Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

(Mención en Ingeniería de Software)

Agente de integración de un LMS y un servicio de mensajería orientado al alumno

Autor:

D. Iván Carreño Calzada

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

(Mención en Ingeniería de Software)

Agente de integración de un LMS y un servicio de mensajería orientado al alumno

Autor:

D. Iván Carreño Calzada

Tutor:

D. César Llamas Bello

Resumen

El propósito de este proyecto es dar la posibilidad al alumnado y al profesorado de utilizar conjuntamente el sistema de aprendizaje Moodle y la plataforma de mensajería Telegram. Para ello se desarrollarán un conjunto de herramientas que permitan sincronizar ambos servicios.

Por una parte, se diseñará *List Management* Bot, un *bot* para Telegram bajo el nombre de @ListManBot que permitirá a sus usuarios acceder a un sistema de ficheros. Se busca ofrecer una interfaz sencilla y cómoda que permita navegar por los árboles de directorios rápida e intuitivamente. Para cumplir este cometido se utilizará la API de programación de bots de Telegram para Python *python-telegram-bot* y un servidor MySQL.

Por la otra, se diseñará un *plug-in* con PHP para Moodle que se conectará con el mismo sistema de almacenamiento con el fin de permitir al cuerpo docente desarrollar y mantener una estructura similar a la de sus cursos sin tener que invertir grandes cantidades de tiempo en ello.

Palabras clave: Python, Telegram, bot, MySQL, Moodle, PHP, sistema de gestión del aprendizaje, integración, alumno, profesor

Abstract

The purpose of this Project is to give students and teachers the possibility to use the learning system Moodle and the messaging platform Telegram. To this end, a set of tools Will be developed to synchronize both services.

On the one hand, *List Management Bot*, a bot for Telegram following the name @ListManBot which Will allow its users to Access a file system, will be designed. It seeks to offer a simple and convenient interface that allows users to quickly and intuitively navigate through directory trees. To accomplish this task, the programming API of Telegram bots for Python *python-telegram-bot* and a MySQL server will be used.

On the other hand, a *plug-in* for Moodle will be designed using PHP. It will connect to the same storage system to allow the faculty to develop and maintain a structure similar to their courses without having to invest large amounts of time on it.

Keywords: Python, Telegram, bot, MySQL, Moodle, PHP, LMS, Learning Manage System, integration, student, professor

Índice de Contenido

Índice de Ilustraciones

Índice de Tablas

Capítulo 1. Introducción, objetivos y memoria

1.1 Introducción

A medida que pasan los años vamos comprobando cómo la digitalización va tocando cada vez más ámbitos de la vida diaria de las personas. Este cambio afecta también a los métodos de estudio y a los métodos de enseñanza, permitiendo incluso que la educación a distancia alcance en prestaciones a la presencial.

La proporción de centros y cuerpo docente que emplea plataformas digitales y plataformas online para facilitar sus tareas y las de sus alumnos se encuentra en constante crecimiento. A estas plataformas se les conoce con el nombre de herramientas de gestión de aprendizaje (LMS, por sus siglas en inglés), y una de las más representativas es *Moodle*.

Por otro lado, las aplicaciones de mensajería instantánea se encuentran en cualquier dispositivo y prácticamente todos los usuarios de las plataformas ya mencionadas emplean también aplicaciones de mensajería. Telegram pone a disposición del público una interfaz de programación de aplicaciones, usualmente llamadas “bots” que permiten desarrollar nuevas funcionalidades para la plataforma y ponerlas a disposición de los usuarios.

1.2 Motivación

El uso que se hace de Moodle por parte del cuerpo docente y por parte del cuerpo estudiantil no es siempre el óptimo, y acceder a los recursos alojados en la plataforma puede llegar a suponer un reto en algunos casos. A esto hay que añadirle la imposibilidad por parte del alumnado de añadir elementos o descripciones que le faciliten las tareas de organización y estudio.

Encontrar un método para permitir a los estudiantes accedes a los contenidos ofrecidos por el profesor de una manera cómoda y sencilla, dar la opción de aumentar estos contenidos y poder compartir estos con los compañeros sería una manera de ofrecer a los alumnos una herramienta que les facilite parte de la vida estudiantil.

La idea de un sistema similar a este no surge para este TFG, sino que el concepto se había formado ya el primer año en el Grado. La idea inicial fue refinada con la ayuda del profesor César Llamas Bello y este TFG será tutorizado por él.

1.3 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es el desarrollo de un bot de Telegram que permita organizar y acceder a apuntes y recursos presentes en Moodle de una manera cómoda para los profesores y estudiantes, así como permitir complementar estos elementos con material externo a la plataforma de LMS.

Para ello se implementará un sistema de directorios y ficheros basado en listas, las cuales tendrán restricciones de acceso y edición para los usuarios, permitiendo así recrear la estructura propia de Moodle y, con la que estamos más familiarizados, estructura de ficheros de los Sistemas Operativos más comunes y utilizados.

También se desarrollará de un plug-in de navegador que permitirá, al menos, añadir elementos a una lista concreta desde el Moodle propio de cada asignatura. Este elemento no es imprescindible, pero facilitaría en gran medida la labor del profesor de reorganizar y compartir los apuntes para con sus alumnos, lo cual podría llevar a un mayor uso y una mejor calidad del material necesario para el estudio.

La implementación de un sistema de notificaciones, que avise a los usuarios con permiso de acceso a una lista, sería también uno de los puntos principales de este proyecto, pues permitiría a los estudiantes estar al tanto de las ampliaciones o cambios realizados en el material de cada una de sus asignaturas.

A parte de esta utilidad, el bot también podría emplearse en aquellas tareas que puedan resolverse con un sistema de ficheros integrado en Telegram, aunque existan opciones más completas o enfocadas a este punto.

Por todo esto, se consideran los siguientes puntos para llevar a cabo este proyecto:

- Planificación de un proyecto software mediante Proceso Unificado.
- Desarrollo de esquemas, diagramas, modelos y tablas para el diseño de la aplicación.
- Implementación de los diseños empleando Python y JavaScript, junto con librerías y APIs que se mencionarán a lo largo de la memoria.
- Pruebas de la aplicación para comprobar su correcto funcionamiento.

1.4 Organización de la memoria

Esta memoria se divide en cinco secciones principales, cada una de ellas tratando una parte distinta del Trabajo de Fin de Grado, pero guardando una relación de orden cronológico con respecto a su aparición.

- Contexto: Descripción de herramientas, ideas y entornos que se emplearán para desarrollar el proyecto.
- Planificación y gestión del proyecto: Explicación de la planificación del proyecto software y modificaciones en esta a lo largo del proyecto.
- Desarrollo: Diagramas, mapas, tablas y esquemas que explican el funcionamiento de la aplicación a desarrollar.

- Despliegue y mantenimiento: Explicación sobre cómo ejecutar la aplicación y las necesidades de esta para que pueda funcionar.
- Conclusiones y posibles mejoras: Conclusiones sobre todas las partes del proyecto y exploración de posibles mejoras a la aplicación que se puedan realizar en un futuro.

A parte de la organización de las secciones sobre la organización de la parte referente al proyecto, al final de la memoria se encontrarán tres secciones más:

- Glosario: Explicación y aparición de algunos términos considerados relevantes.
- Bibliografía: Referencias bibliográficas empleadas durante el proyecto.
- Anexos: Otra información relevante sobre el proyecto.

Capítulo 2. Contexto

2.1 Plataforma Moodle

Moodle, acrónimo de, en inglés, *Modular Object-Oriented Dynamic Learning Enviroment* (Entorno modular dinámico de aprendizaje y orientado a objetos), es sistema de gestión de contenidos empleado en la enseñanza (LMCS, de sus siglas en inglés: *Learning Content Management System*). Es una plataforma de distribución libre que puede ejecutarse en la mayoría de los sistemas operativos. [1]

La primera versión apareció el 20 de agosto de 2002, creada por Martin Dougiamas, que en ese momento trabajaba como administrador de WebTC en la Universidad Tecnológica de Curtin, Australia. Desde entonces, la herramienta ha ido evolucionando y aumentando su número de usuarios. Hoy en día cuenta casi 130 millones de usuarios distribuidos por más de 230 países. [2]

Uno de los mayores atractivos de esta plataforma es la capacidad de personalización que ofrece, tanto por ser su código público, lo cual permite su modificación, como por la posibilidad de desarrollar plug-ins. Los plug-ins permiten aportar nuevas funcionalidades a la plataforma de una forma cómoda y sencilla. Al instalarse como módulos, estos añadidos pueden ser compartidos por la comunidad para que el resto de los usuarios pueda disfrutar de ellos. De esta manera se consigue una mejora en la calidad de la enseñanza, tal como quería su creador. Moodle está programado utilizando PHP, lo que permite acceder a él desde prácticamente cualquier navegador, llegando de esa manera a un público mucho mayor.

Esta plataforma ofrece una gran cantidad de herramientas y utilidades para poder gestionar un curso. Algunos ejemplos pueden ser la jerarquización de los recursos ofrecidos por el cuerpo docente, la posibilidad de realizar pruebas evaluadas (algunas de ellas pueden ser corregidas automáticamente), o la opción de llevar un recuento de las calificaciones de cada alumno. La versión más actual es la 3.5, pero es la versión anterior, la 3.4, la que más se está empleando y aún mantiene casi a la mitad de los usuarios.

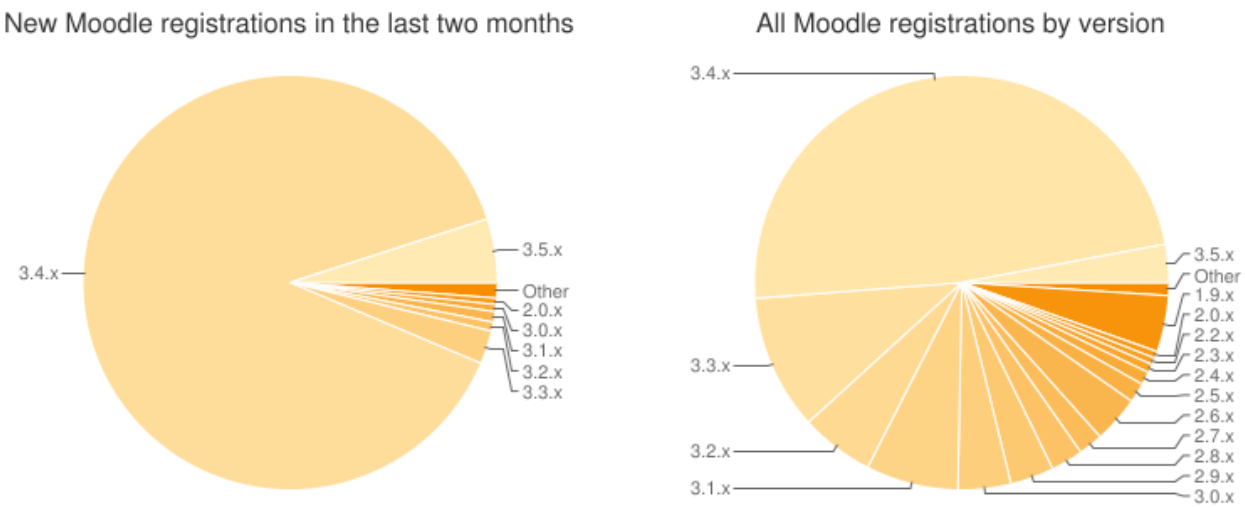

Ilustración 1 - Versiones de Moodle utilizadas por los usuarios [24]

Moodle también ofrece facilidades para que los estudiantes puedan comunicarse entre ellos y con los tutores. Para esto, implementa un servicio de mensajería instantánea y la capacidad de crear foros. Tanto si se recibe un mensaje por el primero como si se crea o responde en un post de un foro, el alumno recibirá una alerta en la que podrá ver el mensaje y, si es el primer caso, responder al momento.

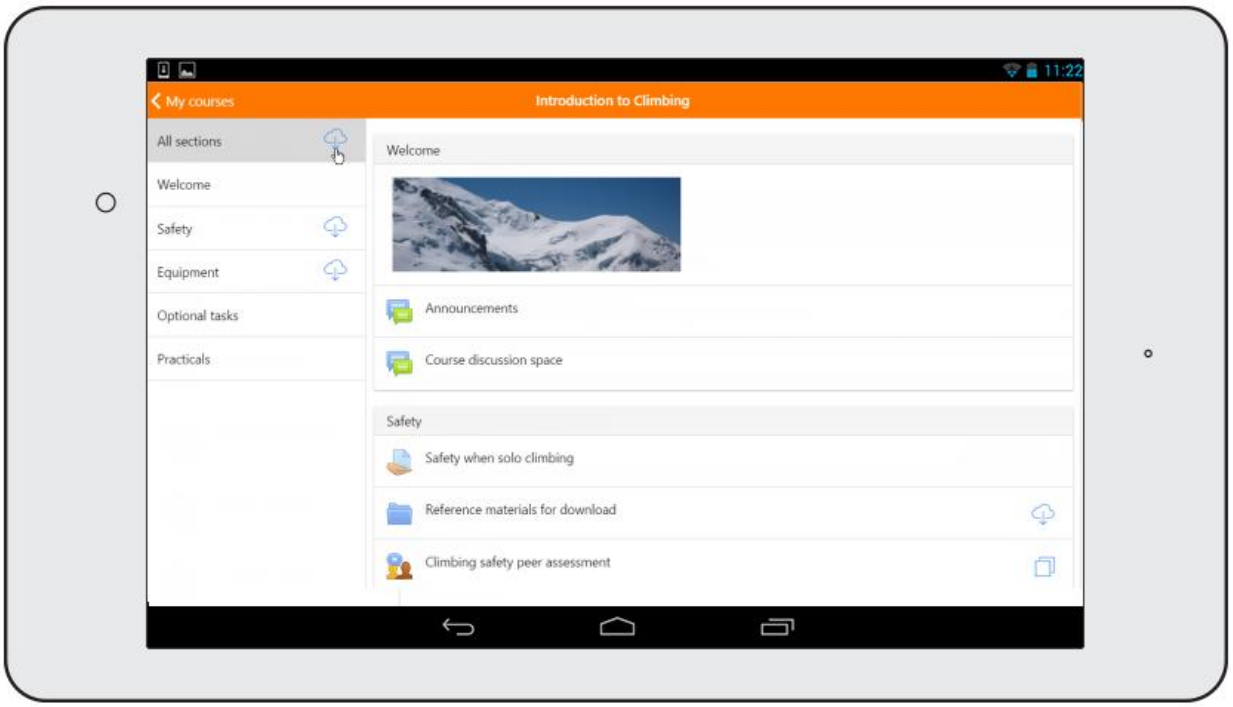

Ilustración 2 - Ejemplo de la aplicación móvil de Moodle [23]

Moodle también posee una aplicación móvil llamada *Moodle Mobile* desde la cual se puede acceder a todos los contenidos empleando una interfaz adaptada a este tipo de dispositivos. Para poder conectarse a un servicio que implemente Moodle únicamente es necesario conocer la URL de dicho servicio y las credenciales para autenticarse contra él (de la misma manera que se haría desde un navegador común).

2.2 Plataforma Telegram

Telegram Messenger es servicio de mensajería instantánea cuya primera versión fue desarrollada en 2013 por los hermanos Nikolái y Pável Dúrov. Actualmente está administrado por la organización sin ánimo de lucro Telegram Messenger LLP, con sede en Dubái, Emiratos Árabes Unidos.

Está enfocado al envío y recepción de mensajes de texto y multimedia, colocando a la velocidad de transmisión y la seguridad de los mensajes como objetivos principales. El contenido multimedia distingue a fecha de hoy varios formatos, que se podrían englobar en las categorías de Imagen (Cámara y Galería), Vídeo, Audio (Música), Contacto telefónico (Contacto), Posición GPS (Ubicación) y Otros tipos de archivo (Archivo).

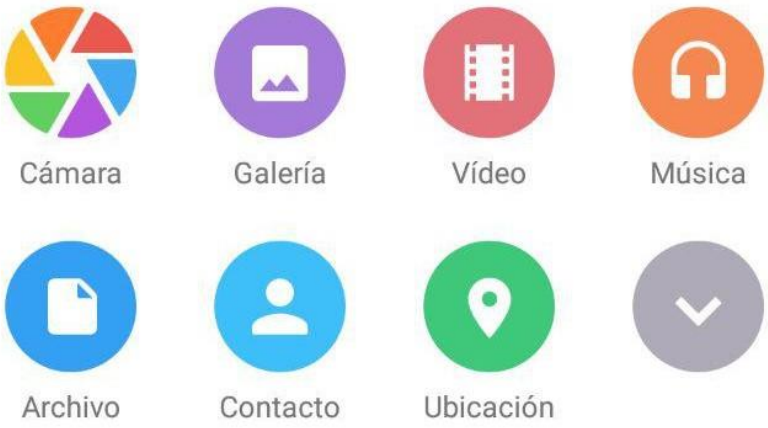

Ilustración 3 - Menú para compartir archivos de Telegram

Telegram almacena los archivos y mensajes enviados, lo cual permite acceder a estos desde otros dispositivos siempre que se hayan compartido usando la aplicación. Para que dejen de ser accesibles pueden eliminarse del chat los mensajes deseados o emplear los chats secretos, que encriptan y desencriptan la información enviada en los dispositivos de emisión y recepción y no almacenan nada en los servidores de Telegram (lo cual significa que estos chats son solo accesibles desde los dispositivos en los que se crean). [3]

Es un servicio multiplataforma. Puede accederse a él a través de un navegador web o con las aplicaciones diseñadas para trabajar de manera nativa sobre algunos Sistemas Operativos. Ejemplos de estas aplicaciones podrían ser la aplicación de Telegram para Android o Telegram Desktop y sus versiones para funcionar en Windows, en macOS o en Linux. Además de ser multiplataforma, permite utilizar la misma sesión simultáneamente desde diversos dispositivos manteniendo todos estos sincronizados a tiempo real. [4]

Las funcionalidades, ya básicas, de las aplicaciones de mensajería tales como la creación de grupos, la realización de llamadas o la configuración de la aplicación, entre otras, también se encuentran disponibles en Telegram. Sin embargo, ofrece la posibilidad de crear bots, aplicaciones que se conectan con Telegram a través de la API proporcionada por la organización y que a, efectos prácticos, actúan casi como usuarios: pueden enviar y recibir mensajes, ser añadidos a grupos, identificar a quienes entran en un grupo, etcétera. Una de las pocas diferencias es que los bots no pueden añadir ellos

mismos a personas a grupos. La combinación de los bots con la posibilidad de editar y eliminar mensajes propios y la capacidad de añadir botones a los mensajes permite desarrollar aplicaciones con interfaces gráficas dinámicas.

Otras peculiaridades de Telegram, menos trascendentales que las ya mencionadas, pero no sin importancia, serían la capacidad de anclar mensajes en un grupo para su rápido acceso por parte de cualquier miembro, la opción de crear canales de difusión a los que los usuarios pueden suscribirse para únicamente recibir mensajes, la opción de enviar mensajes con emoticonos personalizados (*stickers*) o la capacidad de los administradores de los grupos de eliminar cualquier contenido del grupo que administran, haya sido enviado por ellos o no (como aclaración, los bots pueden tener estas mismas capacidades si se les otorga el rango de administrador en un grupo al que pertenezcan).

Telegram también cuenta con una API para desarrollar clientes personalizados, pero es una funcionalidad que, pese a poseer una gran potencialidad, no será usada por gran parte de los usuarios.

2.3 Telegram Bot API

La interfaz de programación de aplicaciones (API, por sus siglas en inglés) de bots de Telegram es el conjunto de herramientas puestas a disposición de todas aquellas personas que deseen desarrollar un bot para este servicio.

Telegram ofrece APIs para distintos lenguajes, aunque todas funcionan traduciendo el mensaje que queramos enviar en el lenguaje que estemos empleando a un mensaje HTTP [5]. La API también permite crear bots con la funcionalidad de aceptar pagos de los usuarios de Telegram. Esta función está disponible desde la versión del 18 de mayo del 2017. Otra peculiaridad de los bots es que pueden conectarse con juegos diseñados con HTML5.

2.3.1 BotFather

Antes de empezar a desarrollar un bot, es necesario solicitar a la plataforma que guarde un nombre para este. Para conseguir esto lo primero que hay que hacer es enviar un mensaje a @BotFather y comenzar a interactuar con él por medio de los comandos y respondiendo a sus solicitudes para completar la información necesaria sobre el bot a crear.

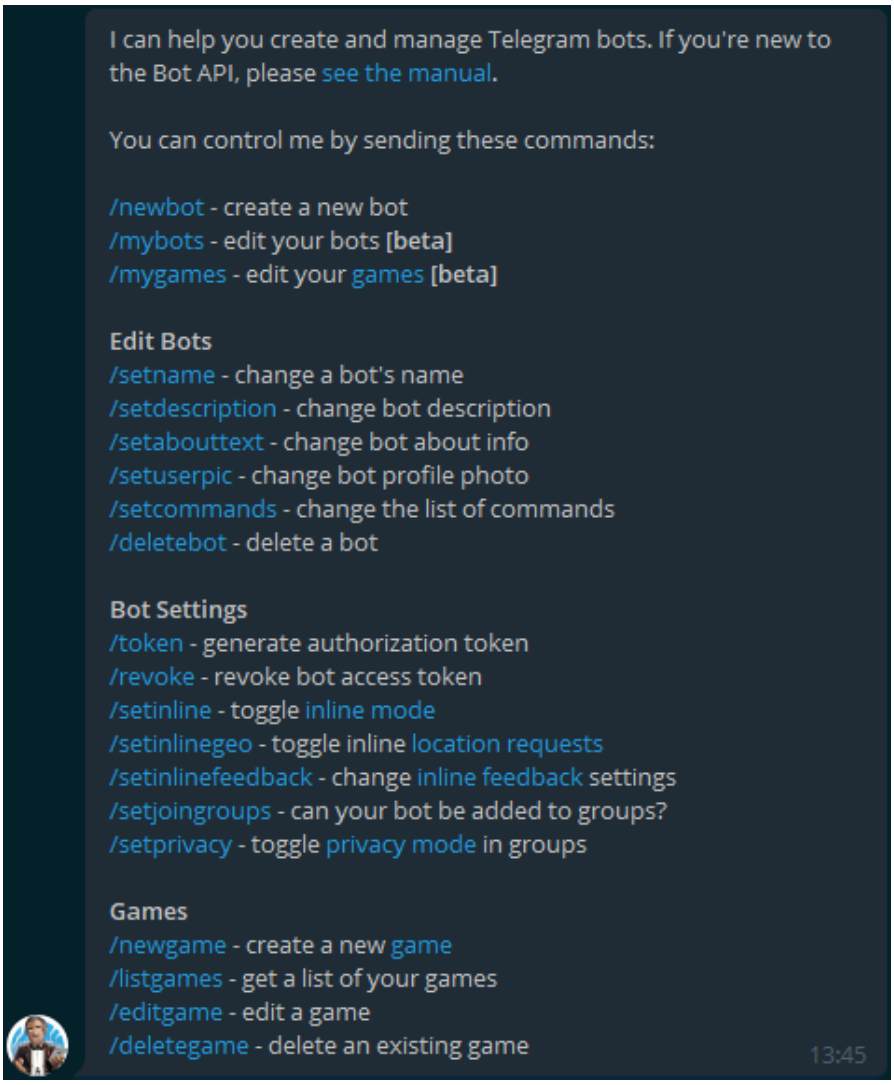

Ilustración 4 - Opciones que ofrece @BotFather

Primero se envía el mensaje "/newbot" para indicar a @BotFather que se desea registrar un nuevo bot. A continuación, se envía un mensaje con el nombre que queramos que tenga y, para finalizar, el "nombre de usuario" que deseamos que utilice. El nombre de usuario tiene que ser único y tiene que terminar en "bot", sin tener en cuenta mayúsculas o minúsculas. @BotFather es el único bot que no cumple esta característica.

Tras registrar el bot, @BotFather enviará un TOKEN único para poder identificar el bot cuando trabajemos con la API. También se podrá, a partir de ahora, gestionar la configuración del bot desde el mismo chat con @BotFather. De esta manera podremos indicar a Telegram qué funcionalidades va a requerir u ofrecer a parte del envío básico de mensajes y archivos.

