaluación por realizar. Es útil para avisar a los estudiantes de que se les ha asignado alguna pregunta o que las evaluaciones de alguna pregunta han comenzado.

Uno de los problemas que se tuvo con esta funcionalidad fue que un estudiante llegó a bloquear al *bot*. Esto hace que los mensajes que envía el *bot* no lleguen al usuario. Cuando esto ocurre, la ejecución se para dando una excepción. Para que esto no ocurra se ha usado la herramienta Curl.

Curl es un sistema que simula un navegador pero sin interfaz gráfica. Esta herramienta puede realizar peticiones HTTP y se puede controlar con PHP. Al usar Curl para enviar los mensajes de notificación a los estudiantes, el fichero no interrumpe su ejecución, ya que cuando un mensaje falla en enviarse, Curl recibe un error que ocurriría al intentar cargar una página web a la que no se puede acceder. Ese error se representa simplemente con el estado de la página, pero puede continuar perfectamente su ejecución haciendo caso omiso del error. De esta manera se consigue avisar a todos los estudiantes aunque alguno haya bloqueado al *bot*.

Junto con los ficheros a usar con Cron, otra de las peticiones de Juanan fue preparar el *bot* para poder cambiar el idioma de los mensajes que el *bot* envía. Esto se planteó porque se tiene como objetivo publicar el proyecto a nivel internacional. Por ello se ha desarrollado un sistema sencillo que permite cambiar el idioma y poder añadir más de los que ya existen. Para ello, se ha creado una carpeta (vendor/longman/telegram-bot/src/Entities/Languages)

en la que se encuentran los ficheros PHP con variables que contienen los mensajes que el *bot* envía. Los ficheros que existen hasta ahora son "Spanish.php", que contiene los mensajes en español, y "English.php", que contiene los mismos mensajes pero en inglés. Además de esos ficheros, se ha creado otro como plantilla para que sean traducidos y se puedan añadir más idiomas fácilmente.

En la carpeta Entities, se ha creado un fichero llamado "Language.php" con el que se accede a los ficheros con los mensajes traducidos. Puede saber el lenguaje que tiene que usar gracias a que en la tabla "*professor*" de la base de datos, hay un campo llamado "*language*". En él, se ha de escribir el nombre del fichero (sin la extensión .php) del idioma que se quiera que el *bot* utilice. Los comandos harán una llamada a la base de datos para recoger ese campo y, después, llamarán al método de "Language.php" que les devuelva los mensajes que ese comando usará, en el idioma correspondiente.

Cuando se terminó de realizar una primera versión del *bot* local, el código implementado para este se subió al servidor que Juanan proporcionó para la realización del proyecto. En este punto empezaron los problemas de versiones.

Como ya se ha comentado anteriormente, antes de empezar con el desarrollo del proyecto en sí, hubo un mes y medio que se dedicó al aprendizaje del uso de la librería php-telegram-bot. En ese momento, se utilizaba como servidor una cuenta de Microsoft Azure[7] con PHP versión 7.0. Todos los comandos que se usaron para aprender a utilizar la librería se realizaron en esa versión. El problema llegó poco más de un mes después, cuando la universidad dejó de proporcionar las licencias de Azure. En este punto todos los *bots* se pasaron otro servidor que no tenía la versión 7.0 de PHP, sino la 5.6. Es por ello que cuando se subió el código local del *bot* al servidor, este no funcionó.

Después de modificar el código para que funcionara en la versión 5.6 de PHP, Juanan empezó a identificar necesidades y funcionalidades que a la larga podrían ser útiles. Además de esto, la librería php-telegram-bot empezo a realizar actualizaciones cada muy poco tiempo. Antes de que se comenzara con el desarrollo del proyecto, los desarrolladores de la librería solían realizar actualizaciones periódicas cada dos meses más o menos. Pero a partir de diciembre, se empezó a actualizar cada poco más de dos semanas. Estas actualizaciones complicaron bastante el desarrollo del proyecto, ya que

[7]Microsoft Azure: Microsoft Azure es una creciente colección de servicios en la nube integrados que los desarrolladores y los profesionales de TI utilizan para crear, implementar y administrar aplicaciones a través de nuestra red global de centros de datos. Su página oficial es: `https://azure.microsoft.com/`.

además de lo ya comentado en el apartado anterior con la base de datos, se modificaban algunas de las clases que se tuvieron que modificar durante la implementación del *bot*. Por ello, cada dos semanas se tenían que analizar los cambios que se habían realizado en la librería, para ver si alguna modificación afectaba a lo que se había realizado en el proyecto, y en tal caso, rediseñar e implementar los cambios que hicieran que funcionase.

Lo de las versiones ha sido una ardua tarea, sobre todo sabiendo que el proyecto comenzó con la versión 0.35 y que ha finalizado con la versión 0.42 de la librería. No se ha seguido actualizando porque en la versión 0.43 se realizaron muchísimos cambios en la librería, y por la falta de tiempo se decidió dejar de actualizar en la versión anterior. Aun así, el tema de las versiones ha hecho que el código se haya intentado realizar de lo más modular posible, para no realizar tantos cambios en cada actualización.

Otro tema a comentar es Botan. Esta es una herramienta que viene integrada con la librería php-telegram-bot que muestra gráficos estadísticos sobre el *bot*. Los gráficos se consultan en un *bot* llamado @botan, que con sus opciones muestra los gráficos sobre el uso de los comandos, sobre nuevos usuarios o sesiones que se han realizado en otros *bots*. Esta herramienta se ha probado y está implementada en la versión con la que ha finalizado este proyecto, pero como sus gráficos no son muy representativos se ha dejado a un lado para la parte de investigación.

8.4. Desarrollo del panel de administración del *bot*

Para el panel de administración, se tuvo en cuenta el requerimiento de que se pudiera acceder a él desde cualquier lugar. Es por ello que se siguió el ejemplo de lo que se realizó en el proyecto "Bot para evaluación colavorativa de ejercicios orales" de Pablo Uribe, y se ha implementado una página web que hace la función de panel de administración.

Actualmente no existe un registro para el profesor, es por ello que para poder utilizarlo, hay que introducir los datos del profesor directamente en la base de datos. En el manual de instalación (Anexo A) se explica como realizar este paso.

Cuando se empezó con el desarrollo de la página, había una cosa clara. Las tablas que se usaran en ella se quería que se pudieran ordenar sus campos y poder buscar elementos en ellas. Al comentar esto con Juanan, él me recomendó usar una librería llamada DataTables. Esta librería hace que las tablas sean interactivas y, además, las funcionalidades de ordenación y

búsqueda de elementos vienen realizadas por defecto.

Para poder usar la librería DataTables, es necesario usar JQuery. Esta es una librería en JavaScript que fácilita la interacción con los elementos de HTML. En el aprendizaje del uso de las librerías se pasó bastante tiempo, ya que DataTables tiene muchas opciones y funcionalidades.

Por cada tabla que utiliza la librería DataTables, existen tres ficheros. El primero es un .php que contiene el código HTML de la tabla junto con la llamada al segundo fichero. Este segundo es un archivo JavaScript. En él, se espera a que la página esté cargada completamente para realizar una llamada al tercer fichero mediante AJAX. El tercer fichero es otro .php que se encarga de imprimir un JSON que es recogido por el archivo JavaScript que lo llamó mediante AJAX. El JSON contiene todos los datos que se pondrán en la tabla. Al recoger esos datos, el fichero JavaScript se encarga de ir introduciéndolos en los campos de la tabla. Para finalizar, se especifican algunas opciones de la tabla, como por ejemplo que tenga paginación, sea *scrollable* o que algunos campos no se puedan ordenar.

Después de conseguir realizar la primera tabla, hacer las demás fue sencillo. Lo que realmente llevó tiempo, fue identificar los campos que serían útiles para el cliente. En algunos casos se acertó y en otros casos hubo que añadir más campos al realizar la evaluación de la herramienta.

Además de la librería DataTables, también se ha usado una librería llamada multi. Esta es una librería en JavaScript que permite crear cajas de selección con una apariencia atractiva y adaptada a dispositivos móviles. Se ha utilizado para crear la caja de selección de los criterios que se evaluarán en una pregunta de voz.

Con esta librería también ha habido algunos problemas, ya que al cargar los datos para hacer una modificación no se cargaban bien los elementos que ya estaban seleccionados. Es por ello que se ha tenido que hacer una modificación en la librería para que la carga funcione correctamente. Pero al hacer esto, se ha tenido que analizar todo el código de la librería para hallar la forma de modificarlo de manera que haga lo que se quería en este proyecto. La modificación consiste en que al llamar a la constructora se carguen las opciones que se encontraban seleccionadas en el cuadro de la derecha (mirar Figura 31). Esto ha llevado más tiempo de lo que se tenía pensado, ya que fue una de las primeras funcionalidades que se implementó y todavía no tenía demasiada soltura con JavaScript.

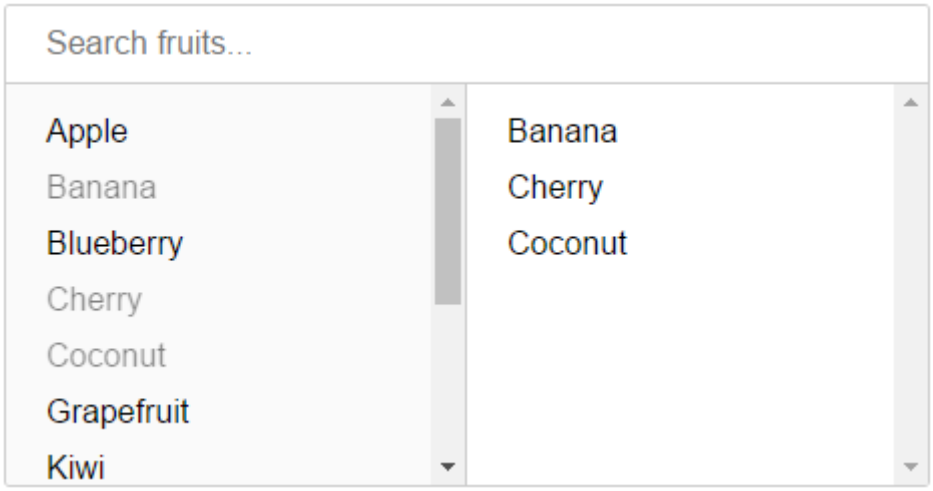


Figura 31: Ejemplo de uso de la librería multi.

9. Pruebas

En este capítulo se muestran las pruebas que se han realizado para la comprobación del correcto funcionamiento de las herramientas desarrolladas durante el proyecto. Hay que comentar que por falta de tiempo, las pruebas no se han podido automatizar.

9.1. Pruebas de las funcionalidades del *bot*

En este apartado se muestran las pruebas que se han realizado para comprobar que las funcionalidades del *bot* funcionan correctamente. Para realizar la comprobación, se intentaron automatizar las pruebas existentes sin éxito. Esto se debe a que, como el usuario ha de estar registrado para usar el *bot*, las pruebas tienen que simular que un estudiante registrado está enviando los mensajes. Por la estructura que en estos momentos la librería php-telegram-bot esto no es posible. Además tampoco ha sobrado demasiado tiempo como para desarrollarlo antes de que termine el plazo de entrega del proyecto.

Comando /test - Temas
Descripción de la prueba: Al enviar el mensaje "/test", el *bot* envía el listado de *inline buttons* con todos los temas de *tests* disponibles.
Salida esperada: Listado de *inline buttons* con todos los *tests* disponibles.
Resultado obtenido: Éxito.

Comando /test - Preguntas de test
Descripción de la prueba: Al pulsar un botón del listado de temas, el *bot* envía la primera pregunta del *test* escogido.
Salida esperada: Mensaje con la pregunta, respuestas y el teclado correctos.
Resultado obtenido: Éxito.

Comando /test - Posición aleatoria de las respuestas de las preguntas de test
Descripción de la prueba: Teniendo la opción de poner las respuestas de los *tests* en posiciones aleatorias activa, se prueba a responder una pregunta de un *test* concreto diez veces. De todas esas preguntas que se envían, como mínimo en una de las veces que se envía, las respuestas han de estar en diferentes posiciones. Además, si se responde correctamente a la pregunta, el *bot* ha de responder que la respuesta es correcta o incorrecta en el caso contrario.
Salida esperada: Las respuestas están en posiciones diferentes. Además, al responder el *bot* actúa correctamente.
Resultado obtenido: Éxito.

Comando /test - Posición fija de las respuestas de las preguntas de test
Descripción de la prueba: Teniendo la opción de poner las respuestas de los *tests* en posiciones aleatorias desactivada, se prueba a responder varias preguntas de diferentes *test*. En ellas, las respuestas siempre estarán en la misma posición y, cuando se responda, el *bot* ha de actuar correctamente tanto si la respuesta es correcta como si no lo es.
Salida esperada: En las diferentes preguntas, las respuestas siempre están en las mismas posiciones. Al responder, el *bot* actúa de manera correcta.
Resultado obtenido: Éxito.

Comando /test - Opciones después de responder las preguntas de test
Descripción de la prueba: Para esta prueba se tendrán en cuanta las siguientes situaciones: ▪ La pregunta de *test* tiene escritos los textos que indican por qué la respuesta es correcta o por qué no. ▪ La pregunta es la última de uno de los *tests*. ▪ La pregunta es la última del último *test* disponible. ▪ La pregunta es la última de uno de los *tests* y el siguiente *test* no tiene preguntas disponibles. Para cada caso han de aparecer las opciones pertinentes.
Salida esperada: Para cada caso aparecen las opciones que han de aparecer.
Resultado obtenido: Éxito.

Comando /status
Descripción de la prueba: Al enviar el mensaje "/status", el *bot* ha de responder con las estadísticas del estudiante que ha enviado el mensaje.
Salida esperada: Se envía un mensaje con las estadísticas.
Resultado obtenido: Éxito.

Comando /voice - Listado de preguntas de voz
Descripción de la prueba: Cuando un estudiante escribe "/voice" se le envían las preguntas de voz que tiene asignadas. Entre esas preguntas no se encontrarán las que se ha pasado su *deadline* ni las que ya se han dado el máximo de respuestas.
Salida esperada: Se envía el listado con las preguntas que se pueden responder.
Resultado obtenido: Éxito.

Comando /voice - Responder una pregunta de voz
Descripción de la prueba: Se responderán varias preguntas de voz. Para responder, se enviarán mensajes de texto, imágenes y grabaciones de voz. El *bot*, ha de mandar un mensaje advirtiendo de que solo se pueden enviar grabaciones de voz en los casos de los mensajes de texto e imágenes. En el caso de las grabaciones de voz, ha de descargarlas en una carpeta especifica del servidor.
Salida esperada: Para los casos de los mensajes y las imágenes, se envía correctamente el mensaje que advierte que solo se permiten grabaciones. En el caso de enviar una grabación, esta se descarga correctamente en la carpeta especifica del servidor.
Resultado obtenido: Éxito.