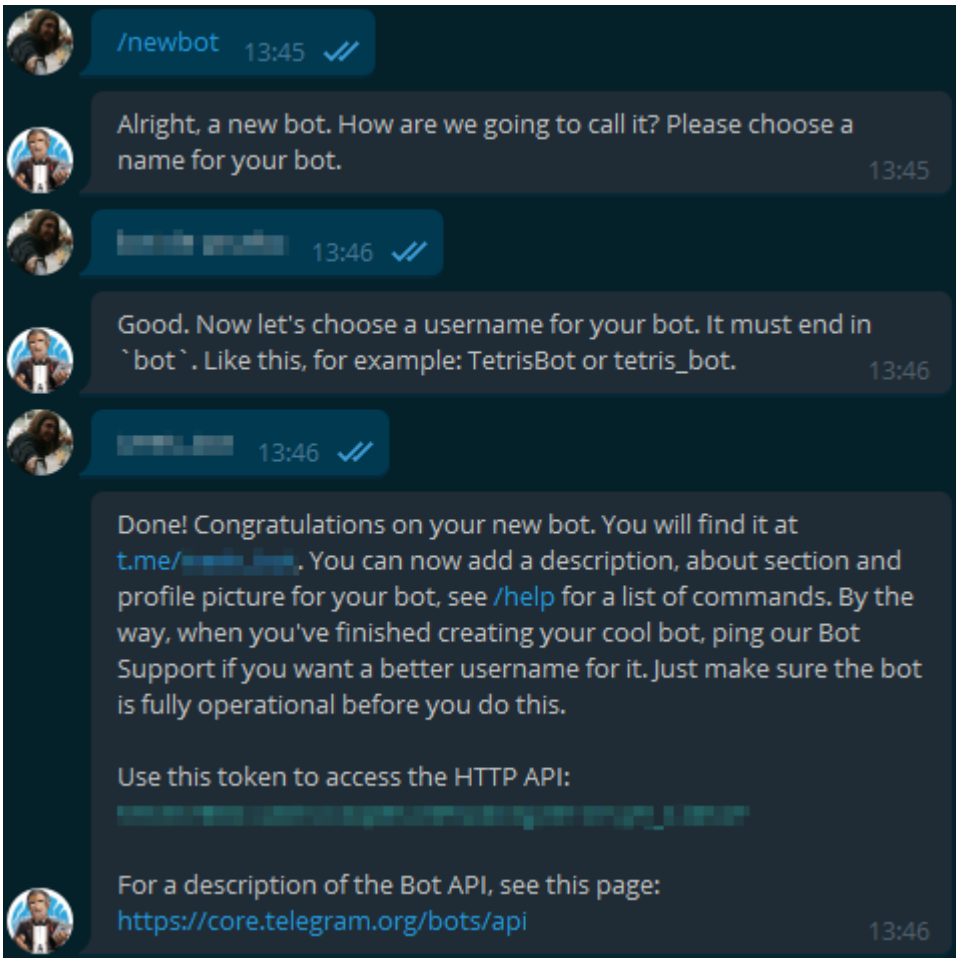

Ilustración 5 - Diálogo con @BotFather para registrar un nuevo bot

Una vez creado un bot, este puede ser configurado en el mismo chat empleando ciertos comandos. Una utilidad interesante es la que ofrece el *modo inline*. Si se informa a @BotFather de que el bot posee esta capacidad, los usuarios podrán emplear el bot en chats ajenos a este (por ejemplo, escribiendo en un chat "@nombredelbot nuevo recordatorio").

2.3.2 Modelo de dominio de la API bot

La API ofrece una serie de clases que permiten a los desarrolladores trabajar cómodamente y de una manera unificada a la hora de implementar los bots que deciden crean. Aquí se tratarán las clases que resultan de mayor interés para este TFG. [6]

2.3.2.1 User

Estos objetos representan tanto a usuarios como a bots de Telegram.

User

This object represents a Telegram user or bot.

Field	Type	Description
id	Integer	Unique identifier for this user or bot
is_bot	Boolean	True, if this user is a bot
first_name	String	User‘s or bot’s first name
last_name	String	*Optional*. User‘s or bot’s last name
username	String	*Optional*. User‘s or bot’s username
language_code	String	*Optional*. IETF language tag of the user's language

Ilustración 6 - Clase User ofrecida por la API Bot

De estos objetos nos interesa extraer un identificador único de cada usuario. Tanto *id* como *username* cumplen esta función. *id* es el identificador único de cada usuario, mientras que *username* es el alias elegido por cada usuario o bot. El identificador es fijo, mientras que el alias puede cambiar. Se usa el identificador para diferenciar a los usuarios, pero se emplea el alias, que es público, para facilitar las conversaciones de los usuarios con el bot. Como el alias es opcional en la plataforma, en caso de que no se disponga de este se utiliza el campo *first_name* (primer nombre del usuario o bot) y, si está disponible, *last_name* (apellido del usuario o bot).

2.3.2.2 Chat

Estos objetos representan los chats de los que el bot formar parte.

De esta clase se emplea únicamente el campo *id* que, como en el caso anterior, representa el identificador único asociado a cada chat.

2.3.2.3 Message

Estos objetos representan los mensajes que puede recibir el bot a través del chat.

La clase Message posee una gran cantidad de campos opcionales. En este trabajo, debido a que los únicos medios del usuario para introducir datos serán mensajes basados en el texto y la utilización de los teclados generados por el bot, se utilizan los siguientes campos:

- *id*: Identificador único para cada mensaje.
- *from*: el tipo de este campo es User. Representa al usuario que ha enviado el mensaje.
- *chat*: el tipo de este campo es Chat. Representa al chat en el que se ha enviado el mensaje.
- *text*: Si se ha recibido una cadena de texto en el mensaje, String que representa ese texto.
- *entities*: Si se ha recibido una cadena de texto, lista de objetos de tipo MessageEntity (explicado más adelante) que almacena los elementos de ese texto que no son texto plano.

- *audio*: Si se ha recibido un archivo de audio, objeto del tipo Audio (explicado más adelante) que hace referencia a ese archivo.
- *document*: Si se ha recibido un archivo, objeto del tipo Document (explicado más adelante) que hace referencia a ese archivo.
- *photo*: Si se ha recibido un archivo de imagen, lista de objetos del tipo PhotoSize (explicado más adelante) que hacen referencia a ese archivo.
- *sticker*: Si se ha recibido un sticker, objeto del tipo Sticker (explicado más adelante) que hace referencia al sticker.
- *video*: Si se ha recibido un archivo de video, objeto del tipo Video (explicado más adelante) que hace referencia a ese archivo.
- *voice*: Si se ha recibido una nota de audio, objeto del tipo Voice (explicado más adelante) que hace referencia a ese archivo.
- *video_note*: Si se ha recibido una nota en vídeo, objeto del tipo VideoNote (explicado más adelante) que hace referencia a ese archivo.

2.3.2.4 MessageEntity

Estos objetos almacenan información sobre los elementos de un mensaje de texto que no son texto plano.

MessageEntity

This object represents one special entity in a text message. For example, hashtags, usernames, URLs, etc.

Field	Type	Description
type	String	Type of the entity. Can be *mention* (`@username`), *hashtag*, *bot_command*, *url*, *email*, *bold* (bold text), *italic* (italic text), *code* (monowidth string), *pre* (monowidth block), *text_link* (for clickable text URLs), *text_mention* (for users without usernames)
offset	Integer	Offset in UTF-16 code units to the start of the entity
length	Integer	Length of the entity in UTF-16 code units
url	String	*Optional.* For "text_link" only, url that will be opened after user taps on the text
user	User	*Optional.* For "text_mention" only, the mentioned user

Ilustración 7 – Clase MessageEntity ofrecida por la API Bot

Los objetos de esta clase no poseen significado por sí mismos, pues hay que almacenarlos junto con el texto al que hacen referencia.

Los elementos que no son considerados texto plano son aquellos que cumplen alguna de las siguientes características:

- Texto en negrita.
- Texto en cursiva.
- Mención a un usuario.
- Comando de bot.

- URL.
- Texto que se puede clicar y conduce a una URL.
- Dirección de correo electrónico.

El significado de los campos de este tipo de objetos es el siguiente:

- *type*: El tipo del elemento al que hace referencia el objeto.
- *offset*: Posición en la que se encuentra el elemento dentro del texto del mensaje al que acompaña.
- *length*: Cantidad de caracteres del elemento.
- *url*: En caso de que el elemento sea un texto sobre el que se puede clicar y conducir a una URL, este campo almacena dicha URL.
- *user*: En caso de que el elemento sea una mención a un usuario que no posee un alias, este campo almacena un objeto de tipo User con los datos de dicho usuario.

2.3.2.5 Document

Los objetos de esta clase representan un archivo que haya sido enviado al bot.

De esta clase utilizaremos los campos *file_id*, que almacena el identificador del archivo dentro de Telegram, y *mime_type*, que contiene el tipo MIME del archivo recibido. Se empleará el tipo MIME del archivo para enviarlo correctamente cuando sea requerido por un usuario. El identificador se utilizará para reconocer el archivo que hay que enviar en caso de que sea necesario.

2.3.2.6 Audio

Los objetos de esta clase representan un archivo de audio que haya sido enviado al bot.

De esta clase utilizaremos los campos *file_id*, que almacena el identificador del archivo dentro de Telegram, y *mime_type*, que contiene el tipo MIME del archivo recibido. Se empleará el tipo MIME del archivo para enviarlo correctamente cuando sea requerido por un usuario. El identificador se utilizará para reconocer el archivo que hay que enviar en caso de que sea necesario.

2.3.2.7 PhotoSize

Los objetos de esta clase representan un archivo de imagen que haya sido enviado al bot.

De esta clase utilizaremos los campos *file_id*, que almacena el identificador del archivo dentro de Telegram, y *mime_type*, que contiene el tipo MIME del archivo recibido. Se empleará el tipo MIME del archivo para enviarlo correctamente cuando sea requerido por un usuario. El identificador se utilizará para reconocer el archivo que hay que enviar en caso de que sea necesario.

2.3.2.8 Sticker

Los objetos de esta clase representan un sticker que haya sido enviado al bot.

De esta clase utilizaremos los campos *file_id*, que almacena el identificador del archivo dentro de Telegram, y *mime_type*, que contiene el tipo MIME del archivo recibido. Se empleará el tipo MIME del archivo para enviarlo correctamente cuando sea requerido por un usuario. El identificador se utilizará para reconocer el archivo que hay que enviar en caso de que sea necesario.

2.3.2.9 Video

Los objetos de esta clase representan un vídeo que haya sido enviado al bot.

De esta clase utilizaremos los campos *file_id*, que almacena el identificador del archivo dentro de Telegram, y *mime_type*, que contiene el tipo MIME del archivo recibido. Se empleará el tipo MIME del archivo para enviarlo correctamente cuando sea requerido por un usuario. El identificador se utilizará para reconocer el archivo que hay que enviar en caso de que sea necesario.

2.3.2.10 Voice

Los objetos de esta clase representan una nota de audio que haya sido enviada al bot.

De esta clase utilizaremos los campos *file_id*, que almacena el identificador del archivo dentro de Telegram, y *mime_type*, que contiene el tipo MIME del archivo recibido. Se empleará el tipo MIME del archivo para enviarlo correctamente cuando sea requerido por un usuario. El identificador se utilizará para reconocer el archivo que hay que enviar en caso de que sea necesario.

2.3.2.11 VideoVoice

Los objetos de esta clase representan una nota en vídeo que haya sido enviada al bot.

De esta clase utilizaremos los campos *file_id*, que almacena el identificador del archivo dentro de Telegram, y *mime_type*, que contiene el tipo MIME del archivo recibido. Se empleará el tipo MIME del archivo para enviarlo correctamente cuando sea requerido por un usuario. El identificador se utilizará para reconocer el archivo que hay que enviar en caso de que sea necesario.

2.3.2.12 InlineKeyboardMarkup

Telegram da a los bots la capacidad de añadir botones en el chat para que la interacción del usuario sea más cómoda e intuitiva. Esta clase indica a la aplicación que los botones que contiene tienen que ser mostrados junto al mensaje de texto enviado.

Esta clase pose un único campo, *inline_keyboard*, que almacena una lista de listas de botones. Entrando más detalle, almacena una lista en la que cada elemento equivale a una fila, y cada fila es una lista que almacena, en orden, los botones.

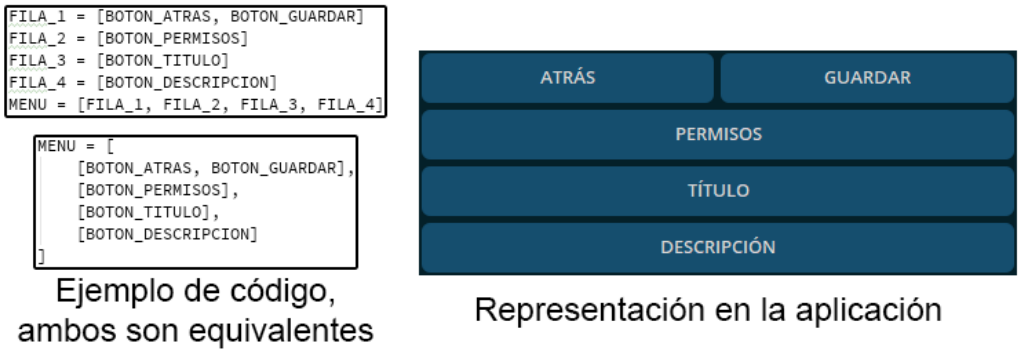

Ilustración 8 - Ejemplo de un objeto de la clase InlineKeyboardMarkup de la API Bot

Debido a esta clase, no se podrá utilizar el bot si la versión de Telegram que se utiliza es anterior al 9 de abril de 2016.

2.3.2.13 InlineKeyboardButton

Cada objeto de esta clase representa un botón de los teclados que aparecen junto al texto. Los objetos de la clase InlineKeyboardMarkup almacenan una lista de listas de objetos de esta clase.

De esta clase los campos que se utilizarán serán:

- *text*: String que almacena el texto que mostrará el botón.
- *url*: URL que se abrirá cuando el botón sea pulsado. Es opcional, se utilizará cuando el valor de una entrada sea una URL.
- *callback_data*: String que almacena la información que será enviada al bot en un objeto del tipo CallbackQuery (explicado más adelante) cuando se pulse el botón.

2.3.2.14 CallbackQuery

Un objeto de esta clase representa la consulta que recibe el bot una vez que se ha presionado un botón.

Esta clase también se emplea en los bots de juegos y por ello tiene un campo específico, complementario a otro campo que emplea el bot diseñado en este TFG. Debido a que ese campo no es utilizado, es ignorado en este caso. Los campos relevantes son:

- *id*: Identificador único de la consulta.
- *from*: Objeto del tipo User que representa al usuario que ha enviado la consulta.
- *inline_message_id*: Si la consulta ha sido originada por un objeto del tipo InlineKeyboardButoon, identificador único del mensaje en el que se encontraba el botón.
- *data*: Si el mensaje no ha sido recibido a causa de un juego contiene el texto que había sido almacenado en el campo *callback_data* del elemento que ha generado la consulta.

2.3.3 Envío de mensajes

La API para bots de Telegram ofrece funciones muy variadas para poder manejar casi cualquier aspecto de las conversaciones que los usuarios mantengan con los bots. Aquí se tratarán las que resultan de mayor interés para este TFG. [7]

Antes de explicar los métodos cabría destacar algunas peculiaridades de algunos de sus parámetros que son compartidos entre varios de ellos.

2.3.3.1 parse_mode (Modo de análisis gramatical)

Los mensajes de texto enviados por el bot, al igual que los de los usuarios, pueden dar formato al texto poniéndolo en negrita, cursiva, justificarlo o interpretándolo como una URL. Para conseguir esto hay que añadir etiquetas en el mensaje, y se he puede hacer de dos maneras. Este campo puede tomar dos valores:

- Markdown: las etiquetas a añadir serán las mismas que las que utilizan los usuarios (asteriscos, guiones bajos, etc.)
- HTML, se emplearán el mismo lenguaje que se emplea en HTML.

2.3.3.2 reply_markup (Opciones adicionales de interfaz)

El medio habitual para enviar mensajes por Telegram es a través del texto de estos, pero el servicio de mensajería también ofrece la posibilidad de añadir botones. Estas opciones pueden ser de gran ayuda a la hora de diseñar diálogos en los que el usuario tiene que escoger entre varias opciones, o si la respuesta de este tiene que relacionarse con un mensaje del bot en concreto. Si este campo tiene valor, este debe ser un objeto de una de las siguientes cuatro clases:

- InlineKeyboardMarkup: Ya explicado anteriormente.
- ReplyKeyboardMarkup: Genera un teclado para introducir datos en lugar del campo de texto.
- ReplyKeyboardRemove: Elimina el teclado que sustituía al campo de texto.
- ForceReply: Fuerza a que el siguiente mensaje del usuario sea una respuesta al mensaje que lleva este parámetro.

2.3.3.3 sendMessage

Este método solicita al bot que envíe un mensaje.

sendMessage

Use this method to send text messages. On success, the sent Message is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
text	String	Yes	Text of the message to be sent
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
disable_web_page_preview	Boolean	Optional	Disables link previews for links in this message
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

Ilustración 9 - Documentación del método sendMessage proporcionado la API Bot

Todos los parámetros, aunque sean opcionales, tendrán importancia en el bot diseñado para este TFG. A continuación, se explica qué función tiene cada parámetro:

- *chat_id*: Identificador único del chat o del canal al que va a ser enviado el mensaje. Los canales pueden ser interesantes para notificar a los usuarios seguidores de cierta lista, así que el identificador podrá ser tanto un entero como un String.
- *text*: Texto que se enviará en el mensaje. Puede contener etiquetas para dar formato.
- *parse_mode*: Explicado anteriormente. Indica al procesador de texto qué conjunto de etiquetas utilizar para dar formato.
- *disable_web_page_preview*: En caso de que haya alguna URL en el mensaje, indica si se desea que se muestre una vista preliminar de esta o no.
- *disable_notification*: Indica si se desea que el usuario reciba una notificación sin sonido o con sonido.
- *reply_to_message_id*: Si el mensaje enviado es una respuesta a otro mensaje, en este campo va el identificador del mensaje que es respondido.
- *reply_markup*: Explicado anteriormente. Si tiene algún valor, ha de ser uno de los ya mencionados. Indica si el mensaje tiene alguna manera concreta de ser contestado.

2.3.3.4 sendPhoto

Este método solicita al bot que envíe una imagen.

sendPhoto

Use this method to send photos. On success, the sent Message is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
photo	InputFile or String	Yes	Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. More info on Sending Files »
caption	String	Optional	Photo caption (may also be used when resending photos by *file_id*), 0-200 characters
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

Ilustración 10 - Documentación del método sendPhoto proporcionado la API Bot

Los parámetros que comparte con el método sendMessage cumplen la misma función. Sin embargo, este método se emplea para enviar imágenes, así que posee dos parámetros que sendMessage no posee:

- *photo*: Puede tener 3 valores distintos. En este parámetro se pasar el propio archivo de la imagen que queremos enviar, una URL desde la cual Telegram pueda descargarse la imagen o el Identificador que Telegram le dio al archivo en caso de que la imagen solicitada haya sido enviada al bot anteriormente.
- *caption*: Representa el texto que se puede adjuntar junto con la imagen enviada.

2.3.3.5 sendAudio

Este método solicita al bot que envíe un archivo de audio de manera que la aplicación pueda abrirlo con el reproductor de audio.

El audio debe estar en formato MP3.

Los parámetros que comparte con el método sendPhoto (*chat_id, caption, parse_mode, disable_notification, reply_to_message_id, reply_markup*) cumplen la misma función. Sin embargo, este método se emplea para enviar audios que se espera reproducir, así que en vez del campo *photo* posee el campo *audio*, que cumple la función análoga.

También existen tres parámetros que aportan información extra sobre el archivo a reproducir: duración (*duration*), intérprete (*performer*) y título (*title*), pero no serán empleados en este trabajo.

2.3.3.6 sendAudio

Este método solicita al bot que envíe un vídeo.

El vídeo debe estar en formato MP4, para otros formatos debe utilizarse sendDcoument.

Los parámetros que comparte con sus métodos análogos (*chat_id, caption, parse_mode, disable_notification, reply_to_message_id, reply_markup*) cumplen la misma función. Sin embargo, este método se emplea para enviar vídeos, así que en vez su campo característico es *video*.

También existen cuatro parámetros que aportan información extra sobre el vídeo enviado: duración (*duration*), ancho (*width*), alto (*height*) y posibilidad de reproducirlo sin descargarlo completamente (*supports_streaming*), pero no serán empleados en este trabajo.

2.3.3.7 sendVoice

Este método solicita al bot que envíe un archivo de audio de manera que la aplicación pueda abrirlo como una nota de audio.

El audio debe estar en un fichero con formato OGG y debe haberse codificado con formato OPUS. Para otros formatos debe utilizarse sendAudio o sendDocument. Los parámetros que comparte con los métodos similares (*chat_id, caption, parse_mode, disable_notification, reply_to_message_id, reply_markup*) cumplen la misma función que en estos.

Este método posee un parámetro extra, para indicar la duración del audio, *duration*.

2.3.3.8 sendVideoNote

Este método solicita al bot que envíe un archivo de vídeo para que la aplicación como pueda abrirlo como una nota de video.

Desde la versión 4.0 de Telegram existe la posibilidad de enviar y recibir vídeos, de hasta un minuto, en formato MP4 en los cuales la imagen tiene forma de círculo en vez del rectángulo que se usa normalmente. Este método se emplea para enviar esos vídeos. Los parámetros que comparte con los últimos métodos mencionados (*chat_id, caption, parse_mode, disable_notification, reply_to_message_id, reply_markup*) cumplen la misma función que en estos.

Este método tiene dos parámetros adicionales, uno para duración (*duration*) y otro para el tamaño del círculo, representando el diámetro de este (*length*).

2.3.3.9 sendSticker

Este método solicita al bot que envíe un sticker.

Comparte con los otros métodos los parámetros *chat_id*, *disable_notification*, *reply_to_message_id* y *reply_markup*. El parámetro propio de este método es *sticker*, que funciona de la misma manera que el resto de los parámetros propios de los últimos métodos tratados (acepta una URL, un identificador o un archivo).

2.3.4 Edición de mensajes

Telegram ofrece la posibilidad de editar los mensajes que el bot ha enviado con anterioridad. Esta funcionalidad será útil a la hora de implementar una interfaz que resulte cómoda para el usuario a la hora de navegar entre las funcionalidades que ofrecerá el bot.

Los únicos mensajes que se pueden editar son aquellos que no tienen un *reply_markup* asociado o aquellos cuyo *reply_markup* es del tipo InlineKeyboardMarkup. Existen cuatro métodos distintos para modificar mensajes ya enviados. Aunque en el proyecto solo se vayan a emplear dos de ellos, se dará una pequeña explicación de todos ellos.

2.3.4.1 editMessageText

Este método permite modificar en su totalidad un mensaje de texto que se encuentre en el chat. También se puede editar el teclado del tipo InlineKeboardMarkup que tenga asociado, ya sea añadiendo uno nuevo, modificando el existente o eliminando el que existiese.

El método posee algunos parámetros complementarios que se explicarán a continuación, la función de estos parámetros es identificar el mensaje que se desea modificar:

- *chat_id*: Es necesario si no se proporciona el parámetro *inline_message_id*. Es el identificador del chat o del canal en el que se encuentra el mensaje que se desea modificar.
- *message_id*: Es necesario si no se proporciona el parámetro *inline_message_id*. Es el identificador del mensaje que se desea modificar.
- *inline_message_id*: Es necesario si no se proporciona los parámetros *chat_id* y *message_id*. Es el identificador único del mensaje en caso de que proceda de haber clicado un botón en un teclado del tipo InlineKeyboardMarkup. Aquí iría el campo del mismo nombre que se recibe en los objetos del tipo CallbackQuery.

editMessageText

Use this method to edit text and game messages sent by the bot or via the bot (for inline bots). On success, if edited message is sent by the bot, the edited Message is returned, otherwise *True* is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Optional	Required if *inline_message_id* is not specified. Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
message_id	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
text	String	Yes	New text of the message
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
disable_web_page_preview	Boolean	Optional	Disables link previews for links in this message
reply_markup	InlineKeyboardMarkup	Optional	A JSON-serialized object for an inline keyboard.

Ilustración 11 - Documentación del método editMessageText proporcionado la API Bot

El resto de los parámetros de este método equivales a sus homónimos en el método sendMessage, con la diferencia de que el usuario no recibirá una notificación en ningún caso puesto que la edición de los mensajes no genera notificaciones.

Cabe también destacar el parámetro *reply_markup*, que en este caso admite únicamente un objeto del tipo InlineKeyboardMarkup.

2.3.4.2 editMessageCaption

Este método permite modificar el texto de aquellos mensajes que hayan llevado asociado un archivo. Igual que en el caso anterior, también se puede añadir, editar o eliminar el teclado asociado al mensaje siempre que este sea del tipo InlineKeyboardMarkup.

Debido a que lleva un archivo asociado, en ningún caso se mostrará una vista preliminar de las URL que pueda haber en el texto del mensaje, así que no posee el parámetro *disable_web_page_preview* que sí puede encontrarse en editMessageText.

También intercambia el parámetro *text* por el parámetro *caption*, pero la modificación que realiza es similar: intercambia el texto que hubiese en el mensaje por el nuevo proporcionado. Sin embargo, así como en el caso anterior el texto era imprescindible (pues no puede enviarse un mensaje que no contenga ningún archivo ni tampoco texto), en este caso el parámetro *caption* es opcional, ya que el mensaje tiene asociado un archivo de antemano, cualidad que este método no puede modificar.

editMessageCaption

Use this method to edit captions of messages sent by the bot or via the bot (for inline bots). On success, if edited message is sent by the bot, the edited Message is returned, otherwise *True* is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Optional	Required if *inline_message_id* is not specified. Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
message_id	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
caption	String	Optional	New caption of the message
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	Optional	A JSON-serialized object for an inline keyboard.

Ilustración 12 - Documentación del método editMessageCaption proporcionado la API Bot

2.3.4.3 editMessageReplyMarkup

Este método modifica únicamente el teclado en pantalla que pudiese tener asociado el mensaje al que hace referencia. Como en los otros casos, puede añadir, editar o eliminar dicho teclado, y este ha de ser del tipo InlineKeyboardMarkup.

editMessageReplyMarkup

Use this method to edit only the reply markup of messages sent by the bot or via the bot (for inline bots). On success, if edited message is sent by the bot, the edited Message is returned, otherwise *True* is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Optional	Required if *inline_message_id* is not specified. Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
message_id	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
reply_markup	InlineKeyboardMarkup	Optional	A JSON-serialized object for an inline keyboard.

Ilustración 13 - Documentación del método editMessageReplyMarkup proporcionado la API Bot

Igual que con los otros dos métodos, los campos *chat_id* y *message_id* complementan al campo *inline_message_id*, y el campo *reply_markup* se emplea para almacenar el nuevo teclado a mostrar.