Comando /change - Listado de preguntas
Descripción de la prueba: Al enviar el mensaje "/change", el *bot* responde con el listado de preguntas asociadas a otra lista de *inline buttons*. En el listado solo aparecen las preguntas a las que ya se les haya dado más de una respuesta.
Salida esperada: Aparece el listado de preguntas en las que se ha dado más de una respuesta.
Resultado obtenido: Éxito.

Comando /change - Respuestas dadas a una pregunta
Descripción de la prueba: Al escoger una pregunta a la que cambiar la respuesta, se envía la pregunta con todas las respuestas que el estudiante ha dado a esa pregunta. La respuesta que el estudiante tiene seleccionada actualmente tiene que estar indicada mediante un mensaje
Salida esperada: Se envían todas las respuestas y la que se tiene seleccionada en cada momento está indicada correctamente con un mensaje.
Resultado obtenido: Éxito.

Comando /change - Cambio de respuesta
Descripción de la prueba: Esta prueba consiste en cambiar la selección de la respuesta a evaluar para comprobar que el cambio se realiza correctamente.
Salida esperada: El cambio de la respuesta se realiza correctamente.
Resultado obtenido: Éxito.

Comando /eval - Listado de evaluaciones a realizar
Descripción de la prueba: Al mandar el mensaje "/eval", se le ha de enviar un listado de las posibles evaluaciones. Solo aparecerán las evaluaciones de las preguntas de las que se haya pasado el *deadline* y de las que se tenga una respuesta que evaluar.
Salida esperada: Se envía el listado de las evaluaciones a realizar se envía correctamente.
Resultado obtenido: Éxito.

Comando /eval - Evaluar una respuesta
Descripción de la prueba: Cuando el estudiante escoge una respuesta a evaluar se le envía toda la información de la respuesta. El estudiante solo podrá evaluar la respuesta con la escala que se indica y que su profesor tiene seleccionada en sus opciones.
Salida esperada: La grabación recibe la nota que el estudiante le ha otorgado y esta se guarda correctamente en la base de datos.
Resultado obtenido: Éxito.

Comando /eval - Explicación de voz
Descripción de la prueba: Después de evaluar todos los criterios, se le pregunta al estudiante si quiere dar una explicación de voz. En esta prueba se comprueba que la grabación que se da como explicación se descargue correctamente en el servidor y que se guarden los datos correctamente en la base de datos.
Salida esperada: La grabación se descarga correctamente en el servidor y los datos se guardan correctamente en la base de datos.
Resultado obtenido: Éxito.

Comando /grades - Listado de preguntas
Descripción de la prueba: Al mandar el mensaje "/grades", el estudiante tiene que recibir un listado de las preguntas que respondió y que como mínimo, se le ha dado una calificación en algún criterio.
Salida esperada: Se envía el listado de las preguntas .
Resultado obtenido: Éxito.

Comando /grades - Mostrar calificaciones
Descripción de la prueba: Al escoger una pregunta de la que se quieren saber las calificaciones, el estudiante ha de recibir la pregunta, la respuesta que dio, las calificaciones en todos los criterios y las explicaciones que los evaluadores dieron a las calificaciones.
Salida esperada: El *bot* envía todos los datos de las evaluaciones que realizaron a la respuesta escogida.
Resultado obtenido: Éxito.

Notificaciones
Descripción de la prueba: Al ejecutar el archivo Notification.php en el servidor, los estudiantes que tienen alguna evaluación o alguna pregunta asignada han de recibir un mensaje que les avise.
Salida esperada: Solo los estudiantes que tienen una asignación reciben el mensaje de notificación.
Resultado obtenido: Éxito.

Conversión de ficheros .oga a .ogg
Descripción de la prueba: Al ejecutar el fichero Fileformat.php en el servidor, todos los ficheros .oga se convierten a .ogg en carpetas ordenadas por grupo y estudiante. Además, en la base de datos se modifican los campos que indican el *path* en el que se encuentran los nuevos ficheros.
Salida esperada: Todos los ficheros se cambian de formato correctamente y los datos en la base de datos se modifican correctamente.
Resultado obtenido: Éxito.

9.2. Pruebas de las funcionalidades del panel de administración

Es este apartado se muestran en tablas las pruebas realizadas para comprobar que el panel de administración funciona correctamente.

Login
Descripción de la prueba: Esta prueba consiste en introducir los datos de un usuario que no exista en la base de datos, los datos de un estudiante y los de un profesor. También se comprobará si en el caso de introducir los datos del profesor entra en el panel y que se ha creado una sesión con ese usuario.
Salida esperada: Mensaje de error al introducir los datos del usuario que no existe y del estudiante. Al introducir los datos del profesor se crea una sesión y se entra al panel de administración.
Resultado obtenido: Éxito.

Preguntas - Crear pregunta de voz
Descripción de la prueba: Se van a crear varias preguntas de voz para ver si los datos se guardan correctamente al guardar.
Salida esperada: Los datos han de estar guardados correctamente en la base de datos.
Resultado obtenido: Éxito.

Preguntas - Eliminar pregunta de voz
Descripción de la prueba: Se eliminarán varias preguntas ya creadas para comprobar que desaparecen de la base de datos.
Salida esperada: Los datos son borrados correctamente dela base de datos.
Resultado obtenido: Éxito.

Preguntas - Subir imagen
Descripción de la prueba: Al crear varias preguntas, se van a subir imágenes al servidor que irán asociadas a ellas. Además de esto se probará que el *bot* envía las imágenes con las preguntas.
Salida esperada: La imágenes han de estar almacenadas en la carpeta especifica del servidor donde esas se guardan. Además el *bot* envía la imagen adjunta a la pregunta.
Resultado obtenido: Éxito.

Preguntas - Crear criterio de evaluación
Descripción de la prueba: En esta prueba se crean varios criterios. Estos han de aparecer en el cuadro de selección.
Salida esperada: Además de aparecer los nuevos criterios en el cuadro de selección, también tienen que estar guardados en la base de datos.
Resultado obtenido: Éxito.

Preguntas - Eliminar criterio de evaluación
Descripción de la prueba: Se eliminarán algunos de los criterios creados.
Salida esperada: Los criterios eliminados han de desaparecer del cuadro de selección y de la base de datos.
Resultado obtenido: Éxito.

Preguntas - Selección automática
Descripción de la prueba: Primero se crea una nueva pregunta y se comprueba que se haya hecho una selección aleatoria de estudiantes. En cada estudiante seleccionado se comprueba si también se han asignado los evaluadores de manera aleatoria.
Salida esperada: Los estudiantes a los que se les ha asignado la pregunta también se les han asignado unos evaluadores de manera aleatoria.
Resultado obtenido: Éxito.

Preguntas - Cargar datos de preguntas de voz
Descripción de la prueba: Al cargar una pregunta, se comprueba que los datos de la pregunta se han cargado correctamente. Además, también se comprueba que los estudiantes a los que se les ha asignado la pregunta y sus evaluadores también se han cargado correctamente.
Salida esperada: Todos los datos se han cargado correctamente en sus campos.
Resultado obtenido: Éxito.

Preguntas - Modificar datos de pregunta de voz
Descripción de la prueba: Al cambiar los datos de una pregunta, se comprueba que los datos también se cambien en la base de datos.
Salida esperada: Los datos de la pregunta modificada han sido cambiados también en la base de datos.
Resultado obtenido: Éxito.

Grupos - Crear grupos de estudiantes
Descripción de la prueba: Se crearán varios grupos de estudiantes.
Salida esperada: Los datos de los nuevos grupos han de estar guardaros en la base de datos.
Resultado obtenido: Éxito.

Grupos - Eliminar grupos de estudiantes
Descripción de la prueba: Se eliminan varios de los grupos creados.
Salida esperada: Los datos de los grupos se han eliminado de la base de datos.
Resultado obtenido: Éxito.

Grupos - Añadir estudiantes a un grupo
Descripción de la prueba: Se añaden varios usuarios a un grupo de alumnos.
Salida esperada: Los estudiantes han de estar registrados como estudiantes del grupo al que se han añadido y ahora podrán usar los comandos del *bot*.
Resultado obtenido: Éxito.

Grupos - Eliminar estudiantes de un grupo
Descripción de la prueba: Se eliminarán varios estudiantes de un mismo grupo.
Salida esperada: Los que antes eran estudiantes, ahora son usuarios normales, no están registrados y no pueden usar los comandos del *bot*.
Resultado obtenido: Éxito.

Evaluaciones - Datos de una evaluación
Descripción de la prueba: Al entrar a la página de una evaluación, los audios han de escucharse y los datos de las tablas de calificaciones tienen que ser correctos.
Salida esperada: Los audios se escuchan y los datos tienen que ser correctos.
Resultado obtenido: Éxito.

Gráficos - Mostrar gráficos
Descripción de la prueba: Al pulsar un botón que lleva a alguno de los gráficos, se ha de abrir una pestaña con el gráfico correcto y con los datos indicados.
Salida esperada: Se ha de mostrar un
Resultado obtenido: Éxito.

Opciones - Modificar opciones
Descripción de la prueba: Al cambiar las opciones estas se tienen que guardar en la base de datos. Además los cambios se tienen que aplicar. Es decir, si modifica la opción de las posiciones de las respuestas de *test* para que sean aleatorias, que esto se cumpla.
Salida esperada: Las opciones se guardan correctamente en la base de datos y los cambios se aplican, ya sea en el *bot* como en el panel de administración.
Resultado obtenido: Éxito.

9.3. Prueba @myqueridobot VS @pruebasbot

Esta prueba consiste en una comparación de tiempos entre @myqueridobot y @pruebasbot. Los tiempos representan lo que tarda cada *bot* en responder al ejecutar un comando.

Esta prueba se ha realizado para comprobar si se hizo bien en descartar lo que Pablo Uribe desarrollo al crear @pruebasbot. Para ello, se ha ejecutado diez veces cada comando con cada uno de los *bots* en buenas condiciones de conexión vía Wi-Fi, con mala conexión vía Wi-Fi y con datos móviles. En la Tabla 6 se muestra la media de los tiempos obtenida en cada ocasión por cada *bot*.

Como se puede comprobar con los datos de la tabla, @myqueridobot tarda menos en responder que @pruebasbot. Esto indica que utilizar PHP en vez del *framework* de Microsoft que usó Pablo ha sido un acierto, ya que por esa misma razón se decidió realizar el proyecto con PHP.

9.4. Evaluación de la herramienta

Como ya se ha comentado durante este documento, la herramienta desarrollada durante el proyecto ha sido utilizada en las clases de la asignatura Desarrollo de Aplicaciones Web Avanzadas (DAWE). Esta asignatura, es una optativa que se imparte de manera no presencial en cuarto curso del grado de Ingeniería Informática de Gestión y Sistemas de Información en la Universidad del País Vasco.

El panel de administración ha sido usado por el profesor de DAWE y director de este proyecto Juanan Pereira. En cuanto al *bot*, lo han usado 26 de los alumnos que estaban matriculados en la asignatura durante el curso 2016-2017. Ellos han realizado los diferentes *tests* que el *bot* tuvo disponibles

@pruebasbot			
Comando\Conexión	Buena Wi-Fi	Mala Wi-Fi	Datos
help	00:00:05.389	00:00:09.104	00:00:09.065
test temas	00:00:04.667	00:00:07.359	00:00:11.688
test escoger tema	00:00:13.193	00:00:14.347	00:00:12.016
test respuesta	00:00:03.184	00:00:08.121	00:00:06.506
test siguiente pregunta	00:00:04.343	00:00:03.008	00:00:14.560
status	00:00:04.960	00:00:07.423	00:00:07.848
voice	00:00:06.008	00:00:05.089	00:00:05.529
evaluate	00:00:05.808	00:00:04.944	00:00:12.506

@myqueridobot			
Comando\Conexión	Buena Wi-Fi	Mala Wi-Fi	Datos
help	00:00:02.760	00:00:02.888	00:00:01.975
test temas	00:00:02.761	00:00:02.256	00:00:03.216
test escoger tema	00:00:00.968	00:00:00.760	00:00:01.831
test respuesta	00:00:01.479	00:00:02.840	00:00:03.344
test siguiente pregunta	00:00:02.745	00:00:03.121	00:00:03.200
status	00:00:02.768	00:00:02.904	00:00:03.529
voice	00:00:02.896	00:00:02.857	00:00:03.448
evaluate	00:00:02.744	00:00:02.432	00:00:02.128

Tabla 6: Tiempos de @myqueridobot VS @pruebasbot.

durante todo el curso. Además de eso, a cada estudiante se le han asignado seis preguntas de voz repartidas en dos tandas.

En la primera, se asignaron tres preguntas a cada estudiante a mediados del cuatrimestre. En aquella ocasión no hubo mucha participación, ya que solo diez estudiantes respondieron las preguntas de voz antes del *deadline*. Aunque algunos de ellos no llegaron a responder todas las que se les asignaron, otros llegaron a enviar más de una respuesta de voz a la misma pregunta. Esto se sabe porque al mirar la base de datos se muestra que se han recibido 33 respuestas en total y varias no están seleccionadas para evaluar.

A la hora de la evaluación, se detectaron varias incidencias. Juanan quería asignar las evaluaciones entre los estudiantes que hubieran respondido la misma pregunta. En aquel momento, él no podía saber quien había respondido cada pregunta. Por ello, se me pidió que en el apartado de asignación pusiera un campo en el que se mostrara quien había respondido la pregunta que se estaba modificando.Después de esto, surgió otro problema que no se había tenido en cuenta.

Al realizar una asignación mientras se utilizaban los filtros propios de la librería DataTables, no se guardaban las asignaciones. Esto se debe a que, para guardar la asignación, se van recorriendo las filas de la tabla mirando si el *checkbox* está marcado o no. Al realizar una búsqueda con el filtro de DataTables, las filas que no entran en la búsqueda desaparecen de la tabla, lo cual no permitía que se guardaran. Para solventar el problema, antes de recorrer la tabla, se realiza una búsqueda vacía mediante código. Con vacía me refiero a que se busca ““. Eso hace que aparezcan todas las filas de la tabla y esta se pueda recorrer entera.

Al terminar con este problema, Juanan modificó la asignación de evaluadores, pero por un fallo del diseño de la interfaz y por un problema de concepto, la asignación no salió como se esperaba. Juanan realizó las modificaciones en el *iframe* de la asignación de evaluadores y pulsó el botón “*save*“. El problema era que, en pantalla de asignación de la pregunta (no en el *iframe* de la asignación de evaluadores), el botón de guardar la pregunta, se encontraba debajo de la tabla en la que aparecen los estudiantes a los que se les asignará la pregunta para responder. Para que realmente se guarden todos los cambios hay que pulsar ese botón, pero como las modificaciones se realizaron desde un portátil de pantalla pequeña, en el cual, el botón no se llegaba a ver. Es por ello que después de que esto pasara, el botón se posicionó encima de la tabla y se ha puesto un mensaje advirtiendo de que para guardar los cambios hay que presionar ese botón.

Al final de esta fase se recibieron 39 evaluaciones, de las cuales, cinco tenían alguna grabación de voz explicando el por qué de las calificaciones dadas.

En la segunda prueba de la herramienta, se realizaron 99 asignaciones repartidas en 10 preguntas. A cada uno de los 32 estudiantes participantes, se le asignaron un mínimo de 3 preguntas para contestar y 15 de máximo. Se dejó un periodo de una semana para contestar a las preguntas, en el cual recibieron 86 respuestas. Durante los 4 días siguientes, se recibieron 136 evaluaciones de todas las respuestas dadas por los estudiantes.

Antes de la segunda prueba, se realizó una encuesta a los estudiantes de DAWE. En ella, el 86 % de los 22 encuestados indicaron que los *tests* son útiles para el estudio de la asignatura. También se recibieron un 65 % de respuestas favorables a la pregunta: “¿Mantendrías las preguntas de respuesta verbal en el futuro? “.

En cuanto a la prueba, se han llegado a varias conclusiones.

- A los estudiantes les ha gustado el apartado de *tests* del *bot*. Se han recibido más de 1000 respuestas a las diferentes preguntas de *test* disponibles en el *bot*, siendo gran parte de ellas correctas.
- Aunque las preguntas de voz no han tenido tanta participación como los *tests*, se han recibido más contestaciones de las que se esperaban. Para haber sido un primer contacto, los estudiantes también han realizado un gran trabajo, ya que la peor calificación que han recibido las respuestas ha sido de 6 puntos sobre 10. Con ello también se ha comprobado que los estudiantes que respondieron en la primera prueba, han respondido más seguros en la segunda prueba.

Aunque se hayan llegado a estas conclusiones, las pruebas se han realizado con un grupo pequeño de estudiantes. Es por ello que, tampoco se ha llegado a una conclusión clara a la hora de evaluar la mejora en la expresión oral, o de relacionar el uso del *bot* con las notas finales de los estudiantes en la asignatura DAWE. Para dar una respuesta clara a si realmente el uso del *bot* mejora las capacidades de los estudiantes, es necesaria la realización de más pruebas.

10. Conclusiones

En este último capítulo, se va realizar un análisis entre la planificación que se estimó al comienzo del proyecto y cómo ha acabado realmente. También se van a plantear las posibles lineas futuras para el proyecto, las licencias utilizadas y, por último, una pequeña reflexión personal.

10.1. Análisis entre planificación estimada y real

A continuación, se exponen los aspectos que no se tuvieron en cuenta a la hora de planificar el proyecto y que han repercutido severamente en el desarrollo de este.

10.1.1. Objetivos

Durante el desarrollo del proyecto, apareció un nuevo objetivo que no se había tenido en cuenta hasta que se tuvo desarrollada una versión inicial del *bot*. Al mostrarle esa primera versión al director del proyecto, Juanan Pereira, me comentó que tenía pensado presentar el *bot* en congresos y distribuirlo de esa manera. Por ello, se identificó el siguiente objetivo para el desarrollo del proyecto:

- **Multiples idiomas y explicaciones en inglés**: Como los congresos no solo serán en España, el *bot* tiene que estar preparado para poder cambiar su idioma fácilmente. Además, los comentarios en el código han de estar completamente en inglés por si a alguien quiere modificarlo.

10.1.2. Herramientas utilizadas

Además de las herramientas mencionadas en el alcance (Sección 3.2), también se han usado las siguientes herramientas:

- **Cron**: Es un administrador regular de procesos en segundo plano en los sistemas operativos Unix. Se utiliza para ejecutar procesos o guiones en intervalos regulares. Este administrador de procesos es utilizado para enviar notificaciones mediante el *bot*. También llama al fichero que se encarga de la conversión de las grabaciones de voz periódicamente.

- **DataTables**: Es un complemento a la librería JQuery que permite un control e interacción avanzados sobre tablas HTML. Permite la búsqueda, paginación y ordenación de los elementos de las tablas de manera sencilla. Es utilizada para casi todas las tablas existentes en el panel de administración. Su página oficial es: `https://datatables.net/`.

- **FFmpeg**: Es una colección de software libre que puede grabar, convertir y hacer *streaming* de audio y vídeo. Tiene desarrolladas librerías como libavcodec, libavutil o libavformat, con las cuales se permite la conversión de tipos en ficheros de vídeo y audio. En este proyecto se utiliza este programa junto con una librería que se llama igual para tratar un problema que repercute en el panel de control. La página web oficial del programa es `https://ffmpeg.org/`, y la de la librería es `https://github.com/FFmpeg/FFmpeg`.

- **JQuery**: Es una librería gratuita de JavaScript que facilita la interacción con elementos HTML, ayuda a la creación de animaciones e incluso permite agregar interacción entre páginas web gracias a AJAX. Es usado por panel de administración para recibir los datos que se necesite en cada página de la Web. Su página web oficial es: `https://jquery.com/`.

- **multi**: Es una librería que hace más sencilla la interacción con las cajas de selección múltiple de HTML. Ha sido utilizada en el panel de administración para la selección de criterios a evaluar en una pregunta de voz. La librería se encuentra en `https://github.com/Fabianlindfors/multi.js?utm_source=frontendfocus&utm_medium=email`.

10.1.3. Alcance

En cuanto al alcance, los bloques de tareas principales que aparecen en la Figura 3 se han mantenido iguales. Pero, dentro de cada bloque si se han tenido que añadir nuevos bloques debido a las nuevas funcionalidades que se identificaron durante el desarrollo.

Aprendizaje

En este bloque se han añadido las tareas de aprendizaje de FFmpeg, DataTables y multi. Las dos últimas librerías son usadas en el panel de administración para añadir usabilidad y estética a algunas de sus funcionalidades. FFmpeg es la librería que se ha usado para la conversión de los ficheros .oga

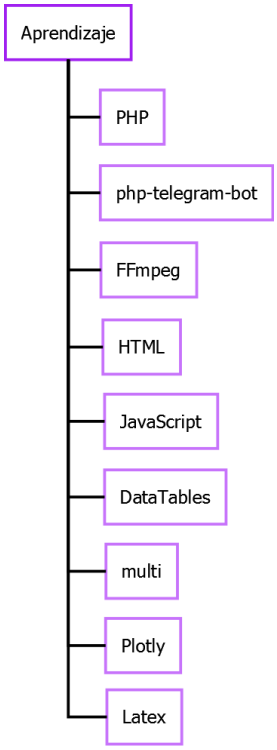


Figura 32: Diagrama EDT final del bloque de aprendizaje.

a .ogg debido al problema que se comenta en el capítulo de implementación y desarrollo (Sección 8.3).

Aprendizaje de FFmpeg
Paquete de trabajo: Aprendizaje.
Descripción: Aprendizaje del uso de la librería y del programa FFmpeg.
Salidas/Entregables: N/A.
Recursos necesarios: Sublime Text 2, PHP, librería FFmpeg, programa FFmpeg y fichero .oga.

Aprendizaje de DataTables
Paquete de trabajo: Aprendizaje.
Descripción: Aprendizaje del uso de la librería DataTables. Para ello también aprender JQuery, AJAX y todas las opciones que la librería DataTables ofrece.
Salidas/Entregables: N/A.
Recursos necesarios: Sublime Text 2, PHP, librería JQuery y librería DataTables.

Aprendizaje de multi
Paquete de trabajo: Aprendizaje.
Descripción: Aprendizaje del funcionamiento de la librería multi.
Salidas/Entregables: N/A.
Recursos necesarios: Sublime Text 2, PHP, Google Chrome y librería multi.

Organización

En este apartado, el tiempo utilizado ha sido mayor al estimado debido a las nuevas funcionalidades que se iban identificando durante el desarrollo y la evaluación de la herramienta. Además, el haber tenido que reinstalar algunas de las herramientas por la diferencia de versiones, también ha hecho que se aumente el tiempo utilizado en las tareas de este bloque.

Captura de requisitos

En la captura de requisitos se ha recortado tiempo respecto al estimado. Esto se debe a que se pensó que se tardaría más tiempo del que realmente se ha tardado en realizar los prototipos.

Análisis y diseño

En este bloque, se ha tardado más de lo esperado. Cada vez que aparecía una nueva funcionalidad o algún cambio en lo ya existente, había que modificar los diagramas que se relacionan con las tareas de este bloque.

Implementación y desarrollo

Este bloque es el que mayor diferencia de duración tiene respecto a la estimación que se realizó al comienzo del proyecto. Todos los problemas, modificaciones y nuevas funcionalidades han repercutido en él. Esto ha creado un desfase de cien horas de duración respecto a lo que se pensó en el planteamiento inicial.

En la Figura 33, se muestra el nuevo apartado, formado por los ficheros que se han preparado para ejecutar con Cron. En las siguientes tablas, se muestran todas las nuevas tareas que se identificaron en este bloque.

Estas primeras tablas representan las nuevas tareas añadidas al apartado del desarrollo del *bot*:

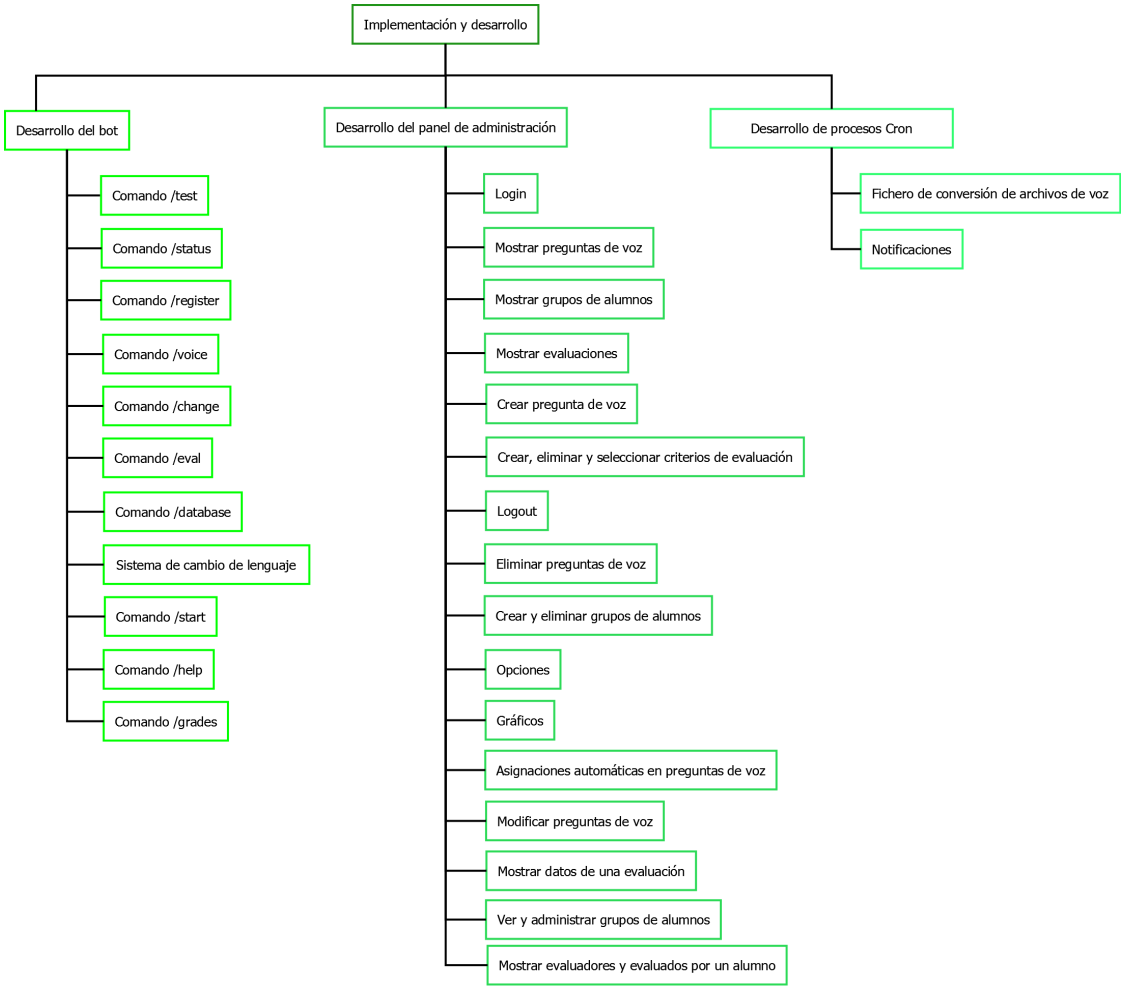


Figura 33: Diagrama EDT final del bloque de implementación y desarrollo.

Comando /database
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*.
Descripción: Implementación de las clases y métodos necesarios para la ejecución de un comando que permita al administrador interactuar con la base de datos directamente desde el *bot*.
Salidas/Entregables: Ficheros que componen el comando /database.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.

Sistema de cambio de lenguaje
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*.
Descripción: Implementación de las clases y métodos necesarios para la creación de un sistema que permita cambiar el lenguaje de los textos que el *bot* envía. Además, se creará una plantilla que permita a traductores añadir nuevos idiomas.
Salidas/Entregables: Ficheros necesarios para la creación del sistema de cambio de lenguaje, ficheros con los mensajes en inglés y español y la plantilla para traductores.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.
Comando /start
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*.
Descripción: Modificación del comando /start que viene por defecto para que envíe un mensaje indicando que es lo que hace el *bot*.
Salidas/Entregables: Ficheros modificados del comando /start.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.
Comando /help
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*.
Descripción: Modificación de los ficheros que ejecutan el comando /help para ocultar los comandos del administrador y el comando de registro.
Salidas/Entregables: Ficheros modificados del comando /help.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.
Comando /grades
Paquete de trabajo: Implementación y desarrollo - Desarrollo del *bot*.
Descripción: Implementación de las clases y métodos necesarios para la ejecución de un comando que permita a los estudiantes ver qué notas les han puesto sus compañeros en las evaluaciones de voz.
Salidas/Entregables: Ficheros que implementen el comando /grades.
Recursos necesarios: Sublime Text 2, librería php-telegram-bot, Apache2, Telegram y un ordenador y *smartphone* con conexión a Internet.