Este método es útil y recomendable usarlo cuando se esté seguro de que el contenido textual del mensaje no va a cambiar o no es necesario que cambie. Se puede emplear para dar por completada una interacción con el usuario o para modificar un menú que haga referencia a información fija, por ejemplificar su uso.

2.3.4.4 deleteMessage

Este método elimina el mensaje que se le indique en sus parámetros, pero hay algunas restricciones a la hora de eliminar un mensaje:

- El mensaje no puede tener más de 48 horas de antigüedad.
- El bot no puede eliminar mensajes del usuario en un chat privado.
- Es necesario darle el permiso *can_post_messages* para que pueda eliminar mensajes salientes en un canal.
- Es necesario dar el rango de administrado a un bot en un grupo para que pueda eliminar mensajes de cualquier usuario.
- Es necesario dale el permiso *can_delete_messages* para que pueda eliminar mensajes de cualquier usuario o bot en un canal o supergrupo.

Si se cumplen las condiciones para ello, este método eliminará el mensaje del chat al que hagan referencia sus parámetros:

- *chat_id*: Identificado del chat o canal del que se quiere eliminar el mensaje.
- *message_id*: Identificado del mensaje que quiere ser eliminado.

deleteMessage

Use this method to delete a message, including service messages, with the following limitations:
- A message can only be deleted if it was sent less than 48 hours ago.
- Bots can delete outgoing messages in groups and supergroups.
- Bots granted *can_post_messages* permissions can delete outgoing messages in channels.
- If the bot is an administrator of a group, it can delete any message there.
- If the bot has *can_delete_messages* permission in a supergroup or a channel, it can delete any message there.

Returns *True* on success.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
message_id	Integer	Yes	Identifier of the message to delete

Ilustración 14 - Documentación del método deleteMessage proporcionado la API Bot

2.3.5 Recibir actualizaciones de los chats y canales

Para que el bot pueda recibir actualizaciones existen dos medios: preparar un Webhook o emplear el método getUpdates.

Tanto si se utiliza un medio como el otro, las actualizaciones que pueda percibir el bot se almacenan en los servidores de Telegram hasta que el bot los solicita, pero serán eliminadas de los servidores de Telegram tras 24 horas desde que han ocurrido.

En ambos casos, cuando se solicita recuperar las actualizaciones se recibirá un conjunto de datos con la forma de un objeto de tipo Update (explicado más adelante). Esta información se recibirá como una lista de objetos serializados con formato JSON.

Cabe destacar que, si se está empleando un webhook, no se podrá utilizar el método getUpdates.

La mayor diferencia radica en que, al utilizar getUpdates es el programa el que se conecta a los servidores de Telegram mientras que, con un webhook, es la plataforma de Telegram la que se conecta al servidor proporcionado para enviarle la última actualización recibida.

2.3.5.1 Webhook

Al utilizar un webhook lo que se hace es solicitar a Telegram que, cada vez que nuestro bot perciba una actualización, envíe una solicitud HTTP con los datos de esta a la URL especificada. Telegram enviará un objeto del tipo Update serializado con formato JSON

Para especificar un webhook para el bot es necesario emplear el método setWebhook.

setWebhook

Use this method to specify a url and receive incoming updates via an outgoing webhook. Whenever there is an update for the bot, we will send an HTTPS POST request to the specified url, containing a JSON-serialized Update. In case of an unsuccessful request, we will give up after a reasonable amount of attempts. Returns *True* on success.

If you'd like to make sure that the Webhook request comes from Telegram, we recommend using a secret path in the URL, e.g. `https://www.example.com/<token>`. Since nobody else knows your bot's token, you can be pretty sure it's us.

Parameters	Type	Required	Description
url	String	Yes	HTTPS url to send updates to. Use an empty string to remove webhook integration
certificate	InputFile	Optional	Upload your public key certificate so that the root certificate in use can be checked. See our self-signed guide for details.
max_connections	Integer	Optional	Maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery, 1-100. Defaults to *40*. Use lower values to limit the load on your bot's server, and higher values to increase your bot's throughput.
allowed_updates	Array of String	Optional	List the types of updates you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See Update for a complete list of available update types. Specify an empty list to receive all updates regardless of type (default). If not specified, the previous setting will be used. Please note that this parameter doesn't affect updates created before the call to the setWebhook, so unwanted updates may be received for a short period of time.

Ilustración 15 - Documentación del método setWebhook proporcionado la API Bot

Si la petición HTTP no es respondida correctamente, Telegram intentará reenviar la petición varias veces antes de considerar que el receptor no puede responderla.

El método setWebhook tiene un máximo de 4 parámetros:

- *url*: URL HTTPS a la que Telegram se conectará para enviar los datos de la última actualización recibida por el bot.
- *certificate*: Certificado que utilizará Telegram para comprobar que el sitio web proporcionado en el parámetro *url* emplea el protocolo SSL o el protocolo TLS. Dependiendo del dominio al que Telegram tenga que enviar los mensajes en este campo pueden ir varios ficheros distintos. En cualquier caso, debe ir codificado en formato PEM.
- *max_connections*: Cantidad máxima de conexiones HTTPS simultaneas que se permite a Telegram realizar con el servidor proporcionado. La cantidad por defecto es 40, pero puede fijarse entre 1 y 100 conexiones.
- *allowed_updates*: Lista con los tipos de actualizaciones que se desea que reciba el bot. Para no tener que estar desechando manualmente peticiones se puede fijar aquí qué tipo de actualizaciones se desea que el bot atienda. Esto puede emplearse para reducir la carga del bot. Si se ha producido alguna actualización de un tipo no deseado antes de añadir el webhook, estas actualizaciones no serán desechadas por Telegram y habrá que procesarlas. Por defecto se atienden todos los tipos de actualización.

Tras emplear ese método, Telegram pasará a enviar las actualizaciones que se le haya solicitado a la URL proporcionada. Si se desea conocer qué webhook se ha proporcionado a Telegram está disponible el método getWebhookInfo.

El método devuelve un objeto del tipo WebhookInfo, que tendrá el camp *url* vacío en caso de que no haya ningún webhook activo o se esté utilizando el método getUpdates.

WebhookInfo

Contains information about the current status of a webhook.

Field	Type	Description
url	String	Webhook URL, may be empty if webhook is not set up
has_custom_certificate	Boolean	True, if a custom certificate was provided for webhook certificate checks
pending_update_count	Integer	Number of updates awaiting delivery
last_error_date	Integer	*Optional.* Unix time for the most recent error that happened when trying to deliver an update via webhook
last_error_message	String	*Optional.* Error message in human-readable format for the most recent error that happened when trying to deliver an update via webhook
max_connections	Integer	*Optional.* Maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery
allowed_updates	Array of String	*Optional.* A list of update types the bot is subscribed to. Defaults to all update types

Ilustración 16 - Clase WebhookInfo ofrecida por la API Bot

El objeto WebhookInfo representa un webhook. Si es el obtenido por el método getWebhookInfo, representará al webhook que esté empleando el bot en ese momento. Contiene los siguientes atributos:

- *url*: URL del webhook, estará vacío si no hay ningún webhook en uso.
- *has_custom_certificate*: Indica si al momento de usar setWebhook se proporción un certificado.
- *pending_update_count*: Cantidad de actualizaciones sin procesar por el bot.
- *last_error_date*: Tiempo Unix (segundos pasados desde medianoche del 1 de enero de 1970 en horario UTC) del último error ocurrido intentado enviar una actualización al webhook.
- *last_error_message*: Mensaje de error generado por el último error ocurrido intentando enviar una actualización al webhook.
- *max_connections*: Cantidad máxima de conexiones HTTPS simultaneas permitidas.
- *allowed_updates*: Lista con los tipos de actualización a los que atiende el bot.

Para solicitar a Telegram que deje de enviar las actualizaciones a webhook proporcionado se puede emplear el método deleteWebhook, que no posee argumentos. Tras utilizarlo, las actualizaciones pasarán a almacenarse en los servidores de Telegram a la espera de que sean solicitadas.

2.3.5.2 GetUpdate

GetUpdate es el método que permite obtener las actualizaciones recibidas por el bot en las últimas 24 horas siempre que se esté utilizando un webhook.

Utiliza tecnología *long polling*. Esta tecnología simula el sondeo tradicional del *polling*, pero cuando no hay información disponible para el cliente el servidor espera a que la haya para responder en lugar de enviar una respuesta vacía.

getUpdates

Use this method to receive incoming updates using long polling (wiki). An Array of Update objects is returned.

Parameters	Type	Required	Description
offset	Integer	Optional	Identifier of the first update to be returned. Must be greater by one than the highest among the identifiers of previously received updates. By default, updates starting with the earliest unconfirmed update are returned. An update is considered confirmed as soon as getUpdates is called with an *offset* higher than its *update_id*. The negative offset can be specified to retrieve updates starting from *-offset* update from the end of the updates queue. All previous updates will forgotten.
limit	Integer	Optional	Limits the number of updates to be retrieved. Values between 1—100 are accepted. Defaults to 100.
timeout	Integer	Optional	Timeout in seconds for long polling. Defaults to 0, i.e. usual short polling. Should be positive, short polling should be used for testing purposes only.
allowed_updates	Array of String	Optional	List the types of updates you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See Update for a complete list of available update types. Specify an empty list to receive all updates regardless of type (default). If not specified, the previous setting will be used. Please note that this parameter doesn't affect updates created before the call to the getUpdates, so unwanted updates may be received for a short period of time.

Notes

1. This method will not work if an outgoing webhook is set up.
2. In order to avoid getting duplicate updates, recalculate *offset* after each server response.

Ilustración 17 - Documentación del método getUpdates proporcionado la API Bot

El método puede emplearse con varios parámetros, todos ellos opcionales:

- *offset*: Identificador de la primera actualización que se desea que Telegram envíe para ser tratada. Por defecto enviará la primera cronológicamente que no se haya sido tratada. El servidor de Telegram considera que una actualización ha sido tratada cuando recibe el método getUpdates y este campo contiene un identificador mayor que el de dicha actualización. También se puede emplear un número negativo, que indicará la cantidad de actualizaciones que queremos recibir empezando por la última acontecida. A modo de ejemplo, si el bot ha recibido 10 mensajes y dichas actualizaciones tienen como identificadores valores del 101 al 110, el uso de getUpdates() no marcaría ninguna como tratada, getUpdates(103) marcaría las 101 y 102 como tratadas y getUpdates(-3) marcaría todas menos la 108, la 109 y la 110.
- *limit*: Indica la cantidad de actualizaciones que se desea recibir. El valor puede estar comprendido entre 1 y 100, siendo 100 el valor por defecto.
- *timeout*: Tiempo en segundos que esperará el servidor de Telegram a recibir datos antes de responder con un mensaje vacío. Por defecto es 0, así que si no se da un valor el método actúa como un sondeo normal en vez de un *long polling*. Se recomienda emplear 0 segundos únicamente con intenciones de realizar pruebas.

- *allowed_updates*: Lista de los tipos de actualizaciones que desean recibirse. Si se ha proporcionado una lista, Telegram descartará las actualizaciones con un tipo distinto mientras no se proporcione una lista distinta. Igual que al añadir un webhook, si se han recibido actualizaciones de un tipo no deseado antes de fijar los tipos deseados, estas también se recibirán. Una lista vacía indica a Telegram que desean recibirse todos los tipos de actualizaciones.

Telegram recomienda recalcular el parámetro *offset* cada vez que se emplea el método para evitar recibir actualizaciones duplicadas.

2.3.5.3 Update

Update es una clase que proporciona la API Bot de Telegram. Representa una actualización perceptible por el bot.

Update

This object represents an incoming update.
At most **one** of the optional parameters can be present in any given update.

Field	Type	Description
update_id	Integer	The update's unique identifier. Update identifiers start from a certain positive number and increase sequentially. This ID becomes especially handy if you're using Webhooks, since it allows you to ignore repeated updates or to restore the correct update sequence, should they get out of order. If there are no new updates for at least a week, then identifier of the next update will be chosen randomly instead of sequentially.
message	Message	*Optional.* New incoming message of any kind — text, photo, sticker, etc.
edited_message	Message	*Optional.* New version of a message that is known to the bot and was edited
channel_post	Message	*Optional.* New incoming channel post of any kind — text, photo, sticker, etc.
edited_channel_post	Message	*Optional.* New version of a channel post that is known to the bot and was edited
inline_query	InlineQuery	*Optional.* New incoming inline query
chosen_inline_result	ChosenInlineResult	*Optional.* The result of an inline query that was chosen by a user and sent to their chat partner. Please see our documentation on the feedback collecting for details on how to enable these updates for your bot.
callback_query	CallbackQuery	*Optional.* New incoming callback query
shipping_query	ShippingQuery	*Optional.* New incoming shipping query. Only for invoices with flexible price
pre_checkout_query	PreCheckoutQuery	*Optional.* New incoming pre-checkout query. Contains full information about checkout

Ilustración 18- Clase Update ofrecida por la API Bot

Como puede apreciarse en la documentación ofrecida por Telegram, hay un único campo obligatorio, *update_id*. Sin embargo, todos los objetos Update tendrán dos campos:

- *update_id*: Identificador único de la actualización. Es un entero incremental, cada actualización incremente el contador en 1. El identificador puede ser muy útil si se emplea un webhook, puesto que puede darse el caso de que se reciba una petición HTTP duplicada por algún problema en la red. Con este campo puede llevarse la cuenta de qué actualizaciones han sido tratadas y cuáles no. Utilizando el método getUpdates el problema también puede presentarse, pero existe la opción de pedir las actualizaciones a partir de una dada, por lo que el valor de este campo pasaría de usarse para compararse con los demás a incrementarse en 1 y solicitar las actualizaciones a partir del nuevo número.
- Uno de los campos opcionales restantes, que identifica el tipo de actualización que se ha producido. Este atributo puede representar los siguientes tipos de actualizaciones:
    - *message*: Representa un mensaje que ha sido recibido en un chat. Es un objeto del tipo Message.
    - *edited_message*: Representa un mensaje que ha sido editado en un chat. Es un objeto del tipo Message.
    - *channel_post*: Representa un mensaje enviado en un canal. Es un objeto del tipo Message.
    - *edited_channel_post*: Representa un mensaje que ha sido editado en un canal. Es un objeto del tipo Message.
    - *inline_query*: Representa un mensaje perceptible por el bot en chat o canal ajeno a este. Es un objeto del tipo InlineQuery.
    - *chosen_inline_result*: Representa la opción elegida por el usuario al utilizar al bot en un chat ajeno a este. Es un objeto del tipo ChosenInlineResult
    - *callback_query:* Representa los datos enviados por el usuario al pulsar un botón del tipo InlineKeyboardButton. Es un objeto del tipo CallbackQuery.
    - *shipping_query*: Representa el mensaje enviado por el usuario cuando, al realizar una compra, la dirección de envío influye en el precio de esta. Es un objeto del tipo ShippingQuery.
    - *pre_checkout_query:* Representa la confirmación del pago tras la confirmación del método de pago y la dirección del usuario. Es un objeto del tipo PreCheckoutQuery.

La parte de la API encargada de gestionar los pagos por servicios puede ser muy útil para cierto tipo de bots, pero en este Trabajo de Fin de Grado no se tratará nada semejante, así que no se profundizará en ello.

2.3.6 Lenguajes de programación y Bot API

La API proporcionada por Telegram emplea únicamente mensajes HTTP. En algunos casos, a la hora de trabajar con la API las consultas pueden volverse muy verbosas y complejas, por lo que desarrollar envoltorios o fachadas que faciliten la interacción con esta es una medida común que facilita el trabajo y lo hace más cómodo.

Por este motivo algunos usuarios de Telegram han desarrollado bibliotecas para diversos lenguajes de programación con el fin de facilitar el trabajo al resto de desarrolladores, permitiéndoles relacionarse con todas las funcionalidades de la API original de una manera adaptada al lenguaje de programación que esté utilizando. Las bibliotecas cuyos autores han hecho públicas pueden encontrarse en el propio sitio web de Telegram, y por el momento podemos encontrarlas para los siguientes lenguajes de programación: PHP, Java, Node.js, Pyhon, C#, Ruby, Go, Lua, Haskell, Scala, Swift y OCaml. Estas bibliotecas adaptan las clases y los métodos ofrecidos por Telegram a la nomenclatura y herramientas ofrecidas por cada lenguaje de programación.

Gracias a la labor de los desarrolladores de estas bibliotecas se acerca la creación de bots a otros desarrolladores que pueden no estar familiarizados con el envío y recepción de peticiones HTTP, ampliando así la cantidad de bots para la plataforma.

Capítulo 3. Plan de desarrollo

3.1 Análisis de riesgos

Como en cualquier proyecto, un Trabajo de Fin de Grado también puede ver alterados sus requisitos específicos, su implementación o incluso su planificación. Debido a estas posibles situaciones adversas que puedan modificar el flujo de trabajo y el desarrollo del proyecto se realiza un Análisis de riesgos con el fin de minimizar el impacto de dichas adversidades.

Elaborar un plan para la aparición de cada riesgo concreto evitaría o reduciría la repercusión de estos en el máximo factor posible. Sin embargo, prevenir todos y cada uno de los riesgos es una tarea que requiere en sí misma tiempo y recursos, por lo que es común clasificarlos y prestar más atención a aquellos que se consideren más relevantes. Los riesgos se han clasificado en función de dos parámetros: probabilidad y repercusión.

La probabilidad hace referencia a cómo de factible es la aparición de dicho riesgo; cuanto mayor sea la probabilidad de que suceda más habrá que tenerlo en cuenta, puesto que pese a poder tener una repercusión pequeña su repetición podría acabar influyendo en gran medida al proyecto. Por su lado, la repercusión hace referencia a cómo de perjudiciales serían las consecuencias en caso de que el riesgo se consumase. Una repercusión alta, aunque sea poco probable, puede implicar modificar la completitud del proyecto.

Teniendo en cuenta estos dos factores se tratará con más detalle aquellos riesgos que presenten tanto una gran probabilidad como una repercusión alta. Los riesgos que se han considerado más relevantes para este proyecto son los siguientes:

Nombre del riesgo	R01 - Falta de tiempo prolongada		
Probabilidad	Media	**Repercusión**	Alta
Descripción			
Debido a factores externos al proyecto y sus elementos más cercanos puede darse el caso de que el alumno no pueda dedicarle tiempo al proyecto durante varios días o semanas consecutivos.			
Consecuencia			
Retraso grande con respecto a la planificación.			
Solución			
Replanificación de la parte restante del proyecto.			

Tabla 1 - Riesgo 01: Falta de tiempo prolongada

Nombre del riesgo	R02 - Falta de tiempo puntual		
Probabilidad	Alta	**Repercusión**	Baja
Descripción			
Debido a factores externos al proyecto y sus elementos más cercanos puede darse el caso de que el alumno no pueda, durante un breve espacio de tiempo, dedicarle al proyecto el tiempo previamente estimado.			
Consecuencia			
Retraso pequeño con respecto a la planificación.			
Solución			
Aumentar en 1 hora las horas de trabajo los días posteriores hasta suplir el tiempo perdido.			

Tabla 2 - Riesgo 02: Falta de tiempo puntual

Nombre del riesgo	R03 - Fallo del hardware		
Probabilidad	Baja	**Repercusión**	Alta
Descripción			
El hardware empleado para el desarrollo del proyecto puede fallar			
Consecuencia			
Retraso con respecto al a planificación. Pérdida parcial o total del proyecto.			
Solución			
Prevención: - Obtención previa de un equipo de repuesto. - Realización periódica de copias de seguridad en dispositivos externos o en la nube. Mitigación: - Reparación de los problemas del hardware. - Sustitución del hardware. - Repetición de la parte del proyecto perdida debido al fallo. - Replanificación de la parte faltante del proyecto o aumento de la carga los días posteriores.			

Tabla 3 - Riesgo 03: Fallo del hardware

Nombre del riesgo	R04 - Desconocimiento del funcionamiento de las herramientas utilizadas		
Probabilidad	Alta	**Repercusión**	Media
Descripción			
Debido a la extensión del proyecto se emplean diversas herramientas. El alumno puede no conocer el funcionamiento total o parcial de alguna de ellas.			

Consecuencia
Retraso con respecto al a planificación. Reducción en la calidad del proyecto.
Solución
Prevención: - Añadir en la planificación etapas de familiarización con las herramientas que se decida utilizar. - Emplear herramientas ya utilizadas previamente, o similares a estas. Mitigación: - Cambio de la herramienta empleada por una con la que ya se esté familiarizado. - Replanificación de la parte faltante del proyecto, añadiendo las etapas de familiarización con las herramientas.

Tabla 4 - Riesgo 04: Desconocimiento del funcionamiento de las herramientas utilizadas

Nombre del riesgo	R05 - Cambio en los requisitos		
Probabilidad	Baja	**Repercusión**	Alta
Descripción			
Los objetivos del TFG no pueden modificarse, pero sí los requisitos concretos acordados para alcanzar dichos objetivos.			
Consecuencia			
Retraso con respecto al a planificación. Desechar parte del proyecto. Modificar parte del proyecto.			
Solución			
Mitigación: - Replanificación de la parte faltante del proyecto junto con los nuevos requisitos. - Investigar posibilidad de reutilización o modificación de las partes ya creadas.			

Tabla 5 - Riesgo 05: Cambio en los requisitos

Nombre del riesgo	R06 - Cambio en los modelos		
Probabilidad	Media	**Repercusión**	Media
Descripción			
Durante la implementación o un posterior diseño puede darse el caso de que sea necesario cambiar alguno de los modelos previamente diseñados para cubrir áreas no cubiertas o para facilitar el posterior desarrollo.			
Consecuencia			
Retraso con respecto al a planificación. Modificar parte del proyecto.			
Solución			

Mitigación: - Replanificación de la parte faltante del proyecto incluyendo los nuevos modelos. - Investigar posibilidad de reutilización o modificación de las partes ya creadas.

Tabla 6 - Riesgo 06: Cambio en los modelos

Nombre del riesgo	R07 - Mala planificación		
Probabilidad	Media	**Repercusión**	Alta
Descripción			
Una mala previsión del tiempo que puede requerir una tarea que se descubre durante la realización de la propia tarea.			
Consecuencia			
Retraso con respecto al a planificación. Reducción de la calidad del proyecto.			
Solución			
Mitigación: Replanificación de la parte faltante del proyecto incluyendo las nuevas mediciones.			

Tabla 7 - Riesgo 07: Mala planificación

3.2 Planificación del proyecto

Basándose en el hecho de que el Trabajo de Fin de Grado equivale a 12 ECTS (por sus siglas en inglés *European Credit Transfer and Accumulation System*) y que cada uno de estos créditos equivale a entre y 25 y 30 horas de trabajo, se puede aproximar el total de la carga del trabajo del TFG a entre 300 y 360 horas. [8]

3.2.1 Primera planificación

En un primer momento la duración del proyecto se estimó en 28 semanas, lo que equivaldría a destinar cada día, de lunes a viernes, algo más de 2 horas (suponiendo una duración del proyecto de 300 horas). La primera planificación realizada se asemejaba más a un esquema que a la planificación de un proyecto software, puesto que ni siquiera estaban considerados las holguras de las tareas ni estas estaban prácticamente desgranadas.

	Nombre de la tarea	**Duración**	**Comienzo**	**Fin**	**Predecesoras**
1.1	Análisis y elicitación de los requisitos del sistema	7 días	mar 12/12/17	mié 20/12/17	
1.2	Diseño del esqueleto del modelo del sistema	10 días	mié 20/12/17	mié 03/01/18	1.1
1.3	Familiarización con el servidor	10 días	jue 04/01/18	mié 17/01/18	1.2
1.4	Selección del lenguaje o lenguajes de programación	10 días	mié 17/01/18	mié 31/01/18	1.3

	Nombre de la tarea	Duración	Comienzo	Fin	Predecesoras
1.5	Desarrollo de la base de datos	10 días	mié 31/01/18	mié 14/02/18	1.4
1.6	Desarrollo de la lógica de negocio del bot	10 días	jue 15/02/18	jue 01/03/18	1.4
1.7	Diseño de la interfaz web del sistema	10 días	jue 01/03/18	mié 14/03/18	1.4
1.8	Desarrollo de las conexiones a la base de datos	10 días	jue 15/03/18	mié 28/03/18	1.5
1.9	Conexión del bot con la base de datos	10 días	jue 29/03/18	mié 11/04/18	1.6 1.7
1.10	Conexión de la interfaz web con el bot	10 días	jue 26/04/18	mié 09/05/18	1.7
1.11	Desarrollo de la interfaz web del sistema	10 días	jue 12/04/18	mié 25/04/18	1.9
1.12	Recopilación de ideas para una posible mejora	10 días	jue 10/05/18	mié 23/05/18	1.8 1.10 1.11
1.13	Implementación de las mejoras	10 días	jue 24/05/18	mié 06/06/18	12
1.14	Comprobación de su funcionamiento	11 días	lun 11/06/18	dom 24/06/18	13

Tabla 8 – Relación de tareas de la primera planificación

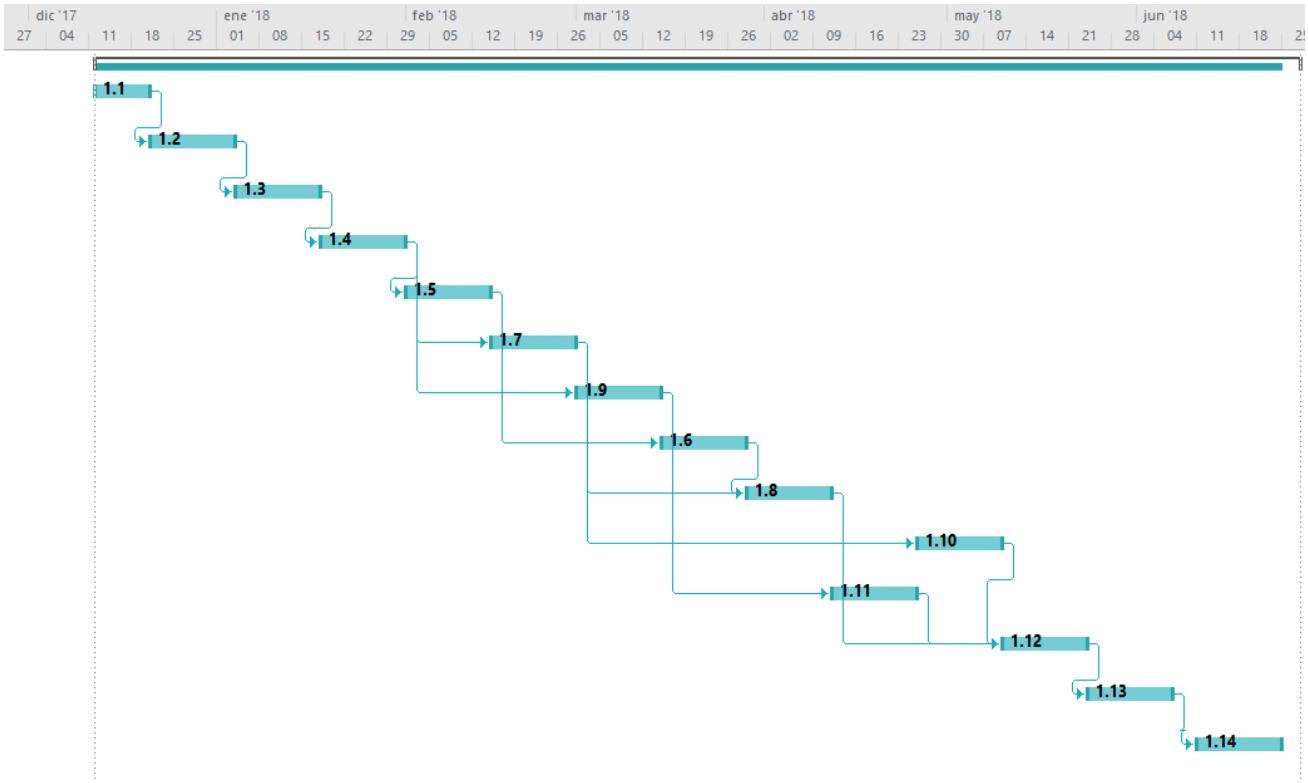

Ilustración 19 - Diagrama de Gantt de la primera planificación

3.2.2 Segunda planificación

Debido a los exámenes de la universidad y a las Prácticas en Empresa con horario completo, el inicio de las tareas se retrasó desde el 12 de diciembre hasta el 28 de marzo. En ese momento se volvió a ajustar la planificación, teniendo que dedicar cada día algo más de 4 horas y media. También se añadió un desglose mayor de las actividades y una reordenación en algunas de las ya existentes. Se puede

apreciar que varios de los puntos existentes en la primera planificación no se encontrarán ya presentes, pues se concluyó que no eran necesarios para el desarrollo de este proyecto.

	Nombre de la tarea	Duración	Comienzo	Fin	Predecesoras
2.1	Análisis y elicitación de los requisitos del sistema	11 días	mié 28/03/18	mié 11/04/18	
2.2	Diseño del modelo de arquitectura	5 días	jue 19/04/18	mié 25/04/18	2.1
2.3	Diseño del modelo de dominio	5 días	jue 26/04/18	mié 02/05/18	2.2
2.4	Análisis de la API de Telegram	5 días	jue 12/04/18	mié 18/04/18	
2.5	Diseño de los modelos de interacción	4 días	jue 03/05/18	mié 09/05/18	2.3 2.4
2.6	Elección del lenguaje o lenguajes de programación	4 días	mié 09/05/18	mar 15/05/18	2.5
2.7	Elección del lenguaje para realizar las consultas a la base de datos	4 días	lun 21/05/18	jue 24/05/18	2.6
2.8	Implementación de la parte de almacenamiento	4 días	mié 06/06/18	lun 11/06/18	2.7
2.9	Implementación de la lógica del bot	4 días	vie 25/05/18	jue 31/05/18	2.6
2.10	Diseño de la interfaz del bot	4 días	mar 15/05/18	lun 21/05/18	2.5
2.11	Desarrollo de la interfaz del bot	4 días	jue 31/05/18	mié 06/06/18	2.6 2.10
2.12	Conectar la lógica del bot con el almacenamiento	4 días	mar 12/06/18	lun 18/06/18	2.8 2.9
2.13	Conectar la interfaz del bot con la lógica	4 días	lun 18/06/18	jue 21/06/18	2.9 2.11
2.14	Realizar pruebas con usuarios reales	3 días	vie 22/06/18	mar 26/06/18	2.12 2.13

Tabla 9 - Relación de tareas de la segunda planificación

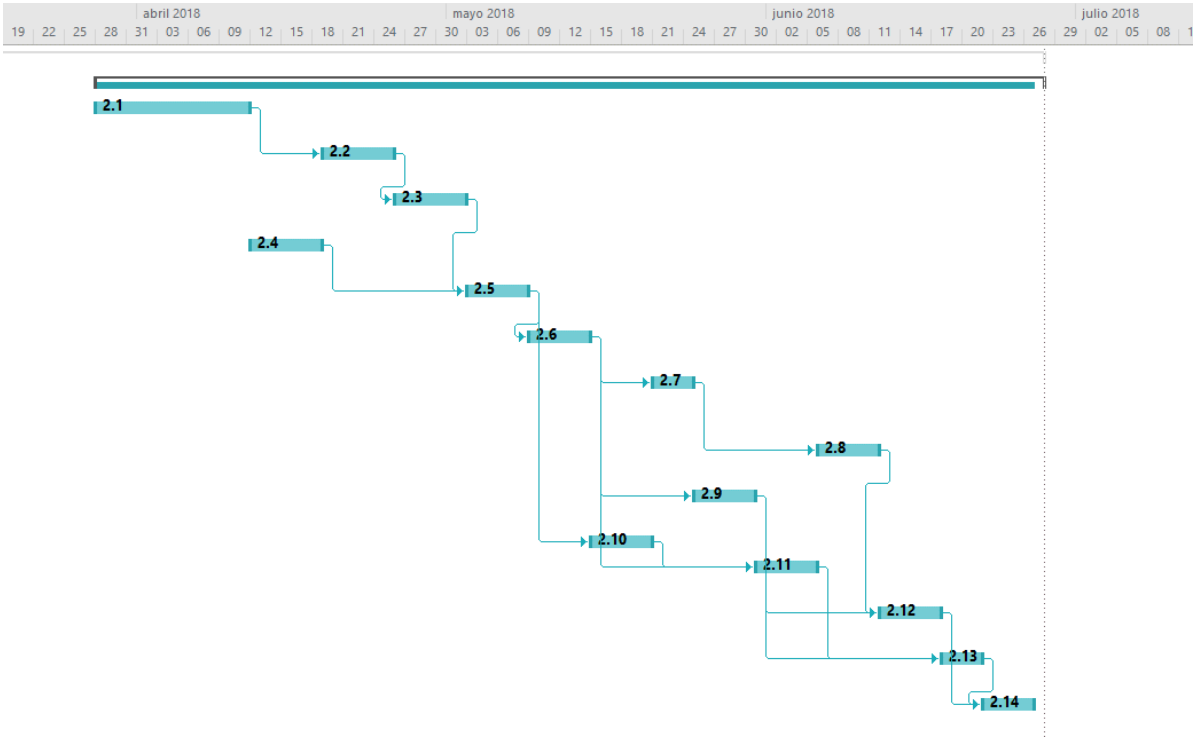

Ilustración 20 - Diagrama de Gantt de la segunda planificación

3.2.3 Tercera planificación

Entre el 8 y el 18 de mayo el desarrollo del proyecto se volvió a pausar, lo que supuso una replanificación de los días restantes. En este momento se decidió completar la memoria al final del proyecto. Debido a estos días sin poder trabajar en el proyecto, en la siguiente planificación cada día a partir del 18 de mayo representa alrededor de 6 horas y media de trabajo.

	Nombre de la tarea	Duración	Comienzo	Fin	Predecesoras
3.1	Análisis y elicitación de los requisitos del sistema	11 días	mié 28/03/18	mié 11/04/18	
3.2	Diseño del modelo de arquitectura	5 días	jue 19/04/18	mié 25/04/18	3.1
3.3	Diseño del modelo de dominio	5 días	jue 26/04/18	mié 02/05/18	3.2
3.4	Análisis de la API de Telegram	5 días	jue 12/04/18	mié 18/04/18	
3.5	Diseño de los modelos de interacción	5 días	jue 03/05/18	vie 18/05/18	3.3 3.4
3.6	Elección del lenguaje o lenguajes de programación	4 días	sáb 19/05/18	mié 23/05/18	3.5
3.7	Elección del lenguaje para realizar las consultas a la base de datos	3 días	mié 30/05/18	vie 01/06/18	3.6
3.8	Implementación de la parte de almacenamiento	2 días	mar 12/06/18	mié 13/06/18	3.7
3.9	Implementación de la lógica del bot	2 días	mar 05/06/18	mié 06/06/18	3.6
3.10	Diseño de la interfaz del bot	3 días	jue 24/05/18	lun 28/05/18	3.5
3.11	Desarrollo de la interfaz del bot	2 días	vie 08/06/18	lun 11/06/18	3.6 3.10

	Nombre de la tarea	Duración	Comienzo	Fin	Predecesoras
3.12	Conectar la lógica del bot con el almacenamiento	2 días	vie 15/06/18	lun 18/06/18	3.8 3.9
3.13	Conectar la interfaz del bot con la lógica	2 días	mar 19/06/18	mié 20/06/18	3.9 3.11
3.14	Realizar pruebas con usuarios reales	2 días	jue 21/06/18	vie 22/06/18	3.12 3.13
3.15	Realización de la memoria	2 días	dom 24/06/18	lun 25/06/18	14

Tabla 10 - Relación de tareas de la tercera planificación

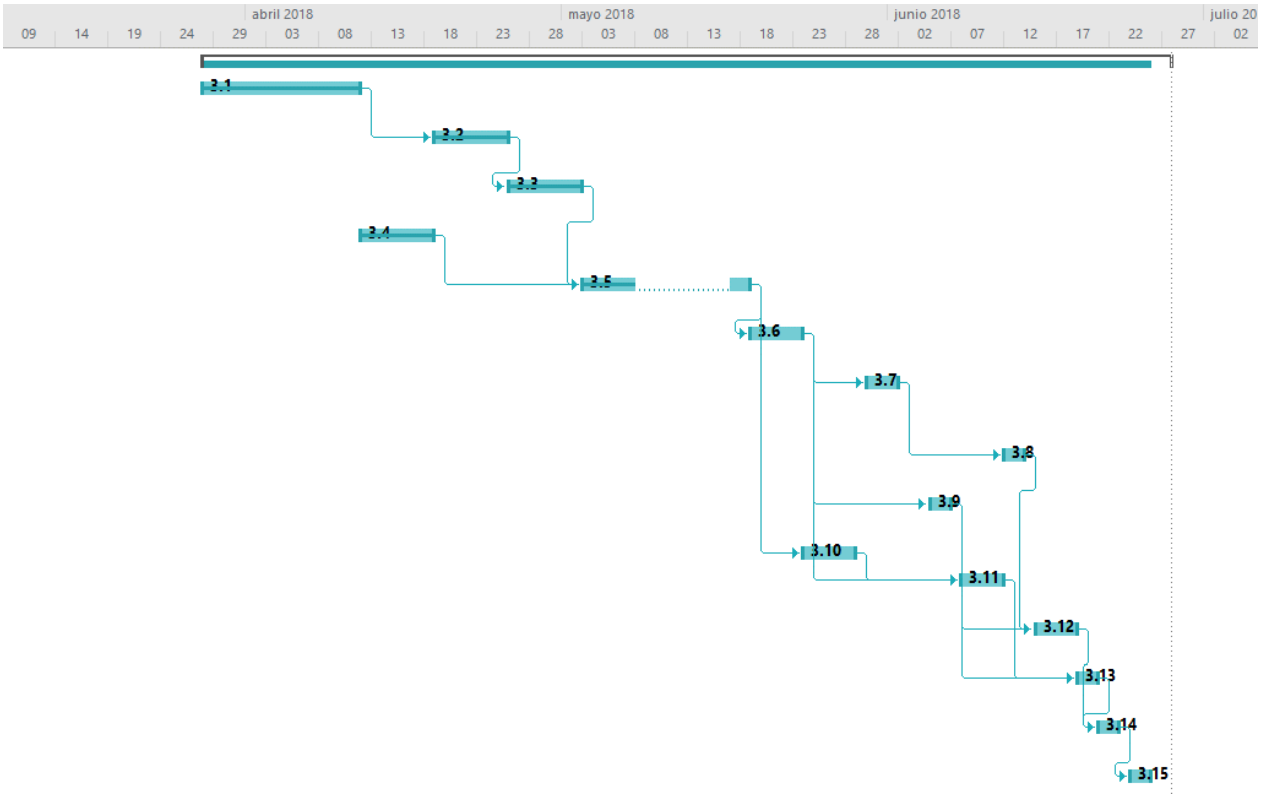

Ilustración 21 - Diagrama de Gantt de la tercera planificación

3.2.4 Cuarta planificación

Tras una reunión con el tutor el 15 de junio, se acuerda añadir algunas funcionalidades al proyecto y se llega a la conclusión de que no hay tiempo suficiente para realizar la memoria antes del plazo límite fijado, por lo que se expande hasta el día 15 de julio, con la clara condición de disponer de la memoria el día 11, y se vuelve a planificar la parte restante. Para ese momento las únicas partes restantes eran la memoria y la prueba con los usuarios, puesto que el resto de las tareas habían consumido menos tiempo del esperado. Se decide eliminar la parte de prueba con usuarios reales para centrarse en añadir las funcionalidades y redactar la memoria.

La planificación inicial tenía en cuenta una memoria de menor envergadura que la actual, por lo que para la redacción de esta se incluirán las 60 horas de diferencia (por la representación en créditos el TFG oscila entre las 300 y 360 horas) que no se tenían en cuenta anteriormente. Desde el 15 de junio, la carga de cada día es de 5 horas.

	Nombre de la tarea	Duración	Comienzo	Fin	Predecesoras
4.1	Análisis y elicitación de los requisitos del sistema	11 días	mié 28/03/18	mié 11/04/18	
4.2	Diseño del modelo de arquitectura	5 días	jue 19/04/18	mié 25/04/18	4.1
4.3	Diseño del modelo de dominio	5 días	jue 26/04/18	mié 02/05/18	4.2
4.4	Análisis de la API de Telegram	5 días	jue 12/04/18	mié 18/04/18	
4.5	Diseño de los modelos de interacción	5,5 días	jue 03/05/18	vie 18/05/18	4.3 4.4
4.6	Elección del lenguaje o lenguajes de programación	4 días	sáb 19/05/18	mié 23/05/18	4.5
4.7	Elección del lenguaje para realizar las consultas a la base de datos	3 días	mar 29/05/18	jue 31/05/18	4.6
4.8	Desarrollo de la interfaz del bot	3 días	dom 03/06/18	mar 05/06/18	4.7
4.9	Implementación de la lógica del bot	2 días	vie 01/06/18	sáb 02/06/18	4.6
4.10	Implementación de la parte de almacenamiento	2 días	sáb 09/06/18	dom 10/06/18	4.5
4.11	Diseño de la interfaz del bot	3 días	jue 24/05/18	lun 28/05/18	4.6 4.10
4.12	Conectar la lógica del bot con el almacenamiento	2 días	lun 11/06/18	mar 12/06/18	4.8 4.9
4.13	Conectar la interfaz del bot con la lógica	2 días	mié 06/06/18	vie 08/06/18	4.9 4.11
4.14	Realización de la memoria	19 días	mar 12/06/18	lun 09/07/18	4.12 4.13
4.15	Modificación del bot y la base de datos para que pueda trabajar con ficheros	2 días	vie 13/07/18	dom 15/07/18	4.14
4.16	Familiarización con PHP	1 día	mar 10/07/18	mar 10/07/18	4.14
4.17	Diseño de la extensión para Moodle	1 día	mié 11/07/18	mié 11/07/18	4.16
4.18	Implementación de la extensión para Moodle	1 día	jue 12/07/18	jue 12/07/18	4.17

Tabla 11 - Relación de tareas de la cuarta planificación

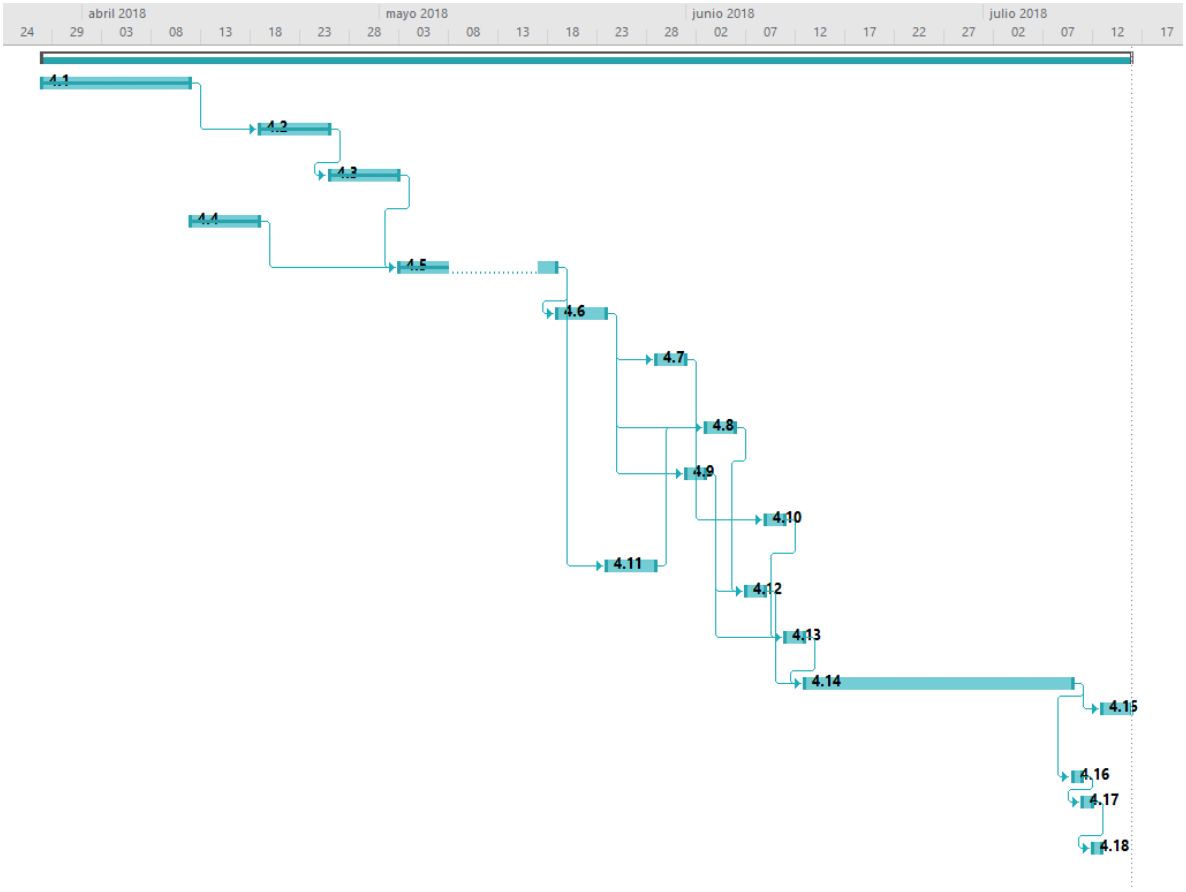

Ilustración 22 - Diagrama de Gantt de la cuarta planificación

3.3 Presupuesto

Aunque el coste salarial del proyecto sea nulo, así como también puede serlo el coste de recursos distintos a la mano de obra debido a que la Universidad o la Escuela puedan proporcionarlos para la realización del Trabajo de Fin Grado, es recomendable estimar el coste que tendría el desarrollo del proyecto en el mercado actual. Esto es útil tanto para mantenerse informado de la situación laboral del sector en el que el alumno se moverá como para ser consciente de la inversión que supondría la realización del trabajo fuera del entorno académico.

Debido a la naturaleza de este proyecto es posible emplear herramientas gratuitas, por lo que el único coste sería el de la mano de obra. Suponiendo un sueldo mensual de 2500 @ para un ingeniero informático, podemos estimar la cuantía que habría recibido dicho profesional durante el desarrollo de este proyecto. En España la jornada laboral se estima en 40 horas semanales y se supondrá que cada mes consta de 4 semanas, para simplificar los cálculos. De ello se obtiene:

$$\frac{2500\ €/mes}{4\ semanas/mes} = 625\ €/semana$$

$$\frac{625\ €/semana}{40\ horas/semana} \cong 15{,}625\ €/hora$$

Si se pretende mantener el servicio en funcionamiento más allá del ámbito del Trabajo de Fin de Grado, sería necesario estimar el coste de los servidores necesarios para mantener la aplicación en

ejecución y el coste del almacenamiento que requeriría. Este tema se abordará más adelante en esta memoria.

3.3.1 Estimación inicial

En la primera planificación se estimó una duración para el proyecto de 300 horas.

$$15{,}625\ {}^{€}/_{hora} \times 300\ horas = 4687{,}5\ €$$

El sueldo percibido por el trabajador por este proyecto sería de unos 2019 €. En España, el sueldo del empleado representa el 55% del coste que le supone al empleador [9], por lo que puede asumirse que el coste para este, sin tener en cuenta el resto de las tasas a las que se encuentre sujeto, sería de $4687{,}5\ €/0{,}55 \cong 8522{,}72\ €$.

3.3.2 Estimación final

Tras la realización del proyecto, el tiempo empleado ha sido de 360 horas.

$$15{,}625\ €/hora \times 360\ horas = 5625\ €$$

El sueldo percibido por el trabajador por este proyecto sería de unos 2423,25 €. De la misma manera que en la estimación inicial, el coste para el empleador, sin tener en cuenta el resto de las tasas a las que se encuentre sujeto, sería de $5625\ €/0{,}55 \cong 10227{,}27\ €$.

Capítulo 4. Análisis y diseño

4.1 Requisitos

Más allá de los objetivos del Trabajo de Fin de Grado, el proyecto ha de poseer unos requisitos formales que permitan comprobar de una manera objetiva que cumple las funcionalidades para las que ha sido diseñado.

4.1.1 Requisitos funcionales

Los requisitos funcionales de la aplicación se corresponden con las funcionalidades y los servicios que tiene que ofrecer. Son los siguientes:

- El sistema deberá permitir crear listas.
- El sistema deberá permitir que una lista tenga anidadas a la vez listas y entradas, así como solo listas o solo entradas.
- El sistema deberá permitir que una lista no tenga nada anidado.
- El sistema deberá permitir a los usuarios con los permisos pertinentes ver las listas o entradas correspondientes a dichos permisos.
- El sistema deberá permitir a los usuarios con los permisos pertinentes editar las listas o entradas correspondientes a dichos permisos
- El sistema deberá permitir a los usuarios con los permisos pertinentes gestionar para cada lista o entrada qué usuarios pueden verla.
- El sistema deberá permitir a los usuarios con los permisos pertinentes gestionar para cada lista o entrada qué usuarios pueden editarla.
- El sistema deberá permitir a los usuarios con los permisos pertinentes crear listas anidadas en otra lista.
- El sistema deberá permitir a los usuarios con los permisos pertinentes crear entradas anidadas en una lista.
- El sistema deberá permitir a los usuarios con los permisos pertinentes vincular listas a cursos en Moodle.
- El sistema deberá permitir a los usuarios con los permisos pertinentes añadir entradas en el árbol de la lista correspondiente a un curso desde la página de dicho curso en Moodle.

4.1.2 Requisitos no funcionales

Los requisitos no funcionales de la aplicación hacen referencia a la manera de implementar los requisitos funcionales. Son los siguientes:

- El usuario podrá interactuar con el bot a través de una interfaz gráfica.
- El bot se implementará con el lenguaje de programación Python, usando la versión 3.
- El bot empleará la API “Python Telegram Bot”.
- El almacenamiento se realizará en una base de datos relacional.
- Se utilizará MySQL para acceder a la base de datos.

4.1.3 Requisitos de información

Los requisitos de información de la aplicación aportan aclaraciones sobre los datos que almacenará la aplicación para su correcto funcionamiento.

- Las listas constarán de un título, una descripción y una cantidad de elementos anidados, que podrán ser tanto listas como entradas.
- Las entradas constarán de un título, una descripción y un valor, que puede tener significado por sí mismo o ser una referencia a un fichero.

4.2 Casos de uso

Los casos de uso describen las operaciones que pueden realizar los usuarios con la aplicación. Se forman a partir de los requisitos funcionales y se emplean para explicar la secuencia de mensajes que intercambian el usuario y el sistema a la hora de realizar alguna tarea.

Con respecto a la aplicación, solo existe un tipo de usuario. Aunque un usuario no tenga los permisos necesarios para poder ver o editar una lista o entrada concreta, eso no significa que no pueda tener esos derechos para otras listas. Todos los usuarios disponen de las mismas funcionalidades.