Las siguientes tablas pertenecen a las nuevas funcionalidades del apartado de desarrollo del panel de administración:

Mostrar grupos de alumnos
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración.
Descripción: Desarrollo de una página web que muestre los grupos de alumnos que un profesor ha creado.
Salidas/Entregables: Ficheros con el código desarrollado para la creación de esta funcionalidad.
Recursos necesarios: Sublime Text 2, Google Chrome y Apache2.
Crear, eliminar y seleccionar criterios de evaluación
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración.
Descripción: Implementación del código necesario para poder realizar la administración de los criterios de evaluación de las preguntas de voz.
Salidas/Entregables: Ficheros de código creados.
Recursos necesarios: Sublime Text 2, Google Chrome y Apache2.
Crear y eliminar grupos de alumnos
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración.
Descripción: Desarrollo del código necesario para la administración de grupos de estudiantes.
Salidas/Entregables: Ficheros de código desarrollados para esta funcionalidad.
Recursos necesarios: Sublime Text 2, Google Chrome y Apache2.
Ver y administrar grupos de alumnos
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración.
Descripción: Implementación del código necesario para poder añadir y eliminar estudiantes de un grupo de alumnos.
Salidas/Entregables: Ficheros desarrollados para esta funcionalidad.
Recursos necesarios: Sublime Text 2, Google Chrome y Apache2.
Mostrar evaluadores y evaluados por alumno
Paquete de trabajo: Implementación y desarrollo - Desarrollo del panel de administración.
Descripción: Implementar el código necesario para poder mostrar la información de cada uno de los estudiantes. La información consta de los estudiantes evaluados y los evaluadores de cada alumno en las diferentes preguntas de voz.
Salidas/Entregables: Ficheros de código implementados para el apartado "*Student Information*".
Recursos necesarios: Sublime Text 2, Google Chrome y Apache2.

Las siguientes tablas representan las tareas del apartado de desarrollo de procesos Cron:

Fichero de conversión de archivos de voz
Paquete de trabajo: Implementación y desarrollo - Desarrollo de procesos Cron.
Descripción: Implementación del fichero que convierte los ficheros de audio de .oga a .ogg y los guarda en carpetas ordenadas por grupo y estudiante.
Salidas/Entregables: Fichero que contiene el código para la conversión de ficheros.
Recursos necesarios: Sublime Text 2, librería FFmpeg, programa FFmpeg y PHP.
Notificaciones
Paquete de trabajo: Implementación y desarrollo - Desarrollo de procesos Cron.
Descripción: Implementación de los ficheros que permiten el envío de notificaciones a los estudiantes que tengan preguntas o evaluaciones asignadas.
Salidas/Entregables: Ficheros que contienen el código necesario para el envío de notificaciones.
Recursos necesarios: Sublime Text 2, CURL y PHP.

Pruebas

Al no haber podido automatizar las pruebas por problemas con la librería php-telegram-bot y por falta de tiempo, en este apartado se ha tardado menos tiempo del estimado.

Documentación

Debido a las modificaciones y a la realización de las funcionalidades que no se tuvieron en cuenta en un primer momento, el tiempo real ha sido mayor al estimado.

Cuando se realizaban modificaciones en cualquiera de los diagramas, esto había que explicarlo en la memoria. Además, cada vez que se identificaba e implementaba una nueva funcionalidad afectaba considerablemente en la redacción de esta. Esto, junto con las correcciones que se han ido realizando, ha hecho que el tiempo real del apartado de documentación, haya superado considerablemente el tiempo que se estimó para su realización.

Duración de las tareas

Por todos los cambios comentados en esta sección, la duración real a sobrepasado con creces lo que se estimó en la planificación inicial del proyecto.

Por las modificaciones y los problemas que se encontraron durante la evaluación de la herramienta, la duración de algunas de las tareas llevaron más de lo previsto. En las Tablas 7, 8 y 9, se muestran todas las tareas, la estimación que se hizo en la planificación inicial y lo que ha durado realmente cada tarea.

Nombre de la tarea	Duración estimada	Duración real
Aprendizaje	95	122
PHP	30	28
php-telegram-bot	10	40
HTML	5	3
JavaScript	15	13
Plotly	5	2
Latex	30	15
FFmpeg	0	4
DataTables	0	13
multi	0	4
Organización	40	42
Planteamiento de la aplicación	3	4
Planificación de las tareas	5	7
Elección de software	2	1
Instalación del software	5	7
Reuniones con el director del proyecto	25	23
Captura de requisitos	29	26
Casos de uso	2	3
Diseño de interfaces	25	20
Modelo de dominio	2	3
Análisis y diseño	16	24
Diagrama de clases	4	6
Diagramas de secuencia	10	16
Diagrama de estados del bot	2	2

Tabla 7: Dependencias y duración de las tareas finales (1).

Como se puede comprobar en las tablas, en el único bloque en el que se ha recortado mucho tiempo ha sido en el de pruebas por no haber tenido tiempo suficiente para automatizarlas. En casi todos los demás bloques, la duración ha sido mayor a la estimada, dando así un total de 624 horas reales respecto a las 510 estimadas en la planificación inicial.

Implementación y desarrollo	156	256
Desarrollo del bot	69	115
Comando /test	20	27
Comando /status	2	4
Comando /register	5	4
Comando /voice	25	36
Comando /change	7	3
Comando /eval	10	16
Comando /database	0	1
Sistema de cambio de lenguaje	0	18
Comando /start	0	1
Comando /help	0	1
Comando /grades	0	4
Desarrollo del panel de administración	87	118
Login	2	1
Mostrar preguntas de voz	15	5
Mostrar evaluaciones	4	2
Crear pregunta de voz	20	26
Asignaciones automáticas en preguntas de voz	7	13
Modificar pregunta de voz	2	9
Logout	1	1
Eliminar preguntas de voz	5	2
Opciones	15	4
Gráficos	10	20
Mostrar datos de una evaluación	6	13

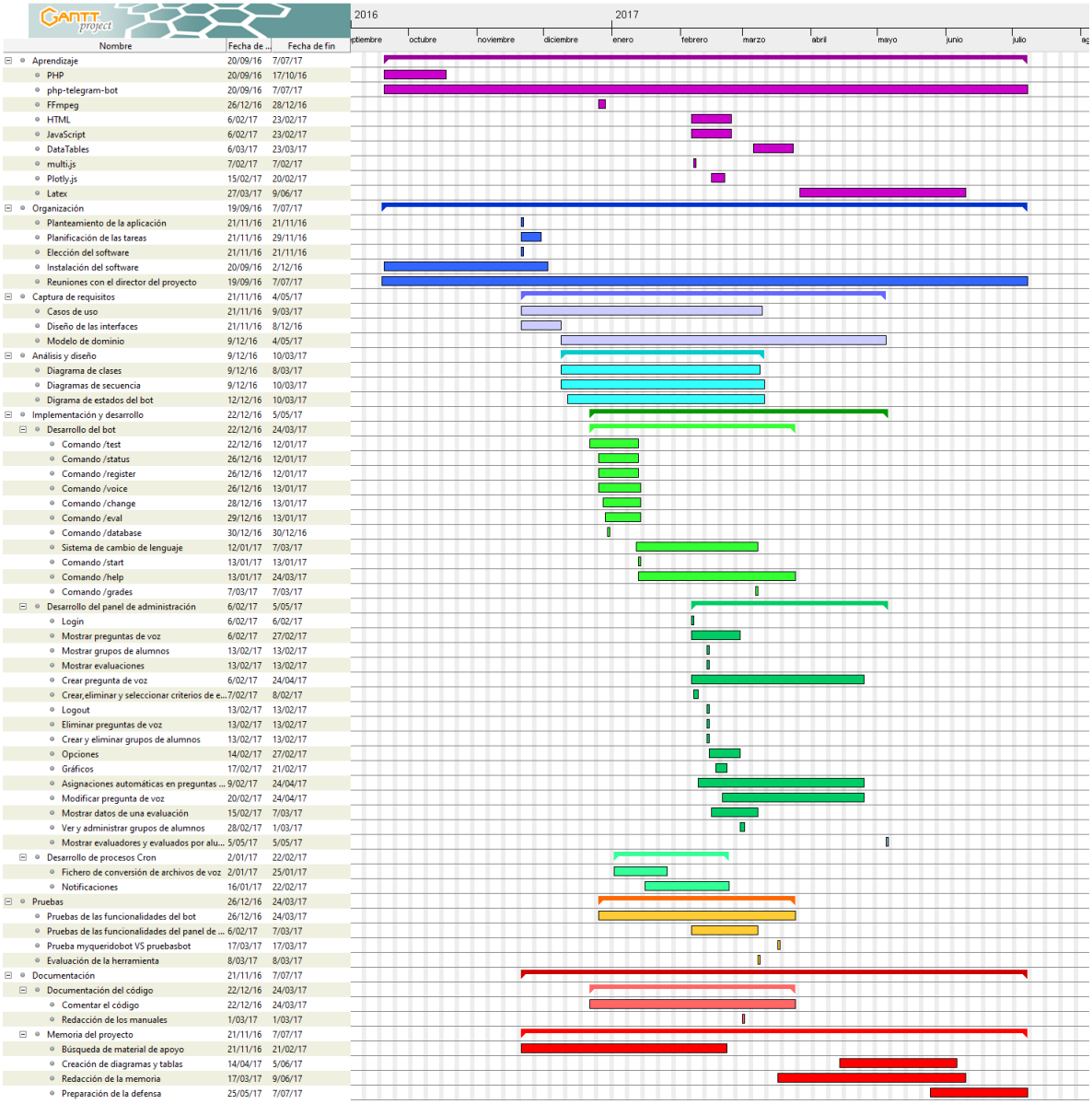
Tabla 8: Dependencias y duración de las tareas finales (2).

10.1.4. Planificación temporal

Como se ha tardado más de lo esperado y han aparecido nuevas tareas, en la Figura 34 se muestra como ha quedado el diagrama Gantt después de todas las modificaciones. Algunas de las tareas, al haber tenido que ser modificadas después de que se tuviera una primera versión, su duración se ha alargado por meses. Pero las horas que han durado ya se han expuesto en las Tablas 7, 8 y 9.

Mostrar grupos de alumnos	0	3
Crear, eliminar y seleccionar criterios de evaluación	0	4
Crear y eliminar grupos de alumnos	0	2
Ver y administrar grupos de alumnos	0	9
Mostrar evaluadores y evaluados por alumno	0	4
Desarrollo de procesos Cron	0	23
Fichero de conversión de archivos de voz	0	7
Notificaciones	0	16
Pruebas	66	22
Pruebas de las funcionalidades del bot	25	3
Pruebas de las funcionalidades del panel de administración	35	8
Prueba myqueridobot VS pruebasbot	1	2
Evaluación de la herramienta	5	9
Documentación	108	132
Documentación del código	25	33
Comentar el código	15	26
Redacción de manuales	10	7
Memoria del proyecto	83	99
Búsqueda de material de apoyo	5	3
Creación de diagramas y tablas	3	3
Redacción de la memoria	50	68
Preparación de la defensa	25	25
TOTAL	**510**	**624**

Tabla 9: Dependencias y duración de las tareas finales (3).



Nombre	Fecha de ...	Fecha de fin
Aprendizaje	20/09/16	7/07/17
PHP	20/09/16	17/10/16
php-telegram-bot	20/09/16	7/07/17
FFmpeg	26/12/16	28/12/16
HTML	6/02/17	23/02/17
JavaScript	6/02/17	23/02/17
DataTables	6/03/17	23/03/17
multi.js	7/02/17	7/02/17
Plotly.js	15/02/17	20/02/17
Latex	27/03/17	9/06/17
Organización	19/09/16	7/07/17
Planteamiento de la aplicación	21/11/16	21/11/16
Planificación de las tareas	21/11/16	29/11/16
Elección del software	21/11/16	21/11/16
Instalación del software	20/09/16	2/12/16
Reuniones con el director del proyecto	19/09/16	7/07/17
Captura de requisitos	21/11/16	4/05/17
Casos de uso	21/11/16	9/03/17
Diseño de las interfaces	21/11/16	8/12/16
Modelo de dominio	9/12/16	4/05/17
Análisis y diseño	9/12/16	10/03/17
Diagrama de clases	9/12/16	8/03/17
Diagramas de secuencia	9/12/16	10/03/17
Digrama de estados del bot	12/12/16	10/03/17
Implementación y desarrollo	22/12/16	5/05/17
Desarrollo del bot	22/12/16	24/03/17
Comando /test	22/12/16	12/01/17
Comando /status	26/12/16	12/01/17
Comando /register	26/12/16	12/01/17
Comando /voice	26/12/16	13/01/17
Comando /change	28/12/16	13/01/17
Comando /eval	29/12/16	13/01/17
Comando /database	30/12/16	30/12/16
Sistema de cambio de lenguaje	12/01/17	7/03/17
Comando /start	13/01/17	13/01/17
Comando /help	13/01/17	24/03/17
Comando /grades	7/03/17	7/03/17
Desarrollo del panel de administración	6/02/17	5/05/17
Login	6/02/17	6/02/17
Mostrar preguntas de voz	6/02/17	27/02/17
Mostrar grupos de alumnos	13/02/17	13/02/17
Mostrar evaluaciones	13/02/17	13/02/17
Crear pregunta de voz	6/02/17	24/04/17
Crear, eliminar y seleccionar criterios de e...	7/02/17	8/02/17
Logout	13/02/17	13/02/17
Eliminar preguntas de voz	13/02/17	13/02/17
Crear y eliminar grupos de alumnos	13/02/17	13/02/17
Opciones	14/02/17	27/02/17
Gráficos	17/02/17	21/02/17
Asignaciones automáticas en preguntas ...	9/02/17	24/04/17
Modificar pregunta de voz	20/02/17	24/04/17
Mostrar datos de una evaluación	15/02/17	7/03/17
Ver y administrar grupos de alumnos	28/02/17	1/03/17
Mostrar evaluadores y evaluados por alu...	5/05/17	5/05/17
Desarrollo de procesos Cron	2/01/17	22/02/17
Fichero de conversión de archivos de voz	2/01/17	25/01/17
Notificaciones	16/01/17	22/02/17
Pruebas	26/12/16	24/03/17
Pruebas de las funcionalidades del bot	26/12/16	24/03/17
Pruebas de las funcionalidades del panel de ...	6/02/17	7/03/17
Prueba myqueridobot VS pruebasbot	17/03/17	17/03/17
Evaluación de la herramienta	8/03/17	8/03/17
Documentación	21/11/16	7/07/17
Documentación del código	22/12/16	24/03/17
Comentar el código	22/12/16	24/03/17
Redacción de los manuales	1/03/17	1/03/17
Memoria del proyecto	21/11/16	7/07/17
Búsqueda de material de apoyo	21/11/16	21/02/17
Creación de diagramas y tablas	14/04/17	5/06/17
Redacción de la memoria	17/03/17	9/06/17
Preparación de la defensa	25/05/17	7/07/17

Figura 34: Diagrama Gantt final
Link a la imagen:
`http://bots.ikasten.io/mikelbot/Imagenes/Gantt/GanttFin.png`

10.1.5. Evaluación económica

En cuanto a la evaluación económica, solo hay que tener en cuenta que el número de horas trabajadas ha sido mayor al previsto. Es por ello que hay que volver a realizar los cálculos del gasto en la mano de obra.

Como los cálculos de cuanto se paga por horas a un programador informático ya se tienen en la valoración económica (Sección 3.6, solo queda cambiar el número de horas por las que se han tardado realmente. Para ello, primero se van a restar las horas que realmente se han usado en el bloque de aprendizaje:

$$Horas\ totales - Horas\ aprendizaje = 624 - 122 = 502\ horas \quad (13)$$

Multiplicando el resultado por el sueldo/hora que se calculó en la evaluación económica, se obtiene el siguiente resultado:

$$Sueldo = Horas\ trabajo * Sueldo/hora = 502 * 8,98 = 4{,}507,96\,e \quad (14)$$

Con esto tenemos que el gasto real en mano de obra sería de 4.507,96 €. Con este nuevo resultado, en la Tabla 10 se muestran los costes finales del proyecto.

Tipo de gasto	***Gasto estimado (en €)***	***Gasto real (en €)***
Mano de obra	**3.726,7 €**	**4.507,96 €**
Gasto en *software*	**110 €**	**110 €**
Gasto en *hardware*	**153,36 €**	**153,36 €**
Gastos indirectos	**150 €**	**150 €**
Total	**4.140,06 €**	**4.921,32 €**

Tabla 10: Costes finales del proyecto

10.2. Lineas futuras

Este proyecto es muy extenso y todavía se puede extender aún más. Durante el desarrollo se han identificado muchas de las funcionalidades con las que ha terminado, pero por las características que tienen las herramientas, este proyecto podría continuar desarrollándose durante años.

Algunas de las lineas futuras que si se han identificado y que son indispensables para el proyecto son las siguientes:

- **Apartado de *tests***: En estos momentos, el panel de administración carece del apartado de *tests*. Estaría bien añadir ese apartado para poder administrar los *tests* que se pueden realizar con el *bot*. Añadir, eliminar y modificar las preguntas de los *tests* como se hace actualmente con las preguntas de voz.
- **Mejorar la selección automática de preguntas de voz**: En estos momentos la selección se puede hacer aleatoriamente o seleccionando a todos los estudiantes. Además de estos métodos, se podrían añadir más. Por ejemplo, escogiendo un número de estudiantes que han de responder dependiendo de la nota media que hayan obtenido en la última evaluación realizada.
- **Añadir más gráficos al panel de administración**: Además de los gráficos que muestra actualmente el panel de administración, se podrían añadir más gráficos o incluso llegar a permitir que el propio profesor pueda crear sus propios gráficos. En un principio, se planteo implementar esta idea, ya que en ese momento todavía se pensaba que se iba a seguir con lo que Pablo Uribe ya había implementado hasta el momento en su proyecto. Pero al final la idea ha tenido que quedar para proyectos futuros.
- **Añadir reconocimiento de voz al *bot***: Juanan comentó en su día que tenía planeado añadir reconocimiento de voz al *bot* para no tener que pulsar tantos botones. Esto podría ser una funcionalidad interesante para implementar en el futuro.
- **Mejorar la administración**: A lo que se quiere llegar con esto es que un solo *bot* pueda usarse en todas las clases de todos los cursos de una universidad. Para poder hacer esto, habría que mejorar la administración de los *tests* y grupos.

En estos momentos, aunque pueda haber más de un profesor con diferentes grupos y dando diferentes asignaturas, un estudiante no puede estar en más de un grupo. Además, puedes realizar todos los *tests*, sean de la asignatura que sean, si estás registrado en el *bot*.

- **Automatización de pruebas**: Como durante este proyecto no se ha podido, estaría muy bien que en el futuro se pudieran realizar las pruebas de manera automática. Esto ahorraría mucho tiempo y ayudaría a encontrar errores con mayor rapidez.

Aunque estas han sido algunas de las ideas que se han planteado durante el desarrollo del proyecto, este tiene infinitas posibilidades. Se le podrían añadir muchísimas funcionalidades y opciones, ya que la herramienta da juego y cada vez se desarrollan nuevas tecnologías y opciones aplicables.

Este es un proyecto que se puede seguir desarrollando y que espero que alguien lo siga haciendo los próximos años.

10.3. Licencias

En este apartado se verán las licencias que utilizan los recursos usados en el proyecto y la licencia que se ha aplicado a las herramientas que se han desarrollado en el proyecto.

10.3.1. Recursos utilizados

Además de los programas gratuitos que se han usado para desarrollar el proyecto, exceptuando la librería FFmpeg todas las demás librerías utilizan licencias MIT. Estas licencias permiten la modificación, distribución, uso personal y uso comercial de lo que protejan.

En cuanto a la librería FFmpeg, esta está compuesta por otras librerías con licencias MIT. En su totalidad FFmpeg está protegido por una licencia LGPL, que es la que se suele usar en librerías de código abierto. Esta permite su uso, modificación y distribución.

Por último, las preguntas que se han usado en este proyecto, ya sean las de voz como las de los *tests* son propiedad del director del proyecto, Juanan Pereira.

10.3.2. Herramientas desarrolladas en el proyecto

La licencia sobre los derechos de uso que se ha establecido para las herramientas creadas en este proyecto es GPL v3. Por lo que si alguien quiere modificarla, seguir con el proyecto y distribuir las herramientas no tendrá ningún problema.

10.4. Reflexión personal

Para concluir con mi memoria del TFG, quiero comentar algunos aspectos que me han hecho disfrutar de este proyecto.

Uno de los aspectos más "frustrantes" de este proyecto fue, que lo empece sin conocimiento alguno de lo que iba a hacer. Mi única experiencia de gran parte de lo que he realizado, era el haber implementado una página web en uno de los proyectos de la asignatura de seguridad que cursé en tercer curso del grado. Todo lo realizado en este proyecto, ha sido a base de un aprendizaje continuo del que me siento orgulloso. He aprendido a usar muchas herramientas, he investigado otras que, aunque no haya llegado a usar, ya tengo conocimientos de ellas y, sobre todo, aun habiendo empezado todo el proyecto desde cero, he conseguido llegar hasta este punto en menos de un año.

Una de las experiencias que me llevo, es la de haber creado algo que, durante su desarrollo, ha estado funcionando y ha sido utilizado por mis compañeros de clase. Aunque muchos de ellos no supieran que era yo el que lo estaba desarrollando, me han ayudado a detectar errores y mejorar la herramienta. También hay que decir que esto ha sido un poco estresante, ya que cuando había algún fallo tenía que intentar arreglarlo lo antes posible. De hecho, creo que me he llegado a anticipar de tal manera a algunos de los errores, que mis compañeros no se han dado cuenta de si había algún error.

Por todo ello, he de dar las gracias a Juanan, por dejarme tener esta experiencia y por ayudarme con los problemas que iban surgiendo durante el proyecto.

Bibliografía

[Ali et al., 2016] Ali, S. M., Gupta, N., Nayak, G. K., and Lenka, R. K. (2016). Big data visualization: Tools and challenges. In *Contemporary Computing and Informatics (IC3I), 2016 2nd International Conference on*, pages 656–660. IEEE.

[Dale, 2016] Dale, K. (2016). *Data Visualization with Python and JavaScript: Scrape, Clean, Explore & Transform Your Data*. .°'Reilly Media, Inc.".

[Espinosa, 2016] Espinosa, C. (2016). ¿importa el número de alumnos en clase? `http://vamoscreciendo.com/2014/11/29/importa-el-numero-de-alumnos-en-clase/`.

[Heller et al., 2005] Heller, B., Procter, M., Mah, D., Jewell, L., and Cheung, B. (2005). Freudbot: An investigation of chatbot technology in distance education. In *Proceedings of the World Conference on Multimedia, Hypermedia and Telecommunications*, pages 3913–3918.

[Pereira, 2016] Pereira, J. (2016). Leveraging chatbots to improve self-guided learning through conversational quizzes. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*, pages 911–918. ACM.

[Prakash, 2016] Prakash, T. G. (2016). Data visualization and communication by big data.

[Sánchez, 2017] Sánchez, J. M. (2017). La mitad del tráfico mundial lo generan «bots». `http://www.abc.es/tecnologia/redes/abci-mitad-trafico-mundial-generan-bots-201703222143_noticia.html`.

A. Anexo 1: Manual de instalación

En este anexo se explica cuales son los pasos a seguir para poder instalar el *bot* y el panel de administración. La instalación en sí es bastante simple, pero si hay algunos puntos que han de ser explicados para no dar margen a errores. Además de esto, se recomienda el uso de sistemas operativos como Ubuntu Server para la instalación. Por ser un sistema gratuito y que además tiene mucha documentación por si hay que consultar algún error.

A.1. Requisitos previos

Antes de empezar a meter en el servidor todo lo que se ha desarrollado durante este proyecto, es necesario instalar las herramientas que permiten que funcione todo.

Las siguientes herramientas, son indispensables para el correcto funcionamiento del *bot* y del panel de administración:

- **PHP5.6**: Es la versión de PHP en la que se han desarrollado el *bot* y el panel de administración.
- **MySQL5.5**: Se ha usado esta versión porque de ella en adelante MySQL soporta la codificación utf8_mb4, que es necesaria para guardar *emojis* en la base de datos.
- **Apache2**: Como ya se ha comentado, Apache2 es el pilar de las comunicaciones, tanto para el *bot* como para el panel de administración web.
- **Certificado SSL**: Es necesario para poder utilizar el *bot* mediante el *webhook*.
- **CURL**: Si no se instala esta herramienta no se podrán enviar notificaciones a los estudiantes.
- **FFmpeg**: Sin este programa no se podrían hacer las conversiones de los ficheros .oga. Esto, provocaría que las grabaciones que se dan como respuesta a las preguntas de voz y las explicaciones de voz, no se puedan escuchar desde el panel de administración ni desde dispositivos con una versión de Android inferior a la 6.0.

- **Git**: Se usará para clonar el repositorio en el que se encuentra el código del proyecto. Para ello tienes que configurarlo de manera que tenga tu usuario de Bitbucket.

- **Telegram**: Para poder crear un *bot*, es necesario realizar algunos pasos en la aplicación de Telegram. Además de eso, habrá que tener la aplicación para usar el *bot* que se cree.

A.2. Preparar la base de datos

Después de tener todos los programas instalados y funcionando, el primer paso es introducir la estructura de la base de datos en MySQL. Para ello, hay que clonar el repositorio de Bitbucket con Git en la carpeta en la que Apache2 tenga acceso completo (por defecto en /var/www en Ubuntu). Situándonos en esa carpeta por consola, ejecutamos el siguiente comando para clonar el repositorio:

```
git clone https://malboniga002@bitbucket.org/malboniga002/myqueridobot.git
```

De esta manera ya tendremos todo el código en la carpeta llamada "myqueridobot" que se ha creado. Ahora lo que hay que hacer es entrar en MySQL con el usuario y contraseña que tengamos. Con ese usuario, creamos una base de datos nueva con el nombre que queramos. Todo esto se hace con los siguientes comandos:

```
# Para entrar en MySQL usamos el siguiente comando e introducimos el password
mysql -u nombre_usuario -p
# Usar el siguiente comando dentro de MySQL para crear la base de datos
create database nombre_base_datos;
# Si todo ha ido bien, salimos de MySQL
exit
```

Ahora, estando en la carpeta myqueridobot, utilizamos el siguiente comando para introducir la estructura de la base de datos:

```
mysql -u nombre_usuario -p nombre_base_datos<structure.sql
```

Con esto ya tendríamos la base de datos lista para la acción.

A.3. Preparar el *bot*

El primer paso es hablar con @BotFather, ya que este ha de darnos un *API_Key* (también llamado *token*) que utilizaremos para identificar al *bot*. Para ello, busca a @BotFather en Telegram y comienza una conversación con él. Introduce el comando /newbot para que te empece a pedir los datos del *bot*, como el nombre que le quieres poner.

Al terminar de introducir los datos, @BotFather te dará un *API_Key* como este: "110201543:AAHdqTcvCH1vGWJxfSeofSAs0K5PALDsaw". Este *API_Key* lo tendremos en cuenta más tarde, pero ahora toca copiar dos ficheros a la carpeta myqueridobot. Los ficheros se encuentran en la carpeta vendor/longman/telegram-bot/examples y se llaman "set.php" y "hook.php". Se copian de la siguiente manera estando situados en la carpeta myqueridobot:

```
cp vendor/longman/telegram-bot/examples/hook.php vendor/longman/telegram-bot/
    examples/hook.php .
```

Ahora que los hemos copiado, abrimos "hook.php" con un editor de texto (yo uso nano). En este archivo hay que modificar varias lineas (no quites las comillas simples, solo cambia el texto que hay entre ellas por lo que se indique):

- Linea 15: En el lugar que pone "*your_bot_api_key*", hay que poner el *API_Key* que nos facilitó @BotFather.
- Linea 16: Donde pone "*username_bot*" hay que poner el nombre de usuario que se le ha puesto al *bot* entre comillas. Este es uno de los datos que le has dado a @BotFather a la hora de crearlo.
- Lineas 22-27: En todas las lineas que hay entre ellas (22 y 27 incluidas), hay que quitar los dos caracteres que tienen al comienzo (caracteres "//"). Además de eso, en la linea 24, cambia "dbuser" por el nombre de usuario de MySQL que has usado antes para crear la base de datos y en la linea 25, cambia "dbpass" por la contraseña que has usado para entrar en MySQL. Por último, en la linea 26 cambia "dbname" por el nombre que le has puesto a la base de datos.
- Lineas 40 y 43: Quita los caracteres "//" al frente de esas lineas.
- Linea 60: Quita los caracteres "//" al frente de esas lineas. Además, entre paréntesis pone "('../Download')", cambia lo por "('Download')".

Después de realizar estos cambios modificaremos "set.php". En ese fichero, como podrás comprobar, las lineas 5 y 6 son iguales que las lineas 15 y 16 de "hook.php". Puedes copiarlas y reemplazarlas en este fichero de lo que se acaba de modificar. En cuanto a la linea 17, entre las comillas tienes que indicar donde se encuentra el fichero "hook.php". Para ello tienes que indicar el nombre de tu dominio (el cual has tenido que obtener cuando has conseguido el servidor). y donde pone "/path/hook.php", en caso de no haberlo movido, sería "/myqueridobot/hook.php".