Los casos de uso pueden dividirse en 4 secciones:

- Gestión del propio usuario: registrar al propio usuario en el sistema para poder interactuar con el bot y eliminar al propio usuario del sistema para que el bot deje de interactuar con el usuario.
- Gestión de listas: creación, modificación, eliminación y consulta de listas sobre las que se tenga permisos.
- Gestión de entradas: creación, modificación, eliminación y consulta de entradas sobre las que se tenga permisos.
- Gestión de permisos: creación, modificación y consulta de permisos siempre que se tenga la potestad para ello.

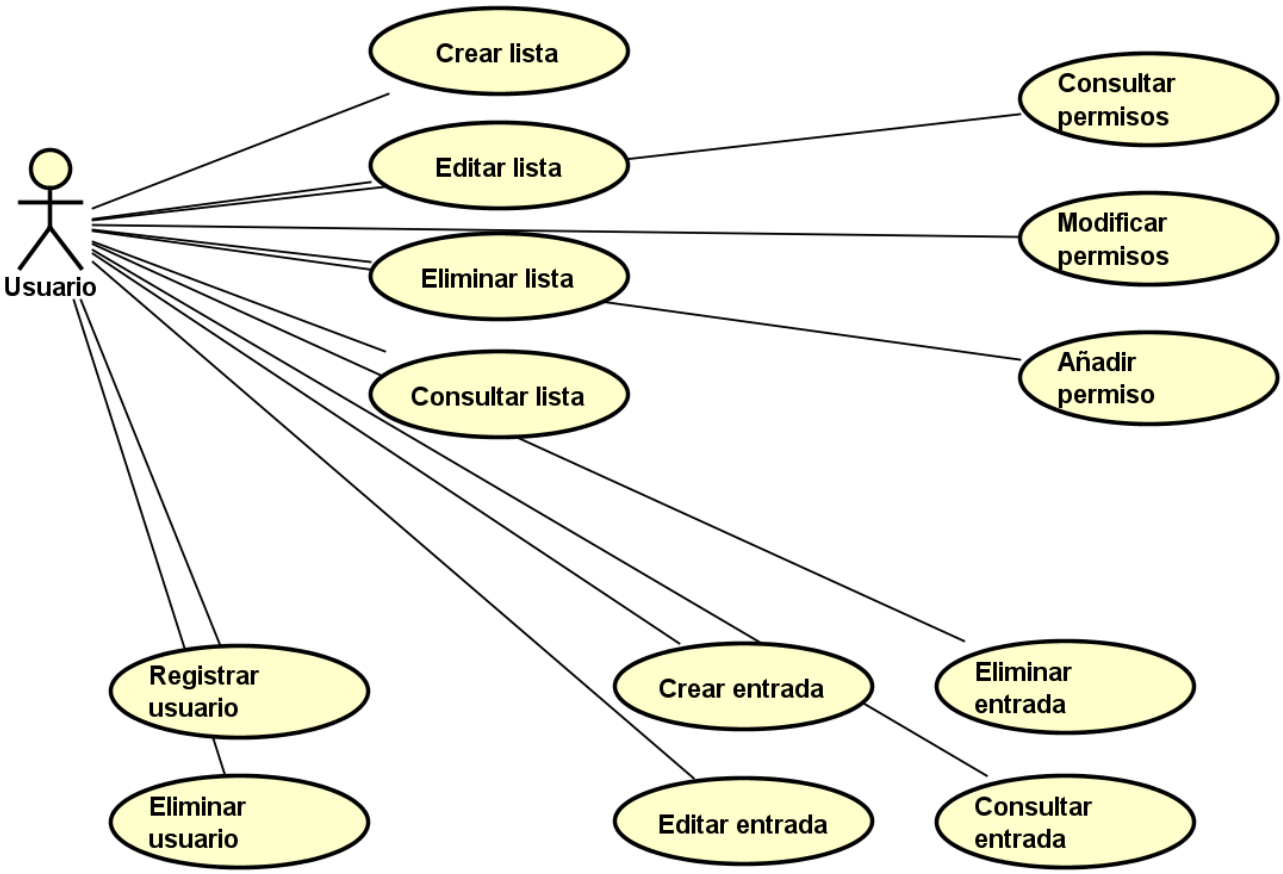

Ilustración 23 - Diagrama de casos de uso

4.2.1 Descripción de los casos de uso

A continuación, se describen los casos de uso presentes en este proyecto, divididos en las secciones ya mencionadas.

4.2.1.1 Gestión del propio usuario

Caso de uso	CU01 - Registrar usuario
Resumen	El sistema debe permitir a los usuarios registrar su nombre de usuario de Telegram. Con esto adquiere el derecho a poder usar todas las funcionalidades, incluida la de poder obtener permisos para acceder o editar el elemento sobre el que se ha dado permiso.
Actor	Usuario
Precondición	
Postcondición	El usuario está registrado en el sistema.
Escenario base	1. El actor Usuario solicita registrar su nombre de usuario en el sistema. 2. El sistema comprueba que el usuario no se encuentra ya registrado. 3. El sistema registra el nombre de usuario en el sistema. 4. El sistema informa al usuario de que el nombre de usuario ha sido registrado correctamente.

Escenario alternativo	
Excepciones	2.a. El usuario ya se encuentra registrado, el sistema informa al usuario y el caso de uso finaliza.

Tabla 12 - Definición del caso de uso 01: Registrar usuario

Caso de uso	CU02 - Eliminar usuario
Resumen	El sistema debe permitir a los usuarios eliminar su usuario del sistema. Con esto pierden el derecho a poder usar cualquiera de las funcionalidades del sistema, incluida la de poder obtener permisos para acceder o editar elementos.
Actor	Usuario
Precondición	El usuario está registrado en el sistema.
Postcondición	El usuario no está registrado en el sistema.
Escenario base	1. El actor Usuario solicita al sistema que elimine su nombre de usuario de la base de datos. 2. El sistema elimina el nombre de usuario de la base de datos. 3. El sistema informa al usuario de que la tarea se ha realizado correctamente.
Escenario alternativo	
Excepciones	

Tabla 13 - Definición del caso de uso 02: Eliminar usuario

4.2.1.2 Gestión de listas

Caso de uso	CU03 - Crear lista
Resumen	El sistema debe permitir a los usuarios crear listas. El sistema debe permitir a los usuarios crear listas anidadas si dispone de los permisos necesarios.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. El usuario tiene permiso para realizar esta acción.
Postcondición	Se ha creado una nueva lista.
Escenario base	1. El actor Usuario comienza el caso de uso Crear lista. 2. El sistema solicita al usuario las propiedades de la nueva lista. 3. El actor Usuario introduce las propiedades de la nueva lista. 4. El actor Usuario solicita guardar la nueva lista 5. El sistema guarda la nueva lista. 6. El sistema informa al actor Usuario de que la lista ha sido guardada.
Escenario alternativo	

Excepciones	3.a. El actor Usuario cancela el caso de uso, quedando sin acción. 4.a. El actor Usuario cancela el caso de uso, quedando sin acción. 5.a. El sistema no puede guardar los cambios, informa al usuario y continúa desde el paso 3.

Tabla 14 - Definición del caso de uso 03: Crear lista

Caso de uso	CU04 - Editar lista
Resumen	El sistema debe permitir a los usuarios editar listas si dispone de los permisos necesarios.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. El usuario tiene permiso para realizar esta acción. La lista que se desea editar existe.
Postcondición	
Escenario base	1. El actor Usuario comienza el caso de uso Editar lista. 2. El actor Usuario solicita editar las propiedades de una lista en concreto. 3. El sistema muestra los valores actuales de las propiedades editables. 4. El sistema solicita al actor Usuario que seleccione una de las propiedades. 5. El actor Usuario selecciona de las propiedades. 6. El sistema informa al usuario de la propiedad que va a modificar. 7. El actor Usuario modifica la propiedad seleccionada. 8. El actor Usuario solicita guardar los cambios. 9. El sistema guarda los cambios realizados. 10. El sistema informa al actor Usuario de que los cambios se han guardado.
Escenario alternativo	8.1. El actor Usuario selecciona una de las propiedades y el caso de uso continúa desde el paso 6.
Excepciones	5.a. El actor Usuario cancela el caso de uso, quedando sin acción. 7.a. El actor Usuario cancela el caso de uso, quedando sin acción. 8.a. El actor Usuario cancela el caso de uso, quedando sin acción. 9.a. El sistema no puede guardar los cambios, informa al usuario y continúa desde el paso 5.

Tabla 15 - Definición del caso de uso 04: Editar lista

Caso de uso	CU05 - Eliminar lista
Resumen	El sistema debe permitir a los usuarios eliminar listas si dispone de los permisos necesarios.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. El usuario tiene permiso para realizar esta acción. La lista que se desea eliminar existe.
Postcondición	La lista ha sido eliminada del sistema.
Escenario base	1. El actor Usuario inicia el caso de uso Eliminar lista. 2. El actor Usuario solicita eliminar una lista concreta. 3. El sistema elimina la lista.

	4. El sistema informa al usuario de que la lista ha sido eliminada.
Escenario alternativo	
Excepciones	3.a El sistema no puede eliminar la lista, informa al usuario y el caso de uso finaliza.

Tabla 16 - Definición del caso de uso 05: Eliminar lista

Caso de uso	CU06 - Consultar lista
Resumen	El sistema debe permitir a los usuarios consultar listas si dispone de los permisos necesarios.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. La lista que se desea consultar existe.
Postcondición	
Escenario base	El actor Usuario solicita ver la información asociada a una lista. El sistema muestra la información al actor Usuario.
Escenario alternativo	
Excepciones	2.a. Si no puede recuperarse la información sobre la lista, el sistema informa al usuario y el caso de uso finaliza.

Tabla 17 - Definición del caso de uso 06: Consultar lista

4.2.1.3 Gestión de entradas

Caso de uso	CU07 - Crear entrada
Resumen	El sistema debe permitir a los usuarios crear entradas anidadas si dispone de los permisos necesarios.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. El usuario tiene permiso para realizar esta acción.
Postcondición	Se ha creado una nueva entrada.
Escenario base	El actor Usuario comienza el caso de uso Crear entrada. El sistema solicita al usuario las propiedades de la nueva entrada. El actor Usuario introduce las propiedades de la nueva entrada. El actor Usuario solicita guardar la nueva entrada. El sistema guarda la nueva entrada. El sistema informa al actor Usuario de que la entrada ha sido guardada.
Escenario alternativo	

Excepciones	3.a. El actor Usuario cancela el caso de uso, quedando sin acción. 4.a. El actor Usuario cancela el caso de uso, quedando sin acción. 5.a. El sistema no puede guardar los cambios, informa al usuario y continúa desde el paso 3.

Tabla 18 - Definición del caso de uso 07: Crear entrada

Caso de uso	CU08 - Editar entrada
Resumen	El sistema debe permitir a los usuarios editar entradas si dispone de los permisos necesarios.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. El usuario tiene permiso para realizar esta acción. La entrada que se desea editar existe.
Postcondición	
Escenario base	1. El actor Usuario comienza el caso de uso Editar entrada. 2. El actor Usuario solicita editar las propiedades de una entrada en concreto. 3. El sistema muestra los valores actuales de las propiedades editables. 4. El sistema solicita al actor Usuario que seleccione una de las propiedades. 5. El actor Usuario selecciona de las propiedades. 6. El sistema informa al usuario de la propiedad que va a modificar. 7. El actor Usuario modifica la propiedad seleccionada. 8. El actor Usuario solicita guardar los cambios. 9. El sistema guarda los cambios realizados. 10. El sistema informa al actor Usuario de que los cambios se han guardado.
Escenario alternativo	8.1. El actor Usuario selecciona una de las propiedades y el caso de uso continúa desde el paso 6.
Excepciones	5.a. El actor Usuario cancela el caso de uso, quedando sin acción. 7.a. El actor Usuario cancela el caso de uso, quedando sin acción. 8.a. El actor Usuario cancela el caso de uso, quedando sin acción. 9.a. El sistema no puede guardar los cambios, informa al usuario y continúa desde el paso 5.

Tabla 19 - Definición del caso de uso 08: Editar entrada

Caso de uso	CU09 - Eliminar entrada
Resumen	El sistema debe permitir a los usuarios eliminar entradas si dispone de los permisos necesarios.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. El usuario tiene permiso para realizar esta acción. La entrada que se desea eliminar existe.
Postcondición	La lista ha sido eliminada del sistema.

Escenario base	1. El actor Usuario inicia el caso de uso Eliminar entrada. 2. El actor Usuario solicita eliminar una entrada concreta. 3. El sistema elimina la entrada. 4. El sistema informa al usuario de que la entrada ha sido eliminada.
Escenario alternativo	
Excepciones	3.a El sistema no puede eliminar la entrada, informa al usuario y el caso de uso finaliza.

Tabla 20 - Definición del caso de uso 09: Eliminar entrada

Caso de uso	CU10 - Consultar entrada
Resumen	El sistema debe permitir a los usuarios consultar entradas si dispone de los permisos necesarios.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. El usuario tiene permiso para realizar esta acción. La entrada que se desea consultar existe.
Postcondición	
Escenario base	El actor Usuario solicita ver la información asociada a una entrada. El sistema muestra la información al actor Usuario.
Escenario alternativo	
Excepciones	2.a. Si no puede recuperarse la información sobre la entrada, el sistema informa al usuario y el caso de uso finaliza.

Tabla 21- Definición del caso de uso 10: Consultar entrada

4.2.1.4 Gestión de permisos

Caso de uso	CU11 - Añadir permiso
Resumen	El sistema debe permitir dar permisos a otros usuarios sobre las listas o entradas siempre que se tenga la capacidad para ello.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. El usuario tiene permiso para realizar esta acción. La lista o entrada de la que se desean dar permisos existe. El usuario al que se quiere dar permisos está registrado en el sistema.
Postcondición	Se ha creado y almacenado un nuevo permiso.

Escenario base	1. El actor Usuario solicita añadir un nuevo permiso a una lista o entrada. 2. El sistema solicita un identificador del usuario al que se quiere dar permisos y los permisos que se desea darle. 3. El actor Usuario proporciona los datos solicitados. 4. El actor Usuario solicita guardar los nuevos permisos. 5. El sistema almacena los nuevos permisos del usuario proporcionado. 6. El sistema informa al actor Usuario
Escenario alternativo	
Excepciones	3.a. El actor Usuario cancela el caso de uso, quedando sin acción. 4.a. El actor Usuario cancela el caso de uso, quedando sin acción. 5.a. El sistema no puede guardar los nuevos permisos, informa al usuario y el caso de uso continúa en el paso 3.

Tabla 22- Definición del caso de uso 11: Añadir permiso

Caso de uso	CU12 - Modificar permiso
Resumen	El sistema debe permitir modificar los permisos de otros usuarios sobre las listas o entradas siempre que se tenga la capacidad para ello. Si se retira el permiso de consulta equivaldría a eliminar el permiso.
Actor	Usuario
Precondición	El usuario está registrado en el sistema. El permiso que se quiere modificar existe en el sistema. El usuario tiene permiso para realizar esta acción.
Postcondición	
Escenario base	1. El actor Usuario inicia el caso de uso Modificar permiso. 2. El sistema muestra los permisos que el actor Usuario tiene permiso para modificar. 3. El actor Usuario modifica los permisos. 4. El actor Usuario solicita guardar los cambios. 5. El sistema guarda los cambios. 6. El sistema notifica al usuario de que los cambios han sido guardados.
Escenario alternativo	
Excepciones	3.a. El actor Usuario cancela el caso de uso, quedando sin acción. 4.a. El actor Usuario cancela el caso de uso, quedando sin acción. 5.a. El sistema no puede guardar los cambios, informa al usuario y el caso de uso continúa en el paso 2.

Tabla 23 - Definición del caso de uso 12: Modificar permiso

Caso de uso	CU13 - Consultar permisos
Resumen	El sistema permitirá consultar los diferentes tipos de permisos sobre una lista o entrada y qué usuarios los poseen.
Actor	Usuario

Precondición	El usuario está registrado en el sistema. El usuario tiene permiso para realizar esta acción.
Postcondición	
Escenario base	1. El actor Usuario solicita ver los permisos que conciernen a una lista. 2. El sistema muestra los usuarios que tienen algún permiso sobre la lista deseada y qué permisos tienen.
Escenario alternativo	
Excepciones	2.a. Si no puede recuperarse la información sobre los permisos, el sistema informa al usuario y el caso de uso finaliza.

Tabla 24 - Definición del caso de uso 13: Consultar permisos

4.3 Modelo de dominio

El modelo de dominio es un diagrama que representa los elementos presentes dentro del sistema y cómo se relacionan entre ellos.

En el primer diagrama se ve refleja la estructura de datos diseñada para dar soporte al sistema de listas y entradas. Sobre el diagrama:

- La clase Estructura posee los atributos comunes a las listas y a las entradas. Además, posee dos tipos de relación consigo misma:
    - Padre: cada estructura puede tener a lo sumo un padre, que sería la lista en la que se encuentra anidada.
    - Hijos: cada estructura puede tener cero o más hijos siempre que sea una lista y no una entrada. Son las listas y entradas anidadas en la propia lista.
- Como expresa la sentencia OCL, si un objeto del tipo Estructura es a su vez una Entrada, dicho objeto tiene que tener un padre y no puede tener hijos. Esto es así porque en las entradas es donde se almacena la información que no tiene que ver con la jerarquía de las listas (ficheros o texto); son los nodos hoja de los árboles que representan este sistema de listas.
- Dependiendo de qué haya puesto el usuario como valor a la hora de crear una entrada, esta puede almacenar dos tipos de dato:
    - Texto: cadena de texto. Puede ser una url, un pasaje de un libro o cualquier tipo de información que pueda compartirse a través de una cadena de texto.
    - Identificador de fichero: identificador único del fichero subido por el usuario. Como Telegram almacena en sus servidores cualquier fichero enviado y permite solicitarlo a través de su identificador, almacenar el identificador permite disponer del fichero sin necesidad de guardarlo en un servidor propio.
- El texto podría emplearse también para almacenar el identificador, pero diferenciar entre un identificador y una cadena de texto normal podría ser complicado. También influye el almacenamiento, pues especificar el tipo del atributo es esencial a la hora de almacenar información en una base de datos.

- Cada estructura se relaciona con cada usuario a través de los permisos que este posee con respecto a la estructura. La mejor manera de representar esta relación es con la clase asociativa Permiso, que posee una referencia tanto a la estructura como al usuario que relaciona. Los campos de clase representan los permisos que puede tener el usuario con referencia a la estructura con la que se relaciona.
- De cada usuario se almacena:
    - Identificador: es único y es lo que se utiliza para compararlos.
    - Un texto comprensivo para los usuarios que les permita identificarse entre ellos. El identificador es un entero mientras que este campo tendrá el valor del nombre de usuario o la combinación nombre y apellido que tenga cada usuario. Son con estos valores con los que se trata normalmente en la plataforma.

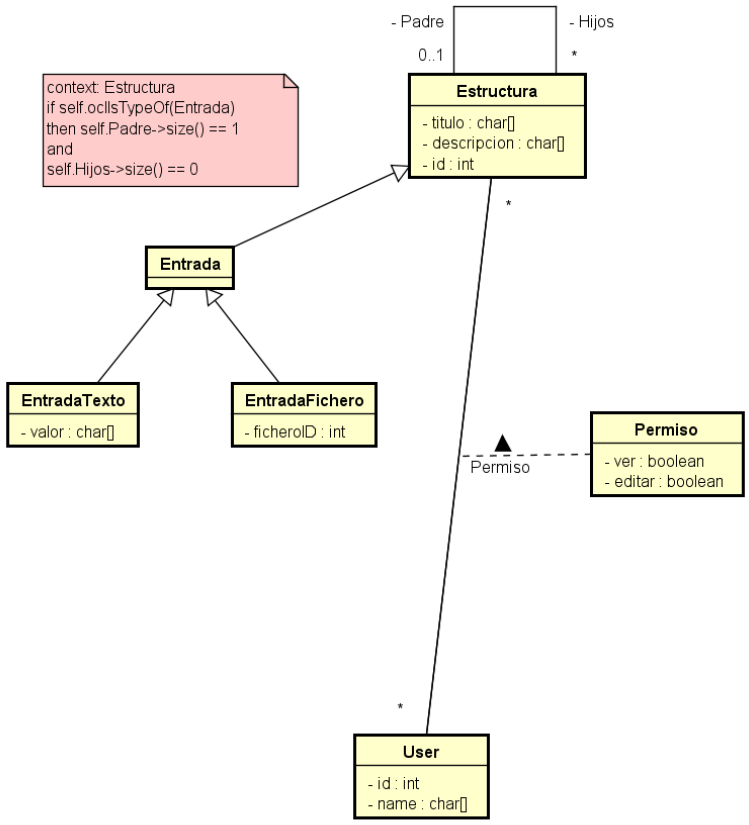

Ilustración 24 - Modelo de dominio del sistema de listas y entradas

Sin embargo, para el funcionamiento de la aplicación son necesarias más clases parar poder seguir el rastro de en qué estado de la interacción se encuentra cada usuario. Para ello se añade la clase Estado, que almacena el identificador del mensaje que se utiliza como interfaz gráfica También se relaciona con la Estructura representada en la vista en la que se encuentre el usuario y con el propio usuario que se encuentra en esa vista.

El último punto remarcable de la propia clase Estado es el hecho de que guarda una referencia a otro objeto de su misma clase. Este objeto será el Estado que haya originado al actual y se utilizará para permitir una navegación hacia niveles superiores de manera sencilla

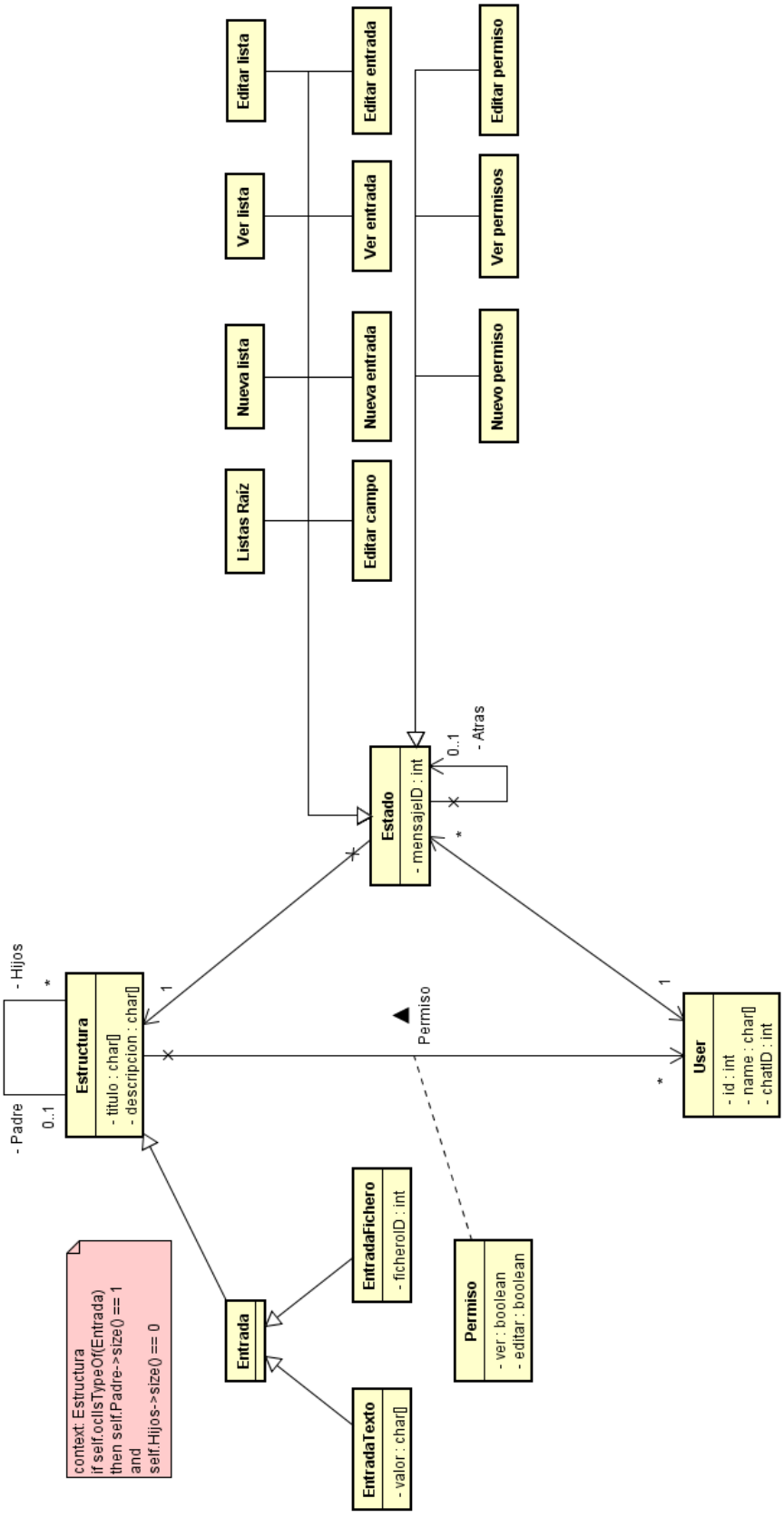

Ilustración 25 - Modelo de dominio de la aplicación

Algunas de las clases que heredan de estado se relacionan entre ellas o poseen algún atributo extra para poder llevar a cabo su cometido, pero la relación entre los estados se verá más adelante y los detalles, casi de implementación, no son relevantes en este punto. Es útil almacenar el identificador del chat que emplea el usuario para comunicarse con el bot para evitar posibles incongruencias en caso de añadir el bot a grupos de Telegram.