Ahora que hemos hecho esto hay que crear las carpetas en las que se guardarán las respuestas de voz de los estudiantes. Para ello, PHP tiene que poder escribir en todas las carpetas o sino podrá dar algún error. Para crear las carpetas hay que escribir los siguientes comandos desde la carpeta myqueridobot:

```
mkdir Download
mkdir Download/voice
```

Ya creadas todas las carpetas, ahora toca modificar otro fichero. Para ello antes hay que cambiarle el nombre de la siguiente manera:

```
mv Globals.php.example Globals.php
```

Ahora hay que modificar el fichero renombrado. Modifica los asteriscos de las lineas con la siguiente información:

- Linea 2: El *API_Key* que te proporcionó @BotFather.
- Linea 3: El nombre de usuario de tu *bot*.
- Linea 4: Los primeros asteriscos por el nombre de tu base de datos. Los siguientes por tu usuario de MySQL y los últimos por tu contraseña de MySQL.
- Linea 5: Para rellenar esta linea tienes que escribir el siguiente comando en consola:

```
# Cambia los asteriscos por lo que te devuelva este comando:
which ffmpeg
```

- Linea 6: Para rellenar esta linea tienes que escribir el siguiente comando en consola:

```
# Cambia los asteriscos por lo que te devuelva este comando:
which ffprobe
```

Llegados a este punto ejecuta lo siguiente en consola estando en la carpeta myqueridobot:

```
php set.php
```

Habiendo hecho esto, ya puedes empezar a conversar con tu *bot*. Para ello, busca en Telegram su nombre y empieza la conversación con él pulsando el botón que hay en la parte baja del *chat*. Al hacer esto, el *bot* debería contestar "*Hi*". Si no lo hace es porque ha habido algún error en lo que se ha realizado anteriormente.

Después de que te conteste, prueba a ejecutar el comando /whoami. Este comando te devolverá tu identificador de Telegram, el cual, tienes que usar para registrarte como profesor en la herramienta. Para ello, ejecuta lo siguiente en la consola del servidor:

```
# Primero entra en MySQL
mysql -u nombre_usuario -p
# Usar el siguiente comando dentro de MySQL para usar tu base de datos
use nombre_base_datos
# Ahora vamos a registrarte
# Cambia los asteriscos por los datos que vienen a continuacion:
# 1. Tu identificador de Telegram
# 2. Un identificador del que te acuerdes para que tus alumnos
# introduzcan al registrarse ellos. Sera el identificador de tus
# grupos de alumnos
insert into professor(id_prof,id_group_selection) values(******,'*****');
```

Con esto ya has hecho todo lo referente a preparar el *bot*. Ahora solo falta el panel de administración.

A.4. Preparar panel de administración web

Para preparar el panel de administración, solo tienes que ejecutar las siguientes lineas en consola desde la carpeta myqueridobot:

```
# Mediante estas lineas se cambian los permisos de las carpetas
chmod 755 -R myweb
chmod 777 -r myweb/uploads
```

Ahora, para comprobar que todo ha funcionado y que te has registrado correctamente como profesor, entra en tu navegador favorito (recomiendo Google Chrome). En la barra de navegación pon tu dominio y /myqueridobot/myweb. Por ejemplo: "http://bots.ikasten.io/myqueridobot/myweb". De esta manera llegarás al *login* del panel de administración.

Para comprobar que te has registrado correctamente, tienes que poner lo siguiente:

- ***User***: En este campo, tienes que escribir tu nombre, lo que pone en *Name* al ejecutar el comando /whoami desde el *bot*. Pero hay que tener en cuenta una cosa, a menos que se haya puesto un nombre compuesto y dos apellidos, solo hay que poner el primer nombre. Es decir, si en ese campo pone "José", "José Pérez" o "José Pérez Díez" solo tienes que introducir "José". Pero en caso de que se haya introducido algo como "Juan José Pérez Díez" habría que introducir "Juan José" en el campo "*User*".
- ***Password***: En este campo tienes que introducir tu identificador de Telegram. El número que pone en "*Your id*" al ejecutar el comando /whoami con el *bot*.

Con esto deberías haber conseguido entrar en el panel de administración de tu *bot*, a menos que hayas escrito algo mal.

Para terminar, estando ya dentro del panel de administración, te recomiendo que te dirijas al apartado de opciones y las modifiques a tu gusto.

B. Manual de usuario

Como manual de usuario, se ha creado un vídeo en el que se muestra el uso de las herramientas desarrolladas durante este proyecto. Este vídeo está subido a YouTube y es público. El *link* es el siguiente:

`https://www.youtube.com/watch?v=EB1rjj-0cbQ`

C. Pantallazos

Este apartado se compone de una secuencia de las imágenes que muestran las interfaces del panel de administración y los procesos que se siguen al ejecutar los comandos del *bot*. Junto con esas imágenes se darán pequeñas explicaciones de como llegar a esas pantallas en el caso del panel de administración.

C.1. *Bot*

Ya que el *bot* está formado por varios comandos, a continuación se muestran las imágenes en las que se observa la secuencia de mensajes que se envía en cada comando.

Comando /help

Cuando un estudiante envía "/help" al *bot*, recibirá el mensaje que se muestra en la Figura 35.

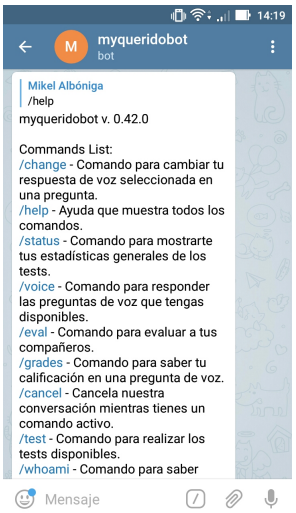


Figura 35: Comando /help en Telegram

Comando /test

Al enviar el mensaje "/test", el alumno recibe un listado de botones como el que aparece en la Figura 36. Cada botón representa uno de los *tests* que se pueden realizar.

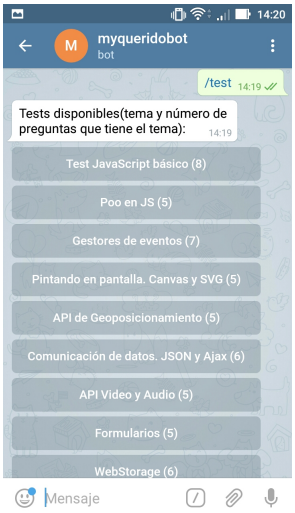


Figura 36: Comando /test en Telegram (1)

Al pulsar uno de los botones, se envía una pregunta como la que aparece en la Figura 37.

Al contestar a la pregunta, el *bot* envía un mensaje indicando si la respuesta ha sido correcta o incorrecta. Además de eso, hace que aparezca un panel de opciones como el de la Figura 38. El *test* continua de una manera similar hasta que el usuario pulsa la opción "Salir" o ejecuta el comando "/cancel".

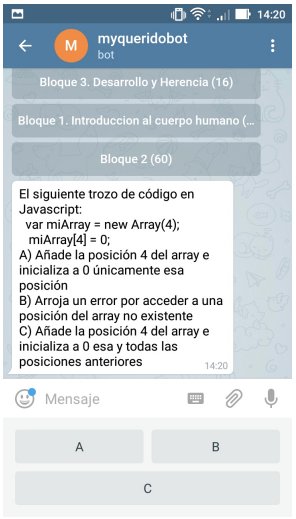


Figura 37: Comando /test en Telegram (2)

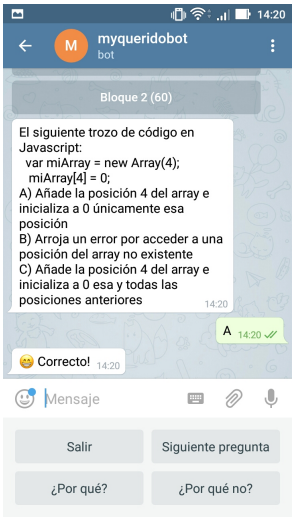


Figura 38: Comando /test en Telegram (3)

Comando /status

Cuando un estudiante manda "/status" en un mensaje al *bot*, este le responde con el último mensaje que aparece en la Figura 39.

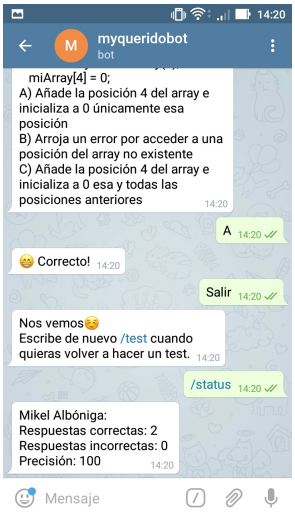


Figura 39: Comando /status en Telegram

Comando /voice

Al enviar "/voice" al *bot*, este envía un listado de mensajes con las preguntas disponibles para responder. Después, envía un listado de botones relacionados con las preguntas por un identificador. En la Figura 40 se muestra el listado de las preguntas (en este caso solo una) y en la Figura 41 se muestra el listado de botones.

Además, en la Figura 41, se muestra lo que sucede cuando se pulsa uno de los botones. Se envía la pregunta seleccionada y se pide al usuario que conteste mediante una grabación de voz. Cuando este envía la grabación, se muestran algunos mensajes que se pueden ver al final de la figura mencionada.

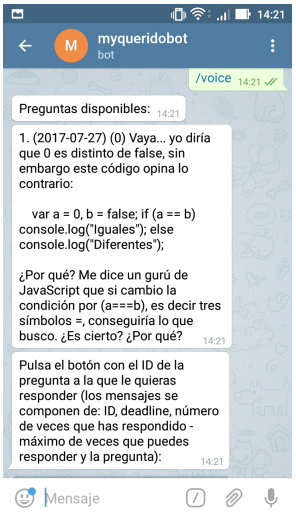


Figura 40: Comando /voice en Telegram (1)

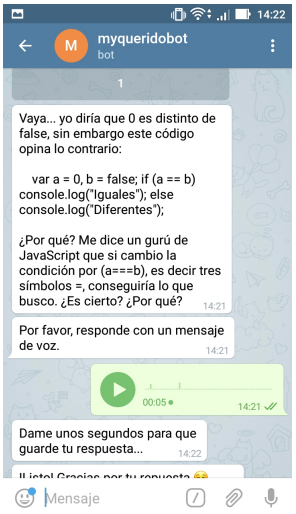


Figura 41: Comando /voice en Telegram (2)

Comando /change

Cuando un estudiante quiere cambiar su respuesta en alguna de las preguntas que ha realizado y en la que ha dado más de una respuesta, solo tiene que mandar el mensaje "/change" al *bot*. Al hacer esto se muestra el listado de las preguntas en las que se puede cambiar la respuesta. Esto se muestra en la Figura 42

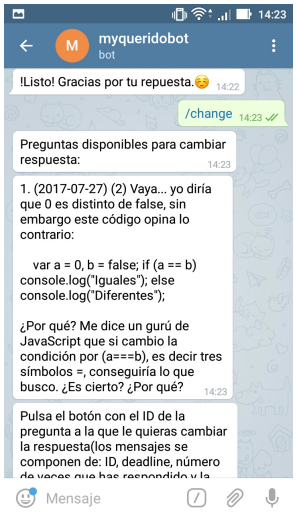


Figura 42: Comando /change en Telegram (1)

Al pulsar uno de los botones, se le envía al estudiante la pregunta y las grabaciones que ha enviado para responderla. Para seleccionar la respuesta, se le pide al usuario que escriba el identificador que se envía junto a la grabación dando igual si la respuesta ya estaba seleccionada. Cuando el usuario envía un mensaje con el identificador, la respuesta seleccionada se cambia.

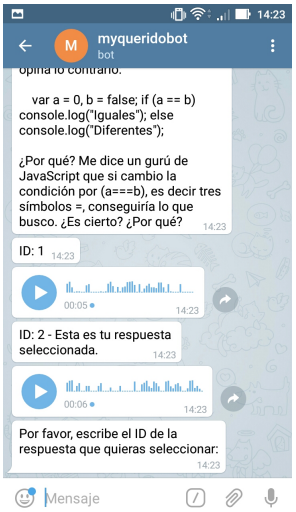


Figura 43: Comando /change en Telegram (2)

Comando /eval

Al recibir el mensaje "/eval" el *bot* envía un listado de las peguntas que se pueden evaluar junto con unos botones que las relacionan. Al pulsar uno de los botones del listado, se le envía al estudiante que ejecutó el comando la pregunta y la respuesta que otro estudiante envió como respuesta.

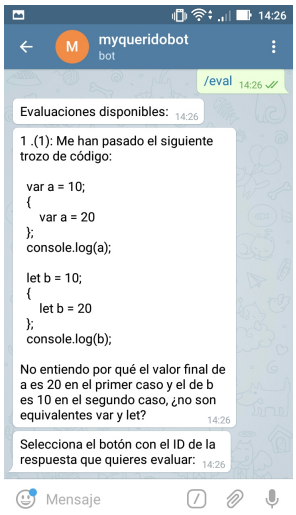


Figura 44: Comando /eval en Telegram (1)

Al usuario se le pide que evalúe en una escala determinada. Después de evaluar la respuesta según todos los criterios que tenía asociados la pregunta, se le pregunta al usuario si quiere dar una explicación. Al pulsar "Sí", se le pide al usuario que envíe una grabación para terminar.

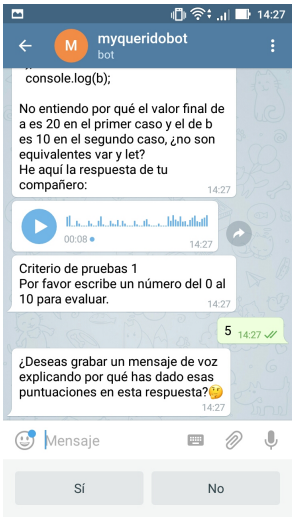


Figura 45: Comando /eval en Telegram (2)

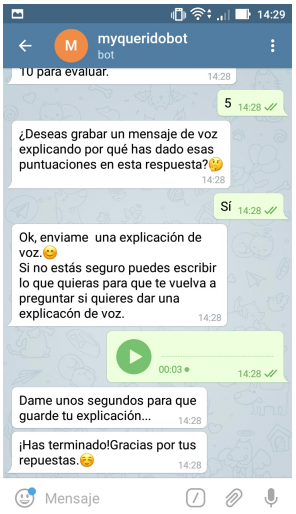


Figura 46: Comando /eval en Telegram (3)

Comando /grades

Cuando un estudiante envía "/grades" en un mensaje al *bot*, este le envía al estudiante el listado de preguntas en las que su respuesta ya ha sido evaluada. Al escribir el identificador asociado a la pregunta, el estudiante recibe la grabación de su respuesta, el listado de criterios evaluados junto con la lista de las calificaciones. Además, también se le envía el listado de explicaciones de voz en caso de que algún evaluador haya dado alguna. Toda esta secuencia se muestra en las Figuras 47 y 48.

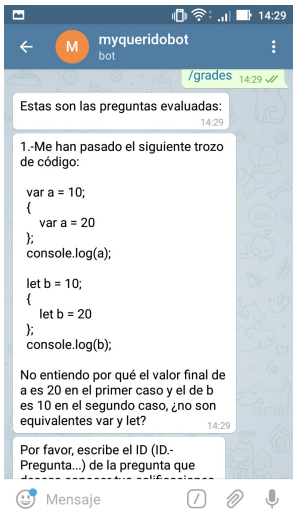


Figura 47: Comando /grades en Telegram (1)

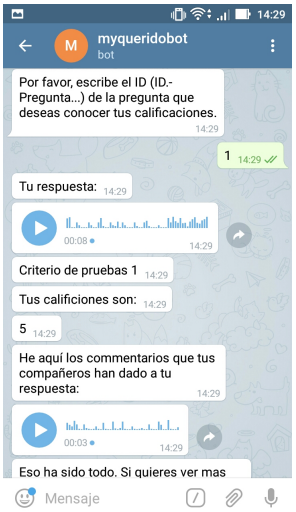


Figura 48: Comando /grades en Telegram (2)

C.2. Panel de administración

A continuación se muestran los pantallazos realizados al panel de administración. En algunos de los casos se ha tenido que dividir la imagen o simplemente se muestran algunas de las lineas de las tablas pero no todas.