4.4 Máquina de estados

Debido a la decisión de realizar una interfaz gráfica para facilitar el uso del bot la solución más sencilla es implementar una máquina de estados que permita al usuario moverse desde una vista a otra. Además, de esta manera se puede agrupar de una manera natural la funcionalidad básica de todas las vistas y después modificar o ampliar cada vista en concreto dependiendo de sus necesidades.

En el diagrama se muestran únicamente los mensajes recibidos a través de la interfaz gráfica.

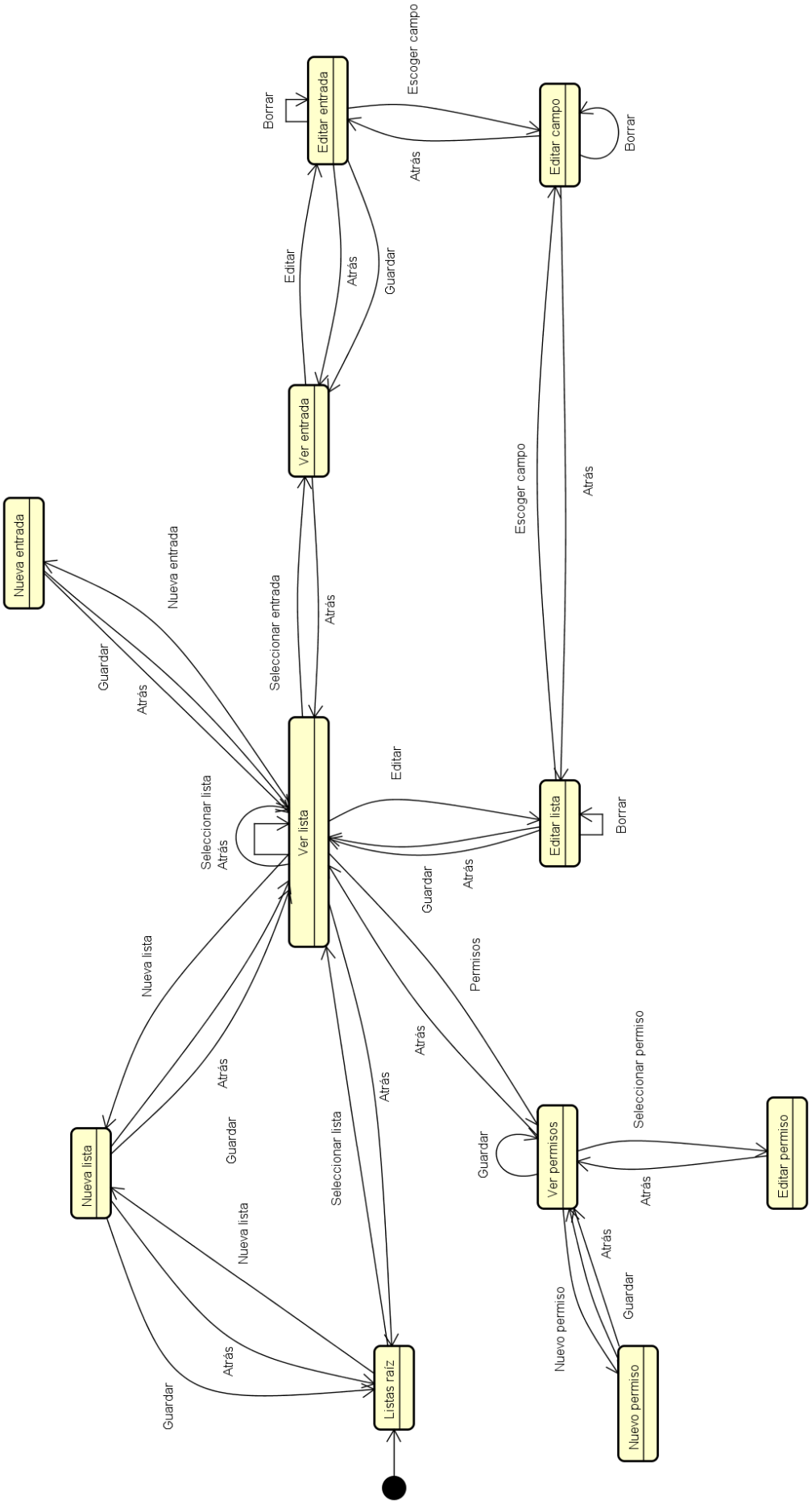
Editar entrada
Editar campo
Ver entrada
Nueva entrada
Ver lista
Editar lista
Nueva lista
Ver permisos
Editar permiso
Listas raíz
Nuevo permiso
Borrar
Escoger campo
Atrás
Editar
Guardar
Seleccionar entrada
Nueva entrada
Seleccionar lista
Permisos
Seleccionar permiso
Nueva lista
Nuevo permiso

Ilustración 26 - Máquina de estados que representa la relación entre las distintas vistas

4.5 Diagramas de interacción

Así como se ha definido la secuencia de intercambio de mensajes entre la aplicación y el usuario en la descripción de los casos de uso, con los diagramas de interacción se acoplarán dichas secuencias al modelo de dominio.

Antes de comenzar con los diagramas, unas aclaraciones sobre esta parte de la memoria:

- Se mostrarán únicamente los diagramas más representativos, pues hay varios cuyo desarrollo es muy similar.
- Debido al método utilizado para recibir los mensajes de los usuarios, el programa recibe de una forma diferente a la que se mostrará en estos diagramas. Sin embargo, todos los argumentos presentes en las llamadas se encuentran tanto dentro del objeto Update como dentro del objeto Bot recibido.
- Para simplificar los diagramas, se supondrá que es el usuario quien envía directamente al manejador los mensajes.

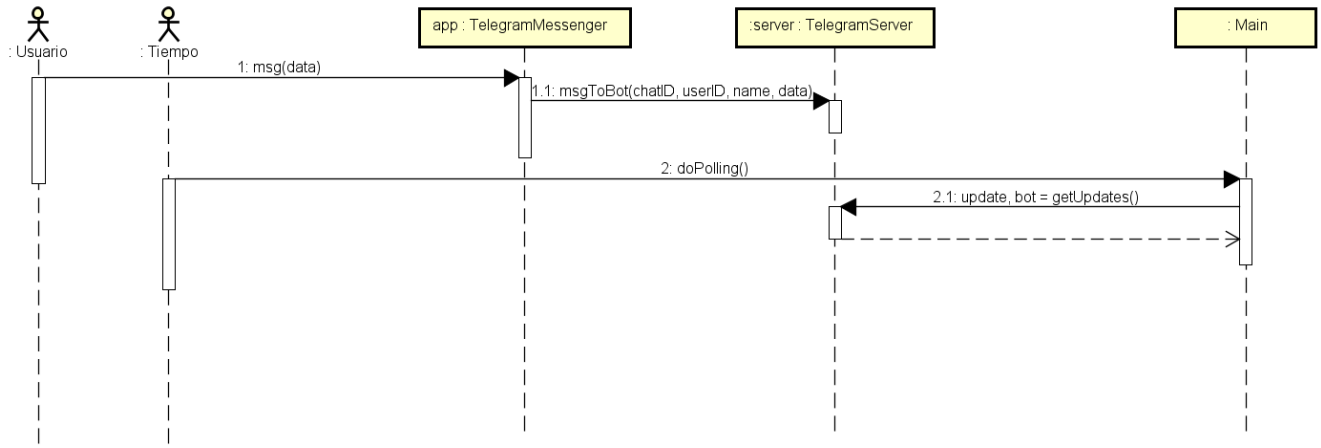

Ilustración 27 - Diagrama explicativo sobre cómo el bot recibe los datos de cada mensaje

- En el objeto update va encapsulada toda la información referente al mensaje recibido por el bot (chat, identificador del usuario y del mensaje, contenido, etc.) y el objeto bot representa el propio bot, con las funcionalidades de este (enviar y editar mensajes, por ejemplo).
- La clase Main representa la clase que se encarga de preguntar al servidor por los mensajes y después manejarlos para que sean tratados.
- El método show() que se aparecerá recurrentemente en estos diagramas solicita al bot que envíe uno o varios mensajes al usuario en los que se represente la vista en la que se encuentra y la Estructura correspondiente.
- En cualquiera de los casos, si ocurre una excepción contemplada en la descripción de los casos de uso, la secuencia volvería hasta el último punto que tuviese una referencia al bot, enviaría un mensaje de error y devolvería los datos necesarios para cumplir con lo explicado en la descripción del caso de uso.

4.5.1 Registrar Usuario

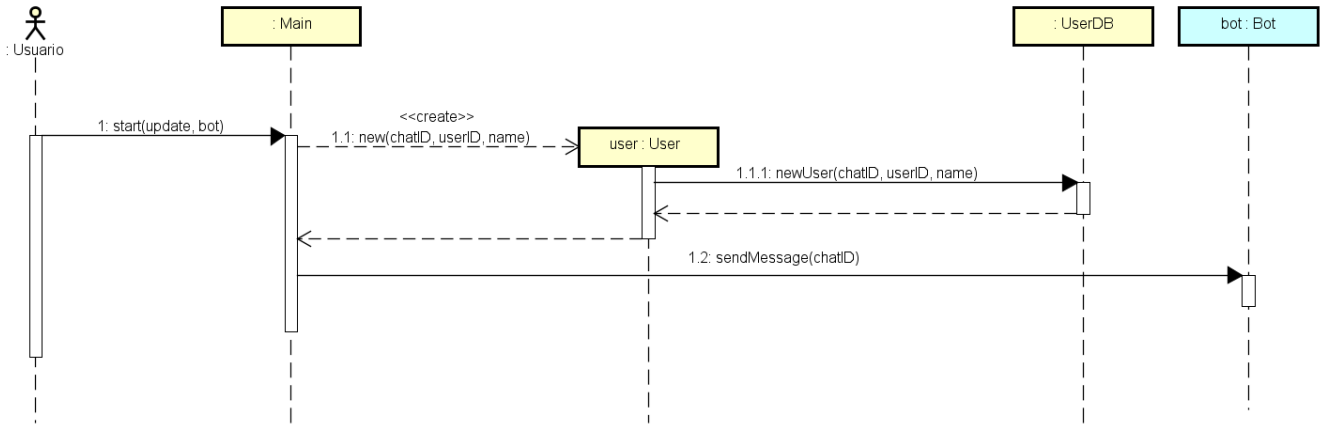

Ilustración 28 - Diagrama de secuencia del caso de uso CU01: Registrar usuario

El usuario envía la solicitud de registro al bot. El bot toma varios parámetros del mensaje recibido (identificador del chat, identificador del usuario y nombre de usuario o nombre y apellido del usuario) y crea un objeto del tipo User. Automáticamente, estos datos se almacenan en la base de datos a través de la clase UserDB, encargada comunicarse con la base de datos tanto para almacenar como para recuperar o actualizar.

Una vez almacenado en la base de datos, el bot envía un mensaje al usuario para que sepa que todo ha ido correctamente.

4.5.2 Eliminar usuario

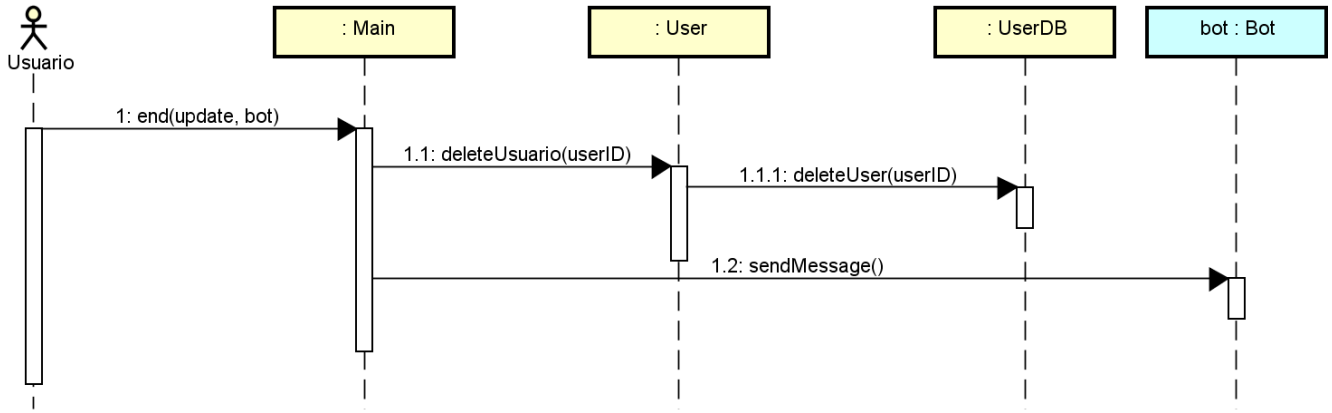

Ilustración 29 - Diagrama de secuencia del caso de uso CU02: Eliminar usuario

A la hora de solicitar la eliminación de un usuario del sistema, el bot toma el identificador único del usuario y lo transmite hasta la clase encargada de eliminarlo de la base de datos junto con los permisos que tuviese asociados.

Después de eso, el bot informa al usuario de que todo ha ido correctamente.

4.5.3 Navigate

Para poder navegar por el sistema de listas y entradas se necesita un punto de partida. Para poder acceder cuantas veces se considere necesario y sin necesidad de buscar ningún botón se implemente en el bot el comando /navigate, que muestra aquellas listas que para el usuario no tienen padre (los nodos raíz desde el punto de vista del usuario).

El diagrama de interacción es el siguiente:

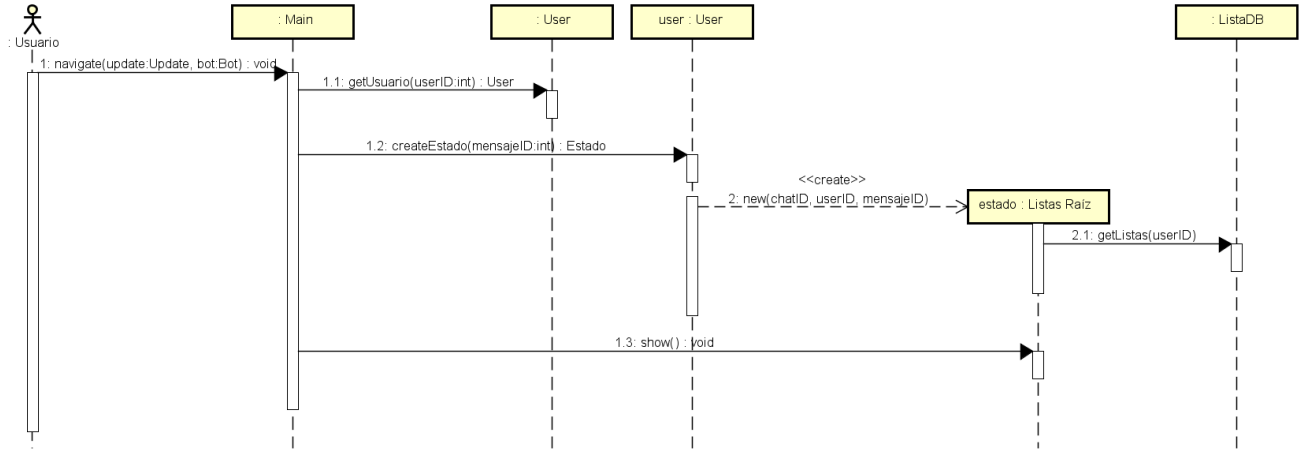

Ilustración 30 - Diagrama de secuencia que muestra cómo un usuario inicia la interfaz gráfica

4.5.4 Crear lista

Si se desea crear una lista primero se ha de pasar al estado de creación de lista. Esto puede realizarse tanto desde el estado Lista Raíz o desde el estado Ver Lista. A continuación, el bot solicitará un título para la lista, seguido de una descripción para la misma. Cuando ambos pasos se hayan completado, el usuario presionará un botón para guardar la nueva lista y esta pasará a almacenarse en la base de datos. Entonces se mostrará la misma vista desde la que se solicitó crear la nueva lista.

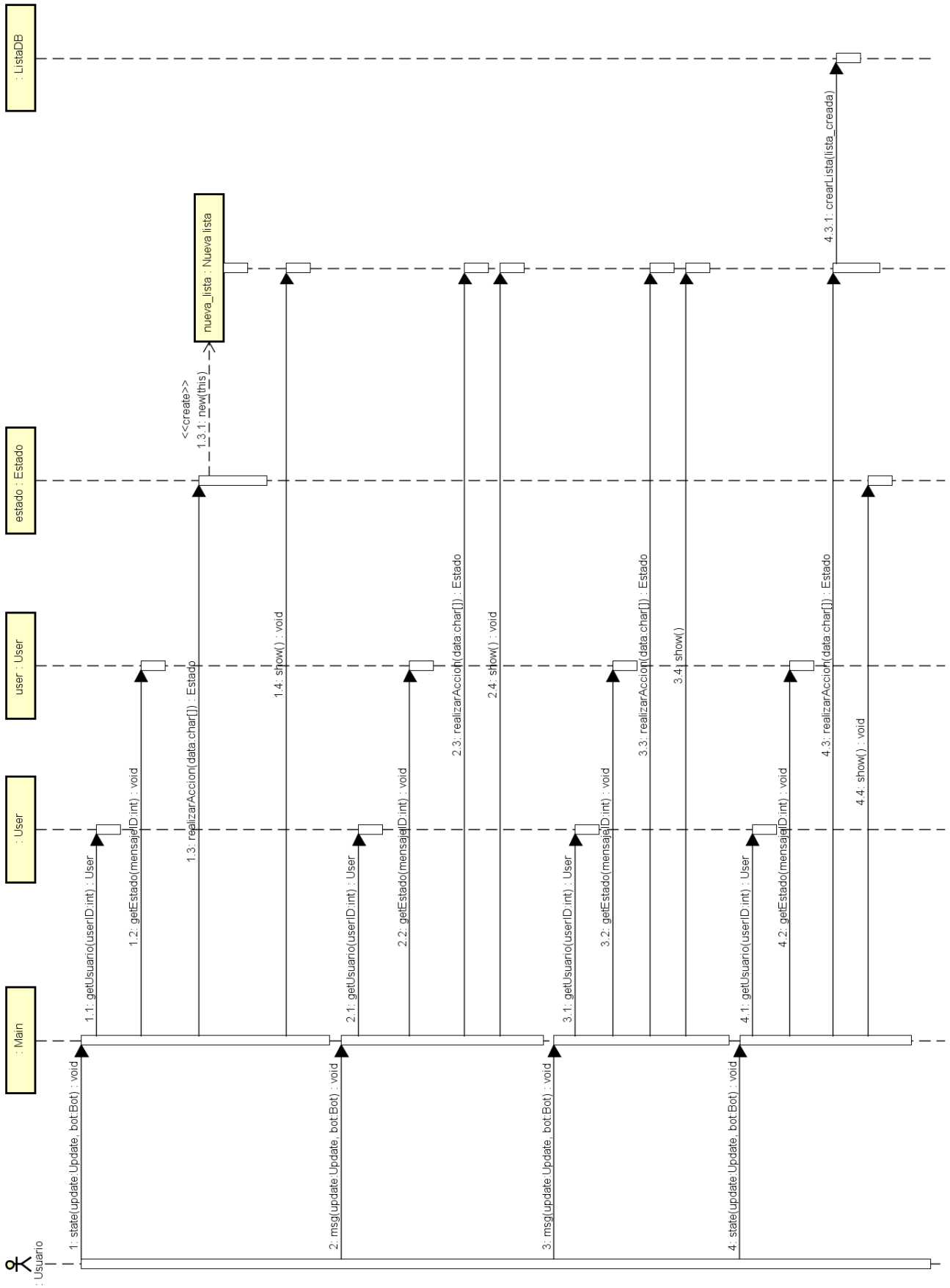

Ilustración 31 - Diagrama de secuencia del caso de uso CU03: Crear lista

4.5.5 Editar lista

Para poder editar una lista primero hay que encontrarse en la vista Ver Lista. Desde ahí se solicita al bot que cargue la vista Editar Lista. Esta opción solo aparece a aquellos usuarios que tiene permiso tanto para ver la lista (pues tiene que llamarse desde Ver Lista) como para editarla. Si no se tiene el permiso, la opción no estará disponible.

En esta vista se debe seleccionar alguno de los atributos de la lista. Una vez seleccionado el bot cargará la vista Editar Campo, donde habrá que introducir el nuevo valor deseado de dicho campo. Tras introducir un valor, se volverá a la visa de Editar Lista, donde podrá modificarse otra propiedad (o la misma si se ha introducido mal algún carácter) o podrá pulsarse el botón para guardar cambios. Pulsar el botón hará que la lista sea actualizada en la base de datos. Después de guardar la lista con los valores actualizados el bot vuelve a la vista Ver Lista.

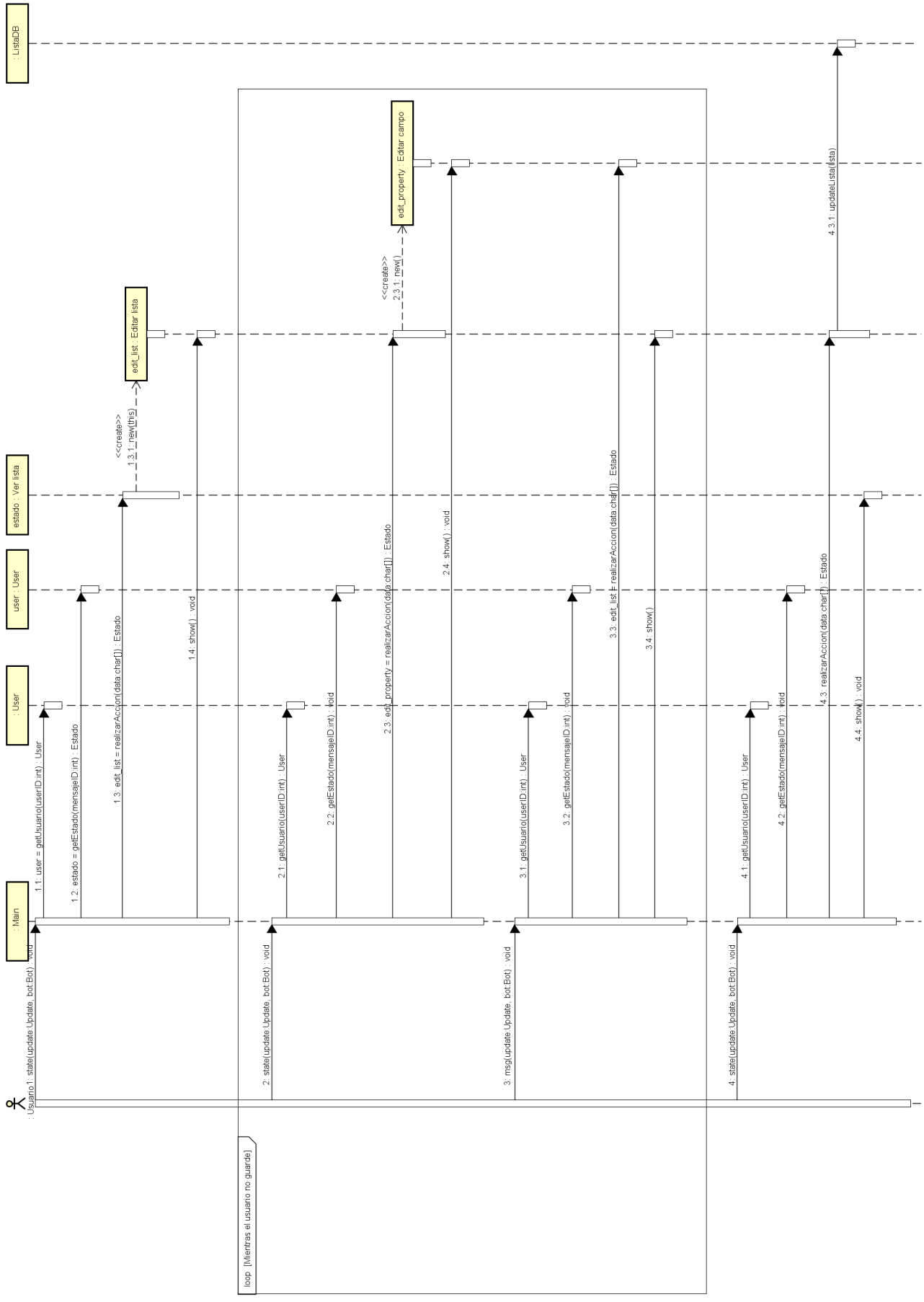

Ilustración 32 - Diagrama de secuencia del caso de uso CU04: Editar lista

4.5.6 Borrar lista

Para poder borrar una lista primero hay que encontrarse en la vista Editar Lista de la lista que se desee borrar. Desde ahí se solicita al bot que elimine la lista. De esta manera, la lista dejaría de existir junto con todos los elementos hijos, de forma recursiva. De la misma manera que para editar o para ver, esta opción únicamente aparece a los usuarios que tienen dicho permiso sobre la lista a tratar, además del permiso de visionado y del de edición.

Tras eliminar una lista se le mostrará al usuario la lista padre de la lista eliminada, si la hubiese, o su vista Listas Raíz, en caso de que la lista no estuviese anidada.

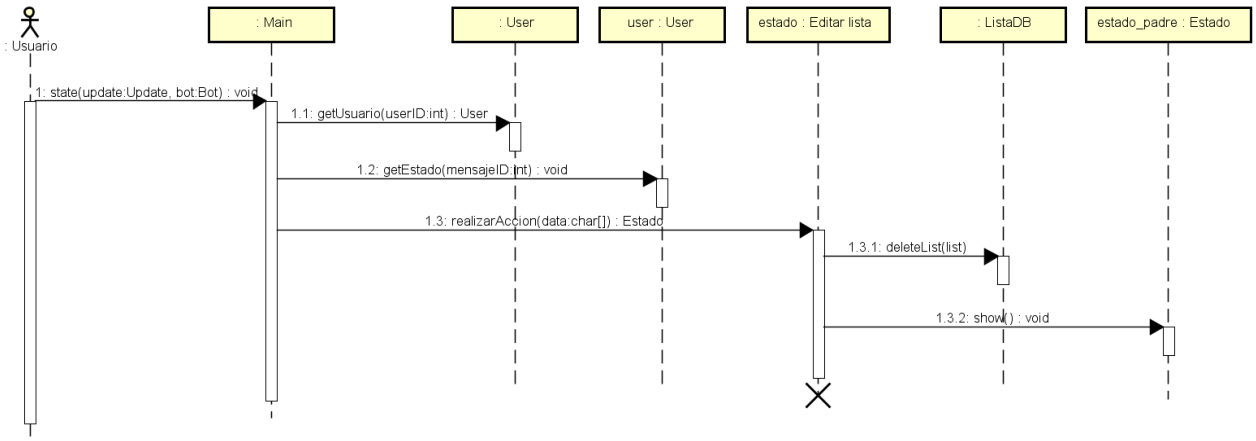

Ilustración 33 - Diagrama de secuencia del caso de uso CU05: Eliminar lista

4.5.7 Consultar lista

Se puede llegar a la vista de Ver Lista desde varios estados y a través de varios métodos:

- Volviendo hacia atrás o cancelando la acción desde varias vistas.
- Finalizando los casos de uso Crea Lista, Editar lista, Borrar Lista, Borrar Entrada.
- Requiriendo acceder a una lista encontrándose en la vista de su lista padre.

Este caso de uso hace referencia al tercer punto, así que dicha interacción es lo que representa el diagrama de secuencia. En los demás casos simplemente se mostraría la vista padre de la vista en la que se encontrase el usuario, descartando esta última. Mantenimiento del proyecto.

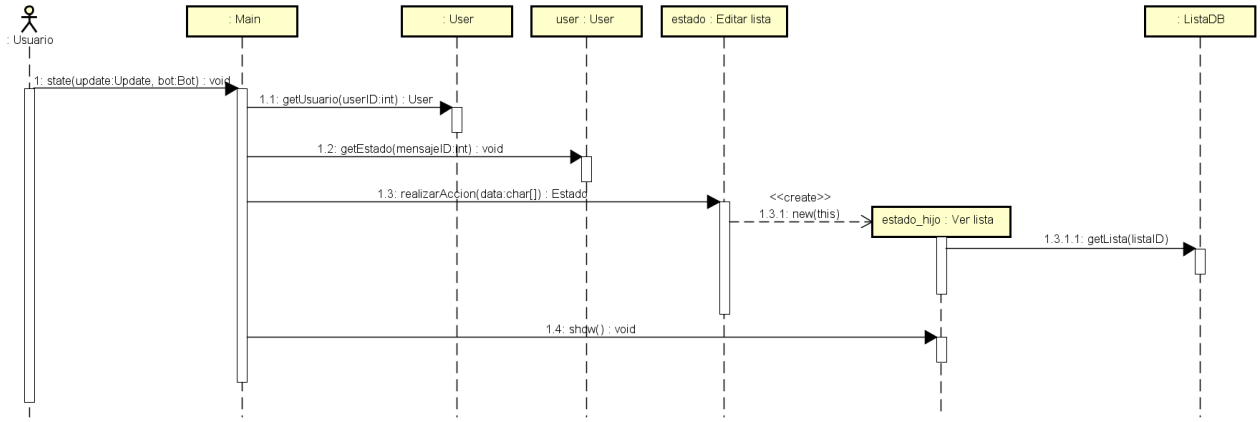

Ilustración 34 - Diagrama de secuencia del caso de uso CU06: Consultar lista

4.6 Alojamiento del bot

Para el que bot pueda cumplir su cometido es necesario que esté funcionando constantemente, pues en caso contrario la interacción sería demasiado lenta comparada con acceder a los ficheros y apuntes navegando por Moodle. Debido a esto, encontrar una manera de mantener al bot en constante ejecución es una tarea importante si se pretende utilizarlo.

Los servidores de almacenamiento y los servidores que permiten ejecutar programas desde la nube son la solución más rentable en cuanto a coste, mantenimiento y comodidad a la hora de mantener o implementar nuevas funcionalidades en el programa. El sistema desarrollado consta de dos partes que pueden almacenarse tanto juntas como separadas, así que se pueden analizar diferentes plataformas para cada una de estas.

Si se decide optar por alguna de estas opciones para mantener en funcionamiento el proyecto, habrá que añadir al presupuesto el costo de la opción elegida. Otra opción sería emplear material propio para alojar la lógica del bot y la base de datos, en cuyo caso habría que tener en cuenta los gastos propios de mantener un sistema así (electricidad, refrigeración, infraestructura, etc.).

4.6.1 Base de datos

El programa precisa de una base de datos en la que almacenar el sistema de listas y entradas creado por los usuarios, así como los tipos de permiso que estos tendrán con respecto a dichas listas y entradas y a los usuarios en sí. Los ficheros enviados por Telegram son almacenados en sus servidores y pueden ser recuperados fácilmente siempre que se posea el identificador de dicho fichero, así que no es necesario almacenar estos ficheros (ni en disco ni en la propia base de datos) siempre que almacenemos el identificador.

4.6.1.1 Cloud SQL

Cloud SQL es un servicio de base de datos englobado en la plataforma Google Cloud. Esta plataforma ofrece una prueba gratuita de 12 meses a todos los usuarios, lo cual es un punto para tener en cuenta a la hora de reducir costes, ya que incluso se podría migrar a otra plataforma una vez cumplido el plazo si los resultados obtenidos no son los esperados o deseados. Los lenguajes soportados por la plataforma son PostgreSQL [BETA] y MySQL, por lo que la base de datos creada para este proyecto podría ser almacenada en este servicio.

El primer paso es crear una instancia de Cloud que actúa como proyecto desde la cual se pueden gestionar las bases de datos que vayan a utilizarse. En este momento se escogen algunas opciones que no pueden modificarse en un futuro, tales como la localización del servidor, el nombre de la instancia creada o la máquina sobre la que trabajará la instancia. No poder cambiar la localización del servidor o la máquina que va a emplear puede causar problemas a largo plazo, pues si el rendimiento es menor del esperado no queda otra opción más que dejar de lado la instancia y crear una nueva.

Los precios ofrecidos por la plataforma [10] varían en función de la máquina (número de CPUs, RAM, capacidad de almacenamiento y número de conexiones), del uso que se le dará a esta (por horas o continuo), del almacenamiento consumido, del lugar desde el que se hace la consulta y del lugar en el que está situado el servidor.

Se utilizará la calculadora de precios que ofrece Google Cloud [11] para la contratación de servicios. Esta calculadora estima el precio mensual basándose en los factores antes mencionados que influyen en el precio. El nivel más básico que se puede contratar debería ser suficiente para una utilización reducida del bot, pero se tendrán también en cuenta el siguiente nivel de servicio ofertado:

Tipo de máquina	CPU virtuales	RAM (GB)	Capacidad máxima de almacenamiento contratada (GB)	Cantidad máxima de conexiones	Precio por hora de uso continuo (USD)
db-f1-micro	Compartidas	0,6	3062	250	0,0126
db-g1-small	Compartidas	1,7	3062	1000	0,0420

Tabla 25 - Tipos de máquinas de Cloud SQL que podrían resultar útiles para el proyecto

Se realizarán varias estimaciones, aunque en todas se escogerá un servidor cercano (Londres), la misma cantidad de instancias (1), y el mismo tiempo de uso (24 horas al día y 7 días a la semana). El factor entre la capacidad mínima y máxima es tan alto debido a que pequeñas diferencias en la necesidad de almacenamiento al principio se verían acrecentadas con el paso del tiempo, pudiendo requerir una ampliación de este límite.

Tipo de máquina	Capacidad de almacenamiento contratada (GB)	Tamaño de la copia de resguardo (GB)	Precio por un mes de uso (EUR)
db-f1-micro	10	0	9,69
		10	10,51
	50	0	16,72
		50	20,86
	100	0	25,51
		50	29,65
		100	33,78
db-f1-small	10	0	28,18
		10	29,01
	50	0	35,22
		50	39,35
	100	0	44,01

Tipo de máquina	Capacidad de almacenamiento contratada (GB)	Tamaño de la copia de resguardo (GB)	Precio por un mes de uso (EUR)
		50	48,14
		100	52,28

Tabla 26 - Precio de la utilización de las máquinas de Cloud SQL

En todos los casos se puede también contratar la opción de Alta Disponibilidad, que asegura que el servicio esté disponible en todo momento, incluso si se cae la instancia principal. Esto se consigue replicando la instancia, lo cual conlleva un aumento en el consumo de recursos y, por tanto, del precio. Si se activa esta opción el precio se ve aumentado en alrededor de un 90%.

4.6.1.2 Oracle MySQL Cloud Service

La empresa desarrolladora de MySQL (entre otras) también ofrece servicio de almacenamiento en la nube para este sistema de gestión de base de datos. De la misma manera que en Cloud SQL, también existe la posibilidad de crearse una cuenta gratuita de prueba, aunque en este caso únicamente durante un mes.

El sistema de pago que ofrece Oracle divide los paquetes en herramientas y método de pago. Cada paquete integra las herramientas del paquete anterior y otras herramientas extra, y los dos tipos existentes de método de pago equivalen a un sistema de pago bajo demanda o a tener un contrato con duración mínima de un año. Dentro de la división de paquetes se diferencia entre utilizar una licencia propia o comprar también una a Oracle, que es lo que se supondrá en este cálculo.

La personalización de los recursos contratados es mucho menor en este caso, así que basta con fijarse en el precio del paquete más pequeño que incluya lo necesario para poder utilizar la plataforma para almacenar la base de datos. En este caso es el paquete básico, *Standard Edition – General Pourpose*, que tiene un coste mensual, en su versión bajo demanda, de 304 euros si se adquiere en su versión más pequeña. Si se firma un contrato de permanencia de al menos un año el precio mensual se reduciría a 204 euros al mes. La opción de personalización que ofrece a mayores Oracle es la de introducir cuántas CPUs se prevé que van a utilizarse cada hora, pero para este tipo de almacenamiento con una, la menor cantidad posible, es suficiente. [12]

4.6.1.3 Amazon RDS para MySQL

Amazon ofrece, de la misma manera que Google, un sistema de instancias con características variadas y un sistema de pago bajo demanda. Existe también un periodo de prueba gratuita de 12 meses, más que suficiente para comprobar si se cumplen las expectativas de rendimiento y costes estimadas y deseadas.

Una gran ventaja de los servicios de Amazon es que la mayoría de las propiedades ya mencionadas (espacio de almacenamiento, memoria RAM y CPUs utilizadas) puede modificarse en cualquier momento, de la misma manera que el ancho de banda. Estas características, con excepción del espacio de almacenamiento, dependen del tipo de instancia contratado, que es el elemento que se puede modificar. También existe una diferenciación entre un servicio con protección ante caídas y uno sin ello. El precio aproximadamente se duplica con respecto a lo mostrado más adelantes si se desea la protección ante caídas.

En este caso, como la cantidad de almacenamiento no está limitada, la decisión recae sobre qué tipo de instancia utilizar. Las más modestas deberían ser suficientes:

Tipo de instancia	CPU virtuales	RAM (GB)	Precio por hora de uso continuo (USD)
db.t2.micro	1	1	0,019
db.t2.small	1	2	0,038
db.t2.medium	2	4	0,077

Tabla 27 - Tipos de instancia de Amazon RDS para MySQL que resultan interesantes para el proyecto

Amazon da la posibilidad de, en vez de pagar por consumo bajo demanda, reservar una instancia. Reservar una instancia consiste en contratar el servicio durante 1 o 3 años, reduciendo en gran medida el coste del servicio. Con un contrato de un 1 año es posible no realizar ningún pago inicial, aunque el ahorro sería únicamente de un 21% [13].

Tipo de instancia	Precio por hora – servicio bajo demanda (USD)	Ahorro con contrato de 1 año (USD)		Ahorro con contrato de 3 años (USD)	
		Pago inicial parcial	Pago total anticipado	Pago inicial parcial	Pago total anticipado
db.t2.micro	0,019	32 %	32 %	53 %	55 %
db.t2.small	0,038	30 %	32 %	50 %	55 %
db.t2.medium	0,077	33 %	34 %	52 %	55 %

Tabla 28 - Precio y ahorro relativo de los distintos tipos de instancia y contratación de Amazon RDS para MySQL

Cubierta la parte del procesamiento de las peticiones, queda analizar los planes para el almacenamiento de los datos. Amazon ofrece distintos tipos de almacenamiento, siendo los que pueden interesar para este proyecto:

Tipo de almacenamiento	Mínimo (GiB)	Máximo (GiB)	Precio por GB/mes (USD)
Almacenamiento de uso general (SSD)	20	16384	0,133
Almacenamiento magnético	20	3072	0,116

Tabla 29 - Precio y capacidad de los tipos de almacenamiento que ofrece Amazon RDS para MySQL y resultan interesantes para el proyecto

Cabe destacar que si se escoge el almacenamiento de uso general (SSD) no se cobrarán las operaciones de E/S, mientras que con el almacenamiento magnético sí. De la misma manera que con el procesado, si el servicio contratado incluye la protección frente a caídas estos precios se duplican.

Con respecto a las copias de respaldo, son gratuitas siempre que el tamaño de estas no supere el espacio utilizado por los datos originales. Según la experiencia de Amazon, en la mayoría de los casos no es necesario sobrepasar ese umbral, por lo que no es un gasto poco probable. Sin embargo, por completar la información, el precio por GiB al mes por el almacenamiento de los respaldos (contando aquí únicamente la parte que supera al original) es de 0,095 USD.

Aunque las opciones de respaldo y protección frente a caídas que han aparecido en los distintos servicios parezcan similares, la protección frente a caídas se encarga de que el servicio se encuentre

siempre disponible, mientras que las copias de seguridad solo se encargan de que no se pierdan los datos y de actuar como una especie de histórico por si es necesario volver a una versión estable. Es decir, en ambos casos no se pierden los datos, pero con la copia de seguridad no se asegura un servicio ininterrumpido.

4.6.1.4 Digital Ocean

Digital Ocean es una plataforma que ofrece el servicio de almacenamiento y computación en la nube enfocándose en la simplicidad y facilidad de uso. Ofrece una prueba gratuita de 2 meses en la parte relacionada con las copias de seguridad (almacenamiento) [14].

La base de datos se almacenaría como un *droplet*, que tienen una tarifa asociada a la memoria RAM, a la cantidad de CPUs, al almacenamiento en disco SSD y a la cantidad de datos transferidos. Para ampliar la cantidad de almacenamiento de esta parte Digital Ocean ofrece la posibilidad de aumentar la capacidad en bloques de 10 GB al precio de 0,10 USD cada GB [15].

RAM (GB)	CPUs	Memoria en disco SSD (GB)	Volumen de datos transferibles por mes (TB)	Precio mensual (USD)
1	1	25	1	5
2	1	50	2	10
3	1	60	3	15
2	2	60	3	15
1	3	60	3	15

Tabla 30 - Características y precios de los planos de Digital Ocean que resultan interesantes para el proyecto

A parte, para almacenar una copia de seguridad de la base de datos ofrece la opción de contratar dicho servicio por un precio equivalente al 20 % del plan contratado para el *droplet*. También puede adquirirse un bloque de almacenamiento extra, no como una ampliación del SSD ni con la necesidad de utilizarlo para una copia de respaldo, que dispone en la versión más básica de 250 GB de almacenamiento y 1 TB de volumen de transferencia mensual al precio de 5 USD por mes. Puede ampliarse este almacenamiento por 0,02 USD por GB almacenado y el volumen de transferencia a razón de 0,01 USD por GB transferido.

4.6.2 Lógica del bot

De la misma manera que es necesario tener disposición en todo momento de la base de datos, es necesario que el bot esté funcionando continuamente para poder dar soporte a los usuarios cuando sea que lo requieran. La vista en la que se encuentra cada usuario, así como los datos del elemento que esté gestionando o consultado y el resteo del árbol hasta llegar a su vista raíz, se encontrará almacenado dentro de la lógica del bot, por lo que es necesario encontrar una plataforma que, además de ofrecer un buen precio, sea fiable, pues un reinicio del bot implicaría que todos los usuarios tuviesen que volver a empezar la navegación a partir del uso del comando.

Son muchas las plataformas que ofrecen servicios de ejecución de código en la nube, así que aquí se tratarán solo una parte de ellas.

4.6.2.1 AWS Lambda

AWS Lambda permite ejecutar código en la nube sin tener que preocuparse de la administración de los servidores. El servicio se encarga de gestionar los recursos y proporcionar los suficientes para que el código proporcionado se ejecute en las condiciones en las que el usuario lo ha solicitado [16].

AWS Lambda divide su coste en dos partes. Primero posee una capa gratuita para todos los usuarios de AWS. Esta capa proporciona soporte para un millón de solicitudes y un tiempo de cómputo de 400000 GBs al mes. A partir de esas cantidades, cada solicitud tiene un coste de 0,0000002 USD (0,20 USD por cada millón de solicitudes) y unos 0,00001667 USD por cada GB/s utilizado [17].

Esta plataforma suma una solicitud cada vez que comienza a ejecutarse como respuesta a una notificación de evento o una llamada de invocación. El tiempo de cómputo se calcula a partir de la cantidad de solicitudes, la memoria asignada a estas y su duración. Para que se comprenda más fácilmente puede emplearse un ejemplo:

$Existe\ una\ función\ que\ tiene\ asignada\ 512\ MB\ de\ memoria\ para\ cuando\ es\ ejecutada.$
$Cada\ ejecución\ de\ esta\ función\ requiere\ un\ 1\ segundo\ de\ uso\ del\ servicio.$
$Se\ ejecuta\ 3\ millones\ de\ veces\ en\ un\ mes.$

$Cargos\ por\ el\ uso\ de\ tiempo\ de\ cómputo:$
$3000000\ ejecuciones \times 1\ {}^{s}/_{ejecucion} = 3000000\ segundos$
$3000000\ segundos \times 512\ MB\ \times \frac{1\ GB}{1024\ MB} = 1500000\ GBs$
$1500000\ GBs - 400000\ GBs\ de\ la\ capa\ gratuita = 1100000\ GBs$
$1100000\ GBs\ \times 0{,}00001667\ {}^{USD}/_{GBs} = 18{,}337\ USD$

$Cargos\ por\ las\ solicitudes:$
$3000000\ solicitudes - 1000000\ solicitudes = 2000000\ solicitudes$
$2000000\ solicitudes \times \frac{0{,}20\ USD}{1000000\ solicitudes} = 0{,}40\ USD$

$Cargos\ totales:$
$18{,}337\ USD + 0{,}40\ USD = 18{,}737\ USD$

Ilustración 35 - Ejemplo del coste económico de AWS Lambda

4.6.2.2 Google App Engine

Google App Engine es la plataforma de Google que permite ejecutar código desde la nube [18]. Tiene dos entornos distintos, el entorno estándar y el entorno flexible:

4.6.2.2.1 App Engine Standard Environment

Las instancias dentro del entorno estándar disponen de un límite diario de uso de recursos. Esta función es gratuita y está delimitada por un conjunto de cuotas. Si se supera dicho límite, se aplicarán cargos a las aplicaciones tal y como se explica más adelante. Se puede establecer un límite de gasto

para controlar los costes de la aplicación. En esta plataforma son 3 los puntos a tratar: tipos de instancia, cuotas y límite de gasto.

Los tipos de instancia se diferencian en el límite del tamaño de la memoria, en la velocidad de la CPU y en la forma de escalar los recursos. Los dos más relevantes para este proyecto con el B1 y el B2. Teniendo el B1 128 MB de memoria y una velocidad de CPU de hasta 600 MHz y, el B2, el doble en ambos casos. El método de escalado en ambos es manual.

Las cuotas se dividen en 3 tipos: gratuitas, de gasto y de seguridad. Las cuotas gratuitas hacen referencia a la cantidad gratuita de cada recurso que recibe cada aplicación, pudiendo superarse el uso de dicho recurso únicamente si la aplicación es de pago. Las cuotas de gasto se alcanzan cuando una aplicación ha consumido los recursos fijados por el propietario del su proyecto, aunque pueden superarse ligeramente cuando se inhabilita la aplicación. Las cuotas de seguridad se encargan de que ninguna aplicación consuma recursos de tal manera que perjudique a otras aplicaciones.

El primer GB de almacenamiento de datos estáticos y de código es gratuito, pero a partir de ese punto se facturará este tipo de almacenamiento a razón de 0,026 USD por GB al mes. También es interesante el límite de despliegues, equivalente a la cantidad máxima de veces que un desarrollador puede actualizar alguno de los ficheros de una aplicación. El límite de despliegues está fijado en 10000 actualizaciones al día, no pudiendo superar ningún archivo los 32 MB. Otra cuota existente hace referencia a la cantidad de E/S de la aplicación:

Recurso	Límite predeterminado gratuito		Límite predeterminado con la facturación habilitada	
	Límite diario	Frecuencia máxima	Límite diario	Frecuencia máxima
Ancho de banda de salida	1 GB	56 MB por minuto	1 GB gratuito (14400 GB como máximo)	10 GB por minuto
Ancho de banda de entrada	1 GB (14400 GB como máximo)	56 MB por minuto	Ninguno	Ninguno

Tabla 31 - Cuotas de E/S para una instancia App Engine Standard Environment

El límite de gasto es el coste máximo de recursos de App Engine que se desea pagar al día en un proyecto determinado. Se trata de un límite aproximado y cabe la posibilidad de que se exceda ligeramente mientras la aplicación está desactivada. También pueden ser aplicados cargos por superar el límite de gasto a la hora de usar otros recursos del entorno Google Cloud. El límite debe fijarse teniendo en cuenta los posibles picos de actividad de la aplicación.

4.6.2.2.2 App Engine Flexible Environment

Este entorno está preparado para reducir al máximo la tarea de gestionar un servidor para permitir a los desarrolladores centrarse en la aplicación [19]. En este caso, los recursos se sirven bajo petición con un coste constante, por lo que es sencillo calcular los gastos que esta opción podría ocasionar:

Recurso	Unidad	Coste de la unidad (USD)
vCPU	Por hora de núcleo	0,063
Memoria	Por GB por hora	0,009
Almacenamiento	Por GB al mes	0,048

Tabla 32 - Precio de los recursos de una instancia App Engine Flexible Environment

Habiendo cubierto ya la infraestructura, que se regulará automáticamente según lo requiera la aplicación, otro coste para tener en cuenta es el tráfico que mueve la aplicación. En la siguiente tabla se muestran los tramos en los que está dividido el tráfico:

Uso mensual	**Salida de red** **Destinos de todo el mundo** **(no se incluyen China y Australia, pero sí Hong Kong)** **(por GB)**	**Salida de red** **Destinos de China (salvo Hong Kong)** **(por GB)**	**Salida de red** **Destinos de Australia** **(por GB)**	**Entrada de red**
0-1 TB	$0.12	0,23 USD	0,19 USD	Gratis
1-10 TB	$0.11	0,22 USD	0,18 USD	Gratis
Más de 10 TB	$0.08	0,20 USD	0,15 USD	Gratis

Tabla 33 - Precio de las operaciones de E/S en una instancia App Engine Flexible Environment

Capítulo 5. Manuales del proyecto

5.1 Manual de instalación

Para que el proyecto desarrollado pueda funcionar en cualquier máquina es necesario instalar en estas algunos componentes, como son el intérprete de Python 3.6, algunas bibliotecas o la base de datos. En este apartado de la memoria se explicará uno de los procedimientos que se puede llevar a cabo para poder conseguir que la aplicación funcione correctamente.

5.1.1 Instalación de Python

El primer paso para poder ejecutar el bot es que el sistema operativo sea capaz de interpretar el código del programa. Para ello será necesario instalar la versión 3.6 de Python, pues es la que se ha empleado para desarrollarlo. En la mayoría de los casos la versión concreta de Python no es relevante, puesto que se comparte gran cantidad de bibliotecas entre las distintas versiones (incluso podría intentar utilizarse las versiones 2.x), pero para no comprometer ningún tipo de dependencia es recomendable instalar la versión 3.6.

Dependiendo del sistema operativo podrá instalarse de unas maneras o de otras, pero accediendo a la página web de Python, en la sección de Downloads puede encontrarse el instalador de la versión requerida. Una vez descargado el instalador, ejecutarlo y seguir la secuencia predeterminada de pasos instalará los componentes necesarios para que el sistema operativo pueda interpretar el lenguaje de programación.

5.1.2 Instalación de bibliotecas de Python

Para que el bot pueda realizar todas las funciones que tiene asignadas es necesario obtener e instalar dos bibliotecas que no vienen incorporadas en la versión básica de Python. Estas bibliotecas son *python-telegram-bot* y *PyMySQL*.

python-telegram-bot es una biblioteca para Python que encapsula las clases y métodos ofrecidos por la API para el desarrollo de bots de Telegram. Para instalar esta biblioteca se puede emplear alguna herramienta o se puede seguir las instrucciones que ofrece el repositorio en el que se encuentra alojada [20]. En este caso utilizaremos desde el intérprete de comandos del sistema operativo un comando que se ha instalado en el paso anterior junto con Python. El comando a ejecutar es el siguiente:

```
$ pip install python-telegram-bot --upgrade
```

Tabla 34 - Comando para instalar la biblioteca python-telegram-bot

Una vez ejecutado el comando, tras una pequeña espera la biblioteca debería haberse instalado en el equipo y el intérprete de Python podrá acceder a ella cuando lo requiera.