Login

Esta página simplemente contiene los dos campos necesarios para acceder al panel de administración. En el primer campo se ha de poner lo que se considera el "*first_name*" de Telegram y la contraseña es el identificador único que Telegram proporciona a los usuarios crear una cuenta en su aplicación.

Figura 49: *Login*

Questions

Como ya se ha comentado en este capítulo, para mostrar las preguntas se ha creado una tabla que se muestra en la Figura 50.

Al pulsar el botón "*Create question*", se envía al administrador a la página para la creación de preguntas. Si se hace *click* en alguna de las filas, mientras no sea en el campo "*Select*", se envía al profesor a la interfaz de modificación de preguntas. Como las funcionalidades de modificación y creación usan las mismas interfaces en las imágenes de esta sección solo se mostrarán las de modificación y se explicarán las diferencias.

La página de modificación se muestra en las Figuras 51 y 52. En ellas se muestran los campos a rellenar durante la creación de preguntas de voz. Se ha de introducir el texto de la pregunta, el número máximo de respuestas que se permiten en esa pregunta,la fecha del *deadline* y se puede subir una imagen que irá asociada a la pregunta (esto es opcional).

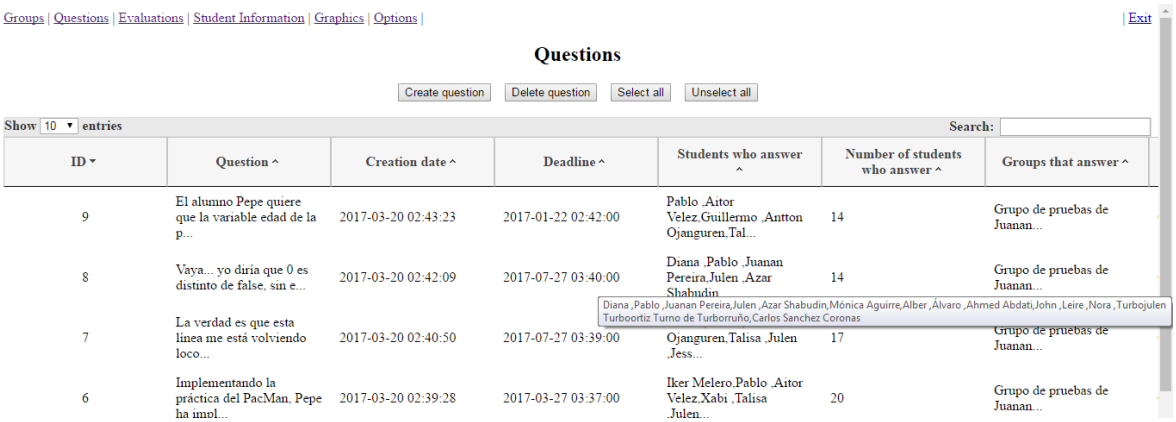


Figura 50: Apartado *Questions*
Link a la imagen:
`http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/Questions.png`.

Además, se ha de seleccionar los criterios que se van a evaluar. Si no se tiene ningún criterio se puede crear directamente desde esa interfaz. También permite eliminar los criterios escribiendo su identificador.

Para terminar la introducción de datos, se ha de escoger los grupos que formarán parte en la evaluación y resolución de la pregunta que se está creando o modificando. Por si no se tiene claro quienes pertenecen a un grupo, se puede hacer *click* en el campo "*Check group members*" para que se muestre un listado de los estudiantes que pertenecen al grupo en una tabla.

Después de seleccionar los grupos, los criterios y rellenar los campos de la pregunta, al pulsar el botón "*Next step*" se lleva al administrador a la página en la que se seleccionan los estudiantes para responder y evaluar. En las Figuras 54 y 55 se muestran las interfaces que intervienen en la selección de los estudiantes que responden y evalúan. Como ya se ha comentado, la selección inicial en la creación de preguntas no llega a ser del todo aleatoria. En caso de que al administrador no le guste la selección aleatoria inicial, este puede realizar más selecciones pulsando los botones de la interfaz o simplemente modificarla de manera manual. Para terminar solo tiene que pulsar el botón "*Save changes*" y, si no ocurre ningún error en las conexiones, se creará o modificará la pregunta.

Las diferencias entre la creación y la modificación, aparte de los títulos de las páginas y en el caso de la modificación la carga de todos los datos, en la modificación, al ir a la página de selección de los estudiantes a los que se les asigna la pregunta, aparece una advertencia a tener en cuenta a la hora

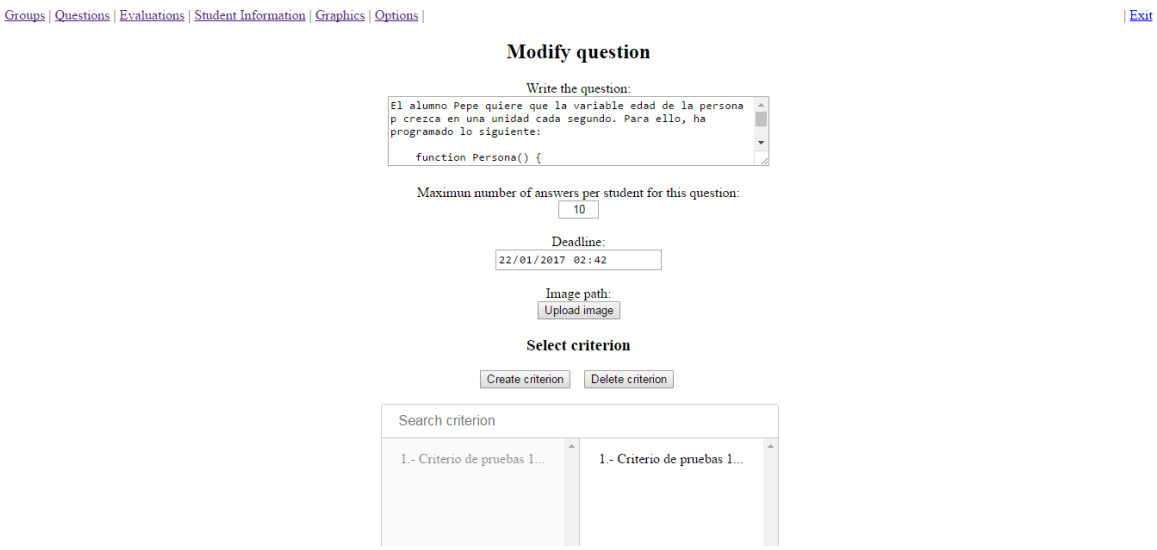


Figura 51: Modificar pregunta de voz (1)
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/ModifQuestion.png`.

de guardar una modificación. Cuando se realiza una modificación se pueden llegar a perder datos si no se tiene cuidado. Esto pasa cuando, por ejemplo, se le quita la asignación de la pregunta a alguien que ya la ha contestado. Esa respuesta se perdería en caso de guardar esa modificación.

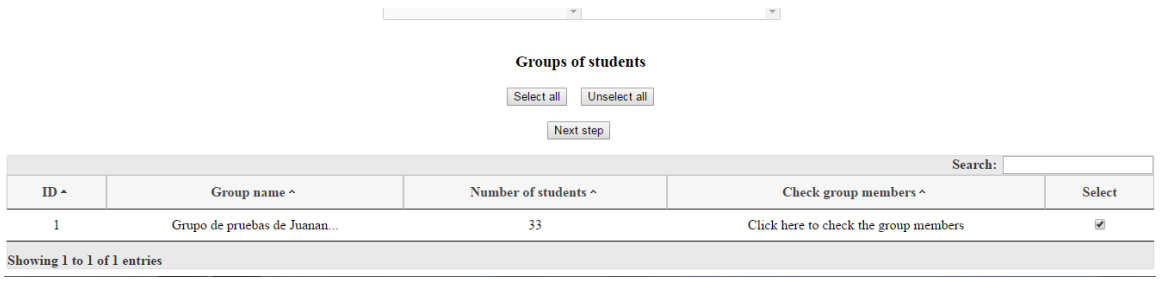


Figura 52: Modificar pregunta de voz (2)
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/ModifQuestionbottom.png`.

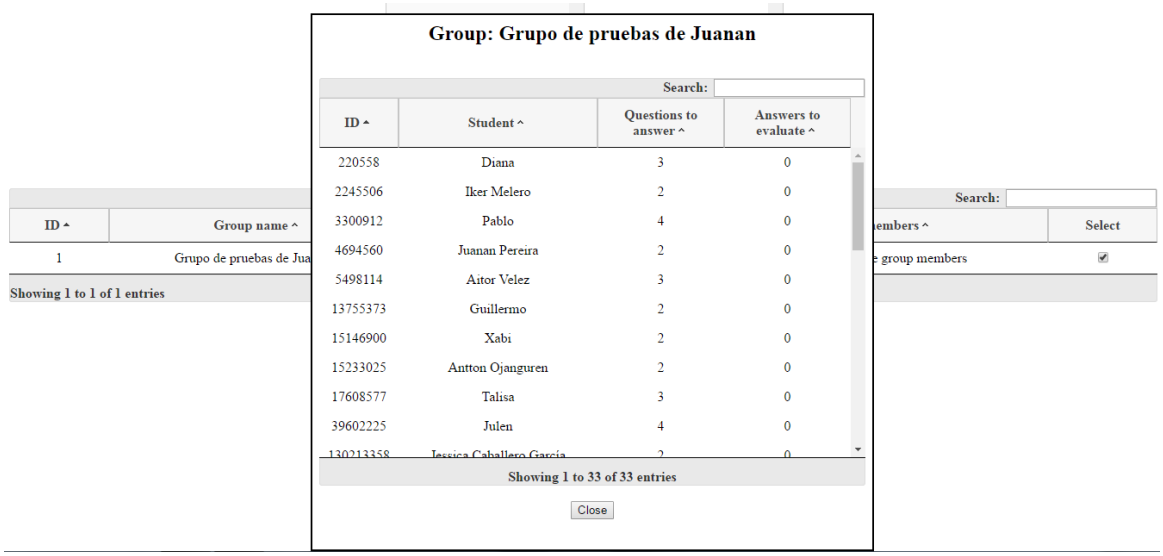


Figura 53: Modificar pregunta de voz (3)
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/ModifQuestionGroups.png`.

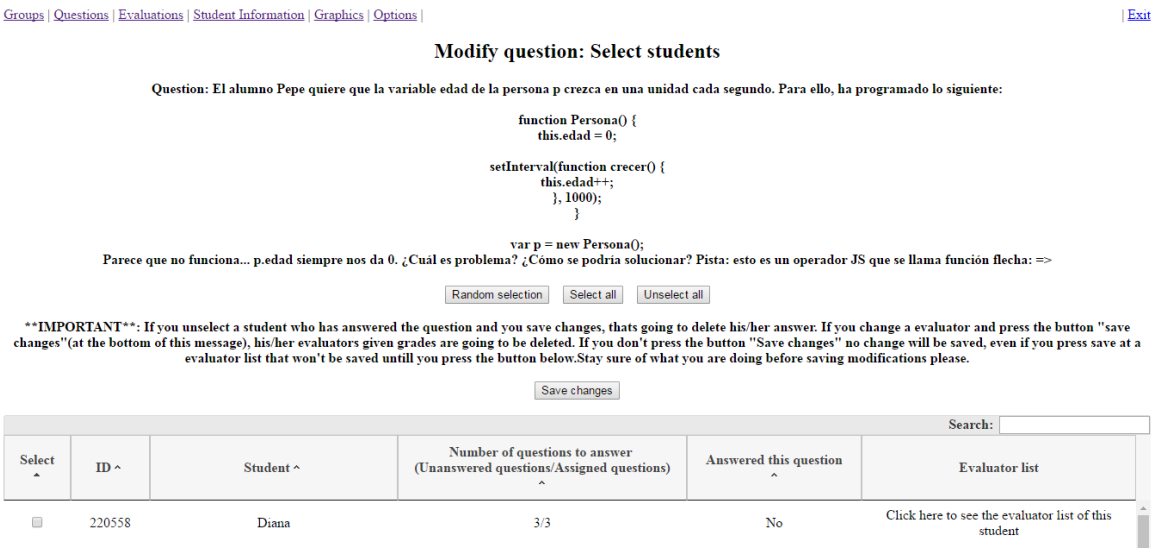


Figura 54: Asignación de pregunta de voz a estudiantes
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/QuestionStudents.png`.

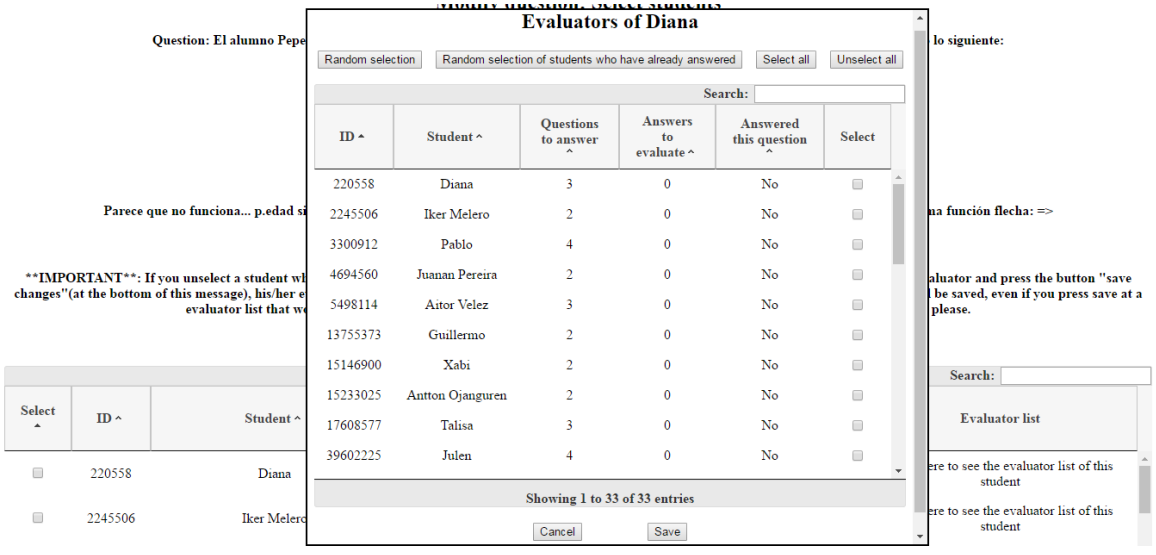


Figura 55: Asignar evaluadores a estudiante
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/QuestionStudentsEvaluator.png`.

Groups

La página en la que se muestran todos los grupos creados por el administrador se muestran en la Figura 56. Desde esa interfaz se pueden crear nuevos grupos o eliminar algunos de los existentes. En caso de pulsar sobre alguna de las filas de la tabla, se envía al administrador a la página en la que se administran los alumnos del grupo de la fila sobre la que se ha hecho *click*.

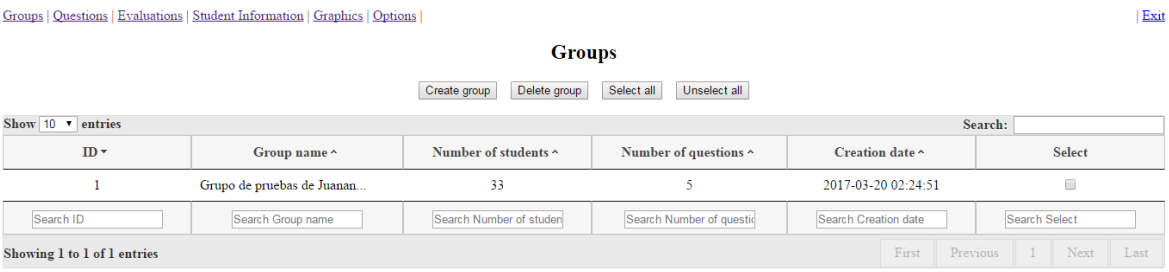


Figura 56: Apartado *Groups*
Link a la imagen:
`http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/Groups.png`.

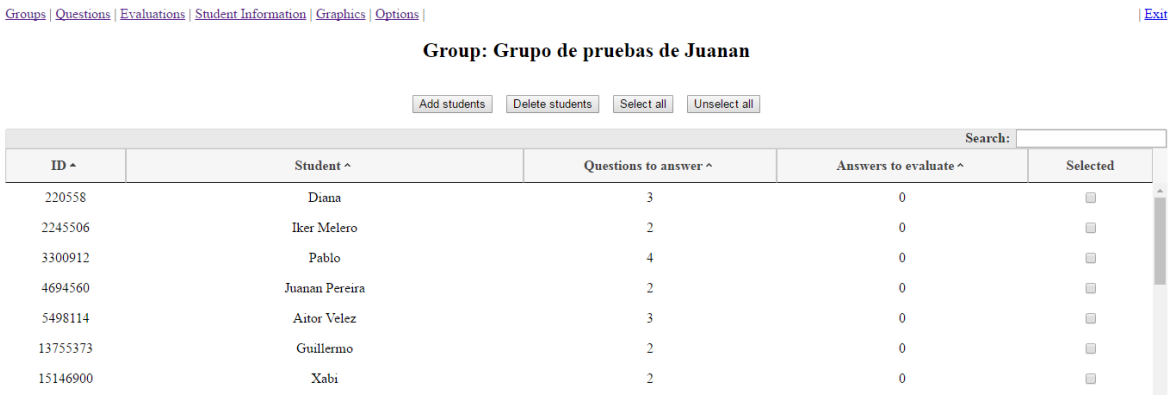


Figura 57: Administración de un grupo
Link a la imagen:
`http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/Group.png`.

Al pulsar el botón "*Add students*" se abre un *iframe* con un listado de los usuarios que han mandado algún mensaje al *bot*, pero que no están registrados en ningún grupo. Desde ese *iframe* se pueden seleccionar los estudiantes que se quieren añadir al grupo que se indica en el título del *iframe*. Cuando se presiona el botón de guardar en el *iframe*, los alumnos seleccionados se añaden al grupo.

Desde la interfaz que aparece en la Figura 57 también se permite eliminar los estudiantes de un grupo.

Evaluations

En el apartado de evaluaciones se presenta la información que muestra la Figura 58. En la tabla aparece casi toda la información de lo que se ha evaluado, pero para ver los detalles de una evaluación, solo hay que hacer *click* en la fila de la tabla en la que se encuentra la evaluación. De esta manera, se envía al administrador a la página que aparece en la Figura 59. En ella se pueden ver las calificaciones y las explicaciones que ha recibido la respuesta de la fila seleccionada.

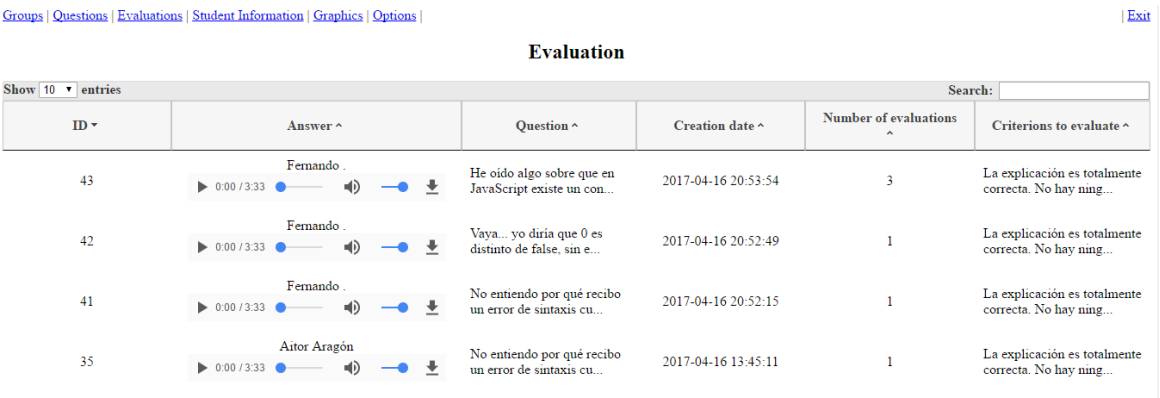


Figura 58: Apartado *Evaluations*
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/Evaluations.png`.

Figura 59: Evaluación de una respuesta
Link a la imagen:
`http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/Evaluation.png`.

Students information

En las Figuras 60 y 61 se muestran las tablas que contienen la información de los estudiantes que puede ser interesante para el administrador. La tabla que muestra los estudiantes evaluados por un estudiante es igual que la de los evaluadores de un estudiante, lo único que cambia es el título del *iframe* en el que aparece la tabla.

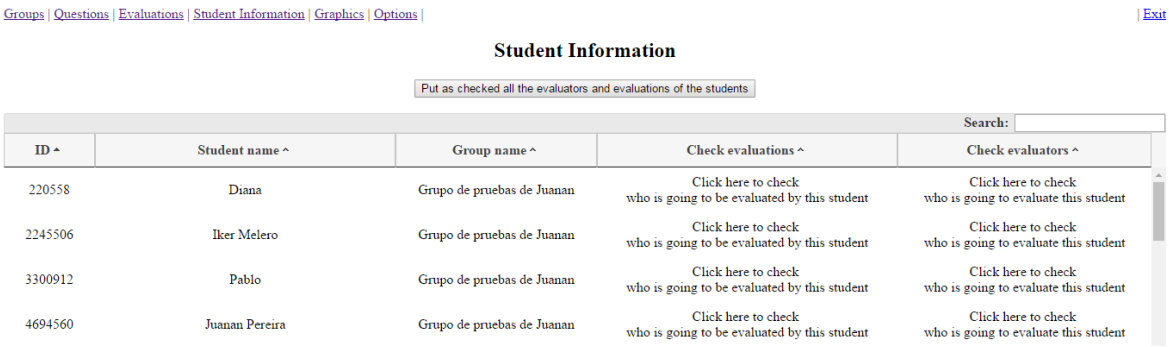


Figura 60: Apartado *Student Information*
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/InfoStudent.png`.

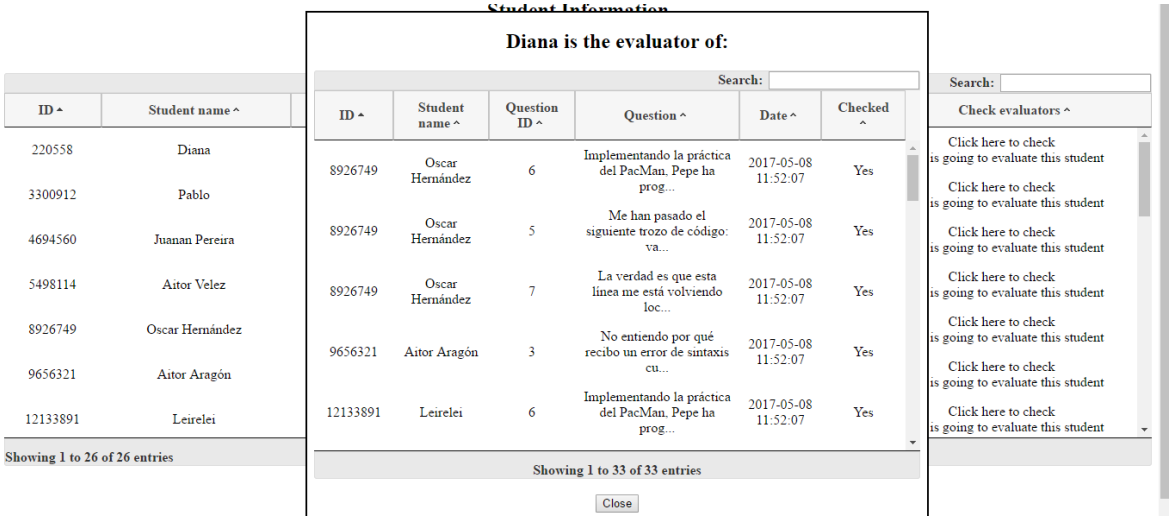


Figura 61: Información de evaluadores de un estudiante
Link a la imagen: `http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/EvaluationEvaluator.png`.

Graphics

Actualmente, el panel de control solo muestra cuatro gráficos del uso del *bot* y del progreso de los estudiantes. Estos, se abren en una nueva pestaña al pulsar uno de los botones de la interfaz que se muestra en la Figura 62.

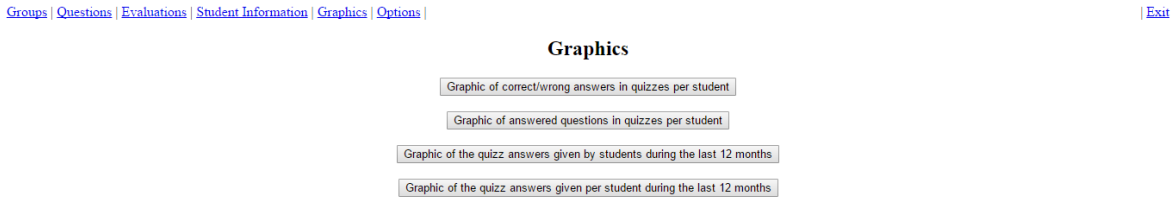


Figura 62: Apartado *Graphics*
Link a la imagen:
`http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/Graphics.png`.

La Figura 63 muestra el ejemplo de uno de los gráficos que permite ver la funcionalidad de gráficos del panel de administración.

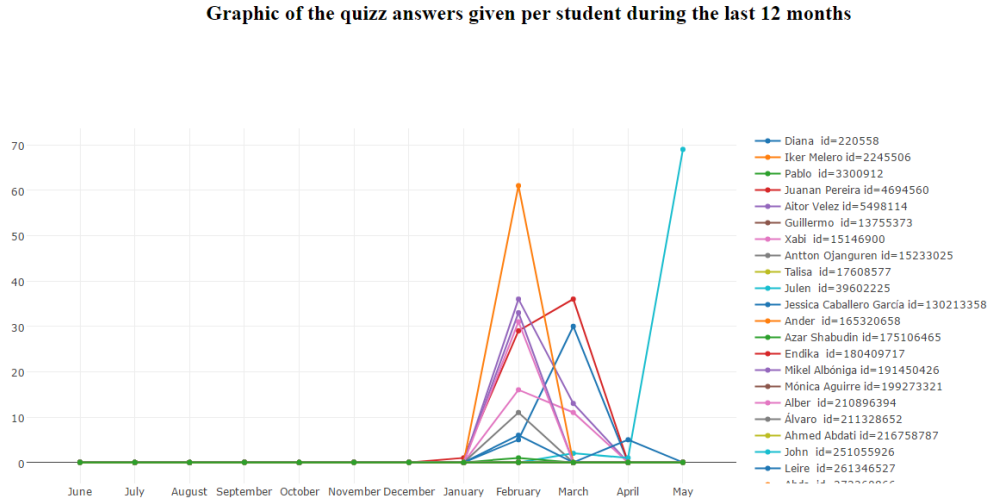


Figura 63: Gráfico de muestra
Link a la imagen:
`http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/Graphic.png`.

Options

En la interfáz del apartado opciones aparecen las opciones a cambiar junto con un botón para guardar los cambios realizados. La Figura 64 muestra las opciones modificables.

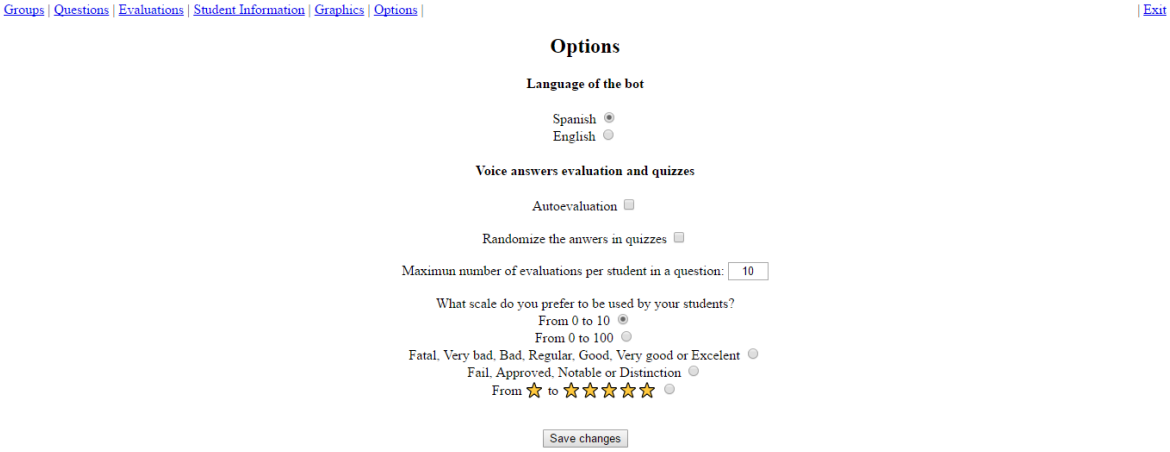


Figura 64: Apartado *Options*
Link a la imagen:
`http://bots.ikasten.io/mikelbot/Imagenes/Pantallazos/Options.png`.