PyMySQL es una biblioteca para Python que permite al desarrollador comunicarse con un servidor MySQL de una manera cómoda y sencilla. De la misma manera que para la biblioteca *python-telegram-bot*, esta biblioteca puede instalarse con la ayuda de herramientas externas o desde la propia línea de comandos, usando el mismo método que en el caso anterior, aunque cambiando los parámetros:

```
$ pip install PyMySQL
```

Tabla 35 - Comando para instalar la biblioteca PyMySQL

Una vez instaladas las dos bibliotecas el sistema operativo tendrá todo lo necesario para poder ejecutar el bot y el intérprete de Python tendrá todo lo necesario para poder realizar todas las llamadas necesarias.

5.1.3 Instalación de MySQL

Debido a que el proyecto se ha desarrollado en un equipo con Windows 10 y que los métodos para instalar MySQL difieren en gran medida de un sistema operativo a otro, se explicará cómo instalarlo en el sistema ya mencionado. En el sitio web de MySQL pueden encontrarse guías para instalarlo en otros sistemas operativos. Dentro de la propia instalación en Windows 10, hay algunos pasos que pueden realizarse tanto por medio de la interfaz gráfica como por la línea de comandos. En esta ocasión se utilizará la interfaz gráfica.

El primer paso es obtener el instalador, que puede descargarse en la sección de Community de la página web de MySQL, la herramienta a descargar es MySQL Community Server [21]. Hay que seleccionar el sistema operativo correspondiente al que se vaya a utilizar (en este caso Micrososft Windows) y descargarlo.

Una vez descargado, se ejecuta el instalador y se siguen los pasos para instalar MySQL Server. Tras finalizar el asistente de instalación, podrá crearse una instancia del servidor con un nuevo asistente. En este asistente se irá pasando por varias fases en las que es requerido realizar varias elecciones. Las opciones relevantes que hay que escoger son “Multifunctional Database” cuando se solicite el uso que se le va a dar a la base de datos, “Enable TCP/IP Networking” cuando se presenten las opciones sobre el uso de la red y “Standard Character Set” o “Best Support For Multilingualism” cuando se solicite el conjunto de caracteres a utilizar. El resto de las opciones no deberían influir en el funcionamiento de la instancia que se ha creado. Información más concreta sobre todo el proceso de instalación puede encontrarse en la sección de desarrolladores de la web de MySQL [22].

5.1.4 Creación de la base de datos en la instancia de MySQL

Para que la aplicación pueda almacenar los datos no basta con tener un servidor donde almacenarlos, sino que hay que crear la base de datos y las tablas necesarias. Para ello, en la instancia de MySQL Server hay que desplazarse a la sección de ”Users and Privileges”, dentro la sección Management, y añadir un nuevo usuario con nombre de usuario “prueba” y contraseña “pass”. Estas son las credenciales del usuario existentes en el programa; puede modificarse, pero habría que modificar los parámetros del método de conexión en el código de la aplicación. A continuación, abrir una pestaña para introducir consultas SQL y se introduce la siguiente consulta:

```
CREATE DATABASE listmanbot;

CREATE TABLE listmanbot.Usuario(
    ID bigint primary key auto_increment,
    Username nvarchar(100)
    );

CREATE TABLE listmanbot.Estructura(
    ID bigint primary key auto_increment,
    ParentID bigint references listmanbot.Estructura(ID),
    Title nvarchar(255),
    Description nvarchar(255)
    );

create table listmanbot.EntradaTexto(
    ID bigint primary key references listmanbot.Estructura(ID),
    Text nvarchar(255)
    );

create table listmanbot.EntradaReferencia(
    ID bigint primary key references listmanbot.Estructura(ID),
    Reference nvarchar(255)
    );

create table listmanbot.Permiso(
    ID bigint primary key auto_increment,
    EstructuraID bigint references listmanbot.Estructura(ID),
    UsuarioID bigint references listmanbot.Usuario(ID),
    edit bool not null,
    del bool not null
    );
```

Tabla 36 - Sentencia SQL para generar la base de datos y las tablas necesarias para el funcionamiento de la aplicación

Tras ejecutar la consulta, la base de datos quedará creada y, de esta manera, el bot ya podrá funcionar plenamente, a falta de ejecutar el fichero *final.py* que se encuentra en el proyecto.

5.2 Manual de usuario

En este apartado se expondrá y desarrollará la información necesaria para que los usuarios puedan comprender el funcionamiento de la aplicación y, con ello, hacer un uso eficiente del bot. Se comenzará explicando la terminología que se utilizará en esta parte y se continuará con cómo pueden interactuar los usuarios con el bot.

5.2.1 Iniciar una conversación con el bot

Para poder relacionarse con el bot será necesario enviarle el mensaje “/start” utilizando Telegram. Para ello, hay que buscar su nombre (@ListManBot) en Telegram y pulsar en el botón que aparece antes de iniciar un chat con un bot. De aquí en adelante ya puede utilizarse el bot.

5.2.2 Terminología y conceptos generales

En este apartado se definirán los términos más importantes que guardan relación con el uso de la aplicación. De esta manera se sentará una base para poder comprender más adelante el resto de las

partes del manual y el usuario conocerá con qué intención se ha diseñado y desarrollado cada elemento.

En el ámbito del bot se considera un fichero a cualquier tipo de dato que puede enviarse por Telegram que no es un mensaje de texto. Se consideran ficheros todos los documentos, audios, imágenes, stickers, vídeos y notas en vídeo.

En el ámbito del bot se considera una *entrada* a la estructura que posee un título, una descripción y un valor. El campo valor puede ser tanto una cadena de texto como un fichero. Se pretende que el título y el valor sean los campos con mayor importancia, relegando la descripción a ser algo puntual que cumpla únicamente su función designada (aportar información extra sobre la entrada cuando el título y el valor no son suficientes). Todas las entradas pertenecen a una lista.

En el ámbito del bot, se considera una *lista* a la estructura que posee un título, una descripción y puede contener entradas u otras listas. En este caso es probable que la descripción sea relevante en más ocasiones, pues puede explicar qué tipo de elementos contiene la lista. Las listas pueden pertenecer a otra lista. Si no lo hacen, se les considerará *lista raíz*.

En el ámbito del bot, se empleará el término *elemento* para referirse a una estructura cuando puede ser tanto una lista como una entrada.

En el ámbito del bot, se considera *árbol de listas* a la estructura formada por una lista que no pertenece a ninguna otra lista y a todos los elementos que pertenecen a dicha lista o a algún elemento que pertenezca al árbol. Dicho de otra manera, un *árbol de listas* representa a una lista raíz y a todos los elementos a los que se puede llegar accediendo recursivamente a las listas pertenecientes a la lista raíz.

En el ámbito del bot, se considera un *permiso* a la capacidad o incapacidad de un usuario para relacionarse de cierto modo con un elemento. Puede haber permiso de consulta (poder ver el elemento), de edición (poder editar el elemento) o de borrado (poder borrar el elemento).

5.2.3 Comandos del bot

Los comandos son mensajes de texto con un cierto formato (comienzan por el carácter / seguido del comando y posibles parámetros separados por espacios) que pueden emplear los usuarios para comunicarse con el bot. Se ha procurado reducir su uso con el fin de que el usuario pueda comunicarse con el bot de una manera con la que se encuentre más familiarizado hoy en día (interfaz gráfica táctil). Los comandos empleados son:

- **/start:** Inicia la comunicación del usuario con el bot. El bot mostrará un mensaje de bienvenida y registrará al usuario para que este pueda utilizar las funcionalidades. Si el usuario ya está registrado se informará de ello. Si no ha sido posible registrar al usuario, se le informará de ello con el fin de que vuelva a intentarlo más adelante.
- **/help:** El bot enviará un mensaje al usuario en el que explicará sus funcionalidades y una lista con los comandos y su utilidad.
- **/deleteuser:** Solicita al bot que elimine el registro del usuario. De esta manera el bot y el usuario no podrán volver a interactuar entre ellos hasta que vuelva a introducirse el comando /start. La

utilización de este comando conlleva la pérdida permanente de todos los permisos concedidos al usuario que lo utiliza.

- **/navigate:** Accede a la vista de todas las listas raíz a las que tiene acceso el usuario. Este es el punto de entrada a la interfaz gráfica que ofrece el bot para navegar por los árboles de listas.

5.2.4 Interfaz gráfica

Para llevar a cabo la mayor parte de las funcionalidades que ofrece el bot es necesario haber efectuado los comandos /start y /navigate. De esta manera el usuario se encontrará registrado en el sistema y el bot será consciente de que ha iniciado la interfaz gráfica. El objetivo de la interfaz gráfica es proporcionar a los usuarios un medio cómodo y sencillo con el que desplazarse por los árboles de listas y utilizar las funcionalidades de la aplicación. Este tipo de interfaz permite que los usuarios realicen las tareas minimizando en gran medida la cantidad de veces que tienen que introducir algún dato (la comparativa más sencilla es la de escribir el nombre de una lista o pulsar sobre ella).

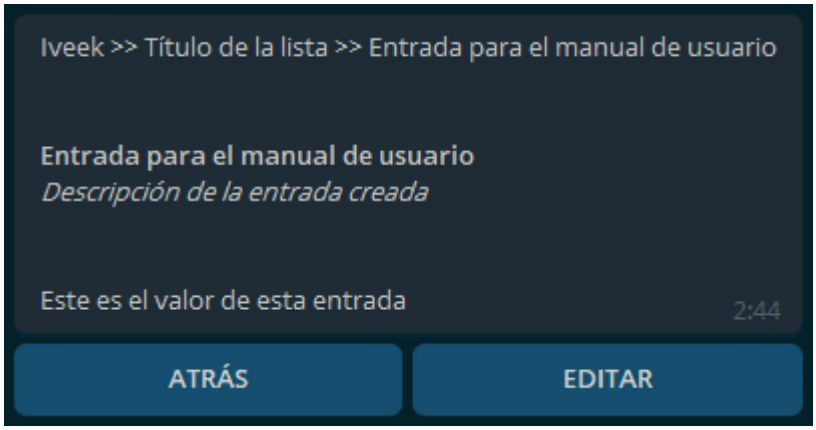

Ilustración 36 - Apariencia de la vista Vista Entrada en web Telegram

La interfaz se encuentra dividida en varias zonas. En la zona superior se muestra la ruta que el usuario ha seguido hasta llegar a la vista en la que se encuentras. Debajo de ello se muestra información sobre el elemento y la vista en la que el usuario se encuentre. Si el usuario está editando algún elemento o permiso y no ha guardado los cambios, habrá un mensaje que le informe debajo de los datos del elemento o permiso.

En la parte de abajo se encuentran los botones que permiten al usuario navegar por los árboles de listas. Dependiendo de los permisos que tenga el usuario sobre el elemento en el que se encuentre es posible que no tenga presente alguno de los botones, como sucedería con el botón de editar si el usuario está viendo una lista para la cual no tiene permiso de edición. Cada una de las vistas tiene una configuración distinta de botones, pero hay botones con funciones similares:

- El botón de **atrás** lleva al usuario a la vista anterior según la ruta que se muestra al principio de los mensajes.
- El botón de **editar** se encuentra únicamente presente cuando el usuario se encuentre viendo algún elemento o permiso, permite acceder a la secuencia para editar alguna propiedad del elemento o permiso.
- El botón de **nueva lista** se encuentra únicamente presente cuando el usuario está viendo una lista o las listas raíz. Permite acceder a la secuencia para crear una nueva lista. Esta lista

pertenecerá a la lista en la que se encuentre el usuario en dicho momento. Si el usuario se encuentra viendo las listas raíz, la nueva lista también será una lista raíz.

- El botón de **nueva entrada** se encuentra únicamente presente cuando el usuario está viendo una lista. Permite acceder a la secuencia para crear una nueva entrada. La nueva entrada pertenecerá a la lista en la que se encuentre el usuario en dicho momento.
- El botón **guardar** se encuentra únicamente presente en los pasos finales de las secuencias de edición y creación de elementos o permisos. Hacer que los cambios realizados se almacenen.
- El botón **permisos** se encuentra únicamente presente cuando el usuario está viendo un elemento. Permite acceder a la vista de los permisos existentes sobre dicho elemento. En esta vista no aparecen los usuarios que ni siquiera tienen permiso para poder ver el elemento (a no ser que antes sí lo tuviese, haya sido editado y aún no se hayan guardado los cambios).
- El botón **nuevo permiso** se encuentra únicamente presente cuando el usuario está viendo los permisos de un elemento. Permite acceder a la secuencia para dar a alguien permisos. El permiso de visión será obligatorio.
- El botón **borrar listas** o **borrar entrada** se encuentra únicamente presente en las vistas de listas y entradas, respectivamente. El botón elimina el elemento en el que se encuentra el usuario y le lleva al elemento al que pertenecía.
- Botones de listas, entradas y permisos. Representan el elemento o permiso que aparece resumido en el texto del botón. Al pulsarlos se accede a la vista de dicho elemento o permiso.

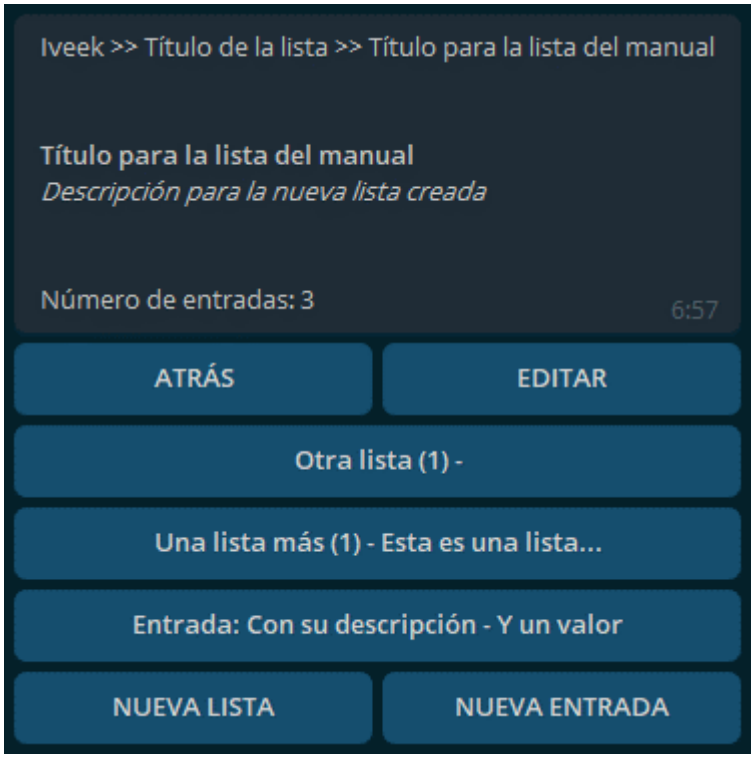

Ilustración 37 - Apariencia de la vista Vista Lista en web Telegram

En las secuencias de creación o edición será necesario que el usuario introduzca datos (ya sea por teclado o enviando algún archivo). En estas interacciones el bot enviará mensajes de texto para indicar qué campo se requiere en cada momento. Una vez finalizadas esas secuencias es recomendable eliminar los mensajes relacionados, para mantener la interfaz al final del chat.

Capítulo 6. Conclusiones

6.1 Sobre el proyecto

Durante el desarrollo del proyecto ha sido necesario modificar la planificación en diversas ocasiones, lo que ha producido ajuste en la carga de trabajo que se ha ido previendo. Sin embargo, incluso con esos cambios se ha conseguido alcanzar y completar los objetivos propuesto al comienzo del proyecto:

- **Implementación de un sistema de ficheros para gestionar el material.** Objetivo alcanzado en su totalidad. A parte del uso previsto para el sistema de ficheros, por la manera en la que se ha llevado a cabo también es posible utilizarlo para otras tareas en las que un sistema simple de directorios y ficheros podría ser una solución o una herramienta útil.
- **Desarrollo de u n *plug-in* para Moodle que se conecte con el bot y permita al menos añadir elementos en una lista seleccionada.** Objetivo alcanzado en su totalidad. El *plug-in* permite navegar por el árbol de listas accesible desde la lista asociada al curso y da la opción de añadir entradas en cualquier parte del árbol.
- **Sistema de suscripción a listas para recibir notificaciones cuando son modificadas.** Objetivo alcanzado en su totalidad. El bot permite a los usuarios suscribirse a cualquier lista que puedan ver y les enviará un mensaje cuando una lista a la que están suscritos reciba algún tipo de cambio.

6.2 Sobre los conocimientos adquiridos

Durante la realización del proyecto se han utilizado diversas herramientas y metodologías para avanzar y, en ocasiones, subsanar los obstáculos que han ido apareciendo. Debido a esto, considero que ha aumentado mi desempeño con el uso de estas herramientas, además de haber tocado de cerca algunos temas que no había previsto en un primer momento. Estas son las herramientas utilizadas y qué me han aportado:

- **PyCharm:** Es un IDE gratuito para desarrollar código utilizando Python. Gracias a la facilidad para importar bibliotecas desde internet, el terminal de rápido acceso y la sencillez para depurar el código mi conocimiento sobre el funcionamiento de Python y las bibliotecas empleadas (*python-telegram-bot* y *PyMySQL*, ambas interesantes por su potencialidad) ha aumentado en gran medida.
- **MySQL Workbench:** Esta herramienta permite gestionar de una manera bastante cómoda instancias de MySQL Server y las bases de datos desarrolladas en estas. El uso de la interfaz gráfica permite realizar pruebas en poco tiempo para comprobar los sistemas diseñados, así que me he familiarizado más con este tipo de aplicaciones y con MySQL.

- **Astah:** Aunque se ha utilizado en muchas ocasiones en diversas asignaturas durante el Grado, todavía hay opciones y atajos que permiten realizar diseños y modelado de una manera mucho más eficiente.
- **Microsoft Office Word:** Empleado para realizar la memoria, debido a la extensión de esta ha habido tiempo más que suficiente para descubrir funciones que facilitan la redacción de informes. También me ha permitido dar un paso hacia el aprendizaje de Visual Basic, pues en alguna ocasión, para facilitar la tarea de dar formato, ha sido necesario desarrollar algún pequeño script.

Como aprendizaje personal ajeno a las herramientas utilizadas y enfocado más en la metodología, he podido concluir que a la hora de planificar un proyecto es recomendable intentar dejar un tiempo de margen al final del plazo con el que poder maniobrar en caso de que sea necesario. También he podido reafirmar la importancia de realizar una lectura de la documentación de los medios a utilizar antes de comenzar a utilizarlos, lo cual puede ahorrar tiempo y una gran cantidad de problemas a medida que transcurre el proyecto; además del aprendizaje personal.

6.3 Sobre los conocimientos previos

Pese a todos los nuevos conceptos aprendidos durante la realización de este proyecto, cabe destacar la cantidad de elementos que ya se habían presentado (en mayor o menor medida) durante el Grado en Ingeniería Informática. Por mencionar algunos de los más relevantes: planificación de proyectos, elicitación de requisitos, modelo de dominio, máquina de estados, diseño de interfaces u orientación a objetos.

6.4 Mejoras futuras

Dando por supuesta la finalización de las tareas restantes, aún hay una gran cantidad de mejoras que podría realizarse, tanto modificaciones como ampliación de las funcionalidades. Algunas de ellas:

- **Poder crear códigos de invitación:** Actualmente la única manera de dar a un usuario un permiso es añadiéndolo a mano desde la interfaz del bot. Permitir crear un código único para cada elemento y tipo de permiso y dar la posibilidad de introducir ese código para obtener ese permiso facilitaría enormemente la tarea de dar permisos similares a muchos usuarios. De esta manera podría crearse un código, compartirlo con todos a los que se quiera dar cierto permiso y que estos usuarios lo introduzcan en el bot. Habría que analizar si añadir un nuevo comando o colocar un botón extra en las listas padre de las afectadas. También sería positivo que los códigos no fuesen fáciles de adivinar.
- **Añadir más tipos de permisos:** En este momento solo existen permisos para ver, editar y borrar. Podría ser interesante que un usuario pudiese dar permiso únicamente de visionado a otros usuarios, pero que no pueda revocarlos. O permitir editar el valor de una entrada (para actualizar el archivo, por ejemplo), pero no editar el título o la descripción. Una mayor variedad de tipos de permiso podría permitir una mejor gestión de los elementos y de los propios permisos.
- **Dar soporte para múltiples idiomas:** Aunque sea algo relativamente sencillo, traducir el texto propio de las interfaces (botones como “atrás” o los mensajes del bot) a otros idiomas permitiría a estudiantes y profesores no hispanohablantes utilizar también la aplicación.

- **Permitir el uso del bot en grupos:** Utilizando los grupos de Telegram, podrían sincronizarse listas a grupos. De esta manera podría informarse de los cambios a una gran cantidad de usuarios a la vez, por ejemplo. También podría utilizarse en canales para este mismo fin.
- **Implementación de interfaces:** La aplicación, pese a estar dividida en varias clases, no hace uso de la potencialidad que ofrecen las interfaces. Si se planea mantener la aplicación durante bastante tiempo esta mejora sería importante implementarla.
- **Añadir ficheros de configuración:** Actualmente toda la configuración se encuentra dentro del código de la aplicación. Si se pretende usar la aplicación con el fin que ha sido diseñada, toda esa información debería almacenarse en ficheros que permitan un acceso y una modificación rápida sin tener que estar deteniendo la ejecución del servicio u otros problemas similares.
- **Análisis y posible modificación de cómo se almacenan los permisos en la base de datos:** Actualmente se almacenan en una única tabla, con una entrada por cada usuario-estructura en la que están los valores para los tipos de permiso existentes. Si se pretenden añadir más tipos de permiso podría ser una idea a analizar el separar la tabla actual en más tablas.

Bibliografía

[1]	Moodle. (n.d), «Computer Desktop Encyclopedia. (1981-2015),» 15 Junio 2018. [En línea]. Available: https://encyclopedia2.thefreedictionary.com/Moodle.
[2]	Moodle, «moodle.net,» 15 Junio 2018. [En línea]. Available: https://moodle.net/stats/.
[3]	Telegram Messenger LLP, «Telegram FAQ,» 16 Junio 2018. [En línea]. Available: https://telegram.org/faq#q-how-are-secret-chats-different.
[4]	Telegram Messenger LLP, «Telegram FAQ,» 16 Junio 2018. [En línea]. Available: https://telegram.org/faq#q-what-is-telegram-what-do-i-do-here.
[5]	Telegram Messenger LLP, «Bots: An introduction for developers,» 16 Junio 2018. [En línea]. Available: https://core.telegram.org/bots.
[6]	Telegram Messenger TTL, «Telegram Bot API,» 17 Junio 2018. [En línea]. Available: https://core.telegram.org/bots/api#available-types.
[7]	Telegram Messenger TTL, «Telegram Bot API,» 17 Junio 2018. [En línea]. Available: https://core.telegram.org/bots/api#available-methods.
[8]	European Comission, «European Comission,» 27 Junio 2018. [En línea]. Available: http://ec.europa.eu/education/ects/users-guide/key-features_en.htm#ectsTop.
[9]	Deloitte, «Deloitte,» 6 julio 2018. [En línea]. Available: https://www2.deloitte.com/content/dam/Deloitte/be/Documents/tax/TaxStudiesAndSurveys/EuropeanSalarySurvey2017.PDF.
[10]	Google, «Precios de Cloud SQL,» 9 julio 2018. [En línea]. Available: https://cloud.google.com/sql/pricing#1st-gen-pricing.
[11]	Google, «Google Cloud Platform Pricing Calculator,» 9 julio 2018. [En línea]. Available: https://cloud.google.com/products/calculator/.
[12]	Oracle, «Oracle Cloud,» 9 julio 2018. [En línea]. Available: https://cloud.oracle.com/es_ES/cost-estimator.
[13]	Amazon, «Orecios de Amazon RDS para MySQL,» 9 julio 2018. [En línea]. Available: https://aws.amazon.com/es/rds/mysql/pricing/.
[14]	Digital Ocean, «Digital Ocean,» 9 junio 2018. [En línea]. Available: https://www.digitalocean.com/.
[15]	Digital Ocean, «Digital Ocean,» 9 julio 2018. [En línea]. Available: https://www.digitalocean.com/pricing/.
[16]	Amazon, «AWS Lambda,» 10 julio 2018. [En línea]. Available: https://aws.amazon.com/es/lambda/.
[17]	Amazon, «Precios de AWS Lambda,» 10 julio 2018. [En línea]. Available: https://aws.amazon.com/es/lambda/pricing/.
[18]	Google, «Precios de App Engine,» 10 julio 2018. [En línea]. Available: https://cloud.google.com/appengine/pricing.
[19]	Google, «Google Cloud,» 10 julio 2018. [En línea]. Available: https://cloud.google.com/appengine/docs/flexible/.
[20]	python-telegram-bot, «Git,» 10 julio 2018. [En línea]. Available: https://github.com/python-telegram-bot/python-telegram-bot#installing.
[21]	MySQL, «MySQL,» 10 julio 2018. [En línea]. Available: https://dev.mysql.com/doc/mysql-windows-excerpt/5.5/en/mysql-installer.html.
[22]	MySQL, «MySQL and Windows,» 10 julio 2018. [En línea]. Available: https://dev.mysql.com/doc/mysql-windows-excerpt/5.5/en/preface.html.

[23]	Moodle, «Moodle Mobile features,» 16 Junio 2018. [En línea]. Available: https://docs.moodle.org/33/en/Moodle_Mobile_features.
[24]	Moodle, 16 Junio 2018. [En línea]. Available: https://moodle.net/stats/.
[25]	Indeed, «Indeed,» 6 julio 2018. [En línea]. Available: https://www.indeed.es/salaries/Ingeniero/a-inform%C3%A1tico/a-Salaries?period=monthly.